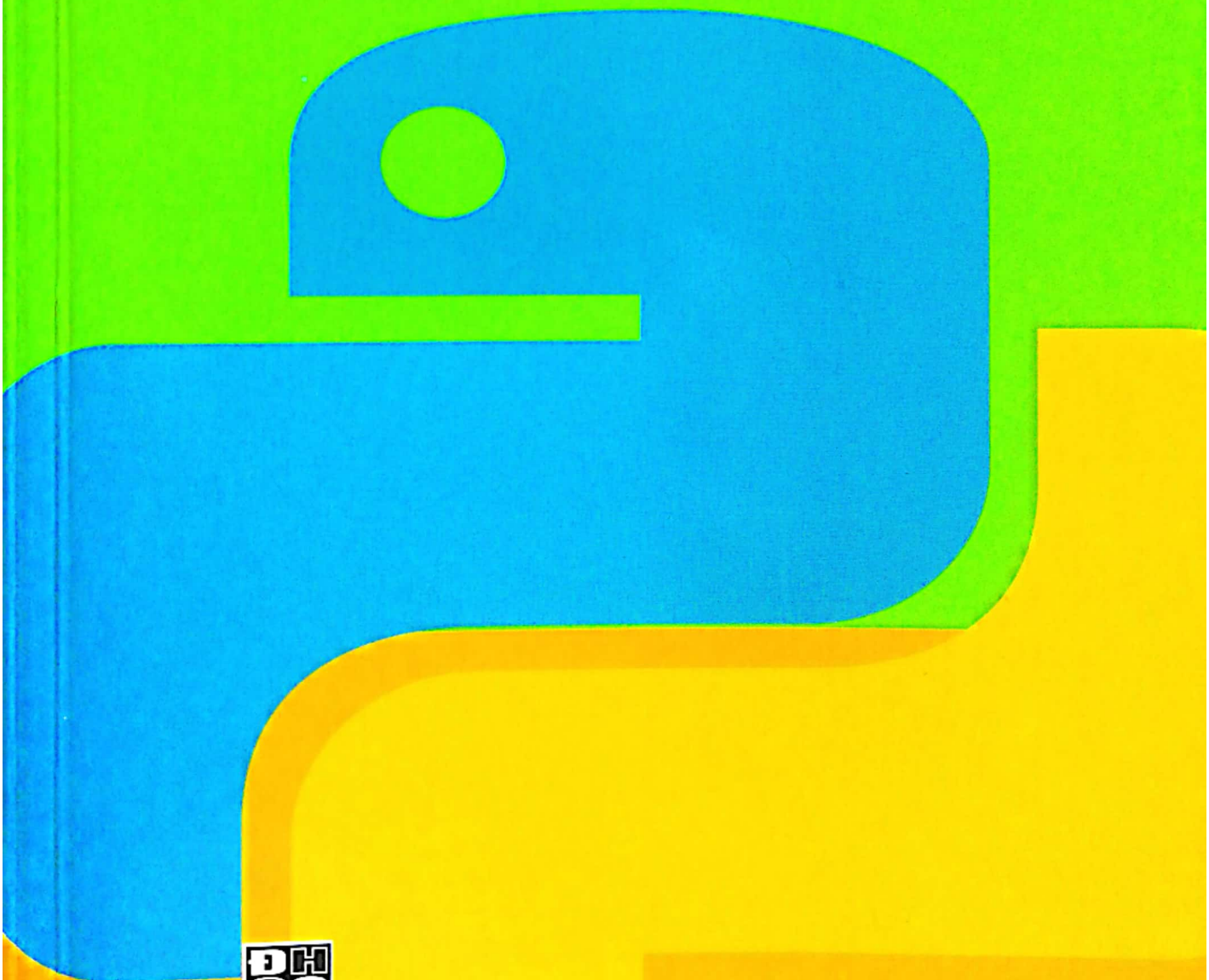
TS. NGUYỄN VĂN HẬU - TS. NGUYỄN DUY TÂN
ThS. NGUYỄN THỊ HẢI NĂNG - ThS. NGUYỄN HOÀNG ĐIỆP

PYTHON CƠ BẢN



ĐHQG Hà Nội
NHÀ XUẤT BẢN ĐẠI HỌC QUỐC GIA HÀ NỘI

TS. NGUYỄN VĂN HẬU – TS. NGUYỄN DUY TÂN
ThS. NGUYỄN THỊ HẢI NĂNG – ThS. NGUYỄN HOÀNG ĐIỆP

PYTHON CƠ BẢN

NHÀ XUẤT BẢN ĐẠI HỌC QUỐC GIA HÀ NỘI

LỜI NÓI ĐẦU

Cách mạng công nghiệp tạo nên sự phát triển vượt bậc của nền công nghiệp, kéo theo sự thay đổi cơ bản các điều kiện kinh tế, văn hóa và kỹ thuật,... trong toàn xã hội. Xã hội loài người đã trải qua ba cuộc cách mạng công nghiệp. Lần thứ nhất (cuối thế kỉ XVIII) bắt đầu bởi nền sản xuất cơ giới (thay thế nền sản xuất cơ bắp được sử dụng từ bao nhiêu năm trước). Cuộc cách mạng công nghiệp thứ hai (cuối thế kỉ XIX) được đánh dấu bằng sự ra đời của động cơ đốt trong và máy móc sử dụng điện. Cuộc cách mạng công nghiệp lần thứ ba (thập niên 1960) ra đời bởi sự phát triển của chất bán dẫn, siêu máy tính, máy tính cá nhân và Internet.

Ba cuộc cách mạng đầu tiên, suy cho cùng vẫn chỉ là sự phát triển mang tính tự động và độ chính xác (ở mức rất cao) của máy móc ("cơ bắp"): lần thứ nhất (sản xuất cơ giới – sử dụng máy móc), lần thứ hai (sản xuất dây chuyền), lần thứ ba (cuộc cách mạng công nghệ cao). Hiện nay, nhiều nhà khoa học và nhà kinh tế đều cho rằng chúng ta đang đứng trước thời gian đầu của cuộc cách mạng công nghiệp lần thứ tư ("Công nghiệp 4.0"), được đánh dấu bởi công nghệ mới đột phá trong nhiều lĩnh vực: robotics, trí thông minh nhân tạo (AI), thực tế ảo (VR), tương tác thực tại ảo (AR), mạng xã hội, điện toán đám mây, di động, phân tích dữ liệu lớn (SMAC). Chúng ta có thể cảm nhận được cuộc cách mạng công nghiệp lần thứ tư là sự kế thừa toàn bộ những ưu điểm của các cuộc cách mạng trước đồng thời được nâng lên bởi một yếu tố chưa từng có: "trí thông minh nhân tạo" (Artificial Intelligence). Trong cuộc cách mạng 4.0 này, Trí tuệ nhân tạo và Máy học (Machine Learning) đóng vai trò tiên phong.

Cũng như mọi cuộc cách mạng công nghiệp trước đó, cách tốt nhất cho xã hội loài người là tìm hiểu và hòa nhập với nó, để rồi tận dụng nó làm thế mạnh cho mình. Lấy cảm hứng từ đó, chúng tôi viết cuốn sách "Python cơ bản"

nhằm cung cấp kiến thức và kĩ năng lập trình căn bản cho người học. Ngôn ngữ Python được coi là ngôn ngữ mạnh mẽ và hiệu quả nhất giúp người học có thể xây dựng được những chương trình Máy học và những ứng dụng Trí tuệ nhân tạo.

Một cách tương đối, có thể chia ra cấp độ của người lập trình thành các mức: cơ bản, trung và nâng cao. Cuốn sách dành cho mức độ đầu tiên – cơ bản, và phục vụ ba đối tượng để có thể tự học: i) cho những học sinh trung học phổ thông yêu thích lập trình; ii) cho những sinh viên (năm đầu và năm thứ hai) trong khối ngành kĩ thuật; iii) cho những người yêu thích lập trình mà chưa được học bài bản. Cuốn sách có ba mục tiêu: i) Cung cấp những nền tảng kiến thức lập trình cơ bản; ii) Giúp người học hình thành kĩ năng lập trình; iii) Khởi đầu cho sự đam mê và yêu thích công nghệ.

Tại sao lại là Python? Có nhiều lý do để Python ngày càng được ưa chuộng. Thứ nhất, Python là một ngôn ngữ rất dễ học cho người mới bắt đầu. Python có cú pháp, cấu trúc đơn giản và dễ đọc. Một học sinh phổ thông cũng có thể tự học được Python (thông qua những cuốn sách như thế này). Điều này là vô cùng quan trọng, Python giúp người mới học lập trình tránh được rất nhiều phiền toái (khi so sánh với các ngôn ngữ khác) vì cú pháp, khai báo, luật phức tạp... không cần thiết, từ đó giúp chúng ta tập trung vào cách thức giải quyết vấn đề. Thứ hai, Python là ngôn ngữ ngày càng phổ dụng và được ứng dụng ngày càng nhiều (công ty Google, Facebook, Instagram, Dropbox đã xây dựng nhiều nền tảng sử dụng Python); và Python có thể xây dựng bất kì ứng dụng nào mà ngôn ngữ khác làm được. Thứ ba, Python có thư viện phong phú có thể kết hợp với nhiều ngôn ngữ lập trình khác (C, C++, Java, C#, PHP, R,…), cộng đồng sử dụng lớn trong nhiều ứng dụng khác nhau (Web Development – ứng dụng web và internet, Data science – khoa học dữ liệu, Machine learning – Máy học, Statistics – thống kê và khoa học).

Trong quá trình viết cuốn sách này, chúng tôi gặp khó khăn về kiến thức ít hơn rất nhiều so với việc tìm cách trình bày, lựa chọn nội dung để truyền tải Python theo cách thân thiện và sinh động. Các tác giả đã cân nhắc rất kĩ để "hi sinh" sự chính xác và đầy đủ, nhằm đổi lại sự dễ dàng và vui thích cho người học. Một lưu ý quan trọng là cuốn sách này nhằm giúp người học có kĩ năng, tức là có khả năng lập trình khi học xong cuốn sách, chứ không chỉ đọc để biết. Và điều này sẽ đạt hiệu quả cao nhất khi người học tiếp cận theo cách: đọc lướt, lập trình (coding), sửa lỗi, và đọc kĩ. Bạn nào chưa học lập trình nên chú ý điều này: việc phát sinh lỗi và sửa lỗi là việc không thể tránh khỏi khi học lập trình. Đó là dấu hiệu bạn đang đi… đúng hướng! Với những người chưa có kinh nghiệm và muốn học lập trình thực sự, gõ lại/tự làm các bài là cách duy nhất và hiệu quả nhất để tích lũy kĩ năng và có được kiến thức chắc chắn nhất.

Cuốn sách chắc chắn có nhiều điểm cần bổ sung và chỉnh sửa. Mọi ý kiến đóng góp xin gửi về địa chỉ email: nvhau66@gmail.com. Các tác giả chúc các bạn tìm thấy nhiều niềm vui trong quá trình khám phá kiến thức và kĩ năng mới.

Các tác giả

MỤC LỤC

Trang

LỜI NÓI ĐẦU 3

Bài 1. BẮT ĐẦU VỚI PYTHON

1.1. PYTHON LÀ GÌ? 13

1.2. CÀI ĐẶT PYTHON 15

1.3. "HELLO WORLD!" 16

1.4. CẤU TRÚC CHƯƠNG TRÌNH PYTHON 24

1.5. GHI CHÚ (COMMENTS) 25

1.6. TỔNG KẾT BÀI HỌC 26

1.7. BÀI TẬP 27

Bài 2. BIẾN, SỐ VÀ CHUỖI

2.1. ĐẶT VẤN ĐỀ 29

2.2. BIẾN VÀ TÊN 32

2.3. CHUỖI 36

2.4. SỐ 55

2.5. TẠO ỨNG DỤNG 65

2.6. TỔNG KẾT BÀI HỌC 68

2.7. BÀI TẬP 69

Bài 3. CẤU TRÚC RẼ NHÁNH

3.1. ĐẶT VẤN ĐỀ 73

3.2. BIỂU THỨC LÔ-GÍC (BOOLEAN) 74

3.3. CẤU TRÚC RẼ NHÁNH VỚI (IF) 75

3.4. TẠO ỨNG DỤNG 81

3.5. TỔNG KẾT BÀI HỌC 84

3.6. BÀI TẬP 84

Bài 4. CẤU TRÚC LẶP

4.1. ĐẶT VẤN ĐỀ 91
4.2. CẤU TRÚC LẶP WHILE 94
4.3. CẤU TRÚC LẶP FOR 98
4.4. LỆNH BREAK VÀ CONTINUE 102
4.5. TẠO ỨNG DỤNG 106
4.6. TỔNG KẾT BÀI HỌC 109
4.7. BÀI TẬP 109

Bài 5. CẤU TRÚC DỮ LIỆU

5.1. ĐẶT VẤN ĐỀ 115
5.2. LIST 118
5.3. TỪ ĐIỂN (DICTIONARY) 126
5.4. TẠO ỨNG DỤNG 135
5.5. TỔNG KẾT BÀI HỌC 140
5.6. BÀI TẬP 141

Bài 6. HÀM

6.1. ĐẶT VẤN ĐỀ 147
6.2. HÀM LÀ GÌ? 150
6.3. CÁCH XÂY DỰNG MỘT HÀM 151
6.4. SỬ DỤNG HÀM 152
6.5. TRUYỀN THAM SỐ CHO HÀM 153
6.6. TẠO ỨNG DỤNG 158
6.7. TỔNG KẾT 165
6.8. BÀI TẬP 165

Bài 7. THAO TÁC VỚI TỆP DỮ LIỆU

7.1. ĐẶT VẤN ĐỀ 171
7.2. TỆP TRONG PYTHON 174
7.3. MỞ TỆP 175
7.4. GHI DỮ LIỆU VÀO TỆP 176
7.5. ĐỌC DỮ LIỆU TỪ TỆP 177
7.6. GIỚI THIỆU THƯ VIỆN PANDAS 180
7.7. TẠO ỨNG DỤNG VỚI TỆP DỮ LIỆU 184
7.8. TỔNG KẾT BÀI HỌC 188
7.9. BÀI TẬP 188

Bài 8. LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG VỚI PYTHON

8.1. LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG 193
8.2. ĐỊNH NGHĨA LỚP TRONG PYTHON 194
8.3. KHAI BÁO VÀ SỬ DỤNG BIẾN ĐỐI TƯỢNG TRONG PYTHON 199
8.4. TẠO ỨNG DỤNG 200
8.5. TỔNG KẾT BÀI HỌC 206
8.6. BÀI TẬP 208
TÀI LIỆU THAM KHẢO 215

DANH SÁCH BẢNG

Bảng 2.1: Các ký tự điều khiển thường dùng trong Python 358

Bảng 2.2: Bảng toán tử số học trong Python 58

Bảng 2.3: Bảng toán tử so sánh trong Python 58

Bảng 2.4: Bảng toán tử logic trong Python..... 59

Bảng 2.5: Bảng toán tử kết hợp trong Python 59

Bảng 2.6: Bảng toán tử thao tác bit trong Python 60

Bảng 2.7: Bảng thứ tự ưu tiên các toán tử trong Python 60

Bảng 7.1: Các chế độ mở tệp..... 176

DANH SÁCH HÌNH

Hình 1.1: Tính phổ biến của các ngôn ngữ lập trình trong năm 2018 14
Hình 1.2: Chương trình Python đã được cài đặt 16
Hình 1.3: Cửa sổ lệnh của Python 16
Hình 1.4: Kết quả chạy từng lệnh của Python 16
Hình 1.5: Kết quả chạy biểu thức của Python 17
Hình 1.6: Cửa sổ các thuộc tính hệ thống 17
Hình 1.7: Cửa sổ thiết lập biến môi trường 18
Hình 1.8: Kết quả chạy chương trình hello.py từ cửa sổ cmd 19
Hình 1.9: Trang web soạn thảo chương trình Python online 20
Hình 1.10: Trang web chạy chương trình Python online 20
Hình 1.11: Giao diện của PyCharm 21
Hình 1.12: Cửa sổ tạo đề án của PyCharm 22
Hình 1.13: Cửa sổ tạo tệp chương trình trong PyCharm 22
Hình 1.14: Cửa sổ nhập tên tệp chương trình trong PyCharm 23
Hình 1.15: Cửa sổ soạn thảo chương trình trong PyCharm 23
Hình 1.16: Cửa sổ chạy chương trình Hello 23
Hình 1.17: Cửa sổ hiển thị kết quả chạy chương trình Hello 24
Hình 3.1: Lưu đồ thuật toán cấu trúc if 75
Hình 3.2: Lưu đồ thuật toán cấu trúc if– else 77
Hình 3.3: Lưu đồ thuật toán cấu trúc if–elif–else 79

Hình 4.1: Lưu đồ thuật toán cấu trúc while 95

Hình 4.2: Lưu đồ thuật toán cấu trúc for 99

Hình 4.3: Lưu đồ thuật toán cấu trúc lặp có chứa lệnh break 102

Hình 4.4: Lưu đồ thuật toán cấu trúc lặp có chứa lệnh continue 104

Bài 1

BẮT ĐẦU VỚI PYTHON

Chào mừng đến với Python. Python là một ngôn ngữ lập trình mạnh mẽ nhưng lại dễ tiếp cận và sử dụng.

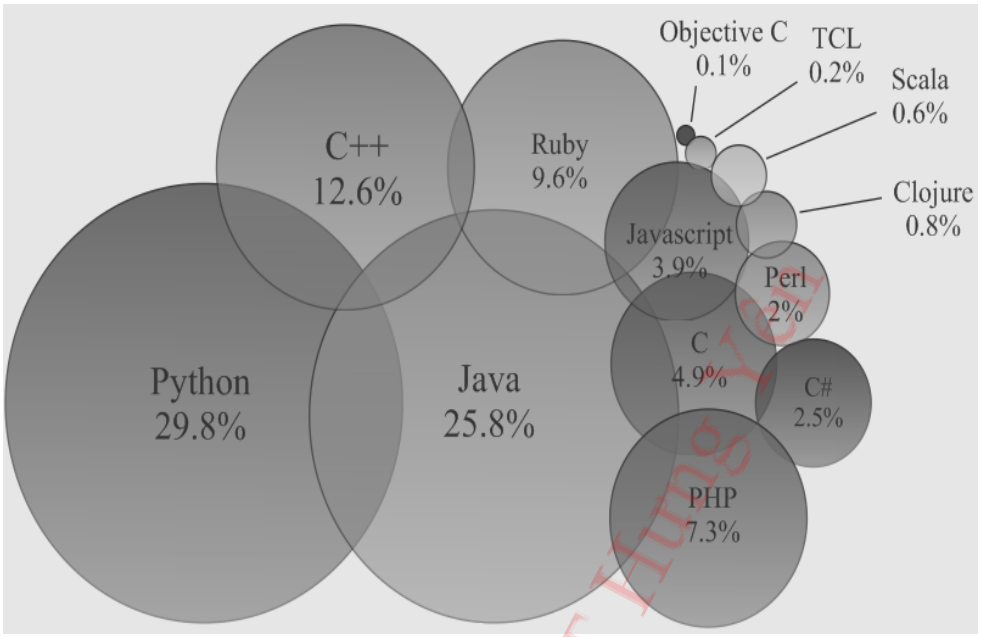
Sau bài học này, người học có thể:

- *Biết thông tin về ngôn ngữ lập trình Python*
- *Biết cách cài đặt môi trường cho Python*
- *Xây dựng và chạy được một chương trình Python ('Hello World!')*
- *Chỉ ra được cấu trúc một chương trình Python*

1.1. PYTHON LÀ GÌ?

Python là một ngôn ngữ lập trình bậc cao hướng đối tượng được Guido van Rossum cùng các cộng sự tạo ra năm 1991 dành cho mục đích lập trình đa năng. Python được thiết kế với ưu điểm mạnh là câu lệnh ngắn gọn, dễ nhớ, dễ hiểu. Cấu trúc chương trình của Python rõ ràng, dễ đọc và viết hơn rất nhiều so với những ngôn ngữ lập trình khác (như C, C++, Java, C#, PHP). Do đó Python được coi là một trong những ngôn ngữ thuận tiện nhất cho người mới học lập trình. Đây là ngôn ngữ lập trình thông dịch, có thể chạy trên nhiều hệ điều hành khác nhau như: Windows, Mac OS, OS/2, Linux và các hệ điều hành khác thuộc họ Unix.

Python hoàn toàn tạo kiểu động và dùng cơ chế cấp phát bộ nhớ tự động; do vậy nó tương tự như Perl, Ruby, Scheme, Smalltalk, và Tcl. Python là ngôn ngữ có mã nguồn mở, và đang có cộng đồng người dùng lớn. Hình 1.1 thể hiện sự phổ biến của Python so với các ngôn ngữ lập trình khác.



Hình 1.1: Tính phổ biến của các ngôn ngữ lập trình trong năm 2018

Một số đặc điểm nổi bật của ngôn ngữ Python:

Đơn giản, dễ học: Python có cú pháp đơn giản, rõ ràng. Nó dễ đọc và viết hơn với những ngôn ngữ lập trình khác (như C, C++, Java, C#, PHP). Python giúp việc lập trình trở nên thú vị, không bị gò bó và cho phép lập trình viên tập trung vào những giải pháp mà không phải tập trung vào cú pháp.

Miễn phí, mã nguồn mở: Bạn có thể tự do sử dụng và phân phối mã chương trình Python, thậm chí là dùng cho mục đích thương mại. Python là ngôn ngữ mã nguồn mở do đó bạn không những có thể sử dụng các phần mềm, chương trình được viết trong Python mà còn có thể thay đổi mã nguồn của nó theo ý thích của mình. Hơn nữa, Python có một cộng đồng lớn người dùng và cộng đồng này không ngừng cải thiện, phát triển.

Khả chuyển: Các chương trình Python có thể chuyển từ hệ điều hành này sang hệ điều hành khác mà không cần bất kỳ thay đổi nào, chương trình có thể chạy tốt trên các hệ điều hành như Windows, macOS, Linux.

Khả năng mở rộng và khả năng nhúng: Giả sử bạn có một ứng dụng đòi hỏi sự phức tạp lớn, bạn có thể dễ dàng kết hợp các khối, mã lệnh bằng C, C++

và những ngôn ngữ khác (có thể gọi được từ C) vào một chương trình của Python. Điều này cung cấp cho ứng dụng của bạn những tính năng tốt hơn cũng như ngôn ngữ kịch bản mà những ngôn ngữ lập trình khác khó có thể làm được.

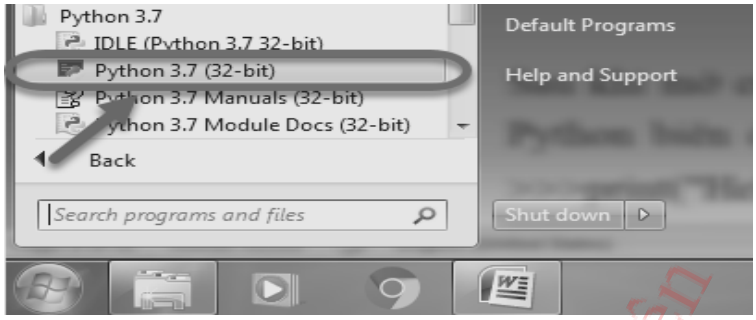
Ngôn ngữ thông dịch cấp cao: Không giống như C hay C++, với Python bạn không phải lo lắng về những vấn đề như tràn bộ nhớ, quản lý bộ nhớ, dọn dẹp rác hay dữ liệu vô nghĩa khác sau khi chạy chương trình. Chương trình dịch của Python sẽ tự động dọn dẹp, giải phóng bộ nhớ giúp chúng ta sau khi hoàn thành chương trình, do đó bạn không cần lo lắng về bất kỳ hoạt động gì ở cấp thấp.

Thư viện ứng dụng lớn giúp giải quyết các công việc phổ biến: Python có một số lượng lớn thư viện tiêu chuẩn, giúp cho công việc lập trình của bạn trở nên dễ dàng hơn rất nhiều vì bạn không phải tự viết tất cả mã lệnh. Những thư viện này đã được kiểm tra kỹ càng và được sử dụng bởi cộng đồng phát triển. Do đó bạn có thể yên tâm với chương trình hay ứng dụng của bạn.

Hướng đối tượng: Python hỗ trợ người dùng phương pháp lập trình hướng đối tượng (Object Oriented Programming – OOP), điều đó giúp lập trình viên giải quyết những công việc phức tạp trong cuộc sống một cách trực quan. Hơn nữa, với OOP, bạn có thể quan sát, phân tích những vấn đề phức tạp trong tự nhiên thành những tập nhỏ hơn bằng cách tạo ra các lớp, các đối tượng trong ngôn ngữ lập trình để giải quyết công việc.

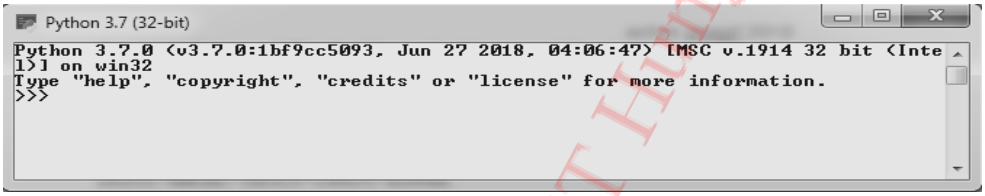
1.2. CÀI ĐẶT PYTHON

Trước khi có thể lập trình được Python, ta cần tải về một phiên bản Python (phiên bản hiện tại trong cuốn sách này là Python–3.7.0) về máy theo địa chỉ https: //www.Python.org/downloads/. Sau khi download, nhấn chuột kép vào tên phần mềm Python–3.7.0 để chạy cài đặt. Chú ý Python–3.7.0 yêu cầu cài trên hệ điều hành Windows từ Windows 7 trở lên. Sau khi cài đặt vào máy, từ màn hình Windows, chọn Start, chọn tiếp Program, chọn tiếp Python 3.7.0 như Hình 1.2.



Hình 1.2: Chương trình Python đã được cài đặt

Chạy chương trình Python ta có kết quả Hình 1.3.



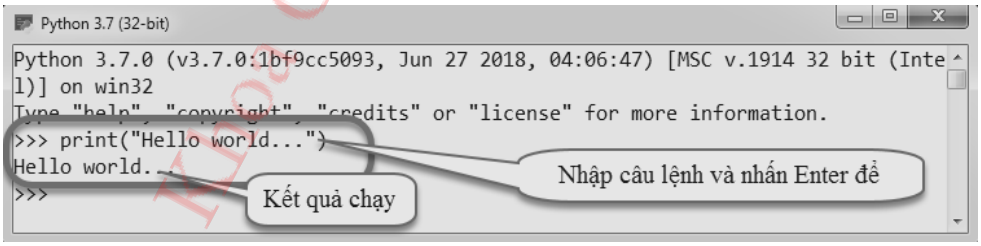
Hình 1.3: Cửa sổ lệnh của Python

1.3. "HELLO WORLD!"

Có thể chạy chương trình Python bằng nhiều cách, chúng ta hãy tìm hiểu từng cách thức một.

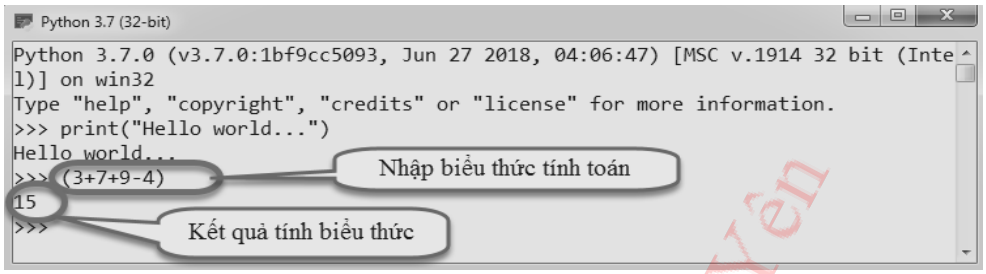
1.3.1. Làm việc với cửa sổ lệnh của Python

Sau khi mở cửa sổ lệnh của Python như Hình 1.2 và Hình 1.3, bạn có thể nhập vào các lệnh, biểu thức để chạy Python biên dịch và chạy trực tiếp trên cửa sổ này, ví dụ, từ cửa sổ lệnh sau dấu nhắc lệnh >>> ta gõ vào: print("Hello world ...") ↵, ta có kết quả như Hình 1.4.



Hình 1.4: Kết quả chạy từng lệnh của Python

Tương tự như vậy, sau dấu nhắc lệnh >>> nhập vào: (3+7+9–4) ↵, ta có kết quả như Hình 1.5.

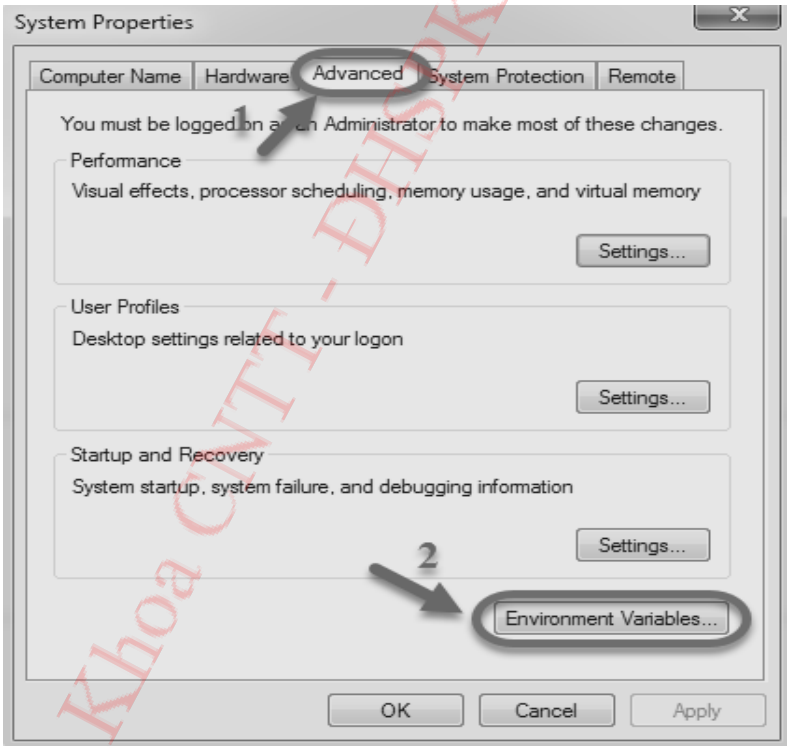


Hình 1.5: Kết quả chạy biểu thức của Python

1.3.2. Chạy chương trình Python từ cửa sổ dòng lệnh

Tạo biến môi trường cho Python:

- Bước 1: Vào Computer/chọn Properties/chọn Advanced System Setting/ta có cửa sổ sau:

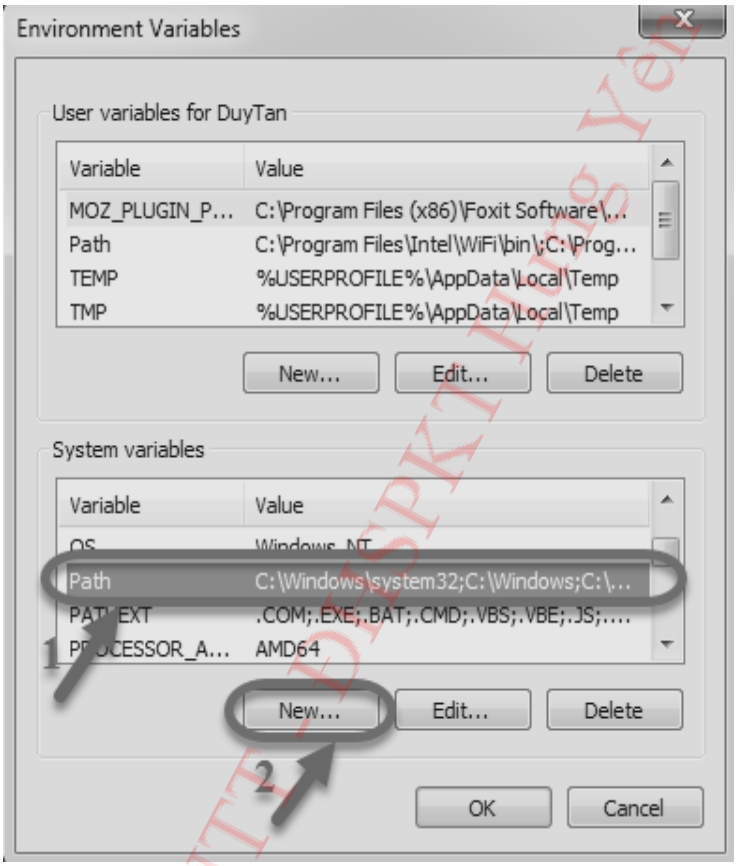


Hình 1.6: Cửa sổ các thuộc tính hệ thống

chọn Environment Variables.../ trong khung cửa sổ System valuables, chọn New.../nhập vào tên biến PythonPath, nhập đường dẫn đến thư mục chứa tệp Python.exe vào ô Variable value.... (ví dụ: C: \Python\Python37–32\) như:

Variable name: PythonPath

Variable value: C: \Python\Python37–32\



Hình 1.7: Cửa sổ thiết lập biến môi trường

- Bước 2: Tìm biến môi trường path trong khung cửa sổ System valuables/ chọn Edit, thêm vào tên biến PythonPath (biến ta vừa tạo hoặc thêm cả thư mục chứa tệp Python.exe vào đây)/sau đó chọn nút OK để hoàn thành.
- Bước 3: Vào cửa sổ cmd, chạy lệnh Python ↵, trên màn hình hiển thị dòng: Python 3.7.0 (v3.7.0: 1bf9cc5093, Jun 27 2018, 04: 06: 47)

[MSC v.1914 32 bit (Intel)] on win32 Type "help", "copyright", "credits" or "license" for more information.

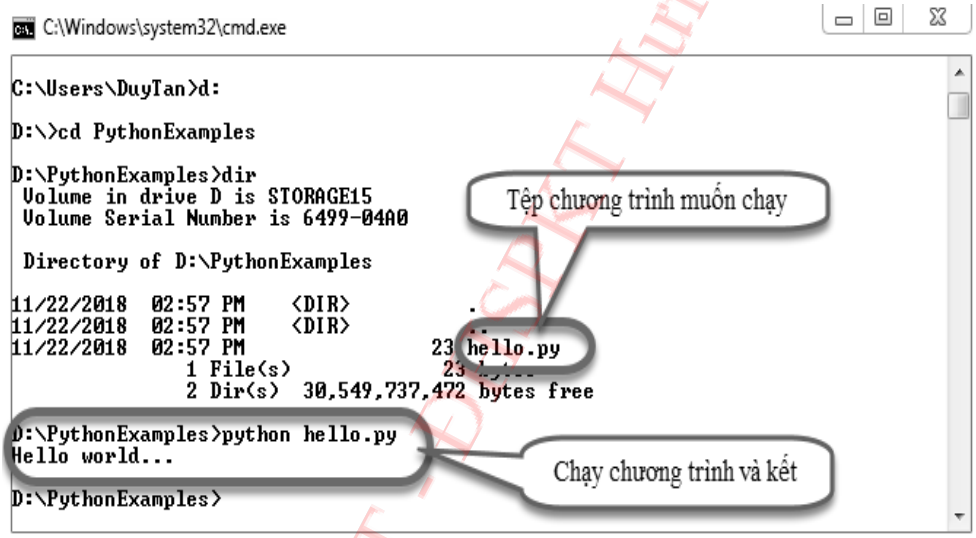
Chạy chương trình Python trên cửa sổ dòng lệnh:

- Bước 4: Chạy chương trình bằng lệnh sau: Python <tên tệp>↵

Ví dụ 1.1: Giả sử tệp hello.py trong thư mục D: \PythonExamples như sau:

hello.py
`print("Hello world...")`

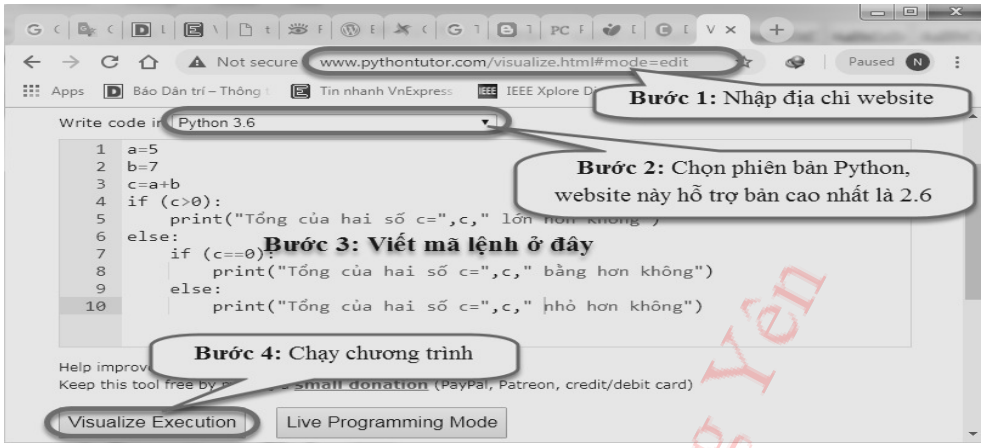
chạy chương trình bằng lệnh: Python hello.py↵ kết quả cho ra như Hình 1.8.



Hình 1.8: Kết quả chạy chương trình hello.py từ cửa sổ cmd

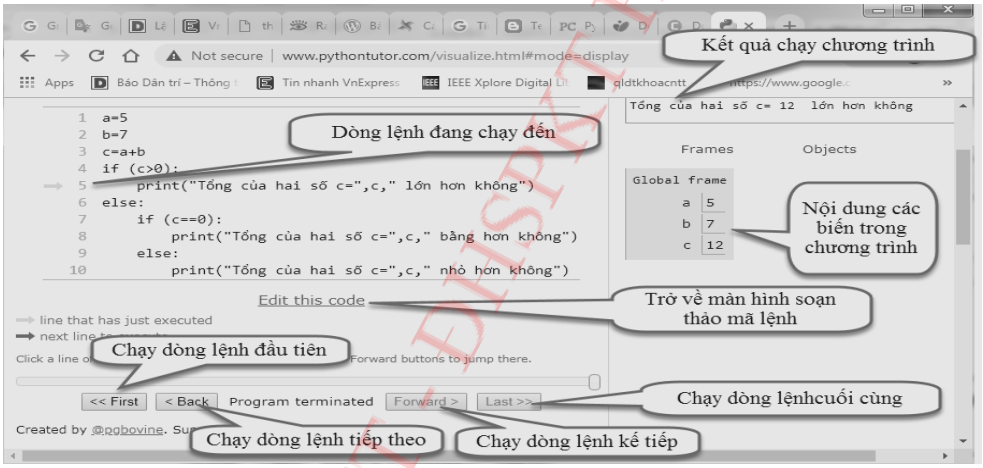
1.3.3. Chạy chương trình Python từ mạng Internet

- Bước 1: Mở trình duyệt web Google chrome hoặc Cốc–cốc sau đó nhập vào địa chỉ: *http: //www.Pythontutor.com/visualize.html#mode=edit.*
- Bước 2: Tại mục Write code in (viết trong mã lệnh): Chọn phiên bản Python hỗ trợ để chạy, ví dụ chọn version Python 2.7 hoặc Python 3.7.0.
- Bước 3: Soạn thảo mã lệnh bên dưới vùng "Write code in".



Hình 1.9: Trang web soạn thảo chương trình Python online

- Bước 4: Kích vào nút Visualize Exercution để chạy để chạy chương trình.



Hình 1.10: Trang web chạy chương trình Python online

Ý nghĩa của các nút lệnh:

– Chọn First để chạy dòng lệnh đầu tiên.

– Chọn Last để chạy dòng lệnh cuối cùng.

– Chọn Back để chạy dòng lệnh trước.

– Chọn Forward để chạy dòng lệnh tiếp theo.

Kết quả từng dòng lệnh hiển thị phần bên phải màn hình.

- Bước 5: Đóng cửa sổ trình duyệt để kết thúc.

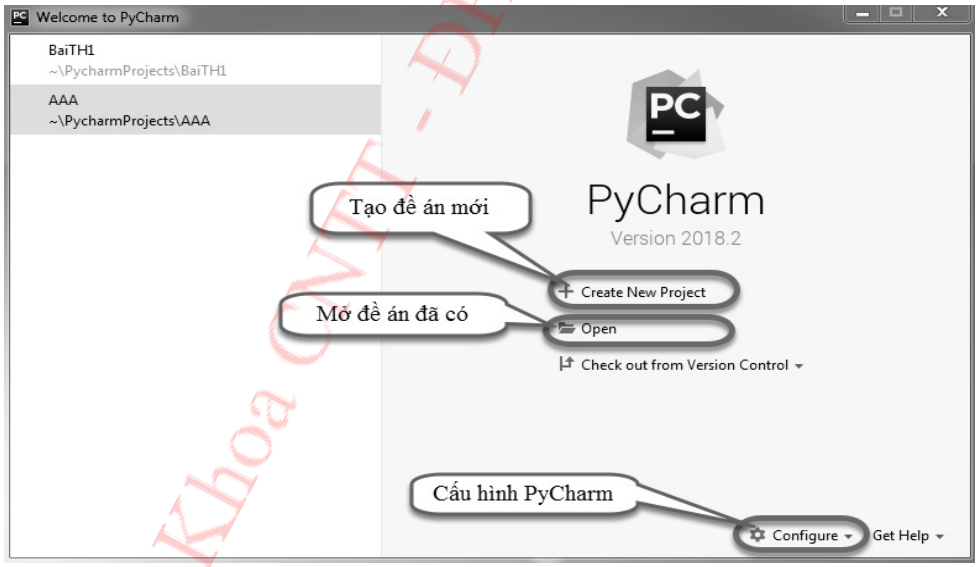
1.3.4. Môi trường lập trình PyCharm

**** Giới thiệu về PyCharm***

PyCharm là một môi trường phát triển tích hợp (IDE) phát triển ứng dụng chuyên nghiệp cho Python. PyCharm được lựa chọn và sử dụng bởi hầu hết các lập trình viên, các nhà nghiên cứu để xây dựng ứng dụng Python bởi nhiều lý do như: giao diện dễ dùng, giúp tăng hiệu xuất công việc khi viết các chương trình lớn,…

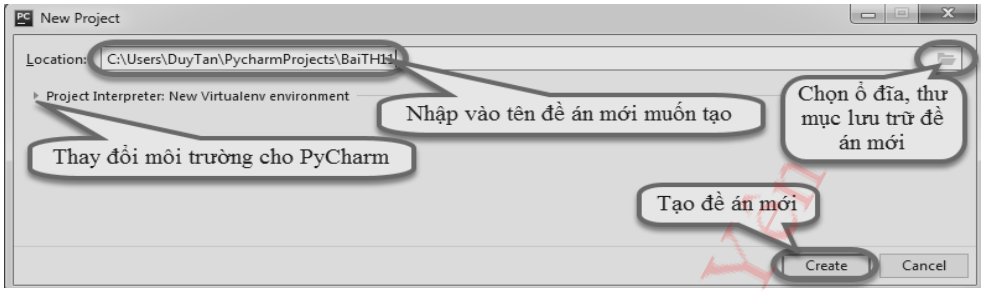
**** Cài đặt PyCharm***

- Bước 1: Tải phiên bản PyCharm mới nhất (phiên bản hiện tại cuốn sách này là PyCharm–Community–2018.2) về máy theo địa chỉ *https: //www.Python.org/downloads/.*
- Bước 2: Sau khi tải về máy, nhấn chuột kép vào tệp PyCharm–community–2018.2.exe để chạy. Sau khi cài đặt ta sẽ thấy xuất hiện ứng dụng "JetBrains PyCharm Community Edition 2018.2" trên máy.
- Bước 3: Tạo ứng dụng trên PyCharm: Từ cửa sổ (windows), chạy ứng dụng "JetBrains PyCharm Community Edition 2018.2", ta có cửa sổ màn hình sau:



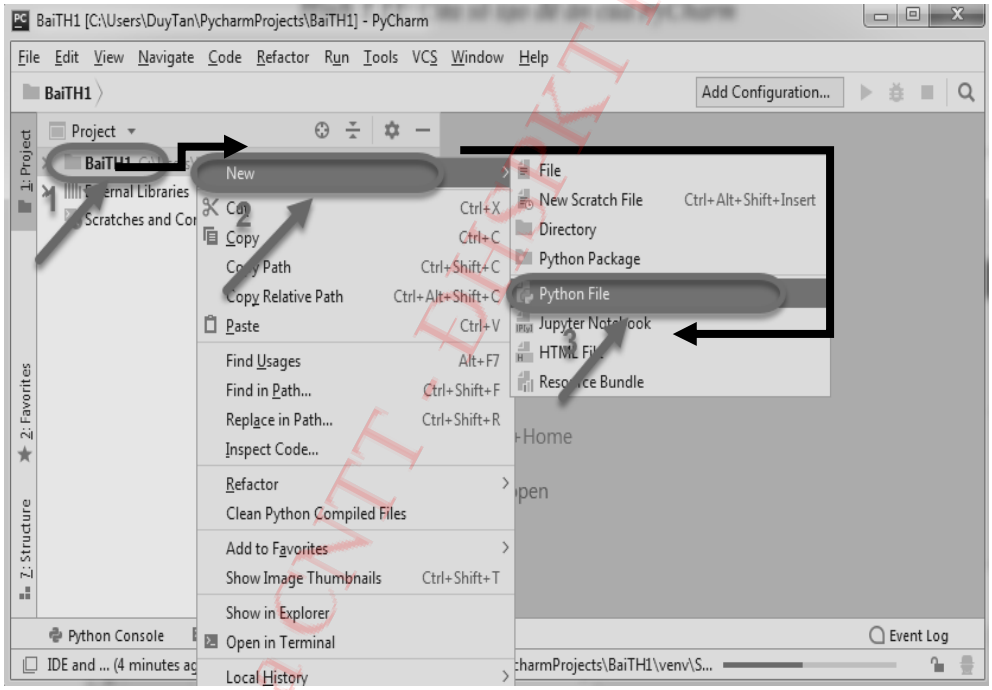
Hình 1.11: Giao diện của PyCharm

- Bước 4: Chọn mục "Create New Project" trên cửa sổ màn hình trong Hình 1.11 ta có màn hình hiển thị ra như sau:



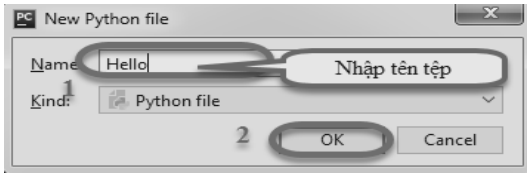
Hình 1.12: Cửa sổ tạo đề án của PyCharm

- Bước 5: Tạo tệp chương trình trong đề án: Nhấn chuột phải vào tên đề án, chọn New/Python File như Hình 1.13 như sau:



Hình 1.13: Cửa sổ tạo tệp chương trình trong PyCharm

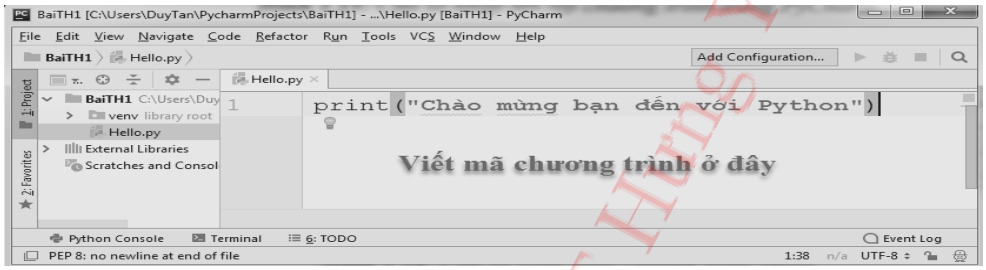
- Bước 6: Nhập vào tên tệp chương trình trong phần *Name* (một đề án trong PyCharm có thể chứa nhiều tệp chương trình) rồi chọn OK để hoàn tất.



Hình 1.14: Cửa sổ nhập tên tệp chương trình trong PyCharm

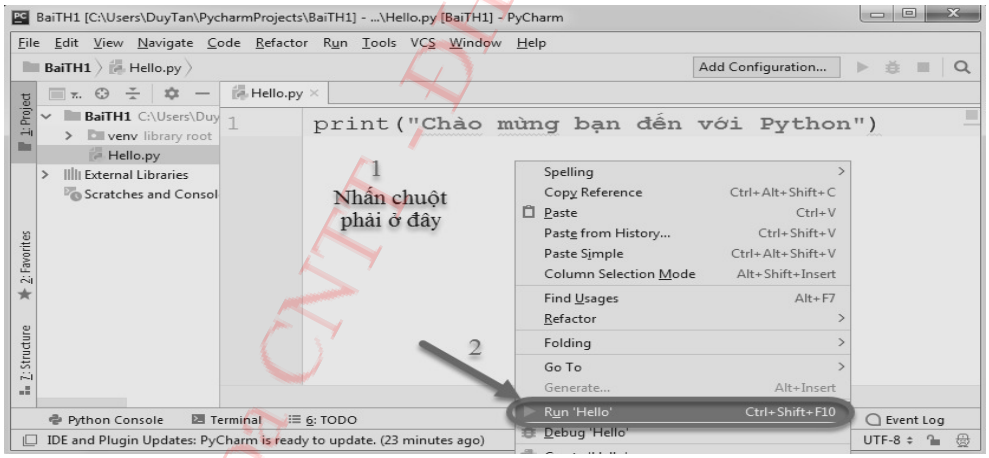
**** Soạn thảo và chạy chương trình trong PyCharm***

- Bước 7: Soạn thảo mã lệnh chương trình



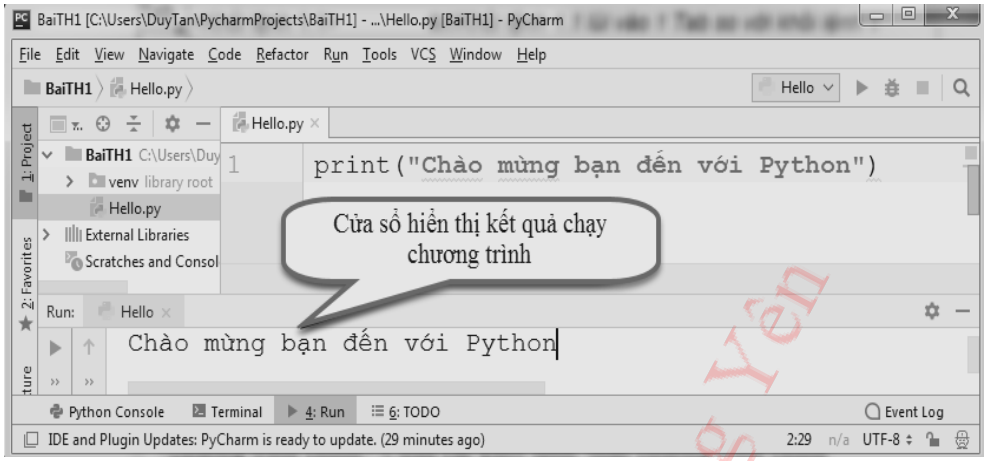
Hình 1.15: Cửa sổ soạn thảo chương trình trong PyCharm

- Bước 8: Biên dịch và chạy chương trình Hello.py bằng cách chọn nút "Run" trên thanh công cụ hoặc nhấn chuột phải trong vùng soạn thảo mã lệnh, chọn mục "Run" để chạy, hoặc nhấn tổ hợp phím Ctrl+Shift+F10.



Hình 1.16: Cửa sổ chạy chương trình Hello

Sau khi biên dịch và chạy chương trình, nếu chương trình có lỗi, Python sẽ thông báo vị trí (dòng) chương trình có lỗi, ngược lại Python mở một cửa sổ nhỏ bên dưới để hiển thị kết quả như Hình 1.17 dưới đây.



Hình 1.17: Cửa sổ hiển thị kết quả chạy chương trình Hello

1.4. CẤU TRÚC CHƯƠNG TRÌNH PYTHON

Cấu trúc chương trình Python bao gồm ba phần chính như sau:

#!/usr/bin/Python #Khai báo sử dụng tiếng Việt trong tệp *.py # -*- coding: utf8 -*- import<tên thư viện>#Khai báo sử dụng thư viện ...
#Định nghĩa các hàm def <tên hàm>([danh sách tham số]) .... #Cấu trúc giống với thân chương trình chính
#Phần thân chương trình chính <Câu lệnh 1>#Các câu lệnh được canh lề trái Tab #Khối lệnh 1.1 lùi vào 1 Tab/khoảng trống so với câu lệnh 1 [<Khối lệnh 1.1>] <Câu lệnh 2>#Các câu lệnh được canh lề trái Tab #Khối lệnh 2.1 lùi vào 1 Tab/ khoảng trống so với câu lệnh 2 [<Khối lệnh 2.1>]

Trong đó:

- *<tên thư viện>:* Là tên thư viện người dùng có sử dụng các hàm, lớp để phục vụ các công việc trong chương trình, ví dụ thư viện math, random, numpy, mathplotlib,...
- *<tên hàm>:* Là tên hàm (chương trình con) do người dùng định nghĩa.
- *<chương trình chính>:* Chứa các khối lệnh, thân chương trình chính.
- Khoảng trống khối lệnh cần thụt vào so với lệnh chứa nó thông thường hay dùng một khoảng Tab.

1.5. GHI CHÚ (COMMENTS)

Không chỉ ngôn ngữ Python, các ngôn ngữ lập trình đều có phần ghi chú, nó dùng để giải thích cho dòng lệnh hoặc khối lệnh trong Python, chương trình biên dịch, thông dịch của Python sẽ bỏ qua các ghi chú này. Ghi chú giúp cho người đọc chương trình dễ hiểu hơn.

– Ghi chú trên một dòng, sử dụng ký tự "#" (ký tự dào) trước chuỗi cần ghi chú:

Ví dụ 1.2: Ghi chú trên một dòng lệnh

```
#Hiển thị ra màn hình chuỗi Hello world
print("Hello world...")
```

hoặc đoạn mã chương trình sau thực hiện phép gán giá trị a = 5 sau đó hiển thị ra màn hình.

```
#Gán giá trị cho biến a
a=5
#Hiển thị ra màn hình giá trị đang có trong biến a
print("Giá trị của a=")
```

– Ghi chú nhiều dòng, sử dụng """ """ (ba cặp nháy đôi liên tiếp để bắt đầu và ba cặp nháy đôi liên tiếp để kết thúc) hoặc ''' ''' (ba cặp nháy đơn liên tiếp để bắt đầu và ba cặp nháy đơn liên tiếp để kết thúc)

Ví dụ 1.3: Sử dụng ba cặp nháy đơn để ghi chú khối lệnh bên dưới:

```
'''
Đây là khối lệnh kiểm tra năm nhuận
Nếu year chia hết cho 4 mà không chia hết cho 100 hoặc chia hết cho 400
thì year là năm nhuận
'''
year=2011
if (year % 4==0 and year %100 !=0) or (year % 400 ==0):
   print("Năm: ",year,"  là năm nhuận")
else:
  print("Năm: ",year,"  không là năm nhuận")
```

Ví dụ 1.4: Sử dụng ba cặp nháy kép để ghi chú khối lệnh bên dưới

```
"""
Đoạn mã giải phương trình bậc nhất dạng: ax+b=0
Có 3 trường hợp có thể xảy ra
Nếu hệ số a =0 và hệ số b=0 thì phương trình có vô số nghiệm
Nếu hệ số a =0 và hệ số b !=0 thì phương trình vô nghiệm
Nếu hệ số a !=0 thì phương trình có nghiệm x=-b/a
"""
a = 0
b = 113
if (a == 0 and b == 0):
        print("Phương trình có vô số nghiệm")
elif(a == 0 and b != 0):
        print("Phương trình vô nghiệm")
else:
        print("Phương trình có nghiệm x=",-b/a)
```

1.6. TỔNG KẾT BÀI HỌC

Trong bài đầu tiên, người học cần nắm được:

- Các bước cài đặt, cấu hình biến môi trường,
- Một chương trình Python có thể được thực thi theo nhiều cách:

- Dịch và chạy chương trình trên cửa sổ dòng lệnh cmd của Windows;
- Chạy trực tiếp chương trình trên mạng Internet;
- Chạy trực tiếp từng dòng lệnh trên cửa sổ lệnh của Python;
- Chạy trên phần mềm lập trình chuyên nghiệp (môi trường tích hợp phát triển – IDE) PyCharm, đây là một công cụ rất dễ dùng và người học cần chú ý nắm bắt.

Các bài tiếp theo sẽ trình bày các thành phần của ngôn ngữ Python, các lệnh nhập, xuất dữ liệu cũng như các cấu trúc dữ liệu giúp bạn đọc có thể thao tác, lập trình thành thạo trên Python.

1.7. BÀI TẬP

A. Câu hỏi ôn tập

Câu 1.1: Cho biết các khẳng định nào sau đây là đúng?

A. Python là một ngôn ngữ lập trình bậc cao.

B. Python là một ngôn ngữ thông dịch.

C. Python là một ngôn ngữ hướng đối tượng.

D. Python hỗ trợ lập trình vi điều khiển.

Câu 1.2: Chọn các lựa chọn đúng với ngôn ngữ lập trình Python?

A. Python là ngôn ngữ lập trình mã nguồn mở.

B. Python có khả năng chạy trên nhiều hệ điều hành khác nhau như: Windows, Linux/Unix, OS/2, Mac và Android.

C. Chúng ta có thể nhúng các thư viện C/C++ vào Python.

D. Python có hỗ trợ soạn thảo mã lệnh và có thể chạy trực tuyến (online).

Câu 1.3: Chọn các đáp án đúng để liên kết các câu lệnh thành khối lệnh trong Python:

A. Cặp dấu ngoặc nhọn {}.

B. Cặp dấu ngoặc đơn ().

C. Căn lề trái các câu lệnh.

D. Dấu chấm phẩy.

Câu 1.4: Chọn các đáp án đúng về ghi chú thích trong Python:

A. Ký tự #.

B. Một dấu ngoặc đơn.

C. Một dấu ngoặc kép.

D. Ba dấu ngoặc đơn.

E. Ba dấu ngoặc kép.

Câu 1.5: Cho biết các khẳng định nào sau đây là đúng?

A. PyCharm là một ngôn ngữ lập trình.

B. PyCharm là giao diện lập trình tích hợp.

C. PyCharm là một ngôn ngữ hướng đối tượng.

D. PyCharm giúp biên dịch mã lệnh Python.

Câu 1.6: Làm sao để chạy một chương trình Python đã soạn thảo trên PyCharm?

A. Nhấn nút "Run" trên thanh công cụ.

B. Nhấn chuột phải trong vùng soạn thảo mã lệnh sau đó chọn mục "Run".

C. Nhấn phím F5 trên bàn phím.

D. Nhấn tổ hợp phím Ctrl+Shift+F10.

Câu 1.7: Cho biết khẳng định nào sau đây là đúng?

A. Tất cả các khai báo như khai báo thư viện, khai báo sử dụng tiếng Việt phải đặt trên đầu tệp chương trình Python và trước định nghĩa tất cả các hàm.

B. Khi định nghĩa tất cả các hàm, các câu lệnh trực thuộc phải được viết sát lề trái.

C. Khi định nghĩa tất cả các hàm, các câu lệnh trực thuộc phải được căn lề trái và lệch sang phải đầu dòng một khoảng trống so với lệnh bao nó.

D. Tất cả những điều trên đều đúng.

Câu 1.8: Cho biết các khẳng định nào sau đây là đúng?

A. Python sẽ bỏ qua chú thích khi dịch chương trình.

B. Chú thích bắt buộc phải có trong chương trình.

C. Chú thích dành cho các lập trình viên để hiểu rõ hơn về chương trình.

D. Chú thích làm thay đổi kết quả chạy dịch chương trình.

Câu 1.9: Hãy chọn các đáp án đúng:

A. Chú thích trên 1 dòng bắt đầu bởi dấu #.

B. Chú thích trên nhiều dòng có thể bắt đầu và kết thúc bởi 3 dấu nháy đơn.

C. Chú thích trên nhiều dòng có thể bắt đầu và kết thúc bởi 3 dấu nháy kép.

D. Tất cả những điều trên đều đúng.

Câu 1.10: Cho biết ưu điểm và nhược điểm khi sử dụng môi trường tích hợp phát triển PyCham so với phương pháp chạy chương trình trực tiếp trên Internet, chạy trên cửa sổ cmd và chạy trên cửa sổ dòng lệnh của Python.

B. Lập trình

Bài tập 1.1: Xây dựng chương trình hiển thị ra màn hình thông tin sau:

```
**************************************
*        LẬP TRÌNH CĂN BẢN           *
*        Ngôn ngữ: PYTHON            *
**************************************
```

Bài tập 1.2: Xây dựng chương trình hiển thị ra màn hình như sau:

```
* * * * *
*       *
*       *
* * * * *
```

Bài tập 1.3: Hãy in ra màn hình hình chữ nhật như dưới đây:

```
**********
**********
**********
**********
```

Bài tập 1.4: Xây dựng chương trình hiển thị ra màn hình các thông tin như sau:

Mời bạn chọn chức năng:

1. Đăng nhập hệ thống
2. Đăng xuất hệ thống
3. Thoát khỏi hệ thống

Bài tập 1.5: Xây dựng chương trình hiển thị thông tin của một người gồm: Họ tên, tuổi, giới tính, số điện thoại, email, địa chỉ, nghề nghiệp; mỗi thông tin trên một dòng.

Bài tập 1.6: Bạn An có 5000000 gửi vào ngân hàng, với lãi suất 0.8%/tháng. Hãy viết chương trình hiển thị ra màn hình thông tin sau:

Tên người gửi: An

Số tiền gửi: 5000000 vnđ

Lãi suất/tháng: 0.8%

Tiền lãi: <số tiền lãi>

Biết rằng <số tiền lãi> được tính theo công thức: 5000000*0.008.

Bài 2

BIẾN, SỐ VÀ CHUỖI

Để lưu trữ được dữ liệu trong thực tế, giống như các ngôn ngữ khác, Python cũng cần lưu trữ dữ liệu để thao tác với chúng thông qua các số và chuỗi. Bài học này sẽ cung cấp cho người học những kiến thức về biến, số, từ khóa, và chuỗi trong Python. Nó là nền tảng để người học sử dụng cho các bài phía sau.

Sau bài học này, người học có thể:

- *Biết cách đặt tên biến và sử dụng biến trong chương trình*
- *Giải thích và sử dụng từ khóa Python*
- *Sử dụng các hàm sẵn có trong chuỗi (string) trong Python*
- *Sử dụng được các số (số nguyên, số thực) trong Python*
- *Khai báo và sử dụng được thư viện (math)*
- *Tạo ứng dụng chuẩn hóa chuỗi họ tên*

2.1. ĐẶT VẤN ĐỀ

Bài toán: Hãy xây dựng chương trình để quản lý thông tin của một thí sinh gồm: Họ tên, năm sinh và điểm thi của thí sinh theo thang điểm 10. Hiển thị lại thông tin và kết quả thí sinh trượt hoặc đỗ, biết thí sinh trượt nếu điểm nhỏ hơn 5 và đỗ nếu điểm lớn hơn hoặc bằng 5.

Xác định yêu cầu bài toán:

– Nhập dữ liệu đầu vào là 3 biến họ tên *name*, năm sinh *year* và điểm *mark* kiểu thực

– Xuất dữ liệu đầu ra là kết quả thí sinh *trượt/đỗ*.

Nhận xét:

– Để giải quyết bài toán trên, trước tiên chúng ta cần phải tìm cấu trúc dữ liệu thích hợp cho bài toán.

– Lưu trữ các thông tin của thí sinh:

+ Họ tên gồm các chữ cái, Python hỗ trợ kiểu chuỗi (str) lưu trữ các ký tự nên kiểu dữ liệu chuỗi thích hợp lưu trữ dữ liệu này.
+ Năm sinh của 1 người là 1 số nguyên, Python hỗ trợ kiểu số nguyên int.
+ Điểm thi theo thang điểm 10 là 1 số hữu tỷ, Python hỗ trợ kiểu số thực float thích hợp lưu trữ dữ liệu này.
+ Thí sinh trượt hay đỗ phụ thuộc vào điểm lớn hơn hoặc bằng 5 là đúng hay sai. Python hỗ trợ kiểu lô-gíc nhận giá trị True hoặc False.

Nói chung, để giải một bài toán trên máy tính, chúng ta cần biểu diễn, lưu trữ nhiều biến với nhiều kiểu dữ liệu khác nhau. Python hỗ trợ lập trình viên thực hiện công việc này một cách đơn giản và hiệu quả.

2.2. BIẾN VÀ TÊN

2.2.1. Khái niệm biến

Biến là một vùng nhớ trong máy tính dùng để lưu trữ một giá trị dữ liệu có thể thay đổi được trong chương trình. Thông thường vùng nhớ này được quản lý bởi một tên (gọi là tên biến) và thay vì truy xuất đến vùng nhớ thì chúng ta truy xuất đến tên biến.

Tùy vào các kiểu dữ liệu mà ngôn ngữ lập trình Python sẽ cấp phát vùng nhớ khác nhau cho một biến, một biến có thể chứa kiểu dữ liệu như số nguyên (int), số thực (float), chuỗi (str), danh sách (list),.v.v

2.2.2. Khai báo biến

Không giống như các ngôn ngữ lập trình C/C++, C# hay Java, để khai báo biến trong Python ta không cần phải khai báo kiểu dữ liệu ta chỉ cần gán một giá trị cho tên biến, Python sẽ tự động suy ra kiểu dữ liệu tương ứng cho biến. Do đó, một biến trong Python có thể có nhiều kiểu dữ liệu khác nhau tùy thuộc vào giá trị mà ta gán cho biến ở từng đoạn mã chương trình. Chúng ta có thể dùng hàm type() để xem kiểu dữ liệu hiện thời của biến.

Cú pháp: *<Tên biến> = <Giá trị>*

Ví dụ 2.1: Khai báo và gán giá trị cho các biến như sau:

Vidu2_1.py

```
"""
    Khai báo và gán giá trị cho biến
"""
x = 15       #Khai báo biến có tên x và gán giá trị bằng 15
#Hiển thị ra màn hình kiểu dữ liệu của biến x trong Python
print('Với x =',x,' ta có kiểu dữ liệu: ',type(x))
x = "utehy"       #Gán giá trị chuỗi cho biến x bằng "utehy"
print('Với x =',x,' ta có kiểu dữ liệu: ',type(x))
x = True           # Gán giá trị lô-gíc cho biến x bằng True
print('Với x =',x,' ta có kiểu dữ liệu: ',type(x))
x = 2.5          # Gán giá trị số thực cho biến x bằng 2.5
print('Với x =',x,' ta có kiểu dữ liệu: ',type(x))
# Gán giá trị số phức cho biến x có phần thực bằng 23, ảo bằng 45
x = complex(23,45)
print('Với x =',x,' ta có kiểu dữ liệu: ',type(x))
```

Kết quả chạy đoạn mã trên lần lượt sẽ có kiểu dữ liệu của x như sau:

Kết quả chạy chương trình ***Vidu2_1.py***

```
Với x = 15  ta có kiểu dữ liệu:  <class 'int'>
Với x = utehy  ta có kiểu dữ liệu:  <class 'str'>
Với x = True  ta có kiểu dữ liệu:  <class 'bool'>
Với x = 2.5  ta có kiểu dữ liệu:  <class 'float'>
Với x = (23+45j)  ta có kiểu dữ liệu:  <class 'complex'>
```

Python còn cho phép chúng ta khai báo và gán một giá trị cho nhiều biến trên cùng một câu lệnh.

Ví dụ 2.2:

```
'''Khai báo các biến a,b,c nguyên và gán giá trị cho các biến bằng 12'''
a = b = c = 12
```

Thêm nữa, chúng ta cũng có thể gán các giá trị khác nhau cho các biến khác nhau trên cùng một câu lệnh như ***Ví dụ 2.3*** dưới đây.

Ví dụ 2.3:

```
a, b, c = 12,50.5, "Lập trình Python rất thú vị"
```

Khi đó, trình thông dịch Python sẽ lấy giá trị bằng 12 có kiểu nguyên (int) gán cho biến a, giá trị 50.5 có kiểu thực (float) gán cho biến b và giá trị "Lập trình Python rất thú vị" có kiểu chuỗi xâu (str) gán cho biến c.

2.2.3. Đặt tên biến

Tên là một định danh do người dùng đặt trong khi lập trình dùng để phân biệt các thành phần với nhau trong cùng một chương trình. Chúng ta có thể đặt tên biến, tên hằng, tên hàm (chương trình con), tên lớp (class), tên tệp, tên thư mục, v.v. Khi đặt tên, chúng ta cần quan tâm đến các quy tắc đặt tên trong mỗi ngôn ngữ lập trình.

Quy tắc đặt tên trong Python:

- Có thể sử dụng các chữ cái viết thường (từ a đến z), các chữ cái viết hoa (A đến Z), các chữ số (từ 0 đến 9) hoặc dấu gạch dưới (_) để đặt tên.
- Không được dùng các ký tự đặc biệt như: @, $, %,...
- Không được bắt đầu bằng một chữ số, ví dụ *1bien* là không hợp lệ, nhưng *bien1* thì hợp lệ.
- Python là ngôn ngữ lập trình có phân biệt chữ viết hoa và chữ viết thường, ví dụ a và A là hai tên khác nhau.
- Đặt tên không được trùng với các từ khóa của Python.

Ví dụ 2.4:

Vidu2_4.py

```
x = 10                          #Đặt tên biến là x
y = 22.5                        #Đặt tên biến là y
hoten =  "Lê Thị Quýt Mơ"       #Đặt tên biến là hoten
dia_chi = "31 Nguyễn Trãi Hưng Yên" #Đặt tên biến là dia_chi
def tinhTong(a, b): #Đặt tên hàm (chương trình con) là tinhTong
    return a+b
```

2.2.4. Từ khóa

Từ khóa là những từ dành riêng của ngôn ngữ lập trình Python, do đó trong các công việc lập trình tên đặt như tên biến, tên hằng, tên hàm,… không được trùng với các từ khóa này.

Dưới đây là tóm tắt danh sách các từ khóa của Python hay gặp trong cuốn sách này.

Từ khóa	Ý nghĩa
and	Toán tử và trong Python dùng để kết hợp 2 biểu thức logic
as	Chỉ định một bí danh được dùng thay cho tên
break	Dùng để thoát khỏi cấu trúc lặp ngay lập tức
class	Dùng để khai báo một lớp mới trong Python
continue	Quay lên đầu cấu trúc lặp và bỏ qua các lệnh sau nó
def	Dùng để khai báo hàm (chương trình con)
elif	Kiểm tra một điều kiện khác trong cấu trúc if... elif...
else	Nằm trong cấu trúc rẽ nhánh if... else
False	Là hằng sai, thường dùng để chỉ điều kiện sai
for	Dùng để bắt đầu cấu trúc lặp for
global	Từ khóa chỉ định khai báo biến toàn cục
if	Dùng để bắt đầu cấu trúc rẽ nhánh
import	Dùng để khai báo sử dụng các thư viện
in	Toán tử dùng để kiểm tra biến có trong một tập hợp hay không
is	Toán tử kiểm tra biến có bằng hay không, toán tử này tương đương với toán tử ==
not	Toán tử phủ định trong Python
or	Toán tử hoặc trong Python
return	Trả về giá trị sau khi kết thúc hàm trở về lệnh gọi

Từ khóa	Ý nghĩa
True	Là hằng đúng, thường dùng để chỉ điều kiện đúng
try	Dùng để bắt lỗi 1 khối lệnh có thể gây lỗi
while	Dùng để bắt đầu cấu trúc lặp while

2.3. CHUỖI

2.3.1. Khái niệm

Trong Python, chuỗi (str) là tất cả giá trị, dữ liệu được đặt trong cặp dấu nháy kép " ", hoặc cặp dấu nháy đơn ' ', hay cũng có thể dùng cặp ba dấu nháy đơn liên tiếp ''' ''' hoặc cặp ba dấu nháy kép liên tiếp """ """. Trong đó, nội dung đặt cặp ba dấu nháy kép liên tiếp hoặc cặp ba dấu nháy đơn liên tiếp sẽ được giữ nguyên định dạng. Do đó chúng ta có thể viết chuỗi trên nhiều dòng mà không cần dùng ký tự nối dòng với cặp ba dấu nháy đơn hoặc kép liên tiếp.

Thông thường, người ta hay sử dụng cặp nháy đơn ' ' hoặc nháy kép " " để biểu diễn một chuỗi ngắn và sử dụng cặp ba dấu nháy đơn hoặc kép liên tiếp để biểu diễn một chuỗi dài hoặc chuỗi có chứa nhiều các ký tự đặc biệt.

Ví dụ 2.5:

Vidu2_5.py

```
"Python"  #Chuỗi dữ liệu có giá trị bằng "Python"
"1234"    #Chuỗi dữ liệu có giá trị bằng "1234"
#Gán cho biến s một giá trị chuỗi bằng "Lập trình Python"
s = "Lập trình Python"
print(type(s)) # Xem kiểu dữ liệu của biến s sẽ thuộc lớp ‘str’
'''Đây là một chuỗi
được trình bày trên nhiều dòng'''    #Chuỗi trên nhiều dòng
```

Kết quả chạy chương trình ***Vidu2_5.py***

```
<class ‘str’>
```

2.3.2. Các ký tự đặc biệt trong chuỗi

Trong cuốn sách này chúng ta hiểu ký tự là một chuỗi có độ dài bằng một và dữ liệu chuỗi (đặt trong cặp nháy đơn hoặc nháy kép) có thể chứa các ký tự thông thường (a–z, A–Z, 0–9,...) hoặc các ký tự đặc biệt (!, @, *, &, %, (,),...). Tuy nhiên, khi chúng ta muốn biểu diễn các ký tự đặc biệt trong chuỗi, Python quy định sử dụng ký tự điều khiển "\" đặt trước ký tự đặc biệt đó, ví dụ khi ta đưa ra câu lệnh print("\a") thì ta sẽ nghe thấy một tiếng bíp được phát ra từ máy tính.

Bảng 2.1. Các ký tự điều khiển thường dùng trong Python.

Tên	Ký hiệu	Ý nghĩa
Alert	\a	Phát ra một tiếng bíp
Backspace	\b	Đưa con trỏ lùi về một khoảng trắng (một ký tự)
Newline	\n	Đưa con trỏ xuống dòng và về đầu dòng tiếp theo
Horizontal tab	\t	In một khoảng trắng bằng phím Tab
ASCII Carriage Return	\r	Đưa con trỏ về đầu dòng
Single quote	\'	In ra ký tự ‘
Double quote	\"	In ra ký tự "
Blackslash	\\	In ra ký tự \

Ví dụ 2.6:

Vidu2_6.py

```
'''Hiển thị ra màn hình:
Lập trình
Python
'''
print("Lập trình \n Python")
# Hiển thị ra màn hình C: \Python32\Bai2
print("C: \\Python37\\Bai2")
```

Kết quả chạy chương trình ***Vidu2_6.py***
Lập trình Python C: \Python37\Bai2

2.3.3. Các toán tử với chuỗi

Toán tử +: Thực hiện việc nối các chuỗi lại với nhau, toán tử này thường được sử dụng.

Cú pháp: *s1 + s2 (*với s1 và s2 đều là chuỗi)

Ví dụ 2.7:

Vidu2_7.py

```
#Khai báo và gán giá trị chuỗi "Hello " cho biến s1
s1 = "Hello "
s2 = "World..."
s12 = s1 + s2     #Biến chuỗi s12="Hello World..."
s1  = s1 + 'Python...'  # s1 = "Hello Python..."
print("s12 = ",s12)
print("s1 = ",s1)
```

Kết quả chạy chương trình ***Vidu2_7.py***
s12 = Hello World... s1 = Hello Python...

*Toán tử * :* Python hỗ trợ toán tử nhân rất đặc biệt, nó giúp chúng ta tạo ra một chuỗi lặp đi lặp lại với số lần bất kỳ.

Cú pháp: *s * n*

Trong đó:

- *s*: Là một giá trị chuỗi hoặc một biến chuỗi
- *n*: Là một số nguyên chỉ số lần lặp lại chuỗi s

Ví dụ 2.8:

Vidu2_8.py

```
s1 = "Hello"
# Tạo ra chuỗi bằng cách lặp lại chuỗi s1 2 lần
s = s1*2        # s = " HelloHello"
print("s = ",s)
s = "Hello"
'''Bất cứ chuỗi nào nhân với 0 (hoặc số âm) cũng đều có kết quả là một
chuỗi rỗng '''
s *= 0
print("s = ",s)
s = "Hello"
s = s*(-2)
print("s = ",s)
```

Kết quả chạy chương trình ***Vidu2_8.py***

```
s =  HelloHello
s =
s =
```

Toán tử in: Toán tử này kiểm tra xem chuỗi con có nằm trong chuỗi mẹ hay không, kết quả trả về bằng True nếu có hoặc False nếu không.

Cú pháp: *s1* **in** *s*

với s và s1 đều là chuỗi

Kiểm tra xem chuỗi *s1* có nằm trong chuỗi *s* hay không? Kết quả trả về bằng True nếu chuỗi *s1* có xuất hiện trong chuỗi *s*, ngược lại bằng False.

Ví dụ 2.9:

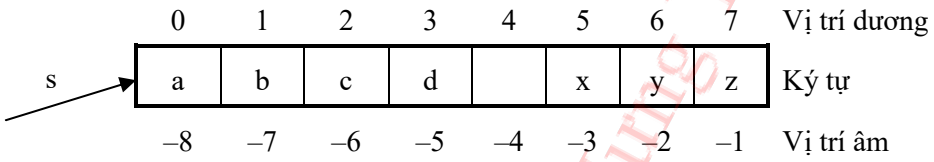
Vidu2_9.py

```
"P" in "Python"      #Kết quả trả về bằng True
"a" in "Python"      #Kết quả trả về bằng False
'ac' in 'abc'        #Kết quả trả về bằng False
```

2.3.4. Các thao tác với chuỗi

Vị trí các ký tự trong chuỗi: Trong một chuỗi của Python, các ký tự bên trong chuỗi được đánh số thứ tự từ 0 đến n – 1 hoặc từ –n đến –1 từ trái sang phải (với n là số ký tự có trong chuỗi).

Ví dụ 2.10: Giả sử ta có một biến kiểu chuỗi s = "abcd xyz", khi đó ký tự "a" sẽ có vị trí bằng 0 (hoặc –8), b sẽ có vị trí bằng 1 (–7) và z có vị trí bằng 7 (–1) trong s.



Truy xuất đến một phần tử trong chuỗi: Từ một biến kiểu chuỗi cho trước, ta có thể truy xuất đến bất cứ ký tự nào có trong chuỗi.

Cú pháp: *<Tên biến chuỗi>*[*<position>*]

Trong đó:

- *<Tên biến chuỗi>:* Là tên biến kiểu chuỗi hoặc một chuỗi, bắt buộc phải có
- *<position>:* Là một số nguyên chỉ vị trí của ký tự trong chuỗi

Ví dụ 2.11:

Vidu2_11.py

```
#Khai báo biến chuỗi và gán chuỗi 'abcd xyz' vào biến s
s = "abcd xyz"
print("s = ",s)
# s[0] hoặc s[-8], cho kết quả bằng ký tự "a"
print("s[0] = ",s[0])
# s[3] hoặc s[-4], cho kết quả bằng ký tự " " (dấu cách)
print("s[3] = ",s[3])
'''Truy xuất đến vị trí không có trong chuỗi, máy sẽ báo lỗi
      IndexError: string index out  of range '''
print("s[8] = ",s[8])
```

Kết quả chạy chương trình ***Vidu2_11.py***

```
s =  abcd xyz
Traceback (most recent call last):
s[0] =  a
s[3] =  d
print("s[8] = ",s[8])
IndexError: string index out of range
```

Truy xuất đến nhiều phần tử trong chuỗi: Dựa trên vị trí của các ký tự trong chuỗi, ta có thể lấy ra nhiều ký tự có trong chuỗi, đây là một điểm mạnh của Python so với các ngôn ngữ lập trình khác.

Cú pháp: *<Tên biến chuỗi>*[*<start>*:*<end>* : *<step>*]

Trong đó:

- *<Tên biến chuỗi>:* Là tên biến kiểu chuỗi hoặc một chuỗi, bắt buộc phải có.
- *<start>:* Là một số nguyên chỉ vị trí bắt đầu lấy trong chuỗi, nếu không có giá trị này, Python sẽ lấy ký tự có vị trí bằng 0.
- *<end>:* Là một số nguyên chỉ vị trí kết thúc lấy trong chuỗi bằng end – 1, nếu không có giá trị này, Python sẽ lấy đến ký tự có vị trí cuối cùng chuỗi.
- *<step>:* Là số bước nhảy sẽ lấy, nếu không có giá trị này, Python sẽ lấy bước nhảy bằng một.

Ví dụ 2.12: Lấy (trích, rút) chuỗi với số bước nhảy bằng 1

Vidu2_12.py

```
s = 'abcd xyz'
#Lấy các ký tự trong chuỗi s có vị trí từ 1 đến 5
s1 = s[1: 6]
print("s1 = ",s1) # s1 = "bcd x"
#Lấy các ký tự trong chuỗi s có vị trí từ 0 đến 2,
```

```
s2 = s[: 3]

print("s2 = ",s2) # s2 = "abc"
#Lấy toàn bỏ chuỗi s trừ ký tự cuối cùng

s3 = s[: -1]
print("s3 = ",s3) # s3 = "abcd xy"
# Lấy các ký tự có vị trí 1 đến hết chuỗi

s4 = s[1:] # s4 = "bcd xyz"
print("s4 = ",s4)
```

Kết quả chạy chương trình ***Vidu2_12.py***

```
s1 =  bcd x
s2 =  abc
s3 =  abcd xy
s4 =  bcd xyz
```

Ví dụ 2.13: Lấy (trích, rút) chuỗi với số bước nhảy lớn hơn 1

Vidu2_13.py

```
s = 'abcd xyz'
#Lấy các ký tự từ 1 đến 7 trong s với bước là 2

s1 = s[1: 7: 2]
print("s1 = ",s1) # s1 = "bdx"
```

Kết quả chạy chương trình ***Vidu2_13.py***

```
s1 =  bdx
```

Ở ví dụ trên, ta đặt số bước bằng 2, khi đó các vị trí được lấy trong khoảng từ 1 đến 6 đó là (2, 4, và 6) ta được các ký tự tương ứng là (bdx).

Ta có thể điều chỉnh việc lấy từ trái sang phải thành từ phải sang trái bằng việc đặt giá trị số bước nhẩy bằng một số âm.

Ví dụ 2.14:

Vidu2_14.py

```
s = 'abcd xyz'
#Lấy bắt đầu từ ký tự tại vị trí 3 trở về đến hết ký tự ở vị trí 1
s1 = s[3: 0: -1]
print("s1 = ",s1) # s1="dcb"
#Lấy bắt đầu từ ký tự tại vị trí 4 trở về đến hết ký tự ở vị trí 1
s2 = s[4: : -1]
print("s2 = ",s2) # s2 = "dcba"
```

Kết quả chạy chương trình ***Vidu2_14.py***

```
s1 =  dcb
s2 =   dcba
```

Chú ý: Chúng ta không được đặt số bước nhảy bằng 0, khi đó câu lệnh sau sẽ sinh ra lỗi.

Ví dụ 2.15:

Vidu2_15.py

```
s =  'abc xyz'
s1 = s[: : 0]     #Lệnh này sinh ra lỗi
        #ValueError: slice step cannot be zero
```

Kết quả chạy chương trình ***Vidu2_15.py***

```
ValueError: slice step cannot be zero
```

2.3.5. Thay đổi nội dung chuỗi

Trong các ngôn ngữ lập trình như Pascal, C, C++, chúng ta có thể thay thế giá trị các ký tự đang có trong chuỗi bằng một lệnh gán. Python không cho phép chúng ta thay đổi nội dung bên trong chuỗi. Tuy nhiên, chúng ta có thể sử dụng kỹ thuật trích rút chuỗi kết hợp với toán tử "+" để tạo ra một chuỗi mới thỏa mãn yêu cầu sau đó gán lại cho một biến cụ thể.

Ví dụ 2.16:

Vidu2_16.py

```
s = 'abcd xyz'
ss = "k" + s[1: ]
# Lấy các ký tự từ vị trí 1 đến hết chuỗi, ss = "kbcd xyz"
print("ss = ",ss)
```

Kết quả chạy chương trình ***Vidu2_16.py***

```
ss =  kbcd xyz
```

2.3.6. Định dạng chuỗi

Phần này sẽ đưa ra hai kiểu định dạng chuỗi cơ bản được sử dụng. Định dạng chuỗi sử dụng toán tử %. Kiểu định dạng này rất thông dụng nếu bạn đã từng làm việc với ngôn ngữ lập trình lập trình C.

Cú pháp: *<chuỗi>* % (*<giá trị 1>, <giá trị 2>,..., <giá trị n>*)

Trong đó:

- *<chuỗi>:* Là một giá trị chuỗi cần định dạng
- *<giá trị 1>, <giá trị 2>,..., <giá trị n>:* Là các giá trị cụ thể hoặc các biến chứa giá trị cần chèn vào chuỗi.

Ví dụ 2.17:

Vidu2_17.py

```
"Tên tôi là: %s" %("Python")
#Kết quả cho ta chuỗi: "Tên tôi là: Python"
a = 5
s = "Giá trị a = %d" %(a)#Kết quả cho ta chuỗi: "Giá trị a = 5"
print("s = ",s)
```

Kết quả chạy chương trình ***Vidu2_17.py***

```
s = Giá trị a = 5
```

Ví dụ 2.18:

Vidu2_18.py

```
a = 1  # Gán giá trị số nguyên 1 cho biến a
b = 2  # Gán giá trị số nguyên 2 cho biến b
f = 3.5  # Gán giá trị số thực 3.5 cho biến f
s = "444"  # Gán giá trị chuỗi "444" cho biến s
# Hiển thị dữ liệu ra màn hình
print("a=%d \nb = %d \nf=%f \ns = %s \n" % (a, b, f, s))
```

Kết quả chạy chương trình ***Vidu2_18.py***

```
a=1
b = 2
f=3.500000
s = 444
```

Một số ký hiệu cơ bản dùng sau toán tử % trong Python

Ký hiệu	Ý nghĩa
%s	Định dạng cho dữ liệu kiểu chuỗi (str)
%d	Định dạng cho dữ liệu kiểu số nguyên hoặc số thực bỏ phần thập phân
%[n]f	Định dạng cho dữ liệu kiểu số thực với n là số chữ số phần thập phân
%r	Định dạng cho dữ liệu chứa cả chuỗi các số

Định dạng bằng phương thức format: Phương pháp này là sử dụng phương thức format() được Python cung cấp sẵn để định dạng.

Cú pháp: *"<chuỗi {}...>"*.format*(<giá trị 1>, <giá trị 2>,..., <giá trị n>)*

Trong đó:

- *<chuỗi {}...>:* Là một giá trị chuỗi cần định dạng
- *<giá trị 1>, <giá trị 2>,..., <giá trị n>:* Là các giá trị cụ thể hoặc các biến chứa giá trị cần chèn vào chuỗi.

Ví dụ 2.19:

Vidu2_19.py

```
a=1
b=2
c=3
x=4.5
#Định dạng chuỗi hiển thị 3 số ra màn hình
print("a={0}, b={1}, x={2}".format(a,b,x))
```

Kết quả chạy chương trình ***Vidu2_19.py***

```
a=1, b=2, x=4.5
```

Ví dụ 2.20: Ta cũng có thể thay đổi thứ tự hiển thị các thành phần trong việc định dạng chuỗi:

Vidu2_20.py

```
a=1
b=2
c=3
x=4.5
#Định dạng chuỗi hiển thị 3 số ra màn hình
print("x={2}, a={0}, b={1}".format(a,b,x))
```

Kết quả chạy chương trình ***Vidu2_20.py***

```
x=4.5, a=1, b=2
```

Thêm nữa, ta cũng có thể đặt một biểu thức tính toán vào trong phương thức format() như ví dụ sau:

Ví dụ 2.21:

Vidu2_21.py

```
a=5.5
b=2.5
#Định dạng chuỗi hiển thị 2 số và một biểu thức ra màn hình
print("a={0}, b={1}, a*b={2}".format(a,b,a*b))
```

Kết quả chạy chương trình ***Vidu2_21.py***
a=5.5, b=2.5, a*b=13.75

2.3.7. Các phương thức xử lý chuỗi

Python là một ngôn ngữ lập trình hướng đối tượng, cho nên, các thành phần biểu diễn trong ngôn ngữ đều thuộc một lớp (class) hoặc một đối tượng (object) nào đó (chúng tôi sẽ giới thiệu lập trình hướng đối tượng trong bài 8. Thành phần chuỗi cũng không phải ngoại lệ, nó thuộc vào lớp str. Phương thức là một hàm thành phần trong lớp, nó tương tự như hàm (chương trình con) trong lập trình hướng thủ tục, để sử dụng nó chúng ta cần phải gọi thông qua một đối tượng được khai báo cụ thể.

Phương thức count(): Đếm số lần xuất hiện của một chuỗi con trong chuỗi, giá trị trả về sau lời gọi hàm là một số nguyên.

Cú pháp: *<chuỗi>*.count*(<substr>[,start [,end]])*

Trong đó:

- *<chuỗi>:* Là một biến hoặc giá trị chuỗi chứa chuỗi con cần đếm
- *<substr>:* Là một chuỗi con hoặc một ký tự cần đếm
- *[,start]:* Là một số nguyên chỉ định vị trí bắt đầu đếm trong chuỗi, nếu không có giá trị này, Python sẽ đếm từ ký tự có vị trí bằng không.
- *[,end]:* Là một số nguyên chỉ định vị trí kết thúc đếm trong chuỗi, nếu không có giá trị này, Python sẽ đếm đến ký tự cuối cùng trong chuỗi.

Ví dụ 2.22:

Vidu2_22.py
#Đếm số dấu cách (ký tự trắng) trong trong chuỗi và gán vào biến n n="**Lập trình Python không khó**".count(" ") *#Hiển thị n ra màn hình* print("**Số dấu cách n =**",n)

Kết quả chạy chương trình ***Vidu2_22.py***
Số dấu cách n = 4

Ví dụ 2.23:

Vidu2_23.py

```
#Khai báo và gán một chuỗi cho biến s
s ="Lập trình Python"
#Đếm số ký tự bằng 't' trong s từ ký tự thứ 2 đến hết 14
n = s.count('t',1,15)
print("Số ký tự 't' trong s=",n)
```

Kết quả chạy chương trình ***Vidu2_23.py***

```
Số ký tự 't' trong s = 2
```

Phương thức capitalize(): Viết hoa ký tự đầu tiên trong chuỗi, các ký tự còn lại viết thường. Giá trị trả về sau lời gọi hàm là một chuỗi.

Cú pháp: <*chuỗi*>.capitalize()

Trong đó: <*chuỗi*>*:* Là một chuỗi hoặc một biến kiểu chuỗi, hay có thể gọi nó là một đối tượng thuộc lớp chuỗi vì khi ta kiểm tra bằng lệnh type(<*chuỗi*>*)* Python sẽ thông báo nó thuộc kiểu lớp str.

Ví dụ 2.24:

Vidu2_24.py

```
s ="lập trình Python" #Khai báo biến chuỗi s và gán chuỗi cho biến s
s1 = s.capitalize()   #Viết hoa ký tự đầu tiên của chuỗi s
print(s1)         #Hiển thị chuỗi s1 ra màn hình
print(type(s))    #Hiển thị kiểu của biến s ra màn hình
```

Kết quả chạy chương trình ***Vidu2_24.py***

```
Lập trình python
<class 'str'>
```

Phương thức upper(): Chuyển tất cả các ký tự thường trong chuỗi thành hoa, giá trị trả về sau lời gọi hàm là một chuỗi.

Cú pháp: <*chuỗi*>.upper()

Ví dụ 2.25:

Vidu2_25.py

```
s ="lập trình Python"
#Chuyển tất cả các ký tự thường thành hoa và hiển thị ra màn hình
print(s.upper())
```

Kết quả chạy chương trình ***Vidu2_25.py***

```
LẬP TRÌNH PYTHON
```

Phương thức lower (): Chuyển tất cả các ký tự hoa trong chuỗi thành ký tự thường, giá trị trả về sau lời gọi hàm là một chuỗi.

Cú pháp: <*chuỗi*>.lower ()

Ví dụ 2.26:

Vidu2_26.py

```
s ="Lập Trình Python"
print(s.lower())#Chuyển các ký tự trong biến s sang ký tự thường
```

Kết quả chạy chương trình ***Vidu2_26.py***

```
lập trình python
```

Phương thức title*(): Viết hoa các ký tự đầu tiên* của mỗi từ trong chuỗi, giá trị trả về sau lời gọi hàm là một chuỗi.

Cú pháp: <*chuỗi*>.title()

Ví dụ 2.27:

Vidu2_27.py

```
s ="lập trình Python rất thú vị"
#Hiển thị chuỗi s với các ký tự đầu của các từ được viết hoa
print(s.title())
```

Kết quả chạy chương trình ***Vidu2_27.py***

```
Lập Trình Python Rất Thú Vị
```

Phương thức lstrip (): Xóa bỏ tất cả các ký tự trắng (dấu cách) hoặc ký tự được chỉ định ở đầu chuỗi (từ bên trái). Giá trị trả về sau lời gọi hàm là một chuỗi.

Cú pháp: *<chuỗi>*.lstrip*([chars])*

Trong đó:

- *<chuỗi>:* Là một biến hoặc giá trị chuỗi chứa ký tự cần xóa
- *[chars]:* Là ký tự cần xóa, hoặc ký tự trắng (dấu cách) nếu không chỉ định tham số này.

Ví dụ 2.28:

Vidu2_28.py

```
s ="*****Lập Trình Python*****"
hoten = "  Lê Thị Quýt Cam   "
#Xóa tất cả các ký tự "*" ở đầu (bên trái) chuỗi trong biến s
# và hiển thị biến s ra màn hình
print(s.lstrip("*"))
# Xóa tất cả các ký tự trắng " " ở đầu (bên trái) chuỗi trong biến hoten
# và hiển thị hoten ra màn hình
print(hoten.lstrip())
```

Kết quả chạy chương trình ***Vidu2_28.py***

```
Lập Trình Python*****

Lê Thị Quýt Cam
```

Phương thức strip (): Xóa bỏ tất cả các ký tự trắng hoặc ký tự được chỉ định ở đầu, cuối chuỗi (bên trái, phải chuỗi). Giá trị trả về sau lời gọi phương thức là một chuỗi.

Cú pháp: *<chuỗi>*.strip*([chars])*

Ví dụ 2.29:

Vidu2_29.py

```
s ="*****Lập Trình Python*****"
hoten = "      Lê Thị Quýt Cam      "
#Xóa tất cả các ký tự "*" ở hai bên trái và phải trong biến s
# và hiển thị s ra màn hình
print(s.strip("*"))
#Xóa tất cả các ký tự trắng " " ở hai bên trái và phải trong biến hoten
# và hiển thị hoten ra màn hình
print(hoten.strip())
```

Kết quả chạy chương trình ***Vidu2_29.py***

```
Lập Trình Python

Lê Thị Quýt Cam
```

Phương thức split(): Tách một chuỗi thành nhiều chuỗi con, các chuỗi con sau khi tách được lưu thành một danh sách. Giá trị trả về sau lời gọi hàm là một danh sách (list) (chúng ta sẽ trao đổi về list ở Bài 5).

Cú pháp: *<chuỗi>*.split*(<char> [,n])*

Trong đó:

- *<chuỗi>:* Là một chuỗi cần tách thành nhiều chuỗi con
- *<char>:* Là một ký tự ngăn cách các chuỗi con với nhau, nếu không có tham số này, Python sẽ dùng dấu cách (ký tự trắng) để tách.
- *n:* Chỉ định số lần tách, nếu n = –1 (mặc định), Python sẽ không giới hạn việc tách (tách hết)

Ví dụ 2.30:

Vidu2_30.py

```
s = "Khoa công nghệ Thông tin"
'''
Tách toàn bộ chuỗi s thành nhiều chuỗi con,
các chuỗi con phân tách nhau bởi dấu cách
'''
khoa = s.split()
#Hiển thị danh sách khoa ra màn hình
print("khoa=",khoa)
'''
Tách chuỗi s thành 3 chuỗi con,
các chuỗi con phân tách nhau bởi dấu cách
'''
khoa = s.split(" ",2)
print("khoa=",khoa)
```

Kết quả chạy chương trình ***Vidu2_30.py***

```
khoa= ['Khoa', 'công', 'nghệ', 'Thông', 'tin']
khoa= ['Khoa', 'công', 'nghệ Thông tin']
```

Phương thức join(): Nối các chuỗi con thành một chuỗi. Các chuỗi con là các phần tử trong các kiểu dữ liệu tuple, list, set,… Giá trị trả về sau lời gọi hàm là một chuỗi.

Cú pháp: *<ký tự nối>*.join*(<tham số>)*

Trong đó:

- *<ký tự nối>:* Là một ký tự sẽ nối với từng chuỗi con (các phần tử) trong tham số truyền vào phương thức.
- *<tham số>:* Là một danh sách, tập hợp chứa chuỗi con (các phần tử) cần nối.

Ví dụ 2.31:

Vidu2_31.py

```
#Sử dụng một tuple để chứa các chuỗi con
s = ("Khoa", "công", "nghệ", "Thông", "tin")
#Nối các phần tử của s thành một chuỗi
khoa = " ".join(s)#Sử dụng ký tự nối là một dấu cách " "
print("khoa = ",khoa)  #Hiển thị biến khoa ra màn hình
khoa = "-".join(s)     #Sử dụng ký tự nối là một dấu trừ "-"
print("khoa=",khoa)
khoa = "".join(s)      #Không sử dụng ký tự nối
print("khoa=",khoa)
```

Kết quả chạy chương trình ***Vidu2_31.py***

```
khoa = Khoa công nghệ Thông tin
khoa = Khoa-công-nghệ-Thông-tin
khoa = KhoacôngnghệThôngtin
```

Phương thức find(): Tìm kiếm chuỗi con trong một chuỗi. Giá trị trả về là vị trí đầu tiên xuất hiện chuỗi con khi tìm từ trái sang phải, nếu chuỗi con không có trong chuỗi thì kết quả trả về sẽ là –1.

Cú pháp: <*chuỗi*>.find*(substr [, start[, end]])*

Trong đó:

- <*chuỗi*>*:* Là một biến hoặc giá trị chuỗi chứa chuỗi con cần tìm kiếm.
- <*substr*>*:* Là một chuỗi con hoặc một ký tự cần tìm kiếm trong chuỗi.
- *[,start]:* Là một số nguyên chỉ vị trí bắt đầu tìm trong chuỗi, nếu không có giá trị này thì Python sẽ tìm từ ký tự đầu tiên trong chuỗi.
- *[,end]:* Là một số nguyên chỉ vị trí kết thúc tìm trong chuỗi, nếu không có giá trị này thì Python sẽ tìm đến ký tự cuối cùng trong chuỗi.

Ví dụ 2.32:

Vidu2_32.py

```
#Khai báo và gán một chuỗi cho biến s
s = "Lập trình Python không khó"
Tìm vị trí xuất hiện đầu tiên chuỗi 'tr' trong chuỗi s
n = s.find("tr")
#Hiển thị giá trị n ra màn hình
print(" n = ",n)
```

Kết quả chạy chương trình ***Vidu2_32.py***

```
n = 4
```

Ví dụ 2.33:

Vidu2_33.py

```
#Tìm chuỗi con "Python" trong chuỗi " Lập trình..."
# từ vị trí 2 đến 10
n="Lập trình...".find("Python",2,10)
print("n =",n)
```

Kết quả chạy chương trình ***Vidu2_33.py***

```
n = -1
```

***Phương thức replace()*:** Thay thế một chuỗi con cũ trong chuỗi bằng một chuỗi con khác. Giá trị trả về là một chuỗi đã được thay thế.

Cú pháp: *<chuỗi>*.replace*(<oldstr>, <newstr> [,count])*

Trong đó:

- *<chuỗi>:* Là một biến hoặc giá trị chuỗi chứa chuỗi con cũ cần thay thế.
- *<oldstr>:* Là một chuỗi con cũ hoặc một ký tự có trong chuỗi cần thay thế.
- *<newstr>:* Là một chuỗi con hoặc một ký tự mới cần thay thế chuỗi cũ.
- *[,count]:* Là một số nguyên chỉ số lượng lần thay thế tính từ trái qua phải, nếu ta không chỉ định tham số này thì Python sẽ thay tất cả.

Ví dụ 2.34:

Vidu2_34.py

```
#Khai báo và gán một chuỗi cho biến s
s = "Lập trình Python ở trình độ cao"
#Thay thế tất cả chuỗi con 'trình' trong s bằng chuỗi 'công'
st = s.replace("trình","công")
print("st = ",st)      #Hiển thị giá trị chuỗi st ra màn hình
```

Kết quả chạy chương trình ***Vidu2_34.py***

```
st =  Lập công Python ở công độ cao
```

Ví dụ 2.35:

Vidu2_35.py

```
#Khai báo và gán một chuỗi cho biến s
s = "Lập trình Python ở trình độ cao"
'''
Thay thế 2 lần chuỗi con " " (dấu cách) trong chuỗi s bằng chuỗi dấu trừ
'-'
'''
st = s.replace(" ","-",2)
print("st = ",st)      #Hiển thị giá trị chuỗi st ra màn hình
```

Kết quả chạy chương trình ***Vidu2_35.py***

```
st =  Lập-trình-Python ở trình độ cao
```

2.4. SỐ

Số tồn tại rất nhiều trong cuộc sống thường ngày của chúng ta như số người trong gia đình, nhiệt độ hôm nay là *30* độ, giá vé xem trận đấu bóng đá Việt Nam và Malaysia là *300.000* đồng,... Các con số này có thể mang tính toán hay xử lý, Python cung cấp cho chúng ta rất nhiều kiểu dữ liệu biểu diễn số để hỗ trợ cho người lập trình giải quyết các công việc, ví dụ như kiểu dữ liệu số nguyên (integer), số thực (float), phân số (fraction), số phức (complex),...

2.4.1. Kiểu số nguyên

Số nguyên bao gồm các số dương, hoặc âm hoặc số 0 không chứa phần thập phân, ví dụ như: 123, –6, 0, 33, 8765,...

Ví dụ 2.36:

Vidu2_36.py

```
# Khai báo và gán giá trị số nguyên bằng 40 cho biến siso
siso = 40
# Hiển thị giá trị biến siso ra màn hình
print("Sĩ số lớp = ",siso)

#Hiển thị kiểu dữ liệu của biến siso

print(type(siso))
```

Kết quả chạy chương trình ***Vidu2_36.py***

```
Sĩ số lớp =  40
<class 'int'>
```

Chú ý: Python hỗ trợ xử lý các số nguyên lớn lên tới trên 16 chữ số

Ví dụ 2.37:

Vidu2_37.py

```
'''

Khai báo và gán giá trị số nguyên rất lớn bằng biểu thức cho biến a '''
a = 1234567890123456 + 123
print("a = ",a)       # Hiển thị giá trị biến a ra màn hình
print(type(a))        #Hiển thị kiểu dữ liệu của biến a
```

Kết quả chạy chương trình ***Vidu2_37.py***

```
a =  1234567890123579
<class 'int'>
```

2.4.2. Kiểu số thực

Số thực bao gồm các số dương, hoặc âm có chứa cả phần nguyên và phần thập phân, ngăn cách giữa chúng là một dấu chấm "." như: 123.89, –6.5, 0.34, 8765.0,... Vậy, tập số nguyên chính là một tập con của tập số thực có phần thập phân bằng không.

Ví dụ 2.38:

Vidu2_38.py

```
# Khai báo và gán giá trị số thực bằng 6.5 cho biến diemtoan
diemtoan = 6.5
# Hiển thị giá trị biến diemtoan ra màn hình
print("Điểm thi = ",diemtoan)
# Hiển thị kiểu dữ liệu của biến diemtoan
print(type(diemtoan))
```

Kết quả chạy chương trình ***Vidu2_38.py***

```
Điểm toan =  6.5
<class 'float'>
```

Chú ý: *Python* hỗ trợ xử lý các số thực có độ chính xác xấp xỉ 15 chữ số phần thập phân

Ví dụ 2.39:

Vidu2_39.py

```
''' Khai báo và gán giá trị số thực bằng biểu thức 10 chia 3 cho biến a
'''
a = 10/3
print("a = ",a)    # Hiển thị giá trị biến a ra màn hình
print(type(a))     # Hiển thị kiểu dữ liệu của biến a
```

Kết quả chạy chương trình ***Vidu2_39.py***

```
a =  3.3333333333333335
<class 'float'>
```

2.4.3. Biểu thức

Toán tử: Là các phép toán chỉ ra thao tác toán học cần thực hiện như: Toán tử số học diễn tả các phép toán cộng "+", phép trừ "–", nhân "*", chia"/",...; toán tử gán; toán tử so sánh; toán tử lô–gíc và toán tử điều kiện.

Bảng 2.2: Bảng toán tử số học trong Python

Toán tử	Diễn giải
+	Phép cộng (5 + 3 = 8)
–	Phép trừ (5 – 3 = 2)
*	Phép nhân (5 * 3 = 15)
/	Phép chia (5 / 3 = 1.6667)
%	Phép chia lấy phần dư (5 % 3 = 2)
**	Phép lấy số mũ (ví dụ 2**3 cho kết quả là 8)
//	Thực hiện phép chia lấy phần nguyên (5 // 3 = 1)

Toán tử so sánh trong Python cho kết quả là một hằng số bằng True hoặc False:

Bảng 2.3: Bảng toán tử so sánh trong Python

Toán tử	Diễn giải
>	So sánh lớn hơn, (3 > 5) = False
<	So sánh nhỏ hơn, (3 < 5) = True
>=	So sánh lớn hơn hoặc bằng (3 >= 5) = False
<=	So sánh lớn hơn hoặc bằng (5 >= 5) = True
==	So sánh bằng (5 == 5) = True, (8 == 9) = False
!= hoặc <>	So sánh không bằng (khác) (5 != 5) = False, (8 != 9) = True

Toán tử logic trong Python:

Bảng 2.4: Bảng toán tử logic trong Python

Toán tử	Diễn giải
and	Phép và, chỉ đúng khi cả hai cùng đúng
or	Phép hoặc, chỉ sai khi cả hai cùng sai
not	Phép phủ định (not True) = False, (not False) = True

Toán tử gán kết hợp với các phép toán:

Bảng 2.5: Bảng toán tử kết hợp trong Python

Toán tử	Diễn giải
=	Phép gán
/=	Phép toán chi gán, s /=5; tương đương s = s/5
+=	Phép cộng và gán, s +=5; tương đương s = s + 5
–=	Phép trừ và gán, s –=5; tương đương s = s – 5
*=	Phép nhân và gán, s *=5; tương đương s = s * 5
%=	Phép chia lấy phần dư và gán, s %=5; tương đương s = s%5
**=	Phép tính số mũ và gán, s **=5; tương đương s = s**5
//=	Phép chia lấy nguyên và gán, s //=5; tương đương s = s//5

Các toán tử thao tác bít trong Python: Các giá trị bít biểu diễn các số (0, 1) trong hệ nhị phân.

Ví dụ 2.40: Chuyển đổi số hệ thập phân và nhị phân

$(60)_{10}$ =(0011 1100)$_2$, $(13)_{10}$ = (0000 1101)2

Bảng 2.6: Bảng toán tử thao tác bit trong Python

Toán tử	Diễn giải
&	Phép và (and), 1110 & 1011 = 1010
|	Phép toán hoặc (or), 1110 | 1010 = 1110
^	Phép toán xor (giống nhau = 0 khác nhau =1), 1110 ^ 1010 = 0100
~	Phép đảo ngược bit (not), ~1110 = 0001
<<	Phép dịch trái, 1110 << 0001 =1100
>>	Phép dịch phải 1 bít nhị phân, 1110 >> 0001 =0111

Thứ tự ưu tiên của các toán tử:

Bảng 2.7: Bảng thứ tự ưu tiên các toán tử trong Python

Thứ tự ưu tiên	Toán tử	Diễn giải
1	function(), (), **	Các hàm, trong ngoặc, toán tử mũ
2	* / % //	Phép nhân, chia, chia lấy phần dư và phép chia lấy phần nguyên
3	+ –	Toán tử cộng, trừ
4	<= <>>=	Các toán tử so sánh lớn nhỏ
5	<> == !=	Các toán tử so sánh bằng, khác
6	= %= /= //= –= += *= **=	Các toán tử gán
7	in, is , is not	Các toán tử kiểm tra
8	not, or, and	Các toán tử lô–gíc

Toán hạng: Biểu diễn các hằng số, các biến, dấu ngoặc "(,)", dấu âm "–",...

Biểu thức: Là một sự kết hợp giữa các toán tử và các toán hạng theo đúng một trật tự nhất định, mỗi toán hạng có thể là một hằng, một biến hoặc một biểu thức khác. Trong trường hợp, biểu thức có nhiều toán tử, ta có thể dùng

cặp dấu ngoặc đơn "()" để chỉ cho máy tính biết toán tử nào được thực hiện trước. Ví dụ, biểu thức tính nghiệm của phương trình bậc hai là: (–b + sqrt(delta))/(2*a), trong đó 2 là hằng số; a, b, delta là biến và sqrt() là hàm tính căn bậc hai.

Ví dụ 2.41:

Vidu2_41.py

```
#Biểu thức gán giá trị nguyên 7 cho biến a
a = 7
print("a = ",a)
#Biểu thức tính tổng hai số nguyên gán vào biến c
b = 5 + 6
print("b = ",b)
#Tính phép toán lấy 8 chia cho 3 lấy phần dư và gán cho biến a
c = 8%3
print("c = ",c)
#Tính phép toán lấy 10 chia cho 3 lấy phần nguyên và gán cho biến b
d = 10//3
print("d = ",d)
```

Kết quả chạy chương trình ***Vidu2_41.py***

```
a = 7
b = 11
c =  2
d =  3
```

Ví dụ 2.42:

Vidu2_42.py

```
#Biểu thức gán giá trị thực 5.5 cho biến x
x = 5.5
#Tính biểu thức có chứa dấu ngoặc phức tạp và gán cho biến y
y = (2*x**2 + 7* x + 13)/(x + 2)
print("y = ",y)
```

Kết quả chạy chương trình ***Vidu2_42.py***

```
y =  14.933333333333334
```

Chúng ta có thể sử dụng hàm eval(<source>) để tính giá trị biểu thức với tham số đầu vào là một chuỗi biểu thức.

Ví dụ 2.43:

Vidu2_43.py

```
#Tính giá trị chuỗi biểu thức   "(3*4-5)*2" và gán cho biến y
y = eval("(3*4-5)*2")
print("y = ",y)        #Hiển thị giá trị biến y ra màn hình
```

Kết quả chạy chương trình ***Vidu2_43.py***

```
y =  14
```

Chú ý: Tham số truyền vào hàm eval() phải là một biểu thức đúng (có thể tính được) nghĩa là các toán tử và toán hạng đặt đúng vị trí, nếu không máy sẽ báo lỗi.

Ví dụ 2.44:

Vidu2_44.py

```
#Chuỗi biểu thức "(3*%4-5)*2"  sai ở vị trí toán tử "*%"
y=eval("(3*%4-5)*2")
print("y = ",y)        #Hiển thị giá trị y ra màn hình
```

Kết quả chạy chương trình ***Vidu2_44.py***

```
File "<string>", line 1    (3*%4-5)*2
SyntaxError: invalid syntax
```

2.4.4. Nhập dữ liệu

Python cung cấp hàm input() cho người dùng nhập vào một chuỗi, xâu từ bàn phím, khi người dùng nhấn phím Enter việc nhập dữ liệu sẽ kết thúc.

Giá trị trả về của hàm là một chuỗi, do đó, ta có thể chuyển kiểu từ kiểu chuỗi sang kiểu dữ liệu như mong muốn không phải kiểu chuỗi.

Cú pháp: *<tên biến>* = input*([thông báo])*

Trong đó:

- *<tên biến>:* Là biến lưu trữ dữ liệu người dùng nhập vào từ bàn phím.
- *[thông báo]:* Là một chuỗi thông báo cho người dùng biết nhập dữ liệu cần nhập, chuỗi này có thể là rỗng.

Ví dụ 2.45:

Vidu2_45.py

```
#Cho người dùng nhập vào một chuỗi từ bàn phím
s = input("Xin mời nhập vào một chuỗi s = ")
#Hiển thị chuỗi vừa nhập ra màn hình
print("Chuỗi bạn vừa nhập s = ",s)
print("Kiểu của s là: ", type(s))
```

Kết quả chạy chương trình ***Vidu2_45.py***

```
Xin mời nhập vào một chuỗi s = Python rất hữu ích
Chuỗi bạn vừa nhập s =  Python rất hữu ích
Kiểu của s là: <class 'str'>
```

2.4.5. Các hàm chuyển đổi kiểu dữ liệu

Trong chương trình Python, chúng ta có thể chuyển đổi qua lại giữa các kiểu dữ liệu số và chuỗi thông qua các hàm được Python tích hợp sẵn như int(), float() và str() để thực hiện các toán tử trong biểu thức tính toán. Điều này còn được gọi là cưỡng chế hay ép kiểu.

Ví dụ 2.46: Viết chương trình cho người dùng nhập vào một số nguyên và một số thực sau đó tính tổng giá trị và hiển thị kết quả ra màn hình.

Vidu2_46.py

```
so_nguyen= int(input('Hãy nhập 1 số nguyên: '))
so_thuc=float(input('Hãy nhập 1 số thực: '))
tong= so_nguyen + so_thuc
print('Tổng của 2 số vừa nhập có giá trị =', tong)
print('kiểu dữ liệu là: ', type(tong))
```

Kết quả chạy chương trình ***Vidu2_46.py***

```
Hãy nhập 1 số nguyên: 3
Hãy nhập 1 số thực: 4.5
Tổng của 2 số vừa nhập có giá trị = 7.5
kiểu dữ liệu là:  <class 'float'>
```

Nhận xét*:* Trong ví dụ 2.46 ta thấy, hàm input() cho người dùng nhập vào một chuỗi số, ngay sau đó ta sử dụng hàm int() với tham số truyền vào là hàm input() để chuyển kiểu dữ liệu chuỗi số nguyên mà người dùng vừa nhập vào sang dạng số nguyên chuẩn và gán cho biến *son*.

Tương tự như vậy, hàm float() sẽ chuyển đổi kiểu dữ liệu chuỗi số thực mà người dùng nhập vào sang dạng số thực chuẩn và gán vào biến *sot*.

Ngược lại, hàm str() với tham số truyền vào là một số (thực hoặc nguyên) sẽ chuyển đổi dữ liệu số sang kiểu chuỗi để thực hiện phép toán cộng chuỗi trong hàm print().

2.4.6. Thư viện math trong Python

Python cung cấp thư viện *math,* trong đó hỗ trợ nhiều hàm tính toán số học, chúng ta có thể sử dụng thư viện này trong chương trình thông qua lệnh import sau:

Cú pháp: import <*tên thư viện*> [as <*tên bí danh*>]

Trong đó:

- <*tên thư viện*>*:* Là tên thư viện chứa các hàm mà chúng ta muốn sử dụng, ví dụ math, numpy, matplotlib,...
- <*tên bí danh*>*:* Là một tên ngắn gọn dùng để thay cho tên thư viện

Để gọi (sử dụng) hàm trong thư viện, chúng ta sử dụng câu lệnh sau:

Cú pháp: *<tên thư viện>.<tên hàm>*([*<danh sách tham số>*])

Trong đó:

- *<tên hàm>:* Là tên hàm chúng ta muốn sử dụng
- *<danh sách tham số>:* Là danh sách tham số truyền vào làm dữ liệu đầu vào cho hàm.

Ví dụ 2.47:

Vidu2_47.py

```
#Khai báo sử dụng các hàm xử lý toán học trong thư viện math
import math
#Khai báo sử dụng hàm sinh số nguyên ngẫu nhiên trong thư viện random
#sử dụng bí danh là rr
import random as rr
#Gọi hàm tính căn bậc hai của một số x=8 và gán vào biến y
y = math.sqrt(8)
print("y = ",y)
'''
Gọi hàm sinh số nguyên ngẫu nhiên trong khoảng từ 1 đến 9 và gán vào biến n
Chú ý: Mỗi lần chạy hàm lại sinh ra một số nguyên khác nhau '''
n =rr.randint(1,9)
print("n = ",n)
```

Kết quả chạy chương trình ***Vidu2_47.py***

```
y =  2.8284271247461903
n =  2
```

2.5. TẠO ỨNG DỤNG

Ví dụ 2.48: Giả sử một người A có số tiền bằng X, đem gửi tiết kiệm với lãi suất tháng là 0,6%; hỏi rằng sau 18 tháng thì A có tất cả bao nhiêu tiền? Hãy viết chương trình cho người dùng nhập vào số tiền X sau đó tính tổng tiền sau 18 tháng. Biết rằng cứ 6 tháng thì tiền lãi được cộng vào gốc và người A không rút tiền.

Hướng dẫn:

- Sử dụng hàm input() kết hợp với hàm float() để nhập một số thực X từ bàn phím biểu diễn tiền gốc của người A.
- Tính tiền lãi cho 6 tháng đầu vì cứ 6 tháng thì tiền lãi lại cộng vào tiền gốc.
- Tính lại tiền gốc sau 6 tháng.
- Tương tự như vậy, tính lại tiền gốc sau 12, 18 tháng.
- Sử dụng hàm print() để hiển thị kết quả ra màn hình.

Vidu2_48.py

```
'''
        Chương trình tính tiền gửi tiết kiệm theo kỳ hạn
'''
#Cho người dùng nhập vào một số tiền lưu vào biến X
X = float(input("Nhập vào số tiền gốc ban đầu = "))
#Lưu lại số tiền gốc vào biến goc để tính toán trên biến này
goc=X

lx=0.006          #Biến Lx lưu lãi xuất hàng tháng
#Tiền lãi sau 6 tháng đầu là
lai=6*goc*lx
print("Tiền gốc sau 6 tháng đầu=",goc, "lãi =",lai, " lãi nhập vào gốc
=",goc+lai)
#Tiền gốc sau 6 tháng tiếp theo là
goc=goc+lai
#Tiền lãi sau 6 tháng tiếp theo là
lai6=6*goc*lx
print("Tiền gốc sau 6 tháng tiếp =",goc, "lãi =",lai, " lãi nhập vào gốc
=",goc+lai)
#Tiền gốc sau 6 tháng tiếp theo là
goc=goc+lai6
#Tiền lãi sau 6 tháng tiếp theo là
lai6=6*goc*lx
print("Tiền gốc sau 6 tháng tiếp =",goc, "lãi =",lai6, " lãi nhập vào gốc
=",goc+lai6)
#Tiền gốc sau 6 tháng tiếp theo là
goc=goc+lai6
print("Tổng tiền gốc cộng lãi sau 18 tháng =",goc)
```

Kết quả chạy chương trình ***Vidu2_48.py***

```
Nhập vào số tiền gốc ban đầu = 30000000

Tiền gốc sau 6 tháng đầu= 30000000.0 lãi = 1080000.0  lãi nhập vào gốc = 31080000.0

Tiền gốc sau 6 tháng tiếp = 31080000.0 lãi = 1080000.0  lãi nhập vào gốc = 32160000.0

Tiền gốc sau 6 tháng tiếp = 32198880.0 lãi = 1159159.68  lãi nhập vào gốc = 33358039.68

Tổng tiền gốc cộng lãi sau 18 tháng = 33358039.68
```

Ví dụ 2.49: Viết chương trình cho người dùng nhập vào một chuỗi họ tên của một người, sau đó hãy chuẩn hóa chuỗi họ tên thỏa mãn các yêu cầu sau:

- Chuỗi họ tên không chứa khoảng trống (dấu cách) ở bên trái và bên phải chuỗi
- Ký tự đầu của họ, đệm và tên phải viết hoa sau.
- Chuỗi họ tên không có khoảng trống thừa ở giữa phần họ, đệm, tên;
- Hiển thị kết quả ra màn hình.

Giả sử nhập vào s = “ nguyễn VĂN hẬu ”

Chuỗi họ tên s sau khi chuẩn hóa là: “Nguyễn Văn Hậu”

Hướng dẫn:

- Sử dụng hàm input() để nhập một chuỗi họ tên từ bàn phím
- Sử dụng phương thức strip() của lớp chuỗi để xóa các dấu cách ở hai bên chuỗi
- Sử dụng phương thức title() của lớp chuỗi để viết hoa các tiếp từ đầu trong chuỗi
- Sử dụng phương thức split() kết hợp với phương thức join() của lớp chuỗi để loại bỏ các dấu cách thừa trong chuỗi
- Sử dụng hàm print() để hiển thị kết quả ra màn hình.

Vidu2_49.py

```
#Cho người dùng nhập vào một chuỗi Lưu vào biến s
s = input("Xin mời nhập chuỗi họ tên =")
s1 = s.strip()        #Xóa các dấu cách ở hai bên chuỗi s
print("s1=",s1)       #Hiển thị kết quả xóa ra màn hình
s2 = s1.lower()       #Chuyển tất cả các ký tự về chữ thường
print("s2=",s2)
s3 = s2.title()       #Viết hoa các ký tự ở đầu mỗi từ
print("s3=",s3)
s4 = s3.split()       #Loại bỏ các dấu cách thừa trong chuỗi
print("s4=",s4)
s5 = " ".join(s4)#Ghép các phần tử lại ngăn cách bởi 1 dấu cách
print("Chuỗi họ và tên sau khi chuẩn hóa s5 = ",s5)
```

Kết quả chạy chương trình ***Vidu2_49.py***

```
Xin mời nhập chuỗi họ tên =   nguyễn   VĂN     hậu
s1= nguyễn   VĂN     hậu
s2= nguyễn   văn     hậu
s3= Nguyễn   Văn     Hậu
s4= ['Nguyễn', 'Văn', 'Hậu']
Chuỗi họ và tên sau khi chuẩn hóa s5 =  Nguyễn Văn Hậu
```

2.6. TỔNG KẾT BÀI HỌC

Bài học đề cập chi tiết những vấn đề cơ bản của ngôn ngữ lập trình Python. Nói chung, các kiểu dữ kiệu số, chuỗi (xâu), biểu thức, các phép toán và thứ tự ưu tiên của các phép toán là khá giống với ngôn ngữ lập trình khác (C/C++, Java, C++, PHP). Tuy nhiên, cách khai báo biến của Python là đơn giản hơn các ngôn ngữ lập trình khác. Với Python, chúng ta chỉ việc gán giá trị cho biến và chương trình thông dịch Python sẽ tự xác nhận kiểu dữ liệu tương ứng với giá trị dữ liệu gán cho biến. Hơn nữa, với Python chúng ta có thể tính toán với các số rất lớn mà không phải lo lắng đến vấn đề tràn bộ nhớ.

Phần tạo ứng dụng đã cung cấp một số ví dụ hoàn chỉnh nhằm minh họa cho vấn đề có thể gặp trong thực tiễn. Để chúng ta có thể ứng dụng được các khái niệm vừa học vào viết các chương trình bằng Python, bài học kế tiếp sẽ tìm hiểu về các cấu trúc điều khiển trong chương trình Python.

2.7. BÀI TẬP

A. Câu hỏi ôn tập

Câu 2.1: Cho biết phát biểu nào sau đây là đúng?

A. Tên biến có thể bắt đầu bằng dấu gạch dưới.

B. Tên biến có thể bắt đầu bằng một chữ số.

C. Tên biến có thể đặt trùng với từ khóa.

D. Tên biến có thể có các ký tự đặc biệt như: @, #, $, v.v..

Câu 2.2: Cho biết các câu lệnh nào sau đây là đúng để nhận được giá trị của biến là số nguyên 10?

A. int x = 10

B. x = 10

C. x = int(“10”)

D. x = int(10.10)

Câu 2.3: Cho biết khối lệnh nào sau đây thực hiện khai báo các biến x, y, z như sau: biến nguyên x =10; biến thực y = 5.5 và xâu z = "Hello"?

A. x = 10;y = 5.5;z = **"Hello"**

B. x = 10
   y = 5.5
   z = "Hello"

C. x = int(10); y = float(5.5); z = "Hello"

D. int x = int(10); float y = float(5.5); string z = "Hello"

Câu 2.4: Cho biết biến n = '5' thuộc kiểu dữ liệu nào?

A. int

B. str

C. float

D. list

Câu 2.5: Cho biết biến n = float('5') thuộc kiểu dữ liệu nào?

A. int

B. str

C. float

D. dictionary

Câu 2.6: Cho biết kết quả hiển thị ra màn hình sau khi python chạy lệnh sau?

```
print(3 >= 3)
```

A. True

B. False

C. None

D. Error

Câu 2.7: Cho biết kết quả hiển thị ra màn hình khối lệnh sau?

```
S = 'Hello'
print( len(S))
```

A. 5

B. 4

C. True

D. False

Câu 2.8: Cho biết kết quả hiển thị ra màn hình sau khi python chạy khối lệnh sau?

```
name = 'PYTHON'
note = 'Thật là hay!'
print('The name is ' + name+ ' - ' + note)
```

A. The name is PYTHON – Thật là hay!

B. The name is PYTHON

C. Thật là hay!

D. PYTHON – Thật là hay!

Câu 2.9: Cho biết kết quả hiển thị ra màn hình sau khi Python chạy khối lệnh sau?

```
mark1=10;mark2=7
mark=(mark1+2*mark2)/3
```

A. 7

B. 7.0

C. 8.0

D. 8

Câu 2.10: Cho biết kết quả hiển thị ra màn hình sau khi chạy câu lệnh sau?

```
print('\t'.join(['LT','Python cơ bản',3.6]))
```

A. LT Python cơ bản

B. Lỗi kiểu (type Error)

C. LT Python cơ bản 3.6

D. Cả 3 ý trên đều sai

B. Lập trình

Bài tập 2.1: Viết chương trình tính giá trị hàm số $f(x)=e^x–x$ tại giá trị x = 0.501 và hiển thị kết quả ra màn hình.

Bài tập 2.2: Viết chương trình cho người dùng nhập vào một số x bất kỳ sau đó tính và hiển thị (in) ra màn hình giá trị x^2, x^3, x^4.

Bài tập 2.3: Viết chương trình cho người dùng nhập vào một số thực là nhiệt độ f, sau đó đổi sang nhiệt độ C theo công thức sau:

$$^0C = \frac{^0F - 32}{1.8}$$

Ví dụ: Nhập vào độ F là 0^oF thì đầu ra độ C là $–17.78^oC$

Bài tập 2.4: Viết chương trình cho người dùng nhập vào từ bàn phím họ và tên của người đó. Hiển thị ra màn hình tên của người đó được tạo thành bởi bao nhiêu tiếng.

Ví dụ: Nếu người đó nhập: Đỗ Nguyễn Diệu Anh

Hiển thị ra màn hình: Tên: Đỗ Nguyễn Diệu Anh được tạo bởi 4 tiếng

Bài tập 2.5: Viết chương trình cho người dùng nhập vào bán kính hình cầu R, tính và in ra màn hình diện tích, thể tích của hình cầu đó.

Hướng dẫn: $S = 4\pi R^2$ và $V = (4/3)\pi R^3$.

Bài tập 2.6: Viết chương trình cho người dùng nhập vào hai chuỗi s1 và s2, sau đó hãy kiểm tra xem chuỗi s2 có trong chuỗi s1 hay không? Nếu có hãy hiển thị vị trí đầu tiên của s2 xuất hiện trong s1 ra màn hình, ngược lại thông báo không tìm thấy.

Bài tập 2.7: Viết chương trình cho người dùng nhập vào một xâu sau đó hiển thị chuỗi đó ra màn hình theo chiều ngược lại:

Ví dụ: Nhập vào: Tran Van Thoa, xuất ra: aohT naV narT

Bài tập 2.8: Viết chương trình cho người dùng nhập vào một chuỗi s sau đó nhập vào một ký tự c và kiểm tra xem ký tự c có xuất hiện trong chuỗi s hay không, nếu có thì xuất hiện bao nhiêu lần? Nếu không thì thông báo không xuất hiện.

Ví dụ: Chuỗi nhập vào: s = "tran van thoa".

Nhập kí tự c = 't', xuất hiện 2 lần trong chuỗi s:

Bài tập 2.9: Xây dựng chương trình cho người dùng nhập một chuỗi ký tự scmt (số chứng minh nhân dân của một người) từ bàn phím, sau đó kiểm tra xem chuỗi scmnd có hợp lệ hay không và hiển thị kết quả ra màn hình? Biết rằng xâu scmt hợp lệ là xâu chứa toàn các ký tự chữ số từ 0 đến 9 và có độ dài từ 9 tới 11 số.

Bài tập 2.10: Xây dựng chương trình cho người dùng nhập một chuỗi là địa chỉ một thư điện tử se từ bàn phím sau đó kiểm tra xem địa chỉ thư se có hợp lệ hay không? Biết xâu (địa chỉ một thư điện tử) se hợp lệ là xâu chứa đúng một ký tự @ và có định dạng x@y.z, trong đó x, y, z phải là các xâu khác rỗng.

Bài 3

CẤU TRÚC RẼ NHÁNH

Bài học này trình bày cấu trúc rẽ nhánh if: là cấu trúc điều khiển được sử dụng trong quá trình lựa chọn nhánh hành động dựa trên kết quả của việc kiểm tra điều kiện.

Sau bài học này, người học có thể:

- *Sử dụng cấu trúc lặp if, if else, if elif trong chương trình*
- *Tạo ứng dụng tìm số lớn nhất (nhỏ nhất)*
- *Tạo ứng dụng tính số ngày (khi nhập tháng và năm)*

3.1. ĐẶT VẤN ĐỀ

Bài toán: Giả sử chúng ta cần xây dựng một chương trình máy tính thực hiện yêu cầu sau:

- Nhập vào tên và điểm thi của một thí sinh.
- Hiển thị thông tin về kết quả thí sinh trượt hoặc đỗ biết rằng thí sinh trượt nếu điểm thi nhỏ hơn 5 và đỗ nếu điểm thi lớn hơn hoặc bằng 5.

Xác định yêu cầu bài toán:

- Nhập dữ liệu đầu vào là 2 biến tên (*name*) kiểu chuỗi và biến điểm (*mark*) kiểu thực.
- Hiển thị biến name ra màn hình và kết quả thí sinh trượt/đỗ.

Cách giải quyết:

Sử dụng cấu trúc rẽ nhánh để xét nếu điểm mark lớn hơn hoặc bằng 5 thì đỗ, ngược lại sẽ bị trượt.

Ta có chương trình minh họa sau:

Chương trình 3.1: *Xét kết quả trượt/đỗ của thí sinh*

```
name=input('Nhập tên: ')          #Nhập giá trị biến chuỗi name
mark=float(input('Nhập điểm: ')))#Nhập giá trị biến thực mark
if mark >=5:
   print(name,' - ĐỖ')
else:
   print(name, ' - TRƯỢT')
```

Trong đó:

Nội dung sau dấu # là phần chú thích, không được dịch trong chương trình.

Hàm input() giúp nhập dữ liệu kiểu chuỗi từ bàn phím.

Hàm float() chuyển dữ liệu kiểu chuỗi thành kiểu thực.

if ...else… cấu trúc rẽ nhánh để xét điểm và đưa ra kết quả trượt hoặc đỗ.

Các lệnh con sau if và sau else được viết thụt lùi dịch sang phải cùng một khoảng trống so với từ khóa if và từ khóa else.

3.2. BIỂU THỨC LÔ-GÍC (BOOLEAN)

Một biểu thức lô-gíc luôn cho kết quả trả về là một trong hai giá trị đúng (True) hoặc sai (False).

Ví dụ 3.1:

Vidu3_1.py

```
#Thực hiện so sánh hai toán hạng và gán kết quả cho biến kt
kt = (5>10)
print("kt = ",kt)
#Thực hiện biểu thức logic phức tạp và gán kết quả cho biến kt2
kt2 = (5>10 and 7 >4 or 10<90)
print("kt2 = ",kt2)
```

Kết quả chạy chương trình ***Vidu3_1.py***

```
kt =  False
kt2 =  True
```

Nhận xét:

Ở ví dụ trên, chúng ta có thể dựa vào thứ tự ưu tiên của các toán tử, đầu tiên máy sẽ lấy 5 so sánh với 10 kết quả bằng False; tiếp đến lấy 7 so sánh với 4 kết quả được True; tiếp đến lấy 10 so sánh với 90 kết quả được True. Sau đó máy tính thực hiện đến phép toán and (và), lấy False và với True ta được False, cuối cùng lấy False hoặc (or) với True ta được giá trị của biểu thức bằng True.

3.3. CẤU TRÚC RẼ NHÁNH VỚI (IF)

if là từ chỉ sự bắt đầu của một cấu trúc câu điều kiện trong tiếng Anh, nghĩa tiếng Việt là "nếu", "giá mà", ..

Ví dụ 3.2:

Nếu tôi thi đỗ đại học thì tôi sẽ chọn học ngành Công nghệ Thông tin.

Nếu hôm nay trời mưa thì tôi sẽ mặc áo mưa.

Nếu m lớn hơn 0 thì m là số dương,...

3.3.1. if

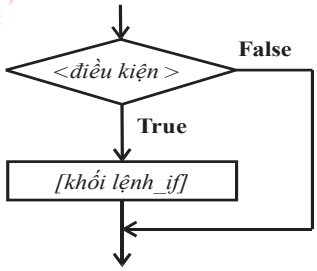
Cú pháp: **if** *(<điều kiện>):*

[khối lệnh_if]

Trong đó:

- *<điều kiện>:* Luôn cho kết quả trả về là một trong hai giá trị đúng (True) hoặc sai (False).
- *[khối lệnh_if]:* Bao gồm một hoặc nhiều lệnh được chạy khi biểu thức lô–gíc có giá trị đúng (True), các câu lệnh trong khối lệnh có lề trái lệch sang bên phải một khoảng trống so với câu lệnh if.

Lưu đồ thuật toán cấu trúc if:



Hình 3.1: Lưu đồ thuật toán cấu trúc if

Ví dụ 3.3:

Vidu3_3.py

```
if (a > 5):       #Kiểm tra nếu biểu thức a lớn hơn 5 = true thì thực hiện
        b = a     #Gán giá trị của biến a cho biến b
```

Trong đó các biến trong biểu thức phải có giá trị cụ thể, trong ví dụ 3.3

Ví dụ 3.4:

Vidu3_4.py

```
x = 9
if  (x > 0):          #Kiểm tra nếu x lớn hơn không thì...
        print(x,": Là số dương")   #Hiển thị ra màn hình x là số dương
```

Kết quả chạy chương trình ***Vidu3_4_.py***

```
9: Là số dương
```

Ví dụ 3.5:

Vidu3_5.py

```
x = 9
 #Kiểm tra nếu x lớn hơn không và nhỏ hơn 50 thì...
#Hiển thị ra màn hình x là số dương nhỏ hơn 50
if (x > 0 and x<50):
        print(x,": Là số dương nhỏ hơn 50")
```

Kết quả chạy chương trình ***Vidu3_5.py***

```
9: Là số dương nhỏ hơn 50
```

3.2.2. if else

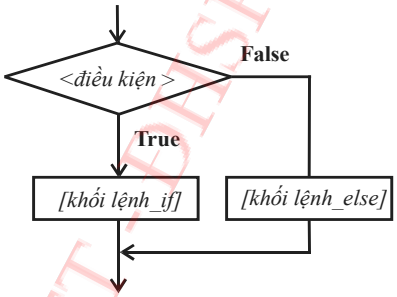
Cú pháp:

```
if (<điều kiện>):
   [khối lệnh_if]
else:
   [khối lệnh_else]
```

Trong đó:

- *<điều kiện>:* Là một biểu thức lô–gíc chỉ điều kiện cho kết quả là đúng (True) hoặc sai (False).
- *[khối lệnh_if]:* Là khối lệnh được chạy khi biểu thức lô–gíc có giá trị đúng (True), các câu lệnh trong khối lệnh có lề trái dịch sang phải một khoảng trống so với câu lệnh if.
- *[khối lệnh_else]:* Là khối lệnh được chạy khi biểu thức lô–gíc có giá trị sai (False), các câu lệnh trong khối lệnh có lề trái thụt vào một khoảng trống so với câu lệnh else.

Lưu đồ thuật toán cấu trúc if–else:



Hình 3.2: Lưu đồ thuật toán cấu trúc if– else

Ví dụ 3.6:

Vidu3_6.py

```
x = -6
if (x > 0):          # Kiểm tra nếu x lớn hơn số 0 thì...
  print(x,": Là số dương") # Hiển thị ra màn hình x là số dương
else:
        # Hiển thị ra màn hình x là số âm hoặc số không
        print(x,": Là số âm hoặc số không")
```

Kết quả chạy chương trình ***Vidu3_6.py***

```
-6: Là số âm hoặc số không
```

Ví dụ 3.7: Ta cũng có thể lồng các cấu trúc if vào nhau (trong trường hợp này bạn cần chú ý tới khối lệnh) như sau:

Vidu3_7.py

```
x = 5
if (x > 0):          # Kiểm tra nếu x lớn hơn số 0 thì...
  print(x,": Là số dương") # Hiển thị ra màn hình x là số dương
else:
  if (x == 0):       # Kiểm tra tiếp nếu x bằng 0 thì...
    print(x,": Là số không") # Hiển thị ra màn hình x là số không
  else:
        print(x,": Là số âm")# Hiển thị ra màn hình x là số âm
```

Kết quả chạy chương trình ***Vidu3_7.py***

```
5 : Là số dương
```

3.2.3. if elif

Cú pháp:

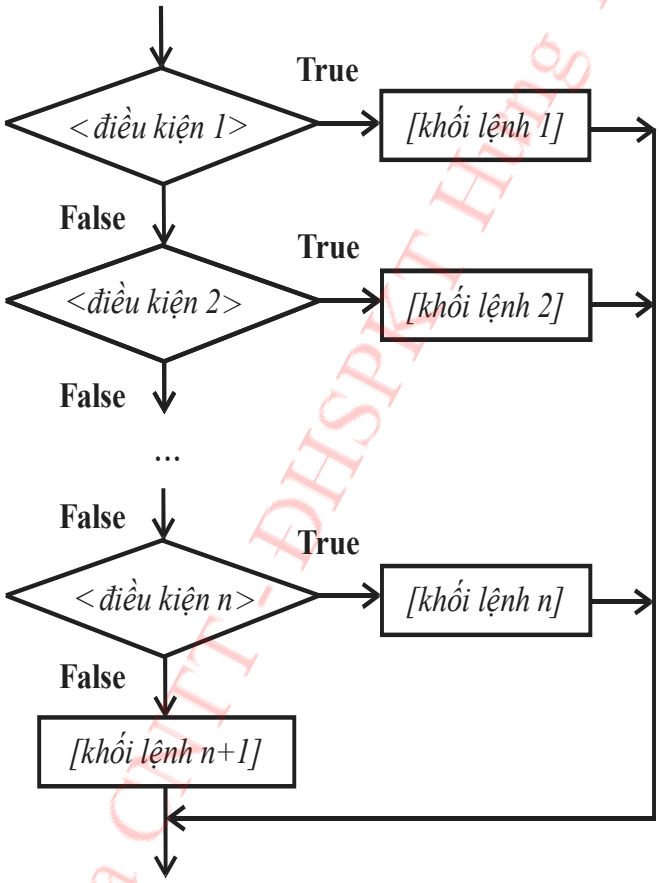
```
if (< điều kiện 1 >):
        [khối lệnh 1]
elif (< điều kiện 2 >):
        [khối lệnh 2]
...
elif (< điều kiện n >):
        [khối lệnh n]
else:
        [khối lệnh n + 1]
```

Trong đó:

- *<điều kiện 1, 2,... n>:* Là các biểu thức lô–gíc chỉ điều kiện cho kết quả là đúng (True) hoặc sai (False)

- *[khối lệnh 1, 2,... n]:* Là khối lệnh được chạy khi biểu thức lô–gíc tương ứng 1, 2,...n có giá trị đúng (True), các câu lệnh trong khối lệnh có lề trái dịch sang phải một khoảng trống so với câu lệnh if hoặc elif.
- *[khối lệnh n+1]:* Là khối lệnh được chạy khi tất cả biểu thức lô–gíc từ 1 đến n có giá trị sai (False), các câu lệnh trong khối có lề trái thụt vào một khoảng trống so với câu lệnh else.

Lưu đồ thuật toán cấu trúc if–elif–else:



Hình 3.3: Lưu đồ thuật toán cấu trúc if–elif–else

Hoạt động: Khi gặp cấu trúc if, elif trong chương trình máy thực hiện các bước như sau:

- Bước 1: Kiểm tra nếu biểu thức lô–gíc <điều kiện 1> có giá trị đúng

(True) thì thực hiện khối lệnh [khối lệnh 1] và thoát khỏi cấu trúc, ngược lại sang Bước 2.

- Bước 2: Kiểm tra nếu biểu thức lô–gíc <điều kiện 2> có giá trị đúng (True) thì thực hiện khối lệnh [khối lệnh 2] và thoát khỏi cấu trúc, ngược lại sang Bước 3.
- Bước 3: Kiểm tra nếu biểu thức lô–gíc <điều kiện n> có giá trị đúng (True thì thực hiện khối lệnh [khối lệnh n] và thoát khỏi cấu trúc, ngược lại sang Bước 4.
- Bước 4: Thực hiện khối lệnh [khối lệnh n+1] và thoát khỏi cấu trúc.

Ví dụ 3.8: Kiểm tra xem số nguyên x là số 0, số dương hay số âm?

Vidu3_8.py

```
x = 5
if (x == 0):          #Kiểm tra nếu x bằng không thì...
  print(x,": Là số không")  #Hiển thị ra màn hình x là số không
elif (x > 0):      #Kiểm tra nếu x lớn hơn không thì...
  print(x,": Là số dương")  #Hiển thị ra màn hình x là số dương
else:    #Nếu tất cả các điều kiện trên đều sai thì...
  print(x,": Là số âm") #Hiển thị ra màn hình x là số âm
```

Kết quả chạy chương trình ***Vidu3_8.py***

```
5 : Là số dương
```

Ví dụ 3.9: Hãy tính giá trị biểu thức y sau:

$$y = \begin{cases} 0, & x < 0 \\ x, & 0 \le x < 1 \\ 2 - x, & 1 \le x < 2 \\ 22 & x \ge 2 \end{cases}$$

Vidu3_9py

```
x=6       #Khai báo biến x và gán giá trị khởi tạo bằng 6
if (x < 0):       #Kiểm tra, nếu giá trị x nhỏ hơn 0 thì
       y=0          # y nhận giá trị 0
elif (x < 1): # Nếu giá trị x lớn hơn 0 và x nhỏ hơn 1 thì
       y = x        # y nhận giá trị là x
elif (x<2): # Nếu giá trị x lớn hơn hoặc bằng 1 và nhỏ hơn 2 thì
       y = 2 – x # y nhận giá trị là 2-x
else : # Nếu giá trị x lớn hơn 2 thì
       y = 22 # y nhận giá trị 22
print("y = ",y)       #Hiển thị giá trị y ra màn hình
```

Kết quả chạy chương trình ***Vidu3_9.py***

```
y =  22
```

3.4. TẠO ỨNG DỤNG

Ví dụ 3.10: Viết chương trình cho người dùng nhập vào ba số a, b, c hãy tìm và hiển thị ra màn hình số lớn nhất trong ba số đó.

Vidu 3.10.py

```
'''
    Chương trình tìm giá trị lớn nhất (GTLN) trong ba số
'''
# Nhập vào ba số a, b, c
'''
Nhập vào một chuỗi số và chuyển sang số nguyên sau đó gán
vào biến a
'''
a = int(input("Nhập vào số a = "))
b = int(input("Nhập vào số b = "))
c = int(input("Nhập vào số c = "))
Max = a  # Đầu tiên coi a là giá trị lớn nhất (Max)
# sau đó mang Max đi so sánh lần lượt với b và c
# Nếu giá trị của Max bé hơn b thì cập nhật lại Max=b
```

```
if (Max < b): Max = b
# Nếu giá trị của Max bé hơn c thì cập nhật lại Max=c
if (Max < c): Max = c
# Hiển thị kết quả ra mà hình
print("Số lớn nhất trong ba số: ", a, ", ", b, ", ", c, "
=", Max)
```

Kết quả chạy chương trình ***Vidu3_10.py***

```
Nhập vào số a = 23
Nhập vào số b = 54
Nhập vào số c = 79
Số lớn nhất trong ba số: 23 ,  54 ,  79  = 79
```

Ví dụ 3.11: Viết chương trình cho nhập vào ba số thực a, b,c, kiểm tra xem ba số thực đó có thỏa mãn là độ dài 3 cạnh của một tam giác hay không, nếu có hãy tính diện tích và chu vi tam giác đó.

Vidu3_11.py

```
'''
    Chương trình tính diện tích và chu vi tam giác
'''
#Khai báo sử dụng thư viện math chứa các hàm toán học
import math

# Nhập từ bàn phím độ dài 3 cạnh của tam giác
a = float(input("Xin moi nhap canh a="))
b = float(input("Xin moi nhap canh b="))
c = float(input("Xin moi nhap canh c="))
#Kiểm tra ba số a, b, c có thoả mãn là 3 cạnh của 1 tam giác?
if (a>0 and b>0 and c>0 and a+b>c and b+c>a and c+a>b):
  p = (a+b+c)/2      #Tính nửa chu vi
  cv = p*2           #Tính chu vi tam giác
  dt = math.sqrt(p*(p-a)*(p-b)*(p-c)) #Tính diện tích tam giác
  # Hiển thị chu vi, diện tích ra màn hình
  print("Dien tich tam giac dt=",dt," chu vi=",cv)
else: #Ngược lại,ba số a, b,c không là 3 cạnh của tam giác
  print("Ba canh a=",a,"b=",b,"c=",c,"khong tao thanh tam giac.")
```

Kết quả chạy chương trình ***Vidu3_11.py***

```
Xin moi nhap canh a=12
Xin moi nhap canh b=9
Xin moi nhap canh c=15
Dien tich tam giac dt= 699.91  chu vi= 36.0
```

Ví dụ 3.12: Viết chương trình cho người dùng nhập vào hai số nguyên dương là tháng và năm của một năm nào đó, hãy tính và hiển thị ra màn hình số ngày tương ứng với tháng và năm đó.

Trong đó:

- Các tháng: 1, 3, 5, 7, 8, 10, 12 có 31 ngày;
- Các tháng 4, 6, 9, 11 có 30 ngày
- Tháng 2 có 28 hoặc 29 ngày tùy theo năm đó có phải là năm nhuận hay không?

Năm nhuận là năm chia hết cho 4 mà không chia hết cho 100. Ví dụ, năm 2018 không phải năm nhuận, năm 2020 là năm nhuận.

Vidu3_12.py

```
'''
    Chương trình tính số ngày của tháng và năm
'''
# Nhập vào tháng và năm của một năm nào đó
thg = int(input("Xin moi nhap vao thang = "))
nam = int(input("Xin moi nhap vao nam = "))
if thg<1 or thg >12 or nam <1900:
    print('Khong ton tai thang ', thg, ' năm ', nam)
else:
    # Kiểm tra điều kiện tháng đó có 31 ngày
    if (thg == 1 or thg == 3 or thg == 5 or thg == 7 or thg
== 8 or thg == 10 or thg == 12):
        so_ngay = 31  # Tháng có 31 ngày

    # ngược lại kiểm tra điều kiện tháng đó có 30 ngày
```

```
    elif (thg == 4 or thg == 6 or thg == 9 or thg == 11):
        so_ngay = 30  # Tháng có 30 ngày
    else : # Tháng 2
        # Nếu năm phải là năm nhuận
        if (nam % 4 == 0 and nam % 100 != 0):
            so_ngay = 29  # Tháng 2 có 29 ngày
        else: # ngược lại,
            so_ngay = 28  # Tháng 2 có 28 ngày
    # Hiển thị kết quả ra màn hình
    print("Tháng: ",thg," năm: ",nam," có: ",so_ngay, " ngày")
```

Kết quả chạy chương trình ***Vidu3–12.py***

```
Xin moi nhap vao thang = 2
Xin moi nhap vao nam = 1996
Tháng:  2  năm: 1996  có: 29  ngày
```

3.5. TỔNG KẾT BÀI HỌC

Bài học này tìm hiểu về cấu trúc điều khiển rẽ nhánh của ngôn ngữ lập trình Python. Nhìn chung, cú pháp câu lệnh và nguyên lý hoạt động của cấu trúc giống với ngôn ngữ lập trình C, Java hay C#. Cấu trúc *if* là một cấu trúc điều khiển rất quan trọng mà ngôn ngữ lập trình nào cũng sử dụng.

Để nâng cao kỹ năng sử dụng cấu trúc *if* trong chương trình, ngoài các ví dụ được nhóm tác giả đưa ra trong bài học, bạn đọc nên tìm hiểu thêm ngoài phần bài tập và nội dung bài học "Cấu trúc lặp" để bổ sung kiến thức và kỹ năng lập trình.

3.6. BÀI TẬP

A. Câu hỏi ôn tập

Câu 3.1: Chọn đáp án đúng sau khi máy chạy đoạn mã lệnh Python dưới đây?

```
if (2 == 2):
    print("Điều tất nhiên!")
```

A. Điều tất nhiên

B. Không hiển thị gì

C. True

D. False

Câu 3.2: Chọn đáp án đúng sau khi máy chạy đoạn mã lệnh Python dưới đây?

```
if ("cat" == "dog"):
    print("Mèo và chó")
else:
    print("Mèo hoặc chó")
```

A. Không hiển thị gì

B. Mèo và chó

C. Mèo hoặc chó

D. False

Câu 3.3: Chọn đáp án đúng sau khi chạy đoạn mã lệnh Python dưới đây?

```
if (True):
       print("")
else:
        print("Mèo và chó")
```

A. Không hiển thị gì

B. Mèo và chó

C. Mèo hoặc chó

D. True

Câu 3.4: Cho biết giá trị của hai biến a và b sau khai máy chạy đoạn chương trình sau?

```
a = 1
b = 10
if (a<b):
       tg = a
       a = b
       b = tg
```

A. a=10, b=1

B. a=1, b=10

C. a=1, b=1

D. tg =10, b=1, a=10

Câu 3.5: Chọn đáp án đúng sau khi máy chạy đoạn mã lệnh Python dưới đây?

```
mark1 = 8
mark2 = 7
mark = (mark1+2*mark2)/3
if (mark >= 5):
        print('Lên lớp')
else:
        print('Trượt – học lại')
```

A. Lên Lớp

B. Trượt – học lại

C. Lên lớp

D. trượt – Học lại

Câu 3.6: Cho biết kết quả hiển thị ra màn hình sau khi máy chạy đoạn mã lệnh Python dưới đây?

```
s = 0
m = 1
n = 2
if (m > n):
        tg = m
        m = n
        n = tg
else:
        m = m + n
print(s + m + n)
```

A. 2

B. 3

C. 4

D. 5

Câu 3.7: Cho biết kết quả hiển thị ra màn hình sau khi máy chạy đoạn mã lệnh Python dưới đây?

```
tk = 'Python'
mk = '123'
if (tk == 'Python' and mk == '123'):
        print('ĐĂNG NHẬP THÀNH CÔNG')
```

A. ĐĂNG NHẬP THÀNH CÔNG

B. 'ĐĂNG NHẬP THẤT BẠI'

C. 123

D. Python

Câu 3.8: Cho biết kết quả hiển thị ra màn hình sau khi máy chạy đoạn mã lệnh Python dưới đây?

```
var1 = True
var2 = False
var3 = False
if (var1 or var2 and var3):
        print("True")
else:
        print("False")
```

A. True

B. False

C. True False

D. False True

Câu 3.9: Cho biết kết quả hiển thị ra màn hình sau khi máy chạy đoạn mã lệnh Python dưới đây?

```
x = True
y = False
z = False
if (not x or y):
        print(1)
elif (not x or not y and z):
        print(2)
elif (not x or y or not y and x):
        print(3)
else:
        print(4)
```

A. 1

B. 2

C. 3

D. 4

Câu 3.10: Cho biết kết quả hiển thị ra màn hình sau khi máy chạy đoạn mã lệnh Python dưới đây?

```
mark1=7
mark2=8
mark=(mark1+2*mark2)/3
mark=4
if (mark<5):
        HocLuc = 'Yếu'
elif (mark<7):
        HocLuc = 'Trung bình'
elif (mark < 8):
        HocLuc = 'Khá'
else:
        HocLuc = 'Giỏi'
print(HocLuc)
```

A. 'Yếu'

B. 'Trung bình'

C. 'Khá'

D. 'Giỏi'

B. Lập trình

Bài tập 3.1: Viết chương trình cho người dùng nhập ba số từ bàn phím sau đó tìm số nhỏ nhất trong ba số đó và hiển thị kết quả ra màn hình.

Bài tập 3.2: Viết chương trình giải phương trình bậc nhất ax+b = 0 với a, b được nhập vào từ bàn phím.

Bài tập 3.3: Viết chương trình giải phương trình bậc hai $ax^2+bx + c = 0$ với a, b, c được nhập vào từ bàn phím.

Bài tập 3.4: Viết chương trình cho người dùng nhập vào từ bàn phím hai số a, b và một ký tự ch. Kiểm tra nếu: ch là "+" thì thực hiện phép tính a + b và in kết quả lên màn hình, nếu ch là "–" thì thực hiện phép tính a – b và in kết quả lên màn hình, nếu ch là "*" thì thực hiện phép tính a * b và in kết quả lên màn hình, nếu ch là "/" thì thực hiện phép tính a / b và in kết quả lên màn hình, nếu ch là ký tự khác các ký tự trên thì hiển thị ra màn hình 'ký tự ' ch ' không phải là một toán tử' .

Bài tập 3.5: Viết chương trình cho người dùng nhập vào thông tin về cán bộ gồm họ tên, chức vụ, sau đó hiển thị ra màn hình phụ cấp mà cán bộ đó được nhận; biết rằng tiền phụ cấp của cán bộ được tính như sau:

– Nếu chức vụ là: Giám đốc, thì phụ cấp là 5000000

– Nếu chức vụ là: Phó giám đốc, thì phụ cấp là 3000000

– Nếu chức vụ là: Trưởng phòng, thì phụ cấp là 500000

– Các trường hợp khác thì phụ cấp là 0

Bài tập 3.6: Xây dựng chương trình nhập vào mã sinh viên (masv), họ tên (hoten) và điểm rèn luyện (diemRL) của sinh viên. Hãy xét hạnh kiểm cho sinh viên biết hạnh kiểm của sinh viên được xét dựa trên điểm rèn luyện như sau:

– Sinh viên có hạnh kiểm: xuất sắc, nếu điểm rèn luyện lớn hơn hoặc bằng 90.

– Sinh viên có hạnh kiểm: tốt, nếu 80 nhỏ hơn hoặc bằng điểm rèn luyện nhỏ hơn 90.

– Sinh viên có hạnh kiểm: khá, nếu 65 nhỏ hơn hoặc bằng điểm rèn luyện nhỏ hơn 80.

– Sinh viên có hạnh kiểm: trung bình, nếu 50 nhỏ hơn hoặc bằng điểm rèn luyện nhỏ hơn 65.

– Sinh viên có hạnh kiểm: yếu, trong trường hợp điểm rèn luyện nhỏ hơn 50.

Bài tập 3.7: Viết chương trình cho người dùng nhập vào số điện tiêu thụ (số ki–lô–wát trên giờ) của một hộ sử dụng điện, sau đó tính và hiển thị ra màn hình tiền điện của chủ hộ tính theo giá bậc thang sau:

– 50 kwh (ki–lô–wát trên giờ) đầu tiên giá 2000đ/kwh

– 100 kwh tiếp theo tính theo giá 2500đ/ kwh

– 100 kwh tiếp theo tính theo giá tính giá 3000đ/ kwh

– 100 kwh tiếp theo tính theo giá tính giá 3500đ/ kwh

– Trên 400 kwh tính theo giá 4000đ/ kwh

Bài 4

CẤU TRÚC LẶP

Bài học này sẽ trình bày hai cấu trúc lặp thường được sử dụng khi lập trình là: while và for. Cấu trúc while, cung cấp một cách để lặp đi lặp lại mã trong vòng lặp nhiều lần. Cấu trúc for, được thiết kế để duyệt qua các mục (items) theo trình tự và chạy một khối lệnh cho tương ứng với mỗi lần duyệt.

Sau bài học này, người học có thể:

- *Sử dụng được cấu trúc lặp while*
- *Sử dụng được cấu trúc lặp for*
- *Sử dụng được lệnh break và continue*
- *Tạo được ứng dụng bảng cửu chương*
- *Tạo được ứng dụng tính giá trị đa thức (bậc chẵn)*
- *Tạo được ứng dụng tính số tiền sau 5 năm gửi tiết kiệm*

4.1. ĐẶT VẤN ĐỀ

Trong phần này, trước tiên ta xét hai bài toán là xây dựng chương trình hiển thị bảng nhân i và xây dựng chương trình hiển thị bảng cửu chương.

Bài toán 4.1: Hãy xây dựng chương trình máy tính để hiển thị bảng nhân i (giả sử chọn i=2).

- Phương án 1: Dùng lệnh *print* để hiển thị bảng nhân i.
- Phương án 2: Sử dụng cấu trúc lặp *for* để xây dựng chương trình giải bài toán này.

Ta xét hai chương trình 4.1 và 4.2 tương ứng với hai phương án trên dưới đây:

Chương trình 4.1: *Hiển thị bảng nhân 2*

```
i=2
print(i,' * ',1,'=', i*1)
print(i,' * ',2,'=', i*2)
print(i,' * ',3,'=', i*3)
print(i,' * ',4,'=', i*4)
print(i,' * ',5,'=', i*5)
print(i,' * ',6,'=', i*6)
print(i,' * ',7,'=', i*7)
print(i,' * ',8,'=', i*8)
print(i,' * ',9,'=', i*9)
print(i,' * ',10,'=', i*10)
```

Chương trình 4.2: *Hiển thị bảng nhân 2 với cấu trúc lặp for*

```
i=2
for j in range (1,11,1):
    print(i,' * ',j,'=', i*j)
```

Nhận xét từ chương trình 4.1 và chương trình 4.2:

- Cả 2 chương trình đều giải quyết được yêu cầu của bài toán 4.1.
- Chương trình 4.2 ngắn gọn hơn chương trình 4.1.
- Với bài toán 4.1 này nên giải quyết theo phương án 2 với chương trình 4.2 như ở trên.

Bài toán 4.2: Hãy xây dựng chương trình máy tính để hiển thị bảng cửu chương

Phương án 1: Chúng ta có thể dùng 90 lệnh *print* để hiển thị bảng cửu chương. Phương án này tốn nhiều thời gian vì mã lệnh dài và phức tạp, bên cạnh đó do số câu lệnh nhiều nên khả năng chương trình bị lỗi cao.

Ta xét phương án 2 với chương trình 4.3 như sau:

Chương trình 4.3: *Hiển thị bảng cửu chương với cấu trúc lặp for*

```
for i in range (1,10,1):
    for j in range (1,11,1):
        print (i,' * ',j,'=',i*j)
```

Bài toán 4.3: Hãy xây dựng chương trình máy tính để giải bài toán gửi tiền tiết kiệm như sau: Giả sử chúng ta có số tiền là a gửi vào một ngân hàng nào đó. Hỏi sau bao nhiêu tháng ta thu được số tiền là b (b>a) biết rằng lãi suất hàng tháng được tính bằng 5%.

Xác định yêu cầu và cấu trúc dữ liệu:

Dữ liệu vào: Biến thực a và b (lưu trữ số tiền gửi và số tiền thu được).

Dữ liệu ra: Số tháng cần gửi t kiểu nguyên.

Cách giải:

Ta chưa biết sau bao nhiêu tháng thì số tiền thu được bằng b (b>a), do đó nếu chỉ dùng cấu trúc rẽ nhánh mà không dùng cấu trúc lặp thì chúng ta sẽ không giải quyết được bài toán này.

Theo đầu bài: Lãi suất hàng tháng là 5%, nên ta có số tiền lãi thu được sau 1 tháng bằng gốc *0.05 và sau mỗi tháng ta có tiền cả gốc và lãi sẽ là: a = a + a*0.05.

Công việc lặp:

a = a + a*0.05

a = t + 1

Điều kiện dừng: Tiền thu được a >= b hay điều kiện lặp là số tiền thu được a < b

Chương trình minh họa giải bài toán gửi tiền tiết kiệm như sau:

Chương trình 4.3: *Bài toán gửi tiền tiết kiệm*

```
# Nhập số tiền cần gửi thoả điều kiện >0
while True:
    a=float(input('số tiền gửi:') )
    if a>0 : break
# Nhập số tiền tối thiểu muốn có >= số tiền gửi
```

```
while True:
    b=float(input('số tiền tối thiểu cần có:'))
    if b >=a: break
t=0 #Mặc định thời điểm xét là lúc gửi tiền
#  Di tim thoi gian cần thiết
while(a<b):
    a = a + a * 0.05
    t+=1
    print(a)
print('Số tháng cần gửi là: ', t)
```

Nhận xét qua 3 bài toán 4.1, bài toán 4.2 và bài toán 4.3 ở trên ta thấy:

- Bài toán 4.1 hiển thị bảng nhân, bài toán 4.2 hiển thị bảng cửu chương ở trên ta thấy với việc sử dụng cấu trúc lặp khi xây dựng chương trình sẽ giúp mã chương trình sáng sủa và ngắn gọn hơn.
- Bài toán gửi tiền tiết kiệm có thể giải quyết tự động bằng cách lặp công việc tính lãi suất theo từng tháng và cộng dồn thêm lãi suất vào gốc. Bài toán này rất khó có thể giải quyết được nếu chỉ dùng lệnh nhập xuất và rẽ nhánh đơn thuần.
- Cấu trúc lặp là cấu trúc điều khiển rất quan trọng trong khi lập trình.

4.2. CẤU TRÚC LẶP WHILE

Trong quá trình lập trình, có những đoạn chương trình được lặp đi lặp lại nhiều lần với một điều kiện nào đó, khi đó ta nên sử dụng các cấu trúc lặp để thực hiện các công việc này. Cấu trúc while sẽ thực hiện các công việc (khối lệnh) khi nào điều kiện còn đúng.

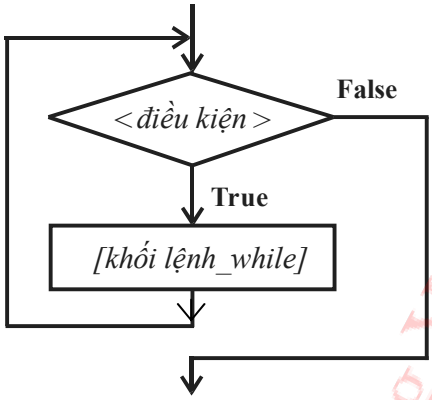
Cú pháp: ***while*** *(<điều kiện>):*

<khối lệnh_while>

Trong đó:

- *<điều kiện>:* Là một biểu thức lô–gíc chỉ điều kiện cho kết quả là đúng (True) hoặc sai (False)
- *<khối lệnh_while>:* Bao gồm một hoặc nhiều câu lệnh được chạy khi biểu thức lô–gíc có giá trị đúng (True), các câu lệnh trong khối lệnh có lề trái dịch sang bên phải một khoảng cách trống so với từ khóa while.

Lưu đồ thuật toán cấu trúc while:



Hình 4.1: Lưu đồ thuật toán cấu trúc while

Giải thích nguyên lý hoạt động của cấu trúc lặp while: Khi gặp cấu trúc while, máy thực hiện các bước sau:

- Bước 1: Kiểm tra giá trị của biểu thức *<điều kiện>*, nếu *<điều kiện>* có giá trị đúng (True) thì thực hiện Bước 2; ngược lại (False) chuyển sang Bước 3.
- Bước 2: Thực hiện các lệnh trong *<khối lệnh_while>*, sau đó quay lên thực hiện tiếp Bước 1.
- Bước 3: Kết thúc lặp while.

Ví dụ 4.1: Xây dựng chương trình hiển thị ra màn hình các tháng trong năm.

Vidu4_1.py

```
'''
Chương trình hiển thị ra màn hình các tháng trong năm
'''
i =1 #Khai báo và gán vào biến i bắt đầu từ tháng 1
#Khi nào biến i còn nhỏ hơn 13 thực hiện (chạy khối lệnh)
while (i <13):
    # Hiển thị thang i
    print("Tháng "+ str(i))
    # Tăng giá trị của biến i lên 1 đơn vị
    i = i+1
print("Trên đây là 12 tháng trong năm")
```

Kết quả chạy chương trình ***Vidu4_1.py***

```
Tháng 1
Tháng 2
Tháng 3
Tháng 4
Tháng 5
Tháng 6
Tháng 7
Tháng 8
Tháng 9
Tháng 10
Tháng 11
Tháng 12
Trên đây là 12 tháng trong năm
```

Ví dụ 4.2: Viết chương trình đếm số ước dương của số nguyên n

Vidu4_2.py

```
'''
    Chương trình  đếm số ước dương của số nguyên n
'''
n=int(input("Nhập vào số nguyên n = ") ) #Nhập n
i=1      #Biến chạy, cần chạy từ 1 tới n
dem=0    #Lưu số ước của n
#  m= |n|
if n < 0:  m = -n
else : m = n
#Khi nào biến i còn nhỏ hơn m thực hiện (chạy khối lệnh)
while (i <= m):
    #Nếu i là ước của n thì tăng biến dem lên 1 đơn vị
    if n%i==0:  dem+=1
    #Tăng giá trị trong biến I lên 1 đơn vị
    i+=1

print(n, " có số ước dương là: ",dem)
```

Kết quả chạy chương trình ***Vidu4_2.py***

```
Nhập vào số nguyên n = 10
10  có số ước dương là:  4
```

Chú ý: Trong cấu trúc *while*, việc xác định điều kiện để thoát khỏi cấu trúc lặp là vô cùng quan trọng, nếu ta không xác định được khi nào điều kiện bằng False, máy sẽ chạy khối lệnh trong cấu trúc lặp mãi mãi. Ở hai ví dụ trên nếu ta không tăng biến i lên một giá trị thì biểu thức điều kiện (i <= 10) sẽ luôn thỏa mãn và chương trình sẽ không bao giờ thoát khỏi vòng *while* được.

Ví dụ 4.3: Xây dựng chương trình nhập vào một số nguyên là tháng trong năm, sau đó hiển thị ra màn hình tháng đó thuộc mùa nào trong năm.

Nhận xét: Ta thấy rằng một năm có 12 tháng tương ứng với các số nguyên từ 1 đến 12, do đó nếu người dùng nhập vào một số không hợp lệ (nhỏ hơn 1 hoặc lớn hơn 12) thì đó là một số không phù hợp với 12 tháng, khi đó chúng ta sẽ đưa ra thông báo lỗi và yêu cầu người dùng nhập lại.

Vidu4_3.py

```
'''
    Chương trình hiển thị mùa trong năm
'''
loop = True
#Biến loop dùng để chỉ cấu trúc while khi nào được thực hiện
while loop:
  thang=int(input("Nhập vào một số nguyên tương ứng với tháng: "))
  if (thang <1 or thang >12):
        print("Tháng "+str(thang)+ " không hợp lệ!")
        print("Xin mời nhập lại:")
  else: # Thay đổi cờ để thoát khỏi cấu trúc lặp while
        loop = False
'''
 Khối lệnh bên ngoài cấu trúc lặp while (có căn lề trái bằng while)
'''
```

```
# Kiểm tra nếu tháng nhập vào bằng 1 hoặc 2 hoặc 3 thì
if(thang ==1 or thang ==2 or thang ==3):
    print ("Tháng "+ str(thang)+" là mùa xuân")
# Ngược lại, Nếu tháng nhập vào bằng 4 hoặc 5 hoặc 6 thì
elif (thang == 4 or thang == 5 or thang == 6):
    print("Tháng " + str(thang) + " là mùa hạ")
#Ngược lại, Nếu tháng nhập vào bằng 7 hoặc 8 hoặc 9 thì
elif (thang == 7 or thang == 8 or thang == 9):
    print("Tháng " + str(thang) + " là mùa thu")
else: #Ngược lại, tháng nhập là 10 hoặc 11 hoặc 12 thì
        print("Tháng " + str(thang) + " là mùa đông")
```

Kết quả chạy chương trình ***Vidu4_3.py***

```
Nhập vào một số nguyên tương ứng với tháng: 13
Tháng 13 không hợp lệ!
Xin mời nhập lại:
Nhập vào một số nguyên tương ứng với tháng: 6
Tháng 6 là mùa hạ
```

4.3. CẤU TRÚC LẶP FOR

Chúng ta có thể sử dụng cấu trúc lặp while để viết chương trình thực hiện các công việc được lặp đi lặp lại nhiều lần. Ngoài ra, Python còn cung cấp cho chúng ta cấu trúc lặp for để giúp cho người lập trình có thể thao tác với nhiều kiểu dữ liệu khác nhau một cách thuận tiện và dễ dàng.

Cú pháp:

for *<tên biến>* **in** *<tập hợp>:*

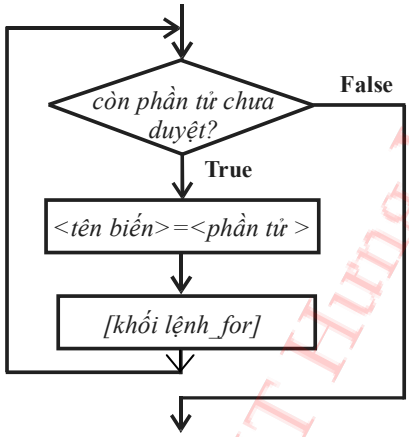
<khối lệnh_for>

Trong đó:

- *<tên biến>:* Là một tên do người dùng đặt theo quy tắc đặt tên.
- *<tập hợp>:* Gồm một tập các phần tử của một List, Chuỗi, Tuple,….

- <*khối lệnh_for*>: Bao gồm một hoặc nhiều câu lệnh được chạy khi duyệt qua các phần tử trong tập hợp, các câu lệnh trong khối lệnh có lề trái dịch sang bên phải một khoảng cách (một khoảng trống) so với câu lệnh for.

Lưu đồ thuật toán cấu trúc for:



Hình 4.2: Lưu đồ thuật toán cấu trúc for

Giải thích nguyên lý hoạt động của cấu trúc lặp for: Khi gặp cấu trúc for, máy thực hiện các bước sau:

- Bước 1: Kiểm tra xem trong <tập hợp> có còn phần tử chưa được duyệt hay không? Nếu còn (True) thì thực hiện Bước 2; ngược lại (False) sang Bước 5.
- Bước 2: Lấy ra phần tử đầu tiên chưa được duyệt trong <tập hợp> và gán cho <*tên biến*>.
- Bước 3: Thực hiện [*khối lệnh _for*].
- Bước 4: Quay lên thực hiện tiếp Bước 1.
- Bước 5: Kết thúc cấu trúc lặp for.

Cấu trúc lặp for thường được dùng để thực hiện duyệt qua lần lượt các phần tử trong một tập hợp, danh sách hoặc chuỗi, tương ứng với mỗi phần tử được duyệt, máy tính sẽ thực hiện các lệnh trong cấu trúc lặp một lần. Như vậy số lần các câu lệnh trong thân cấu trúc lặp được thực hiện đúng bằng số phần tử trong tập hợp, danh sách hoặc chuỗi.

Ví dụ 4.4: Viết chương trình tính tổng các phần tử từ 1 đến 5, sau đó hiển thị kết quả ra màn hình.

Vidu4_4.py

```
'''
    Chương trình tính tổng các số nguyên từ 1 đến 5
'''
s = 0        #Khởi tạo biến lưu trữ tổng các phần tử
for i in [1,2,3,4,5]:#Duyệt qua tất cả các phần tử trong tập hợp
    s = s + i
#Khối lệnh bên ngoài cấu trúc lặp for (có căn lề trái bằng for)
print("Tổng = ", s)
```

Kết quả chạy chương trình ***Vidu4_4.py***

```
Tổng =  15
```

Ví dụ 4.5: Viết chương trình hiển thị ra màn hình 100 số tự nhiên đầu tiên.

Vidu4_5.py

```
'''
    Chương trình hiển thị 100 số tự nhiên đầu tiên
'''
#Duyệt qua tất cả các phần tử trong tập hợp từ 0 đến 99
for i in range(100):
    print(i)
```

Kết quả chạy chương trình ***Vidu4_5.py***

```
0
1
2
3
4
….
98
99
```

Trong ví dụ trên, chúng ta thấy có hàm range(), hàm này được sử dụng để tạo ra một danh sách, dãy số.

Cú pháp:

range([<*start*>,] <*end*> [, <*step*>])

Trong đó:

- Hàm range sẽ tạo ra một dãy các phần tử có giá trị từ <*start*> đến <*end*> – *1* theo bước nhảy giữa các phần tử tạo ra là <*step*>
- <*start*>*:* Chỉ định phần tử bắt đầu, tạo ra (nhỏ nhất) trong dãy, nếu không có thì Python mặc định bắt đầu từ 0.
- <*end*>*:* Chỉ định phần tử kết thúc, tạo ra (lớn nhất) trong dãy, bắt buộc phải có phần tử này.
- <*step*>*:* Chỉ định bước nhẩy (khoảng cách) giữa hai số <*start*> và <*end*>, nếu không có thì Python sẽ sử dụng giá trị mặc định, nó bằng 1.

Ví dụ 4.6: Sử dụng hàm range() để tạo một dãy (danh sách) các số.

Vidu4_6.py

```
#Tạo ra một danh sách các phần tử và hiển thị ra màn hình
# Tạo các phần tử từ start = 0 tới end-1=10-1=9, step mặc định =1
print(range(0,10))
# Tạo các phần tử từ start mặc định 0 tới end-1=9-1=8, step mặc định =1
print(list(range(9)))
# Tạo các phần tử từ start 2 tới end-1=4, step mặc định =1
print(list(range(2, 5)))
# Tạo các phần tử từ start = 0 tới end-1=14, step =5
print(list(range(0, 15, 5)))
```

Kết quả chạy chương trình ***Vidu4_6.py***

```
range(0, 10)
[0, 1, 2, 3, 4, 5, 6, 7, 8]
[2, 3, 4]
[0, 5, 10]
```

4.4. LỆNH BREAK VÀ CONTINUE

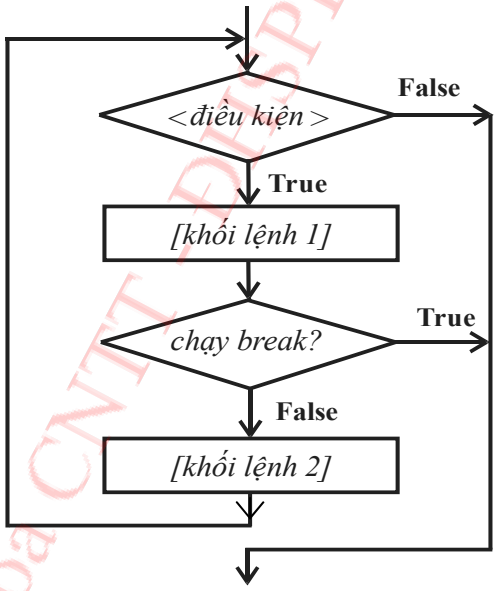
Các cấu trúc lặp thực hiện lặp đi, lặp lại một khối lệnh bên trong nó cho đến khi điều kiện bằng *False*, nhưng trong thực tế, đôi khi chúng ta muốn chấm dứt công việc lặp (thoát ra khỏi cấu trúc lặp) hoặc bỏ qua phần còn lại của khối lệnh bên trong cấu trúc lặp để chuyển lên kiểm tra biểu thức điều kiện ngay lập tức, khi đó chúng ta cần sử dụng lệnh *break* và *continue*.

4.4.1. Lệnh break

Python cung cấp lệnh *break* để thoát khỏi cấu trúc lặp ngay lập tức mà không cần kiểm tra điều kiện, nếu lệnh *break* được đặt trong nhiều cấu trúc lặp lồng nhau (cấu trúc lặp bên trong một cấu trúc lặp khác), thì lệnh *break* sẽ kết thúc cấu trúc lặp chứa nó và chuyển điều khiển đến lệnh tiếp theo sau cấu trúc lặp đó.

Cú pháp: **break**

Lưu đồ thuật toán cấu trúc lặp có chứa lệnh break:



Hình 4.3: Lưu đồ thuật toán cấu trúc lặp có chứa lệnh break

Ví dụ 4.7: Chương trình đọc tất cả các ký tự trong chuỗi "Python ngôn ngữ lập trình" và kiểm tra điều kiện, nếu chương trình đọc được chữ cái "n"

đầu tiên trong chuỗi thì chương trình kết thúc; ngược lại thì in ký tự ra màn hình.

Vidu4_7.py

```
#Chương trình minh họa sử dụng lệnh break
#Duyệt qua tất cả các phần tử trong chuỗi
for i in "Python ngôn ngữ lập trình":
    print(i)       #Hiển thị ký tự ra màn hình
    if (i == "n"): #Kiểm tra xem ký tự đọc được có bằng "n"?
        break  #nếu đúng, thoát ra ngoài cấu trúc for ngay
print("Kết thúc!")
```

Kết quả chạy chương trình ***Vidu4_7.py***

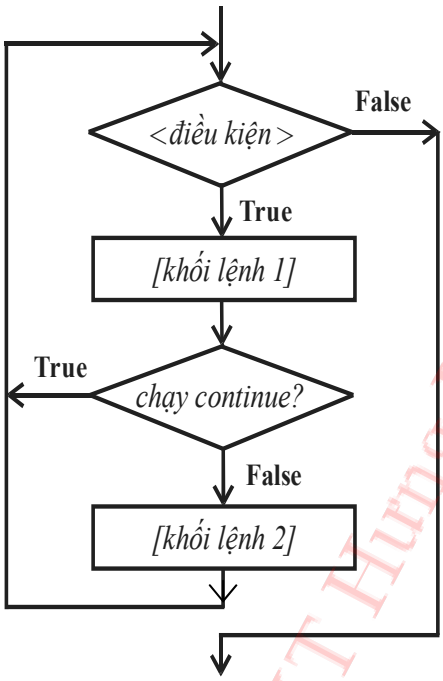
```
P
y
t
h
o
n
Kết thúc!
```

4.4.2. Lệnh continue

Khác với lệnh *break*, lệnh *continue* bỏ qua các lệnh phía dưới nó trong cấu trúc lặp và quay lên đầu kiểm tra điều kiện lặp ngay lập tức để tiếp tục với lần lặp kế tiếp.

Cú pháp: **continue**

Lưu đồ thuật toán cấu trúc lặp có chứa lệnh *continue*:



Hình 4.4: Lưu đồ thuật toán cấu trúc lặp có chứa lệnh *continue*

Ví dụ 4.8: Chương trình hiển thị ra màn hình các số từ 9 đến 0 và kiểm tra điều kiện, nếu bằng 5 thì chương trình không hiển thị mà quay lên đầu cấu trúc lặp.

Vidu4_8.py

```
#Chương trình minh họa sử dụng lệnh continue
a = 10       #Khai báo và khởi tạo biến nguyên a bằng 10
while (a > 0):  #Khi biến a còn lơn hơn không thì thực hiện
    a = a - 1   #Giảm a đi 1 đơn vị
    if (a == 5):#nếu a đúng bằng  5
      continue #Quay lên đầu cấu trúc lặp, kiểm tra điều kiện a>0
    print('Giá trị biến hiện tại là: ', a)
print("Kết thúc...")
```

Kết quả chạy chương trình ***Vidu4_8.py***

```
Giá trị biến hiện tại là:  9
Giá trị biến hiện tại là:  8
Giá trị biến hiện tại là:  7
Giá trị biến hiện tại là:  6
Giá trị biến hiện tại là:  4
Giá trị biến hiện tại là:  3
Giá trị biến hiện tại là:  2
Giá trị biến hiện tại là:  1
Giá trị biến hiện tại là:  0
Kết thúc...
```

Ví dụ 4.9: Viết chương trình cho người dùng nhập vào một chuỗi họ tên với điều kiện dữ liệu nhập vào phải khác trống (rỗng), nếu người dùng cố tình nhập dữ liệu rỗng thì chương trình yêu cầu nhập lại; ngược lại thì hiển thị họ tên vừa nhập ra màn hình.

Vidu4_9.py

```
#Chương trình minh họa sử dụng lệnh break
#Cấu trúc lặp vô hạn, cần có lệnh break để thoát ra khỏi lặp
while True:
        #Cho nhập vào một chuỗi và gán vào biến hoten
       hoten = input("Xin mời nhập họ tên: ")
       if (hoten != ""):   #Kiểm tra nếu biến hoten khác “” thì
              break        #thoát khỏi cấu trúc lặp while
       print("Xin mời nhập lại, họ tên phải khác rỗng !")
print("Họ tên bạn vừa nhập = ",hoten)
```

Kết quả chạy chương trình ***Vidu4_9.py***

```
Xin mời nhập họ tên: ↵
Xin mời nhập lại, họ tên phải khác rỗng !
Xin mời nhập họ tên: Đào Thị Hồng Lê Mận
Họ tên bạn vừa nhập =  Đào Thị Hồng Lê Mận
```

4.5. TẠO ỨNG DỤNG

Ví dụ 4.10: Viết chương trình cho người dùng nhập một số nguyên n, hãy kiểm tra xem số vừa nhập có phải là số nguyên tố hay không.

Phân tích:

– Nhập vào một số nguyên n từ bàn phím ta sử dụng lệnh

n = int(input('Nhập số nguyên: '))

– Ta biết rằng số nguyên tố là một số dương và chỉ có 2 ước số dương khác nhau là 1 và chính nó. Do vậy để kiểm tra một số nguyên n có phải là số nguyên tố hay không ta thực hiện các bước sau:

- Bước 1: Nếu n < 2, thông báo n không phải là số nguyên tố.
- Bước 2: Nếu n >=2, nếu tồn tại một số dương nào đó khác 1 và n là ước số của n thì n không là số nguyên tố; ngược lại n là số nguyên tố.
- Bước 3: Kết quả.

Vidu4_10.py

```
'''
    Chương trình kiểm tra n có phải là số nguyên tố hay không
'''
n = int(input("Xin mời nhập n = "))
# Giả sử n là số nguyên tố
is_snt = True;
# Nếu n <2 thì n không là số nguyên tố
if n<2:
    is_snt=False
else:
    i=2
    while (i <= n/2):
        if (n % i == 0):  # i là ước của n thì
            is_snt = False  # n không là số nguyên tố
            break; #Kết thúc vòng lặp
        i = i + 1  # Tăng giá trị của i lên 1
if is_snt== True:
    print(n, ' là số nguyên tố')
else:
    print(n, ' không là số nguyên tố')
```

Kết quả chạy chương trình ***Vidu4_10.py***

```
Xin mời nhập n = 10
10  không là số nguyên tố
```

Ví dụ 4.11: Viết chương trình cho người dùng nhập vào hai số x và n sau đó tính tổng $S(x, n) = x^2 + x^4 + x^6 + \ldots + x^{2n}$ rồi hiển thị kết quả ra màn hình.

Phân tích:

- Nhập vào số nguyên dương n và số thực x.
- Dùng cấu trúc lặp for hoặc while để duyệt qua tất cả các phần tử i nhận từ 1 tới n.
- Tính tổng $S(x, n) = x^2 + x^4 + x^6 + \ldots + x^{2n}$ (chúng ta có thể sử dụng toán tử ** hoặc hàm pow() trong thư viện math để tính số mũ).
- Thoát khỏi cấu trúc lặp, hiển thị kết quả ra màn hình.

Vidu4_11.py

```
'''
    Chương trình tính tổng S(x, n) = x2 + x4 + x6 +...+ x2n
'''
#Nhập vào số nguyên gán vào biến n
n = int(input("Xin mời nhập n = "))
#Nhập vào số thực gán vào biến x
x = float(input("Xin mời nhập x = "))
s = 0                   #Khởi tạo biến s để lưu trữ tổng
for i in range(1,n+1):   #Duyệt qua các phần tử từ 1 đến n + 1
       s = s + x**(2*i) #Tính tổng gán vào biến s
print("Tổng s = ", s) #Hiển thị giá trị tổng S(x,n) ra màn hình
```

Kết quả chạy chương trình ***Vidu4_11.py***

```
Xin mời nhập n = 11
Xin mời nhập x = 2
Tổng s =  5592404.0
```

Ví dụ 4.12: Một sinh viên tiết kiệm được số tiền 8,000,000 (tám triệu đồng). Sau khi cân nhắc, sinh viên này quyết định tạo một sổ tiết kiệm và gửi vào ngân hàng với lãi suất 9% một tháng.

Hãy viết chương trình tính xem sau 5 năm sinh viên này sẽ nhận được số tiền bằng bao nhiêu (được làm tròn đến hàng nghìn) biết rằng trong khoảng thời gian 5 năm này sinh viên không thực hiện giao dịch rút tiền tại ngân hàng này.

Phân tích:

- Ta thấy số dư cuối tháng tại tài khoản ngân hàng của anh sinh viên đó được tính bằng số dư đầu tháng cộng với tiền lãi nhận được của tháng đó, tiền lãi tháng = số dư đầu tháng * 9%.
- Để tính số dư sau 5 năm (60 tháng), ta thực hiện tính số dư cuối của tháng thứ 60 (số dư đầu tháng sau chính là số dư cuối của tháng trước đó).

Vidu4_12.py

```
'''
    Chương trình tính lãi suất tiền gửi tiết kiệm cho sinh viên
'''
#Khởi tạo biến du_dau để lưu trữ tiền gửi tiết kiệm
du_dau = 8000000;
so_thang = 60      #Khởi tạo biến so_thang , tính cho 5 năm gửi
lai_xuat =0.9/100  #Khởi tạo biến lưu lãi xuất trên tháng
for i in range(so_thang):
        # Tính số dư cuối tháng
        du_cuoi = du_dau+ du_dau * lai_xuat
# Số dư cuối tháng trước sẽ là số dư của đầu tháng tiếp theo    du_dau =
du_cuoi
#Sau khi tính xong, hiển thọ kết quả ra màn hình
print("Sau "+ str(so_thang/12)+" năm số dư trong tài khoản của sinh viên
là: "+ str(round(int(du_cuoi),-3)))
```

Kết quả chạy chương trình ***Vidu4_12.py***

```
Sau 5.0 năm số dư trong tài khoản của sinh viên là: 13695000
```

4.6. TỔNG KẾT BÀI HỌC

Bài học đã giới thiệu với bạn đọc hai cấu trúc lặp khác nhau là *while* và *for* trong khi lập trình. Với cấu trúc *while*, các câu lệnh trong thân cấu trúc được thực hiện khi nào biểu thức điều kiện có giá trị bằng *True*, do đó người dùng cần xác định khi nào điều kiện lặp có giá trị bằng *False* để thoát khỏi lặp; nếu không khối lệnh sẽ lặp vô hạn (máy sẽ chạy mãi mãi). Với cấu trúc *for*, chúng ta thường sử dụng để duyệt qua một tập gồm nhiều phần tử (ta có thể biết trước số phần tử), ứng với mỗi phần tử máy sẽ chạy khối lệnh trong cấu trúc lặp một lần và máy sẽ thoát khỏi cấu trúc lặp khi duyệt qua hết các phần tử. Bài học cũng giới thiệu về câu lệnh *break* để kết thúc cấu trúc lặp ngay lập tức và câu lệnh *continue* bỏ qua các lệnh sau nó và quay lên kiểm tra điều kiện lặp.

Để hiểu thêm về cách ứng dụng các cấu trúc lặp vào viết chương trình trong Python, bài học tiếp theo trong cuốn sách nói về kiểu dữ liệu có cấu trúc trong Python sẽ giúp người đọc thấy rõ hơn.

4.7. BÀI TẬP

A. Câu hỏi ôn tập

Câu 4.1: Chọn các phát biểu đúng sau đây đối với cấu trúc lặp while (cho phép chọn nhiều đáp án)

A. Khối lệnh lặp trong cấu trúc lặp while được thực hiện ít nhất một lần.

B. Khối lệnh lặp trong cấu trúc lặp while có thể không được thực hiện lần nào.

C. Các câu lệnh thực hiện công việc lặp phải được viết thẳng hàng dọc và căn lề trái một khoảng so với từ khóa while.

D. Biểu thức điều kiện chỉ được nhận giá trị True hoặc False.

Câu 4.2: Phát biểu nào là sai trong các phát biểu sau đây đối với cấu trúc lặp for

A. Bắt buộc phải có từ khóa in khi sử dụng cấu trúc lặp for.

B. [*Khối lệnh_for*] có thể không chứa câu lệnh nào.

C. Các câu lệnh trong [*khối lệnh_for*] phải cùng viết dịch sang phải một khoảng cách so với từ khoá for.

D. Các câu lệnh trong [*khối lệnh_for*] phải được căn lề trái thẳng hàng.

Câu 4.3: Cho biết giá trị của biến *dem* sau khi máy thực hiện đoạn chương trình Python dưới đây?

```
dem = 0
while (True):
        dem += 1
        if (dem == 3):
                break
```

A. 1

B. 2

C. 3

D. 4

Câu 4.4: Cho biết giá trị của biến S sau khi máy thực hiện đoạn chương trình Python sau?

```
dem = 1
s =
while (False):
        s = s + 1
        if (dem < 3):
                break;
```

A. 1

B. 2

C. 3

D. 0

Câu 4.5: Cho biết kết quả hiển thị ra màn hình sau khi máy chạy đoạn chương trình Python sau?

```
i = 1
s=0
while (i<9):
    if (i%3 == 0):
        s=s+i
    i + = 1
print(i)
```

A. 0

B. 9

C. 18

D. Chương trình không dừng lại.

Câu 4.6: Cho biết kết quả hiển thị ra màn hình sau khi máy chạy đoạn chương trình Python sau?

```
i = 1
while (True):
        if (i > 7):
                break
        i = i + 1
print(i)
```

A. 1

B. 9

C. 7

D. 8

Câu 4.7: Cho biết kết quả hiển thị ra màn hình sau khi máy chạy đoạn chương trình Python sau?

```
i = 5
while (True):
    if (i% 11) > 0:
            break
        print(i)
        i += 1
```

A. 5

B. 6

C. Chương trình không hiển thị giá trị nào

D. Chương trình báo lỗi

Câu 4.8: Cho biết kết quả hiển thị ra màn hình sau khi máy chạy đoạn chương trình Python sau?

```
for item in [1,3,5,8]:
        print(item)
```

A. 1,3,4,5,8

B. [1,3,5,8]

C. 1
   3
   5
   8

D. 1358

Câu 4.9: Cho biết đoạn chương trình nào tính tổng các số nguyên từ 1 tới 10 S=1+2+...+10, với S khởi được tạo bằng 0:

A. **for** i **in** range (0,10,1): S = S + i

B. **for** i **in** range (1,10,1): S = S + i

C. **for** i **in** range (11): S = S + i

D. **for** i **in** range (10): S = S + i

Câu 4.10: Cho biết đoạn chương trình nào tính tích các số nguyên lẻ từ 1 tới 10 (S=1 * 3 * 5 * 7 * 9, với S khởi được tạo bằng 0):

A. **for** i **in** range (0,10,2): S = S * i

B. **for** i **in** range (1,10,2): S = S * i

C. **for** i **in** range (1,9,2): S = S * i * i

B. Lập trình

Bài tập 4.1: Xây dựng chương trình hiển thị ra màn hình 100 dòng 'Hello!'.

Bài tập 4.2: Xây dựng chương trình cho phép người dùng nhập vào năm sinh của một người nào đó từ bàn phím thoả điều kiện năm sinh lớn hơn 1900 và nhỏ hơn hoặc bằng năm hiện tại, sau đó hiển thị ra màn hình tuổi của người đó.

Gợi ý: Tuổi của một người được tính bằng = năm hiện tại – năm sinh.

Để lấy về năm hiện tại trong máy, ta thực hiện đoạn lệnh sau:

```
from datetime import datetime
year= datetime.now().year
```

Bài tập 4.3: Viết chương trình cho người dùng nhập vào giờ, phút, giây dạng (hh:mm:ss) và một số k từ bàn phím sau đó chương trình cộng thêm k giây và hiển thị ra kết quả dưới dạng (hh:mm:ss) ra màn hình.

Bài tập 4.4: Xây dựng chương trình nhập vào 2 số nguyên khác 0 từ bàn phím; hiển thị ra màn hình ước số chung lớn nhất của hai số đó.

Bài tập 4.5: Xây dựng chương trình để giải bài toán dân gian sau:

Trăm trâu, trăm cỏ
Trâu đứng ăn năm
Trâu nằm ăn ba
Ba trâu già ăn một
Hỏi số trâu mỗi loại?

Bài tập 4.6: Xây dựng chương trình

Nhập một số n nguyên dương từ bàn phím.
Cho biết n có bao nhiêu chữ số?
Tính tổng các chữ số của nó.

Bài tập 4.7: Viết chương trình tính tổng của dãy số sau và hiển thị kết quả ra màn hình:

S(n) = 1*2 + 2*3 + 3*4 +…+i*(i+1) +…+2003*2004

Bài tập 4.8: Viết chương trình tính tổng của dãy số sau và hiển thị kết quả ra màn hình:

$S(x, n) = x + x^3 + x^5 + \ldots + x^{(2n+1)}$, với x, n được nhập từ bàn phím.

Hướng dẫn:

- Dùng vòng lặp for, do while để duyệt biến i từ 0 tới n.
- Tính tổng: s = s + x^(2*i + 1).
- Dùng hàm pow() trong thư viện math để tính x^(2*i + 1) hoặc toán tử **.

Bài 5

CẤU TRÚC DỮ LIỆU

Trong các bài học trước chúng ta đã tìm hiểu về kiểu dữ liệu số (gồm có số thực, số nguyên) và chuỗi (xâu). Các kiểu dữ liệu này chỉ mang một giá trị nhất định và ta có thể gọi chúng là kiểu dữ liệu nguyên thủy, ví dụ tên của cô ấy là "Hoa", đây là một chuỗi, tuổi bằng 21, đây là một số nguyên,...

Bây giờ giả sử chúng ta muốn lưu trữ thông tin về một sinh viên gồm: Họ tên, tuổi, địa chỉ, điểm toán, điểm lý, điểm hóa, hay thông tin về một mặt hàng bao gồm: Tên hành, mã hàng, số lượng, đơn giá thì chúng ta dùng kiểu dữ liệu gì để biểu diễn chúng?

Python cung cấp cho chúng ta các kiểu dữ liệu có cấu trúc như: Danh sách (list) và từ điển (dictionary) để biểu diễn dữ liệu gồm nhiều phần tử gọi là tập hợp (collection), mỗi phần tử trong nó là một kiểu dữ liệu nguyên thủy hoặc cũng có thể là một kiểu dữ liệu có cấu trúc.

Sau bài học này, người học có thể:

- *Sử dụng được hai cấu trúc danh sách (list), và từ điển (dictionary): tạo, thêm, sửa, xóa.*
- *Tạo ứng dụng tính giá trị đa thức (bậc n) tại một giá trị bất kì.*
- *Tạo ứng dụng quản lí một cửa hàng thời trang.*

5.1. ĐẶT VẤN ĐỀ

Viết chương trình quản lý chuỗi cửa hàng của thương hiệu thời trang Yody_Fashion gồm các công việc chính sau (biết rằng Yody_Fashion gồm10 cửa hàng trên cả nước):

- Nhập doanh số bán hàng từ mỗi cửa hàng.
- Hiển thị cửa hàng có doanh số bán hàng lớn nhất.
- Tìm kiếm cửa hàng có doanh số bán hàng lớn nhất.

Để viết chương trình thực hiện các công việc trên, ta viết chương trình gồm:

- Sử dụng 10 biến để lưu doanh số bán hàng của 10 cửa hàng.
- Sử dụng các phép toán so sánh để tìm ra cửa hàng có doanh số bán hàng cao nhất, thấp nhất và sắp xếp giá trị các biến.
- Sử dụng lệnh print() để hiển thị kết quả.

Khi đó đoạn chương trình thực hiện nhập doanh số bán hàng tại các cửa hàng như sau:

Chương trình 5.1: *Nhập doanh số của 10 cửa hàng*

```
doanh_so_1 = int(input('Nhập doanh số bán hàng của cửa hàng 1: '))
doanh_so_2 = int(input('Nhập doanh số bán hàng của cửa hàng 2: '))
doanh_so_3 = int(input('Nhập doanh số bán hàng của cửa hàng 3: '))
doanh_so_4 = int(input('Nhập doanh số bán hàng của cửa hàng 4: '))
doanh_so_5 = int(input('Nhập doanh số bán hàng của cửa hàng 5: '))
doanh_so_6 = int(input('Nhập doanh số bán hàng của cửa hàng 6: '))
doanh_so_7 = int(input('Nhập doanh số bán hàng của cửa hàng 7: '))
doanh_so_8 = int(input('Nhập doanh số bán hàng của cửa hàng 8: '))
doanh_so_9 = int(input('Nhập doanh số bán hàng của cửa hàng 9: '))
doanh_so_10 = int(input('Nhập doanh số bán hàng của cửa hàng 10: '))
```

Và đoạn chương trình tìm kiếm cửa hàng có doanh số bán hàng lớn nhất

Chương trình 5.2: *Tìm cửa hàng có doanh số bán hàng lớn nhất*

```
doanh_so_max = doanh_so_1
if (doanh_so_max < doanh_so_2):
    doanh_so_max = doanh_so_2;
if (doanh_so_max < doanh_so_3):
    doanh_so_max = doanh_so_3;
if (doanh_so_max < doanh_so_4):
    doanh_so_max = doanh_so_4;
if (doanh_so_max < doanh_so_5):
    doanh_so_max = doanh_so_5;
```

```
if (doanh_so_max < doanh_so_6):
    doanh_so_max = doanh_so_6;
if (doanh_so_max < doanh_so_7):
    doanh_so_max = doanh_so_7;
if (doanh_so_max < doanh_so_8):
    doanh_so_max = doanh_so_8;
if (doanh_so_max < doanh_so_9):
    doanh_so_max = doanh_so_9;
if (doanh_so_max < doanh_so_10):
    doanh_so_max = doanh_so_10;
print('Doanh số bán hàng cao nhất =', doanh_so_max)
print('Các cửa hàng có doanh số bán hàng cao nhất là: ')
if (doanh_so_max == doanh_so_1):
    print('Cửa hàng 1')
if (doanh_so_max == doanh_so_2):
    print('Cửa hàng 2')
if (doanh_so_max == doanh_so_3):
    print('Cửa hàng 3')
if (doanh_so_max == doanh_so_4):
    print('Cửa hàng 4')
if (doanh_so_max == doanh_so_5):
    print('Cửa hàng 5')
if (doanh_so_max == doanh_so_6):
    print('Cửa hàng 6')
if (doanh_so_max == doanh_so_7):
    print('Cửa hàng 7')
if (doanh_so_max == doanh_so_8):
    print('Cửa hàng 8')
if (doanh_so_max == doanh_so_9):
    print('Cửa hàng 9')
if (doanh_so_max == doanh_so_10):
    print('Cửa hàng 10')
```

Chương trình này sẽ chạy và cho kết quả đúng, nhưng bạn thử hình dung giả sử thương hiệu thời trang Yody_fashion không phải chỉ có 10 cửa hàng, mà chuỗi cửa hàng của họ lên đến hàng ngàn cửa hàng thì chương trình của chúng ta sẽ trở nên phức tạp hơn, cần hàng ngàn biến để lưu trữ, cần hàng ngàn câu lệnh để nhập dữ liệu, cần hàng nghìn câu lệnh if để tìm ra doanh số bán hàng lớn nhất …. Đó không phải là cách mà chúng ta muốn thực hiện đối với dạng dữ liệu này.

5.2. LIST

Danh sách (List) là một thùng chứa một hoặc nhiều phần tử, các phần tử trong một danh sách có thể có các giá trị thuộc các kiểu dữ liệu khác nhau, danh sách gồm các đặc điểm sau:

- Các phần tử trong danh sách được đặt trong cặp ngoặc vuông [].
- Các phần tử trong danh sách phân cách nhau ra bởi dấu phẩy dưới ",".
- Danh sách có khả năng chứa mọi giá trị, đối tượng trong Python.
- Danh sách có thể chứa một danh sách khác như một phần tử của nó.

Ví dụ 5.1: Ví dụ về danh sách

Vidu5_1.py

```
[1, 2, 3, 4, 5]          # Một danh sách chứa 5 số nguyên
['a', 'b', 'c', 'd']     # Một danh sách chứa 4 chuỗi
# Một danh sách chứa 2 danh sách khác là [1, 2] và [3, 4]
[[1, 2], [3, 4]]

# Danh sách chứa số nguyên, số thực chuỗi, và  danh sách
[1, 4.0,'one', [2, 'two']]
```

5.2.1. Khởi tạo một List

Để tạo một danh sách trong Python chúng ta có thể sử dụng một trong các cách sau:

Cách 1: Tạo danh sách sử dụng lệnh gán:

Cú pháp: *<tên biến>* = [*<giá_trị_1>, <giá_trị_2>, …, <giá_trị_n>*]

Trong đó:

- *<tên biến>:* Là tên của một biến do người dùng đặt theo quy tắc đặt tên.
- *<giá_trị_1>, <giá_trị_2>, ..., <giá_trị_n>*: Là dãy các phần tử khởi đầu được lưu trữ trong danh sách và được quản lý bởi <tên biến>. Nếu ta không đưa giá trị nào vào danh sách thì ta sẽ khởi tạo một danh sách rỗng.

Ví dụ 5.2: Tạo một danh sách để lưu thông tin về một sinh viên như sau:

Vidu5_2.py

```
sv = ["Lê Thị Hoa",21, "41 Hàng Trống Hà Nội", 8.0, 6.0,
7.0]
print("sv = ",sv) #Hiển thị danh sách sv ra màn hình
print(type(sv)) #Hiển thị kiểu dữ liệu sv ra màn hình
```

Kết quả chạy chương trình ***Vidu5_2.py***

```
 sv = ['Lê Thị Hoa', 21, '41 Hàng Trống Hà Nội', 8.0, 6.0, 7.0]
<class 'list'>
```

Ví dụ 5.3: Cách 1: Khai báo và khởi tạo giá rị cho danh sách:

Vidu5_3.py

```
lst1 = []      #Khởi tạo danh sách rỗng
print("lst1 = ",lst1)
#Khởi tạo danh sách với 10 phần tử nguyên
lst2=[0,1, 2, 3, 4, 5, 6, 7, 8, 9]
print("lst2 = ",lst2)
#Khởi tạo danh sách với các phần tử khác kiểu
lst3 = [1,2,3, [1,2,5],"Python"]
print("lst3 = ",lst3)
#Tạo danh sách dùng cấu trúc for
lst4 = [item for item in range(3)]  #Lst4 = [0, 1, 2]
print(lst4)
```

```
#Khởi tạo danh sách gồm 3 danh sách con
lst5 = [[n, n * 1, n * 2] for n in range(1, 4)]
print("lst5 = ",lst5)
'''
Tạo danh sách gồm các phần tử từ 0 đến 9 và lớn hơn 2 và nhỏ hơn 6 và khác 5
'''
lst6 = [x for x in range(0,10) if x > 2 and x < 6 and x !=5]
print("lst6 = ",lst6)
```

Kết quả chạy chương trình ***Vidu5_3.py***

```
lst1 =  []
lst2 =  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
lst3 =  [1, 2, 3, [1, 2, 5], 'Python']
lst4 =  [0, 1, 2]
lst5 =  [[1, 1, 2], [2, 2, 4], [3, 3, 6]]
lst6 =  [3, 4]
```

Cách 2: Sử dụng hàm list() để tạo danh sách:

Cú pháp: **list(col)**

Trong đó: col có thể là một xâu ký tự, một list, tuple hoặc set,…; nếu ta không truyền vào đối số cho list() thì chương trình Python sẽ tạo ra một danh sách rỗng.

Ví dụ 5.4:

Vidu5_4.py

```
lst = list()
print('lst = ', lst)     #Hiển thị danh sách sv ra màn hình
print(type(lst))         #Hiển thị kiểu dữ liệu sv ra màn hình
```

Kết quả chạy chương trình ***Vidu5_4.py***

```
lst = []
<class 'list'>
```

5.2.2. Truy cập đến các phần tử trong danh sách

Để truy cập đến các phần tử trong danh sách, chúng ta sử dụng tên biến kết hợp với chỉ số của phần tử đó trong danh sách theo cú pháp:

<tên biến>[<chỉ số>]

Để lấy về một danh sách con chứa các phần tử liên tục trong danh sách, ta thực hiện theo cú pháp:

<tên biến>[<start: end : step>]

Trong đó:

- *<tên biến>:* Là tên của danh sách
- *<chỉ số>:* Nhận các giá trị từ –n đến n–1, với n là số phần tử trong danh sách.
- *<start>:* Là một số nguyên chỉ vị trí của phần tử bắt đầu lấy trong danh sách, nếu không có giá trị này thì Python sẽ lấy phần tử có vị trí bằng 0.
- *<end>:* Là một số nguyên chỉ vị trí của phần tử cuối cùng lấy trong danh sách, nếu không có giá trị này thì Python sẽ lấy giá trị là số phần tử của danh sách –1.
- *[<step>]:* Là số bước nhẩy sẽ lấy, nếu bước nhẩy bằng một thì ta không cần chỉ định giá trị này.

Ví dụ 5.5: Chương trình minh hoạ truy cập các phần tử trong danh sách

Vidu5_5.py

```
L=[1,2,3,4,5,6]     #Tạo một List có 6 phần tử
print(L[0])    #Hiển thị phần tử đầu tiên trong danh sách
print(L[-5])   #Hiển thị phần tử thứ hai trong danh sách
print(L[5])    #Hiển thị phần tử thứ năm trong danh sách
print(L[-2])   #Hiển thị phần tử thứ hai tính từ cuối danh sách
#Hiển thị danh sách con không chứa phần tử đầu, cuối của L
print(L[1: -1])
print(L[1: 0])      #Hiển thị danh sách rỗng
print(L[1: 3])   #Hiển thị danh sách con [2,3,4] của L
```

Kết quả chạy chương trình ***Vidu5_5.py***

```
1
2
6
5
[2, 3, 4, 5]
[]
[2, 3]
```

5.2.3. Thêm phần tử vào danh sách

Để thêm một phần tử vào danh sách, chúng ta sử dụng một trong các phương thức sau đây:

- append(x): Thêm phần tử có giá trị x vào cuối danh sách
- insert(i, x): Thêm phần tử có giá trị x tại vị trí có chỉ số i trong danh sách. Nếu i = 0 hoặc i =< –n thì x sẽ được thêm vào đầu danh sách. Nếu i > n – 1 thì x sẽ được thêm vào cuối danh sách. Với n là số phần tử của danh sách.

Ví dụ 5.6: Chương trình minh hoạ thêm một phần tử vào danh sách

Vidu5_6.py

```
L = [1]         #Khởi tạo danh sách L có một phần tử
L.append(2)     #Thêm phần tử 2 vào cuối danh sách
print(L)        # [1,2]
L.insert(0,3)  #Chèn thêm phần tử 3 vào đầu danh sách
print(L)        # [3,1,2]
L.insert(5, 4)
print(L)        #[3,1,2,4]
```

Kết quả chạy chương trình ***Vidu5_6.py***

```
[1, 2]
[3, 1, 2]
[3, 1, 2, 4]
```

Ví dụ 5.7: Đoạn chương trình để nhập doanh số bán hàng tại các cửa hàng của chuỗi cửa hàng của chương trình 5.1 được viết lại như sau:

Vidu5_7.py

```
lst_ch =[]
for i in range(1, 11):
    doanh_so = float(input("Nhập doanh số của cửa hàng "+str(i)+" ="));
    lst_ch.append(doanh_so)
```

Kết quả chạy chương trình ***Vidu5_7.py***

```
Nhập doanh số của cửa hàng 1 =12.6
Nhập doanh số của cửa hàng 2 =63.5
Nhập doanh số của cửa hàng 3 =44.6
Nhập doanh số của cửa hàng 4 =22.7
Nhập doanh số của cửa hàng 5 =87
Nhập doanh số của cửa hàng 6 =22.6
Nhập doanh số của cửa hàng 7 =3.5
Nhập doanh số của cửa hàng 8 =44.2
Nhập doanh số của cửa hàng 9 =20.7
Nhập doanh số của cửa hàng 10 =81
```

5.2.4. Xoá phần tử trong danh sách

Để thực hiện xoá một phần tử trong danh sách, chúng ta có thể xóa theo chỉ số (vị trí) của phần tử trong danh sách hoặc xóa theo giá trị của phần tử.

Xóa theo chỉ số của phần tử trong danh sách:

Cú pháp: *< tên biến>.__delitem__(<index>)*

Trong đó:

- *<tên biến>:* Là tên biến danh sách do người dùng đặt mà ta muốn xoá phần tử trong nó
- *<index>:* Là chỉ số, vị trí của phần tử trong danh sách muốn xoá, phần tử đầu tiên trong danh sách có chỉ số bằng 0 hoặc –n. Nếu index $\notin$[–n, n–1] thì câu lệnh không làm gì và không báo lỗi.

Ví dụ 5.8: Minh họa xóa phần tử có chỉ số bằng 1 trong danh sách lsn = [1, 2, 3, 4]

Vidu5_8.py

```
lsn = [1,2,3,4]
print("Danh sách lsn chưa xóa = ",lsn)
lsn.__delitem__(1)
print("Danh sach lsn đã xóa = ",lsn)
```

Kết quả chạy chương trình ***Vidu5_8.py***

```
Danh sách lsn chưa xóa =  [1, 2, 3, 4]
Danh sách lsn đã xóa =  [1, 3, 4]
```

Xóa theo giá trị của phần tử trong danh sách: Ta sử dụng phương thức remove() có của lớp danh sách sau:

<*tên biến*>.remove(x): Xoá phần tử có giá trị bằng x ra khỏi danh sách <*tên biến*>. Nếu trong <*tên biến*> có nhiều hơn một phần tử có giá trị bằng x thì phần tử x đầu tiên trong danh sách sẽ bị xoá khỏi danh sách; nếu trong danh sách không có phần tử x thì trình dịch Python sẽ phát sinh một ngoại lệ với thông báo lỗi như sau: "ValueError: list.remove(x): x not in list" và chương trình sẽ bị dừng lại.

Ví dụ 5.9: Minh họa xóa phần tử có giá trị bằng x trong danh sách lsn = [1, 2, 3, 4]

Vidu5_9.py

```
lsn = [1,2,3,4]
print("Danh sach L chưa xóa = ",lsn)
lsn.remove(4)                  #Xóa phần tử có giá trị bằng 4
print("Danh sach L đã xóa = ",lsn)
lsn.remove(7)                  #Xóa phần tử có giá trị bằng 7
print("Danh sach L đã xóa = ",lsn)
```

Kết quả chạy chương trình ***Vidu5_9.py***

```
Danh sach L chưa xóa =  [1, 2, 3, 4]
  File "E:/DOCUMENT/2018-ML/untitled/vd.py", line 29, in <module>
Danh sach L đã xóa =  [1, 2, 3]
    lsn.remove(7)                    #Xóa phần tử có giá trị bằng 7
ValueError: list.remove(x): x not in list
```

5.2.5. Duyệt qua các phần tử trong danh sách

Để duyệt qua các phần tử trong danh sách chúng ta sử dụng cấu trúc lặp for hoặc while đã được trình bày trong bài học trước.

Ví dụ 5.10: Duyệt qua tất cả các phần tử trong danh sách sinh viên *sv* và hiển thị kết quả ra màn hình như sau:

Vidu5_10.py

```
sv = ["Lê Thị Hoa",21, "Hà Nội"]
for i in sv:
      print(i)        #Hiển thị ra màn hình phần tử i của sv
```

Kết quả chạy chương trình ***Vidu5_10.py***

```
Lê Thị Hoa
21
Hà Nội
```

Ví dụ 5.11: Duyệt qua tất cả các phần tử trong danh sách sinh viên *sv* cùng chỉ số và hiển thị kết quả ra màn hình như sau:

Vidu5_11.py

```
sv = ["Lê Thị Hoa",21, "Hà Nội"]
for i, xi in enumerate (sv):
      #Hiển thị ra màn hình chỉ số i và phần tử xi
       print("i = ",i, ", xi = ",xi)
```

Kết quả chạy chương trình ***Vidu5_11.py***

`i= 0,  xi = Lê Thị Hoa`
`i= 1,  xi = 21`
`i= 2,  xi = Hà Nội`

5.2.6. Một số phương thức khác

- len*(<list>):* Trả về số phần tử trong danh sách
- sum*(<list>):* Trả về tổng giá trị các phần tử trong danh sách
- sorted(list): Trả về một danh sách chứa các phần tử trong danh sách đã được sắp xếp theo thứ tự tăng dần về giá trị
- sorted(list, reverse=True): Trả về một danh sách chứa các phần tử trong danh sách đã được sắp xếp theo thứ tự giảm dần về giá trị.
- max(list): Trả về phần tử có giá trị lớn nhất trong danh sách
- min(list): Trả về phần tử có giá trị nhỏ nhất trong danh sách

Trong đó list là tên biến danh sách.

5.3. TỪ ĐIỂN (DICTIONARY)

Với cấu trúc dữ liệu kiểu danh sách, chúng ta có thể lưu trữ mọi đối tượng với số lượng không hạn chế trong đó và việc truy xuất đến từng phần tử thông qua chỉ số của phần tử. Tuy nhiên với dữ liệu có cấu trúc ví dụ như thông tin của một sinh viên gồm có nhiều trường thông tin: Mã sinh viên, Họ tên, tuổi,… ta muốn lưu trữ dữ liệu này một cách tường minh thì danh sách không làm được điều đó.

Python cung cấp cho chúng ta thêm kiểu dữ liệu từ điển (Dictionary) để lưu trữ tập dữ liệu có cấu trúc, mỗi phần tử trong từ điển lưu trữ bởi một cặp *<key>: <value> (<khóa>: <giá trị>)* với ràng buộc khóa là duy nhất trong từ điển, một từ điển bao gồm các đặc tính sau:

- Được giới hạn bởi cặp ngoặc nhọn {}, tất cả những gì nằm trong đó là những phần tử của từ điển.

– Các phần tử của từ điển được phân cách nhau ra bởi dấu phẩy dưới ",".
– Mỗi phần tử của từ điển là một cặp *<khóa>: <giá trị>*
– Ngăn cách giữa thành phần *<khóa>* và thành phần *<giá trị>* bởi dấu hai chấm ":"
– Các *<khóa>* trong một từ điển buộc phải là một đối tượng băm (duy nhất).

5.3.1. Tạo từ điển

Để tạo một từ điển ta thực hiện một trong hai cách sau:

Cách 1: Sử dụng câu lệnh gán để tạo từ điển

Cú pháp:

<Tên biến> = {<khóa_1: giá trị_1>, <khóa_2: giá trị_2>,..., <khóa_n: giá trị_n>}

Trong đó:

- *<tên biến>:* Là tên của một biến do người dùng đặt theo quy tắc đặt tên
- *< khóa_1>, < khóa_2>, ... ,< khóa_n>:* Là danh mục các khóa được dùng như chỉ số của các phần tử trong từ điển và các khóa này là không trùng nhau.
- *<giá_trị_1>, <giá_trị_2>, ..., <giá_trị_n>:* Là dãy các giá trị được lưu trữ trong từ điển đi theo khóa và được quản lý bởi <tên biến>. Nếu ta không đưa cặp khóa, giá trị nào vào danh sách thì Python sẽ khởi tạo một từ điển rỗng.

Ví dụ 5.12: Tạo một từ điển để lưu thông tin về một môn học gồm các trường thông tin mã môn học, tên môn học, số tín chỉ:

Vidu5_12.py

```
#Khởi tạo từ điển lưu thông tin về một môn học
monhoc= {"Ma": "201191", "Ten": "Lập trình căn bản", "soTC": 3}
#Hiển thị từ điển môn học ra màn hình
print("Thông tin môn học: ",monhoc)
print(type(monhoc))
```

Kết quả chạy chương trình ***Vidu5_12.py***

```
Thông tin môn học:  {'Ma': '201191', 'soTC': 3, 'Ten': 'Lập trình căn bản'}
<class 'dict'>
```

Ví dụ 5.13: Tạo một từ điển để lưu thông tin về các khoa trong trường đại học:

Vidu5_13.py

```
#Khởi tạo từ điển tên khoa
khoa= {1: "Công nghệ thông tin", 2: "Ô tô", 3: "Kinh tế"}
print("khoa = ",khoa)    #Hiển thị từ điển khoa ra màn hình
print(type(khoa))
```

Kết quả chạy chương trình ***Vidu5_13.py***

```
khoa =  {1: 'Công nghệ thông tin', 2: 'Ô tô', 3: 'Kinh tế'}
<class 'dict'>
```

Cách 2: Sử dụng hàm dict() để tạo danh sách:

Cú pháp: *<Tên biến> = dict()*

Ví dụ 5.14: Sử dụng hàm khởi tạo để tạo từ điển:

Vidu5_14.py

```
#Khởi tạo từ điển tên khoa sử dụng hàm khởi tạo
khoa= dict()
print("khoa = ",khoa)
print(type(khoa))
```

Kết quả chạy chương trình ***Vidu5_14.py***

```
khoa =  { }
<class 'dict'>
```

Ví dụ 5.15: Sử dụng từ điển khai báo thông tin về một người

Vidu5_15.py

```
#Khởi tạo từ điển tên person
person = {
    'Ten': 'Lê Thị Mận',
    'dia chi': "Hà Nội",
    'tuoi': 22,
    'gioi tinh': 'Nữ',
    'tinh trang hon nhan': 'Độc thân'
    }
print("person = ",person)  #Hiển thị biến person ra màn hình
print(type(person))  #Hiển thị kiểu của biến person ra màn hình
```

Kết quả chạy chương trình ***Vidu5_15.py***

```
person =  {'Ten': 'Lê Thị Mận', 'dia chi': 'Hà Nội', 'gioi tinh': 'Nữ',
'tinh trang hon nhan': 'Độc thân', 'tuoi': 22}
<class 'dict'>
```

5.3.2. Truy cập đến các phần tử trong từ điển

Lấy giá trị gắn với khóa của phần tử trong **từ điển**

Cú pháp: *<Tên biến>[<khóa>]*

#Truy xuất tự nhiên

hoặc *<Tên biến>.get(<khóa>)*

#Sử dụng phương thức get()

Ví dụ 5.16: Lấy giá trị có khóa bằng name với biến từ điển person trên

Vidu5_16.py

```
person = {
    'Ten': 'Lê Thị Mận',
    'dia chi': "Hà Nội",
    'tuoi': 22,
    'gioi tinh': 'Nữ',
```

```
    'tinh trang hon nhan': 'Độc thân'
    }
#Hiển thị tên của người đó ra màn hình
print("Tên=",person['Ten'])
#Hiển thị tuổi của người đó ra màn hình
print("Tuổi =",person.get('tuoi'))
```

Kết quả chạy chương trình ***Vidu5_16.py***

```
Tên= Lê Thị Mận
Tuổi = 22
```

Lấy tất cả các khóa trong **từ điển.**

Cú pháp: *<Tên biến>.keys() #Truy xuất đến thuộc tính keys*

Ví dụ 5.17: Lấy tất cả các khóa có trong biến từ điển person trên

Vidu5_17.py

```
person = {
    'Ten': 'Lê Thị Mận',
    'dia chi': "Hà Nội",
    'tuoi': 22,
    'gioi tinh': 'Nữ',
    'tinh trang hon nhan': 'Độc thân'
    }
print("person = ",person)
#Hiển thị tất cả khóa trong person ra màn hình
print("Các khóa =",person.keys())
```

Kết quả chạy chương trình ***Vidu5_17.py***

```
person =  {'tinh trang hon nhan': 'Độc thân', 'dia chi': 'Hà Nội',
'Ten': 'Lê Thị Mận', 'tuoi': 22, 'gioi tinh': 'Nữ'}
Các khóa = dict_keys(['tinh trang hon nhan', 'dia chi', 'Ten', 'tuoi',
'gioi tinh'])
```

Lấy tất cả các giá trị trong **từ điển.**

Cú pháp: *<Tên biến>.values()* *#Truy xuất đến thuộc tính values*

Ví dụ 5.18: Lấy tất cả các giá trị có trong biến từ điển person trên

Vidu5_18.py

```
person = {
    'Ten': 'Lê Thị Mận',
    'dia chi': "Hà Nội",
    'tuoi': 22,
    'gioi tinh': 'Nữ',
    'tinh trang hon nhan': 'Độc thân'
    }
print("person = ",person)
#Hiển thị tất cả giá trị trong person ra màn hình
print("Giá trị =",person.values())
```

Kết quả chạy chương trình ***Vidu5_18.py***

```
person =  {'tinh trang hon nhan': 'Độc thân', 'gioi tinh': 'Nữ', 'Ten':
'Lê Thị Mận', 'tuoi': 22, 'dia chi': 'Hà Nội'}
Giá trị = dict_values(['Độc thân', 'Nữ', 'Lê Thị Mận', 22, 'Hà Nội'])
```

Duyệt qua tất cả các phần tử trong từ điển: Để duyệt qua tất cả các phần tử trong từ điển, chúng ta có thể sử dụng cấu trúc *for* hoặc *while* như sau:

Ví dụ 5.19: Duyệt qua tất cả các phần tử có trong biến từ điển person

Vidu5_19.py

```
person = {
    'Ten': 'Lê Thị Mận',
    'dia chi': "Hà Nội",
    'tuoi': 22,
    'gioi tinh': 'Nữ',
    'tinh trang hon nhan': 'Độc thân'
    }
```

```
# Duyệt qua tất cả các phần tử trong biến person
for i in person:
    print( i, ": ", person[i])
```

Kết quả chạy chương trình ***Vidu5_19.py***

```
dia chi :  Hà Nội
Ten :  Lê Thị Mận
gioi tinh :  Nữ
tuoi :  22
tinh trang hon nhan :  Độc thân
```

5.3.3. Thêm/Sửa một phần tử trong từ điển

Cú pháp: *<Tên biến>.[<khóa>]= <giá trị>*

Trong đó:

- *<tên biến>:* Là tên một biến từ điển đã được khai báo trước đó
- *< khóa>:* Chỉ định khóa được lưu trữ trong từ điển, nếu khóa này đã có thì giá trị cũ sẽ mất đi và giá trị mới sẽ thay thế; nếu khóa chưa có trong từ điển thì một phần tử mới *<khoá>: <giá trị>* sẽ được thêm vào từ điển.
- *<giá trị>:* Là giá trị được lưu trữ trong từ điển đi theo khóa và được quản lý bởi *<tên biến>*.

Ví dụ 5.20: Sửa lại địa chỉ của khóa address và thêm vào phần tử hệ số lương (hsl: 3.99) trong biến từ điển person

Vidu5_20.py

```
person = {
    'Ten': 'Lê Thị Mận',
    'dia chi': "Hà Nội",
    'tuoi': 22,
    'gioi tinh': 'Nữ',
    'tinh trang hon nhan': 'Độc thân'
```

```
    }
for i in person:#Duyệt qua tất cả các phần tử trong biến person
    print( i, ": ", person[i])
person = {
    'Ten': 'Lê Thị Mận',
    'dia chi': "Hà Nội",
    'tuoi': 22,
    }
print('Thông tin ban đầu')
print("person = ",person)
person['tuoi']=20  # Sửa lại tuổi của người đó thành 20
print('Sau khi cập nhật tuổi')
print("person = ",person)
person['gioi tinh']='Nữ' # Thêm trường giới tính vào từ điển
print('Sau khi thêm giới tính')
print("person = ",person)
```

Kết quả chạy chương trình ***Vidu5_20.py***

```
tuoi :  22
gioi tinh :  Nữ
Ten :  Lê Thị Mận
dia chi :  Hà Nội
tinh trang hon nhan :  Độc thân
Thông tin ban đầu
person =  {'tuoi': 22, 'Ten': 'Lê Thị Mận', 'dia chi': 'Hà Nội'}
Sau khi cập nhật tuổi
person =  {'tuoi': 20, 'Ten': 'Lê Thị Mận', 'dia chi': 'Hà Nội'}
Sau khi thêm giới tính
person =  {'tuoi': 20, 'gioi tinh': 'Nữ', 'Ten': 'Lê Thị Mận', 'dia
chi': 'Hà Nội'}
```

5.3.4. Xoá phần tử trong từ điển

Cú pháp: *<Tên biến>.__delitem__(<khóa>);*

 # Xóa phần tử có khóa ra khỏi từ điển

< Tên biến >.clear();

#Xoá tât cả các phần tử có trong từ điển

Ví dụ 5.21: Xóa phần tử có khóa bằng address trong biến từ điển person

Vidu5_21.py

```
person = {
    'Ten': 'Lê Thị Mận',
    'dia chi': "Hà Nội",
    'tuoi': 22,
    }
print('Thông tin ban đầu')
print("person = ",person)
person.__delitem__('tuoi') # Xoá phần tử có khoá là 'tuoi'
print('Sau khi xoá bỏ tuổi')
print("person = ",person)
person.clear()        #Xoá tất cả các phần tử của từ điển
print('Sau khi xoá hết các phần tử')
print("person = ",person)
```

Kết quả chạy chương trình ***Vidu5_21.py***

```
Thông tin ban đầu
person =  {'Ten': 'Lê Thị Mận', 'dia chi': 'Hà Nội', 'tuoi': 22}
Sau khi xoá bỏ tuổi
person =  {'Ten': 'Lê Thị Mận', 'dia chi': 'Hà Nội'}
Sau khi xoá hết các phần tử
person =  {}
```

5.4. TẠO ỨNG DỤNG

Ví dụ 5.22: Xây dựng chương trình cho người dùng nhập vào thông tin của một đa thức P_n một biến bậc n, sau đó chương trình thực hiện tính giá trị của đa thức và hiển thị kết quả ra màn hình.

$$P_n = a_0 + a_1x + a_2x^2 + ... + a_nx^n$$

Phân tích:

Các hệ số của đa thức là một dãy số, do đó chúng ta có thể sử dụng một danh sách để lưu trữ các hệ số của của đa thức khi người dùng nhập vào.

a_0	a_1	a_2	a_3	a_4	…..	a_{n-1}	a_n

Ví dụ: $P_7 = -1 + x^2 + 3x^7$

Khi đó ta có danh sách hệ số của đa thức P_7 như sau:

–1	0	1	0	0	0	0	3

Ta thấy rằng với đa thức bậc 7, chúng ta cần danh sách có 8 phần tử để lưu các hệ số của đa thức, từ đó, nếu đa thức bậc n thì ta cần sử dụng một danh sách gồm có n+1 phần tử để lưu trữ các hệ số của đa thức bao gồm cả các phần tử có hệ số bằng 0.

Giả sử để lưu đa thức: $P_{100} = 25 + x^{100}$, chỉ có 2 phần tử, nhưng chúng ta vẫn cần phải dùng đến danh sách gồm 101 phần tử để lưu các hệ số của đa thức này, điều này gây lãng phí bộ nhớ và chương trình chạy không tối ưu.

Nếu chúng ta biểu diễn đa thức P_n như một tập gồm nhiều cặp số mũ và hệ số được coi như một phần tử thì chương trình sẽ tối ưu hơn với các đa thức có nhiều hệ số bằng không. Từ đó với đa thức $P_{100} = 25 + x^{100}$ ta chỉ cần sử dụng một từ điển gồm hai phần tử như: {0: 25, 100: 1} để biểu diễn và nếu ta so sánh với việc sử dụng danh sách để lưu trữ đa thức thì sử dụng từ điển hiệu quả hơn trong trường hợp này.

Vidu5_22.py

```
'''
    Chương trình tính giá trị của đa thức
'''
#Nhập đa thức từ bàn phím
pn = dict();
sm=0
while (True):
      # Người dùng phải nhập mũ của phần tử thoả điều kiện >=-1
        while (True):
            sm = int(input('Nhập số mũ hoặc -1 để kết thúc: '))
            if (sm >= -1): break;
        if (sm ==-1):       #Nếu mũ = -1
            break       #Thì kết thúc việc nhập
        # Nhập hệ số ứng với số mũ
        hs = float(input('Nhập hệ số của x^'+ str(sm) + ' : '))
        if (hs!=0):         pn[sm]=hs
     # Phần tử của đa thức được lưu dạng mu: heso; nếu hệ số<>0
#Hiển thị đa thức ra màn hình
strpol=''
for pow in sorted(pn.keys()):
        if (pn[pow]>0):
                strpol = strpol +' + ' + str(pn[pow]) + ' * x^' +
str(pow)
        else: strpol = strpol + ' ' + str(pn[pow]) + ' * x^' + str(pow)
print('P = ',strpol.strip(' +').strip())

#Tính giá trị của đa thức tại điểm x
x = float(input("Nhập giá trị x = "))
val =sum(pn[pow] * x ** pow for pow in pn)
print('Giá trị của đa thức tại ', str(x), ' là: ', str(val))
```

Kết quả chạy chương trình ***Vidu5_22.py***

```
Nhập số mũ hoặc -1 để kết thúc: 1
Nhập hệ số của x^1 : 1
Nhập số mũ hoặc -1 để kết thúc: 2
Nhập hệ số của x^2 : 2
Nhập số mũ hoặc -1 để kết thúc: 3
Nhập hệ số của x^3 : 3
Nhập số mũ hoặc -1 để kết thúc: 4
Nhập hệ số của x^4 : 4
Nhập số mũ hoặc -1 để kết thúc: -1
P =  1.0 * x^1 + 2.0 * x^2 + 3.0 * x^3 + 4.0 * x^4
Nhập giá trị x = 2
Giá trị của đa thức tại  2.0  là:  98.0
```

Ví dụ 5.23: Một thương hiệu thời trang Yody_Fashion bao gồm một chuỗi 10 cửa hàng được đặt trên khắp cả nước. Hãy viết chương trình giúp chủ cửa hàng giải quyết các công việc sau:

1. Nhập doanh số bán hàng từ 10 cửa hàng gửi về;
2. Sắp xếp doanh số bán hàng từ bé đến lớn;
3. Tính doanh số trung bình trên 10 cửa hàng;
4. Hiển thị doanh số bán hàng lớn nhất.

Từ đó, chủ của chuỗi cửa hàng này lập kế hoạch kinh doanh và tặng thưởng cho cửa hàng có doanh số cao nhất.

Phân tích: Để viết chương trình giải quyết các công việc trên, bạn có thể:

- Sử dụng 10 biến để lưu trữ doanh số bán hàng của 10 cửa hàng;
- Sử dụng các phép toán so sánh để tìm ra cửa hàng có doanh số bán hàng cao nhất, thấp nhất;
- Sử dụng phương thức sort() để sắp xếp giá trị trong danh sách;
- Sử dụng lệnh print() để hiển thị kết quả.

Tuy nhiên bạn thử hình dung giả sử thương hiệu thời trang Yody–fashion không phải chỉ có 10 cửa hàng, mà chuỗi cửa hàng của họ có nhu cầu mở rộng

kinh doanh lên đến hàng trăm cửa hàng thì chương trình của chúng ta sẽ trở nên rất phức tạp, cần hàng trăm biến để lưu trữ, cần hàng trăm câu lệnh để nhập dữ liệu, cần hàng trăm câu lệnh if để tìm ra doanh số bán hàng lớn nhất,… Đó không phải là cách mà chúng ta muốn thực hiện đối với bài toán này.

Giải pháp của chúng ta là xây dựng chương trình giải quyết công việc cho n cửa hàng, thông qua việc sử dụng danh sách hoặc từ điển, nội dung chương trình như ***Ví dụ 5_23*** dưới đây.

Vidu5_23.py

```
'''
    Chương trình quản lý cửa hàng thời trang
'''
Lst_fashion = []                    #Biến lst_fashion lưu trữ doanh thu
của các cửa hàng
#Nhập thông tin doanh số của 10 cửa hàng
for i in range(1, 11):
        doanh_so = int(input('Nhập doanh số của cửa hàng '+ str(i)+':
'));
        Lst_fashion.append(doanh_so)
#Hiển thị thông tin vừa nhập
print("Lst_fashion = ",Lst_fashion)
#Hiển thị menu để thực hiện chức năng
while (True):
        print('Nhập 1: Hiển thị doanh số bán hàng cao nhất.')
        print('Nhập 2: Hiển thị doanh số trung bình của 10 cửa hàng.')
        print('Nhập 3: Hiển thị doanh số bán hàng được sắp xếp từ thấp
lên cao.')
        print('Nhập 0: Kết thúc chương trình.')
        chon = int(input('Bạn chọn công việc: '))
        if (chon == 1):
                # Tìm cửa hàng có doanh số bán hàng cao nhất
                doanh_so_max = 0;
                for doanh_so in Lst_fashion:
                        if (doanh_so > doanh_so_max):
                                doanh_so_max = doanh_so;
```

```
                # Hiển thị kết quả ra màn hình
                print('Doanh số bán hàng cao nhất = ', doanh_so_max)
        elif (chon == 2):
                # Tính tổng doanh số của 10 cửa hàng
                s = 0;
                for doanh_so in Lst_fashion:
                        s = s + doanh_so
                # Tính trung bình các doanh số của 10 cửa hàng
                doanh_so_tb = s / 10
                print('Doanh số trung bình = ', doanh_so_tb)
        elif (chon == 3):
        # Hiển thị doanh số bán hàng từ thấp đến cao
                print(' Doanh số bán hàng được sắp xếp từ cao xuống
thấp:')
                lst_high2low = sorted(Lst_fashion, reverse = True)
                print('\t'.join([str(doanh_so) for doanh_so in
lst_high2low]))
        else:
                break;              # Kết thúc chương trình
```

Kết quả chạy chương trình ***Vidu5_23.py***

```
Nhập doanh số của cửa hàng 1: 10
Nhập doanh số của cửa hàng 2: 50
Nhập doanh số của cửa hàng 3: 30
Nhập doanh số của cửa hàng 4: 20
Nhập doanh số của cửa hàng 5: 90
Nhập doanh số của cửa hàng 6: 70
Nhập doanh số của cửa hàng 7: 80
Nhập doanh số của cửa hàng 8: 120
Nhập doanh số của cửa hàng 9: 60
Nhập doanh số của cửa hàng 10: 96
Lst_fashion = [10, 50, 30, 20, 90, 70, 80, 120, 60, 96]
Nhập 1: Hiển thị doanh số bán hàng cao nhất.
```

```
Nhập 2: Hiển thị doanh số trung bình của 10 cửa hàng.
Nhập 3: Hiển thị doanh số bán hàng được sắp xếp từ thấp lên cao.
Nhập 0: Kết thúc chương trình.
Bạn chọn công việc: 1
Doanh số bán hàng cao nhất =  120
Nhập 1: Hiển thị doanh số bán hàng cao nhất.
Nhập 2: Hiển thị doanh số trung bình của 10 cửa hàng.
Nhập 3: Hiển thị doanh số bán hàng được sắp xếp từ thấp lên cao.
Nhập 0: Kết thúc chương trình.
Bạn chọn công việc: 2
Doanh số trung bình =  62.6
Nhập 1: Hiển thị doanh số bán hàng cao nhất.
Nhập 2: Hiển thị doanh số trung bình của 10 cửa hàng.
Nhập 3: Hiển thị doanh số bán hàng được sắp xếp từ thấp lên cao.
Nhập 0: Kết thúc chương trình.
Bạn chọn công việc: 3
 Doanh số bán hàng được sắp xếp từ cao xuống thấp:
120     96      90      80      70      60      50      30      20      10
Nhập 1: Hiển thị doanh số bán hàng cao nhất.
Nhập 2: Hiển thị doanh số trung bình của 10 cửa hàng.
Nhập 3: Hiển thị doanh số bán hàng được sắp xếp từ thấp lên cao.
Nhập 0: Kết thúc chương trình.
Bạn chọn công việc: 0
```


5.5. TỔNG KẾT BÀI HỌC

Trong bài học này chúng ta đã tìm hiểu hai cấu trúc dữ liệu nâng cao trong Python là danh sách và từ điển. Với danh sách (List), chúng có thể chứa nhiều kiểu dữ liệu khác nhau trong cùng một tập như chuỗi, số nguyên, số thực, số thập phân,v.v.. với số lượng không hạn chế. Đặc biệt, chúng ta có thể đưa ra các thao tác như: Thêm, sửa, xóa các phần tử trong danh sách một cách trực tiếp hoặc thông qua phương thức của lớp list rất thuận tiện. Loại thứ hai là kiểu dữ liệu từ điển (Dictionary), đây là một loại cấu trúc dữ liệu

được sử dụng để lưu trữ thông tin được kết nối theo nhiều cách khác nhau. Mỗi phần tử trong từ điển được định nghĩa thành hai phần là khóa và giá trị (Key và Value), trong đó.

Khóa: Là một phần tử tồn tại duy nhất trong từ điển, Python không cho phép trùng hai khoá bất kỳ và các khóa (key) phải có cùng kiểu dữ liệu trong cùng một biến từ điển.

Giá trị: Có thể là một danh sách hoặc giá trị có kiểu dữ liệu bất kỳ.

5.6. BÀI TẬP

A. Câu hỏi ôn tập

Câu 5.1: Cho biết phát biểu nào sau đây là đúng với kiểu dữ liệu danh sách

A. Các phần tử trong danh sách bắt buộc phải có cùng kiểu dữ liệu;
B. Các phần tử trong danh sách có thể có các kiểu dữ liệu khác nhau;
C. Các phần tử trong danh sách có thể là chính kiểu danh sách;
D. Số lượng các phần tử trong danh sách là không giới hạn.

Câu 5.2: Cho biết các phát biểu nào sau đây là đúng với kiểu dữ liệu từ điển (dict)

A. Các phần tử trong từ điển được phân biệt thông qua khóa;
B. Các phần tử trong từ điển được phân biệt thông qua giá trị;
C. Kiểu dữ liệu từ điển gồm nhiều phần tử mà mỗi phần tử là một cặp <khóa:giá trị>, trong đó giá trị phải có kiểu xâu;
D. Kiểu dữ liệu từ điển gồm nhiều phần tử mà mỗi phần tử là một cặp <khóa:giá trị>, trong đó khóa phải duy nhất.

Câu 5.3: Cho biết kết quả hiển thị ra màn hình sau khi máy chạy đoạn chương trình Python sau?

```
x = ['ab', 'cd']
for i in x:
    i.upper()
print(x)
```

A. [‘ab’, ‘cd’];

B. [‘AB’, ‘CD’];

C. [‘ab’, ‘CD’];

D. [‘AB’, ‘cd’].

Câu 5.4: Câu lệnh nào cho phép khởi tạo một danh sách

A. List1 = list();

B. List1 = [];

C. List1 = List([1, 2, 3]);

D. List1 = ["abc",123,[4,5]].

Câu 5.5: Giả sử L = [1, 3, 5, 7, 6, 2, 10], cho biết câu lệnh nào thực hiện hiển thị ra màn hình danh sách [3, 5, 7, 6, 2, 10]?

A. print(L[1:len(L)+1]);

B. print(L[:–1]);

C. print(L[1:]);

D. print(L[1:7:2].

Câu 5.6: Giả sử L=[1,3,5,7,6], cho biết phần tử L[1] trả về giá trị nào trong các giá trị sau

A. 1

B. 3

C. 5

D. 7.

Câu 5.7: Cho biết kết quả hiển thị ra màn hình sau khi máy chạy câu lệnh Python sau?

```
print( list('hello'))
```

A. [‘h’, ‘e’, ‘l’, ‘l’, ‘o’]

B. hello

C. h e l l o

D. ‘h’ ‘e’, ‘l’, ‘l’, ‘o’

Câu 5.8: Cho biết kết quả hiển thị ra màn hình sau khi máy chạy khối lệnh Python sau?

```
L = [1, 10, -5]
L.append(90)
print(max(L))
```

A. 1

B. 10

C. –5

D. 90

Câu 5.9: Cho biết kết quả hiển thị ra màn hình sau khi máy chạy khối lệnh Python sau?

```
L = [1,10,-5,90]
L.remove(-1)
print(min(L))
```

A. 1

B. 10

C. –5

D. 90

Câu 5.10: Cho biết kết quả hiển thị ra màn hình sau khi máy chạy khối lệnh Python sau?

```
L=[1,10,-5,90]
print(sum(L))
```

A. 1

B. 10

C. 90

D. 96

B. Lập trình

Bài tập 5.1: Xây dựng chương trình thực hiện các công việc sau:

- Nhập số dương n.
- Nhập một dãy số thực x1, x2,..., xn và lưu vào danh sách L.
- Tìm phần tử có giá trị lớn nhất trong danh sách L.
- Tính tổng các phần tử có trong danh sách L.
- Sắp xếp các phần tử trong danh sách tăng dần.
- Đếm trong dãy có bao nhiêu phần tử là số dương, bao nhiêu phần tử là số âm.
- Hiển thị các phần tử trong L ra màn hình.

  Gợi ý:

  - Sử dụng danh sách để lưu các số thực.
  - Sử dụng hàm min(), max(), sum() để tìm các phần tử nhỏ nhất, lớn nhất và tổng các phần tử trong danh sách.
  - Sử dụng phương thức sort() để sắp xếp các phần tử trong danh sách.
  - Sử dụng cấu trúc lặp for để duyệt qua các phần tử trong danh sách.
  - Sử dụng lệnh print() để hiển thị thông tin ra màn hình.

Bài tập 5.2: Xây dựng chương trình thực hiện các yêu cầu sau:

- Nhập vào một danh sách sinh viên gồm các thông tin: Mã sinh viên, tên sinh viên, viết lớp.
- Hiển thị danh sách sinh viên vừa nhập ra màn hình, thông tin của mỗi sinh viên trên một dòng.

Bài tập 5.3: Xây dựng chương trình quản lý hàng hóa cho cửa hàng tạp hoá Nam Tuyên với các công việc chính sau:

- Nhập thông tin các mặt hàng hiện có tại cửa hàng bao gồm: Tên hàng, số lượng, giá bán.
- Tính tổng số lượng hàng hoá của cửa hàng và hiển thị ra màn hình.
- Hiển thị ra màn hình thông tin các mặt hàng có số lượng <10.
- Hiển thị ra màn hình thông tin các mặt hàng có số lượng >50.

Gợi ý:

Sử dụng từ điển để lưu thông tin của một mặt hàng: Ví dụ mặt hàng: Mì chính Vedan 450g, có số lượng 10 gói, giá bán 25000 được lưu trong từ điển có cấu trúc như sau:

{'Ten':'Mì chính vedan 450g', 'so luong': 10, 'gia' : 25000}

Sử dụng ListHH – danh sách để lưu thông tin danh sách các mặt hàng, mỗi mặt hàng có cấu trúc như trên.

– Tính tổng số lượng hàng hoá có trong cửa hàng được thực hiện như sau:

```
soLuong =0
for hh in ListHH:
    soLuong = soLuong + hh['so luong']
```

– Hiển thị thông tin các mặt hàng có số lượng <10:

```
print('Danh sách các mặt hàng có số lượng <10')
for hh in ListHH:
   if hh['so luong']<10:
        print(hh[ten], '\t', hh['so luong'],'\t', hh['gia']
```

Bài tập 5.4: Xây dựng chương trình quản lý kỳ thi tuyển sinh vào lớp 10 của một trường THPT gồm các thông tin:

Số báo danh là một chuỗi chỉ gồm các ký số và có độ dài bằng 5.

Họ và tên thí sinh là một xâu ký tự có độ dài không quá 25.

Điểm Toán là một số thực có giá trị từ [0 đến 10].

Điểm Tiếng Việt là một số thực có giá trị từ [0 đến 10].

Thực hiện các công việc sau:

– Nhập danh sách thí sinh từ bàn phím.
– Hiển thị danh sách thí sinh ra màn hình.
– Hiển thị danh sách thí sinh có tổng điểm >10 ra màn hình.
– Hiển thị danh sách thí sinh có điểm liệt ra màn hình (các thí sinh có ít nhất một điểm thi là điểm 0).

Gợi ý:

Dùng kiểu dữ liệu từ điển để lưu thông tin của mỗi thí sinh, mỗi từ điển gồm bốn thành phần với khóa lần lượt là SBD, Ten, DToan và DTiengViet, giá trị tương ứng là thông tin của thí sinh.

Bài tập 5.5: Xây dựng chương trình quản lý cửa hàng băng đĩa nhạc với các công việc chính sau:

– Nhập vào thông tin của các loại băng đĩa nhạc gồm: *Tên, thể loại, số lượng, giá tiền.*

– Hiển thị thông tin đĩa nhạc ra màn hình theo định dạng sau:

Tên Thể loại Số lượng Giá tiền

– Hiển thị ra màn hình danh sách các băng đĩa rẻ nhất (giá tiền là nhỏ nhất), trong đó thông tin hiển thị gồm:

Tên Thể loại Giá tiền nhỏ nhất

Gợi ý:

– Dùng kiểu từ điển để lưu thông tin về băng đĩa nhạc.

– Dùng danh sách để lưu các băng đĩa nhạc của cửa hàng, trong đó mỗi phần tử của danh sách là 1 từ điển.

Để hiển thị được các đĩa nhạc có giá nhỏ nhất thực hiện theo các bước sau:

- Bước 1: Tìm băng đĩa có giá tiền nhỏ nhất có trong danh sách.
- Bước 2: Duyệt danh sách, với mỗi phần tử được duyệt so sánh giá tiền của đĩa nhạc đó với giá tiền nhỏ nhất vừa tìm được ở Bước 1, nếu bằng nhau thì hiển thị thông tin của đĩa nhạc đó ra màn hình. Ngược lại, duyệt tiếp cho đến hết danh sách.
- Bước 3: Kết thúc.

Bài 6

HÀM

Sau bài học này, người học có thể:

- *Biết cách xây dựng và sử dụng một hàm trong Python*
- *Biết cách truyền tham số cho hàm (theo bốn cách)*
- *Tạo ứng dụng tính tổng hai đa thức*
- *Tạo ứng dụng sắp xếp loại học sinh (kết quả học tập)*

6.1. ĐẶT VẤN ĐỀ

Bài toán 6.1: Xây dựng chương trình thực hiện tính tổ hợp chập k của n (C_n^k).

Biết rằng : $C_n^k = \frac{n!}{k!(n-k)!}$. (1)

Ví dụ: $C_n^k = \frac{4!}{2!(4-2)!} = \frac{24}{2\times 2} = 6.$

Thuật toán:

- Bước 1: Nhập n và k từ bàn phím thoả điều kiện n >= k> 0.
- Bước 2: Tính n giai thừa = 1*2*3*…*n.
- Bước 3: Tính k giai thừa = 1*2*3*…*k.
- Bước 4: Tính (n-k) giai thừa = 1* 2*3*…* (n – k).
- Bước 5: Giá trị tổ hợp chập k của n được tính bằng công thức (1) và hiên thị kết quả ra màn hình.
- Bước 6: Kết thúc.

Chương trình 6.1: Tính tổ hợp chập k của n phần tử

```
"""
Chương trình thực hiện tính tổ hợp chập k của n.
"""
#B1 Nhập n và k từ bàn phím thoả điều kiện n>=k và k>0
while(True):
    n =int( input("Nhập n: "))
    k = int(input("Nhập k : "))
    if (n>=k) and (k>0):
        break;
#B2 Tính n! = 1*2*3*...*n
giai_thua_n =1;
for i in range(1, n+1):
    giai_thua_n = giai_thua_n *i;
#B3 Tính k! = 1*2*3*...*k
giai_thua_k =1;
for i in range(1, k+1):
    giai_thua_k = giai_thua_k *i;
#B4 Tính (n-k)! = 1*2*3*...*(n-k)
giai_thua_nk =1;
for i in range(1, n+1-k):
    giai_thua_nk = giai_thua_nk *i;
# B5 TÍnh tổ hợp chập k của n
Cnk = giai_thua_n/(giai_thua_k*giai_thua_nk)
print('Tổ hợp chập %d của %d là %d'%(k,n,Cnk))
```

Kết quả sau khi thực hiện ***chương trình 6.1***

```
Nhập n: 2
Nhập k : 5
Nhập n: 5
Nhập k : 2
Tổ hợp chập 2 của 5 là 10
```

Nhận xét:

Chương trình 6.1 ta nhận thấy đoạn chương trình để tính giai thừa của n, k và (n–k) có mã chương trình giống nhau, và được viết ba lần. Vậy có cách nào để viết đoạn chương trình trên mà ta có thể sử dụng lại đoạn mã lệnh giống nhau không?

Để tránh việc viết lại đoạn mã lệnh tính giai thừa của một số nhiều lần, chúng ta nên đặt đoạn mã lệnh này vào một khối riêng. Sau đó, mỗi khi cần tính giai thừa của một số n nào đó chúng ta chỉ cần gọi khối lệnh này và truyền cho nó một tham số n như trong chương trình 6.2 dưới đây.

Chương trình 6.2: Tính tổ hợp chập k của n phần tử sử dụng hàm

```
"""
Chương trình thực hiện tính tổ hợp chập k của n.
"""
#Tính giai thừa của một số n
def giai_Thua(n):
    #  Tính n! = 1*2*3*...*n
    giai_thua_n = 1;
    for i in range(1, n + 1):
        giai_thua_n = giai_thua_n * i;
    return giai_thua_n;
#Nhập n và k từ bàn phím thoả điều kiện n>=k và k>0
while(True):
    n =int( input("Nhập n: "))
    k = int(input("Nhập k : "))
    if (n>=k) and (k>0):
        break;
#Tính n! = 1*2*3*...*n
giai_thua_n = giai_Thua(n);
#Tính k! = 1*2*3*...*k
giai_thua_k =giai_Thua(k);
#Tính (n-k)! = 1*2*3*...*(n-k)
giai_thua_nk =giai_Thua(n-k);
# TÍnh tổ hợp chập k của n
Cnk = giai_thua_n/(giai_thua_k*giai_thua_nk)
print('Tổ hợp chập %d của %d là %d'%(k,n,Cnk))
```

Kết quả thực hiện ***chương trình 6.2***
Nhập n: 5 Nhập k : 2 Tổ hợp chập 2 của 5 là 10

Nhận xét: Ta nhận thấy chương trình 6.2 ngắn gọn hơn chương trình 6.1, tránh việc viết lại đoạn mã giống nhau.

6.2. HÀM LÀ GÌ?

Trong khi lập trình, chúng ta hay gặp những đoạn chương trình giống nhau xuất hiện ở nhiều nơi. Người lập trình phải viết lại nhiều lần đoạn mã đó, công việc này lãng phí nhiều thời gian và khó kiểm soát lỗi. Do đó, để tránh phải viết lại nhiều lần cùng một đoạn mã chương trình, tạo ra một khối lệnh và gán cho nó một tên gọi (tên hàm), mỗi lần cần thực thi đoạn mã đó thì chúng ta chỉ việc gọi tên hàm và truyền tham số vào cho nó. Hay chương trình chúng ta viết hôm nay có công việc mà chúng ta đã viết và đóng gói vào hàm từ trước (cách đây vài năm), vậy thay vì phải viết lại đoạn mã giống nhau thì chúng ta chỉ việc gọi lại hàm đã viết từ trước, do đó tiết kiệm được rất nhiều thời gian và công sức.

Hàm là một chương trình con gồm một tập các câu lệnh giải quyết một công việc cụ thể từ các tham số, giá trị truyền vào hàm sau khi thực hiện các câu lệnh hàm kết thúc có hoặc không trả về một giá trị cụ thể.

Python cung cấp cho chúng ta nhiều hàm đã được xây dựng sẵn (chúng ta đã sử dụng ở những bài trước) như hàm *print*(), *sum*(), *len*(), *input*(),.v.v. Chúng ta có thể sử dụng các hàm này trong chương trình hoặc cũng có thể tự tạo ra các hàm của riêng chúng ta.

Việc sử dụng hàm trong chương trình giúp lập trình viên tiết kiệm được nhiều thời gian, chương trình dễ đọc, dễ gỡ rối và đặc biệt là rất hiệu quả trong việc bảo trì và nâng cấp.

6.3. CÁCH XÂY DỰNG MỘT HÀM

Để định nghĩa một hàm, chúng ta sử dụng cú pháp sau:

def *<tên hàm> ([<dsts>]):*
<khối lệnh>
[**return** *<giá trị tv>]*

Trong đó:

- *<tên hàm>:* Tên của hàm do người dùng đặt theo quy tắc đặt tên, nên đặt tên hàm gợi nhớ đến nhiệm vụ hàm.
- *<dsts>:* Danh sách các tham số truyền vào làm giá trị đầu vào cho hàm để giải quyết công việc, *<dsts>* có thể có hoặc không, nếu có nhiều thì các tham số cách nhau bởi dấu phẩy.
- *<khối lệnh>:* Gồm hữu hạn các lệnh trong thân hàm thực hiện một công việc nào đó, nó được viết dịch, lùi sang phải vào cùng một khoảng trống so với từ khóa def
- *<giá trị tv>:* Là giá trị trả về của hàm sau khi hoàn thành công việc nếu có lệnh return.

Ví dụ 6.1: Xây dựng hàm tính tổng hai số với tham số truyền vào hàm là hai số a và b, giá trị trả về của hàm là tổng của a và b.

Vidu6_1.py

```
'''
    Xây dựng hàm tính tổng hai số a và b
'''
#Định nghĩa hàm có tên tong2So, tham số đầu vào hàm là a và b
def  tong2So(a,b):
        c = a + b
        return c     #Giá trị trả về của hàm là c
print("Tổng của 4 + 5 = ",tong2So(4,5))
```

Kết quả chạy chương trình ***Vidu6_1.py***

```
Tổng của 4 + 5 =  9
```

6.4. SỬ DỤNG HÀM

Sau khi định nghĩa hàm, chúng ta có thể gọi hàm để cho máy tính chạy khối lệnh trong thân hàm theo cú pháp sau:

[<tên biến> =] <tên hàm>([<dsts>])

Trong đó:

- *<tên biến>:* Biến được dùng để lưu trữ dữ liệu sau khi kết thúc hàm
- *<dsts>:* Danh sách các tham số truyền vào làm giá trị đầu vào cho hàm, <dsts> có thể có hoặc không tùy vào hàm đã xây dựng, nếu có nhiều thì các tham số cách nhau bởi dấu phẩy dưới.

Ví dụ 6.2: Xây dựng hàm tính diện tích hình chữ nhật với tham số đầu vào là độ lớn hai cạnh a và b, giá trị trả về của hàm là diện tích của hình chữ nhật đó. Chương trình chính cho nhập vào chiều dài và chiều rộng hình chữ nhật, sau đó gọi gàm tính diện tích và hiển thị kết quả ra màn hình.

Vidu6_2.py

```
'''
    Xây dựng hàm tính diện tích hình chữ nhật
'''
#Định nghĩa hàm có tên dientichHCN, tham số đầu vào hàm là a và b
def dientichHCN(a,b):
        return a*b  #Giá trị trả về của hàm
#Cho người dùng nhập vào hai cạnh của hình chữ nhật
while True:
    x = float(input("Nhập chiều dài = "))
    if x>0: break;
while True:
    y = float(input("Nhập chiều rộng = "))
    if y > 0: break;
#Gọi hàm tính diện tích, kết quả được lưu ở biến dt
dt = dientichHCN(x,y)
# Hiển thị kết quả ra màn hình
print("Diện tích của HCN: ",dt)
```

Kết quả chạy chương trình ***Vidu6_2.py***

```
Nhập chiều dài = -2
Nhập chiều dài = 4
Nhập chiều rộng = 5
Diện tích của HCN:  20.0
```

6.5. TRUYỀN THAM SỐ CHO HÀM

Trong Python, các hàm do người dùng định nghĩa có thể nhận bốn loại đối số khác nhau đó là: Truyền tham số với giá trị bắt buộc; truyền tham số mặc định; truyền tham số từ khóa và truyền tham số lựa chọn. Các kiểu tham số này cũng có thể được sử dụng riêng rẽ hoặc trộn lẫn cùng nhau khi xây dựng hàm.

6.5.1. Tham số với giá trị bắt buộc (Required arguments)

Tham số bắt buộc là các tham số yêu cầu chúng ta buộc phải truyền vào trong lời gọi hàm theo đúng thứ tự nếu không chương trình dịch sẽ thông báo lỗi. Để tránh trường hợp nhầm lẫn, chúng ta cần quan tâm đến thứ tự của các tham số truyền vào trong lời gọi hàm và nên kết nối chính xác với phần định nghĩa hàm.

Ví dụ 6.3: Viết hàm thực hiện hoán vị giá trị của hai số a và b.

Vidu6_3.py

```
'''
    Xây dựng hàm hoán vị hai số
'''
#Hàm để hoán đổi giá trị của 2 số
def hoanVi(a,b):
        return b,a
# Chương trình chính minh hoạ cách sử dụng hàm
a1=2
a2=5
print("Trước hoán vị, a1 = ",a1," a2 = ",a2)
a1,a2 = hoanVi(a1, a2)
print("Sau hoán vị, a1 = ",a1," a2 = ",a2)
#Gọi hàm hoán vị mà không truyền tham số bắt buộc, máy sẽ báo lỗi
a1,a2 = hoanVi()
```

Kết quả chạy chương trình ***Vidu6_3.py***
Trước hoán vị, a1 = 2 a2 = 5
Sau hoán vị, a1 = 5 a2 = 2
a1,a2 = hoanVi()
TypeError: hoanVi() missing 2 required positional arguments: 'a' and 'b'

6.5.2. Tham số với giá trị mặc định (Default arguments)

Tham số là dữ liệu đầu vào cho một hàm, trong nhiều trường hợp người dùng không truyền tham số trong lời gọi hàm, điều này dẫn tới chương trình chạy bị lỗi. Do đó, nhằm đảm bảo một hàm luôn thực thi bình thường (ngay cả khi người dùng không truyền tham số vào khi gọi hàm), chúng ta có thể thiết lập giá trị mặc định cho các tham số đầu vào trong quá trình xây dựng hàm.

Cú pháp:

def <tên hàm> *([<thamsố1=giá trị1>, <thamsố2=giá trị2>,...]):*
    <khối lệnh>
    [**return** *<giá trị tv>]*

Trong đó:

- *<tham số1, tham số2,... >:* Là các tham số làm dữ liệu đầu vào của hàm do người dùng đặt theo quy tắc đặt tên, nên đặt tên ngắn gọn, dễ nhớ.
- *<giá trị1, giá trị2>:* Là danh sách các giá trị khởi tạo cho tên tham số, trong đó gọi hàm nếu người dùng không truyền tham số thì Python sẽ sử dụng các giá trị này cho việc tính toán.

Ví dụ 6.4: Xây dựng hàm tính diện tích hình tròn với tham số đầu vào là bán kính đường tròn, đặt mặc định bằng không.

Vidu6_4.py

```
'''
    Xây dựng hàm tính diện tích hình tròn với bán kính mặc định
'''
def dientDT(r = 0):
        c = 3.14*r*r
        return c
dt = dientDT()
print("Diện tích đường tròn bán kính mặc định = ",dt)
dt = dientDT(9)
print("Diện tích đường tròn bán kính r = 9 là, dt = ",dt)
```

Kết quả chạy chương trình ***Vidu6_4.py***

```
Diện tích đường tròn bán kính mặc định =  0.0
Diện tích đường tròn bán kính r = 9 là, dt =  254.34
```

Ví dụ 6.5: Xây dựng hàm tính tổng ước số dương của một số nguyên với đối số mặc định của hàm bằng 1.

Vidu6_5.py

```
'''
    Xây dựng hàm tính tổng các ước số
'''
def tongUoc(a =1):
    if (a <0): a = -1*a
    tong_uoc_a = 0
    for uoc in range(1, a+1):
       if (a % uoc == 0):       # Nếu a chia hết cho uoc
           # thì cộng uoc vào tổng ước của a
              tong_uoc_a = tong_uoc_a + uoc

    return tong_uoc_a
# Gọi hàm
print("Tính tổng ước với tham số mặc định = ",tongUoc())
print("Tổng ước số dương của  6 = ",tongUoc(6))
```

Kết quả chạy chương trình ***Vidu6_6.py***
Tính tổng ước với tham số mặc định = 1 Tổng ước số dương của 6 = 12

Ghi chú:

- Trong hàm *tongUoc*() có tham số a có giá trị mặc định bằng 1, do đó nó là tùy chọn trong khi gọi hàm. Tuy nhiên, nếu một giá trị được cung cấp trong lời gọi hàm, giá trị này sẽ ghi đè lên giá trị mặc định và hàm sẽ chạy với giá trị truyền vào hàm.
- Python quy định các tham số mặc định phải đứng ở vị trí sau cùng trong danh sách tham số khi định nghĩa một hàm.

6.5.3. Tham số từ khoá (keyword arguments)

Với các kiểu truyền tham số ở mục trước khi gọi hàm, chúng ta chủ yếu dựa vào vị trí của tham số trong phần định nghĩa hàm để truyền theo đúng thứ tự, ví dụ hàm với dientichHCN(a,b) ở trên khi chúng ta gọi hàm dt = dientichHCN(12,33) thì giá trị 12 sẽ được truyền vào tham số a và 33 sẽ được truyền vào tham số b. Vấn đề này đôi khi dẫn đến bất tiện cho người lập trình, do đó Python cho phép chúng ta không cần nhớ chính xác vị trí của các tham số mà chỉ cần nhớ tên tham số (vấn đề này đã được các IDE hỗ trợ theo dạng "gợi nhớ" khi chúng ta soạn thảo mã chương trình ví dụ trong PyCharm chẳng hạn) và chúng truyền theo cú pháp sau:

[<tên biến> =] <tên hàm>([thamsố1=<giá trị 1>, thamsố2=<giá trị 2>,...])

Trong đó:

- *<thamsố1, 2,...>:* Là tham số chứa giá trị đầu vào cho hàm.
- *<giá trị 1, 2,...>:* Là danh sách các giá trị tương ứng với tên tham số trong hàm, không cần quan tâm đến vị trí của chúng.

Ví dụ 6.6: Gọi hàm tính diện tích hình chữ nhật và truyền tham số theo kiểu từ khóa

Vidu6_6.py

```
'''
    Gọi hàm tính diện tích hình chữ nhật truyền tham số theo kiểu từ khóa
'''
print("Diện tích hình chữ nhật (5x25) = ", dientichHCN(a=5, b=25))
print("Diện tích hình chữ nhật (3x5) = ", dientichHCN(b=5, a=3))
```

Kết quả chạy chương trình ***Vidu6_6.py***

```
Diện tích hình chữ nhật (5x25) =  125
Diện tích hình chữ nhật (3x5) =  15
```

Chú ý: Nếu trong hàm có chứa nhiều tham số, mà ta muốn truyền tham số theo mặc định, và truyền tham số theo từ khóa thì các tham số truyền theo mặc định phải đặt ở phía cuối của danh sách tham số khi xây dựng hàm.

6.5.4. Tham số tuỳ ý (arbitrary arguments)

Phương pháp này rất hữu ích đối với người lập trình khi chúng ta xây dựng một hàm mà không biết người dùng có nhu cầu truyền vào bao nhiêu tham số, ví dụ, với bài toán tìm số lớn nhất hoặc số nhỏ nhất của n số hoặc bài toán tính tổng, tính giá trị trung bình của n số nhưng n bằng bao nhiêu? Python hỗ trợ chúng ta giải quyết các loại bài toán như vậy thông qua định nghĩa hàm với số lượng tham số tùy ý, nó khác với các kiểu tham số trên là chúng ta sử dụng dấu hoa thị (*) đặt trước tên tham số để biểu thị kiểu tham số này.

Ví dụ 6.7: Xây dựng hàm tìm số lớn nhất của n số truyền vào hàm với n là tùy ý với người dùng (ta không biết trước).

Phân tích:

Vidu6_7.py

```
'''
    Xây dựng hàm tìm số lớn nhất trong n số truyền vào
'''
def timMax(*args):  #Định nghĩa hàm tìm Max với tham số tùy chọn
   if (len(args) == 0):  #Nếu không truyền tham số gì
      return; #thì kết thúc hàm
   else:        # ngược lại có truyền tham số
      max = args[0]  #Giả sử, coi tham số đầu tiên là số max
      for arg in args:   #Duyệt qua tất cả các số còn lại
           if (max < arg):  #Nếu có số nào đó lớn hơn max thì
                max = arg # cập nhật lại giá trị max
        return max;       #Kết thúc hàm trả về giá trị Max
#Chương trình chính gọi hàm timMax()
print("Số lớn nhất của (2,3)=",timMax(2,3))
print("Số lớn nhất của (12,-3, 6)=",timMax(12,-3, 6))
print("Số lớn nhất của (2,1,5,74,3)=",timMax(2,1,5,74,3))
print(timMax())
```

Kết quả chạy chương trình *Vidu6_7.py*

```
Số lớn nhất của (2,3)= 3
Số lớn nhất của (12,-3, 6)= 12
Số lớn nhất của (2,1,5,74,3)= 74
None
```

6.6. TẠO ỨNG DỤNG

Ví dụ 6.8: Viết chương trình có sử dụng hàm thực hiện nhập vào hai đa thức một biến, sau đó tính tổng hai đa thức này và hiển thị kết quả ra màn hình.

Phân tích: Chúng ta thấy các công việc cần thiết để giải bài toán và có thể xây dựng các hàm tương ứng với các công việc đó như:

- Nhập dữ liệu cho đa thức;
- Tính tổng hai đa thức;
- Hiển thị kết quả ra màn hình;
- Chương trình chính thực hiện gọi các hàm trên.

Vidu6_8.py

```
'''
    Chương trình tính tổng hai đa thức sử dụng hàm
'''
#Nhập đa thức từ bàn phím
def NhapDT():
    pol = dict();
    while (True):
      # Người dùng phải nhập mũ của phần tử thoả điều kiện >=-1
       while (True):
           mu = int(input('Nhập số mũ hoặc -1 để kết thúc: '))
           if (mu ==-1):   #Nếu mũ = -1
              break;  # Thì kết thúc việc nhập
            # Nhập hệ số ứng với số mũ
       heso = float(input('Nhập hệ số của x^' +str( mu)+' : '))
       if (heso!=0):  # nếu hệ số <> 0 thì lưu Phần tử
         pol[mu]=heso # của đa thức theo dạng mu: heso
    return pol
#Hiển thị đa thức ra màn hình
def hienThiDT(pol):
    strpol=''
    #Sắp xếp các phần tử của đa thức theo mũ tăng dần
    for pow in sorted(pol.keys()):
        if (pol[pow]>0):
          strpol = strpol +' + '+ str(pol[pow])+' * x^'+str(pow)
        else:

          strpol = strpol +' '+ str(pol[pow]) +' * x^'+ str(pow)
    # Hiển thị đa thức đã loại bỏ ký tự + và khoảng trắng thừa
    print('P = ',strpol.strip(' +').strip())
#Tính tổng 2 đa thức
def tongDT(pol1, pol2):
        pol = dict()
        for key in pol1:
```

```
                if (key in pol2):
                        pol[key]= pol1[key]+pol2[key]
                else: pol[key] = pol1[key]
        for key in pol2:
                    if key not in pol1:
                            pol[key] = pol2[key]
        return pol
#Tính giá trị của đa thức tại điểm x0
def giaTriDaThuc(pol, x0):
       return sum(pol[pow]*x0**pow for pow in pol)

"""-------Chương trình chính------------"""
#Nhập đa thức pol
print('Nhập đa thức 1')
pol1 = NhapDT()  #Gọi hàm nhập dữ liệu cho đa thức thứ nhất
print('Nhập đa thức 2')
pol2 =NhapDT()   #Gọi hàm nhập dữ liệu cho đa thức thứ hai
#Gọi hàm tính tổng hai đa thức gán vào biến pol
pol = tongDT(pol1, pol2)
x=10
#Hiển thị các đa thức
print('Đa thức 1: ')
hienThiDT(pol1)  #Gọi hàm hiển thị đa thức thứ nhất ra màn hình
print('Đa thức 2: ')
hienThiDT(pol2)  #Gọi hàm hiển thị đa thức thứ hai ra màn hình
print('Đa thức tổng: ')
hienThiDT(pol)  #Gọi hàm hiển thị đa thức tổng nhất ra màn hình
#Hiển thị giá trị các đa thức tại điểm x0 =10
print(' Giá trị đa thức 1: ')
print(giaTriDaThuc(pol1, x))
print(' Giá trị đa thức 2: ')
print(giaTriDaThuc(pol2, x))
print(' Giá trị đa thức tổng: ')
print(giaTriDaThuc(pol, x))
```

Kết quả chạy chương trình ***Vidu6_8.py***

```
Nhập đa thức 1
Nhập số mũ hoặc -1 để kết thúc: 0
Nhập hệ số của x^0 : 2
Nhập số mũ hoặc -1 để kết thúc: 5
Nhập hệ số của x^5 : 3
Nhập số mũ hoặc -1 để kết thúc: 9
Nhập hệ số của x^9 : -3
Nhập số mũ hoặc -1 để kết thúc: -1
Nhập đa thức 2
Nhập số mũ hoặc -1 để kết thúc: 8
Nhập hệ số của x^8 : 6
Nhập số mũ hoặc -1 để kết thúc: 2
Nhập hệ số của x^2 : 7
Nhập số mũ hoặc -1 để kết thúc: 5
Nhập hệ số của x^5 : 8
Nhập số mũ hoặc -1 để kết thúc: -1
Đa thức 1:
P =  2.0 * x^0 + 3.0 * x^5 -3.0 * x^9
Đa thức 2:
P =  7.0 * x^2 + 8.0 * x^5 + 6.0 * x^8
Đa thức tổng:
P =  2.0 * x^0 + 7.0 * x^2 + 11.0 * x^5 + 6.0 * x^8 -3.0 * x^9
 Giá trị đa thức 1:
-2999699998.0
 Giá trị đa thức 2:
600800700.0
 Giá trị đa thức tổng:
-2398899298.0
```

Ví dụ 6.9: Viết chương trình quản lý học sinh thực hiện các chức năng sau:

- Nhập danh sách học sinh gồm các thông tin: *Mã học sinh*, *tên học sinh*, *điểm toán*, *điểm tiếng việt.*
- Hiển thị danh sách học sinh vừa nhập ra màn hình theo định dạng:

Mã học sinh Tên học sinh Điểm toán Điểm tiếng việt Điểm trung bình Xếp loại

Biết rằng Điểm trung bình = (Điểm toán+ Điểm tiếng việt)/2.

Xếp loại học sinh dựa vào điểm trung bình của học sinh như sau:

- Nếu điểm trung bình lớn hơn hoặc bằng 9 thì xếp loại xuất sắc.
- Nếu điểm trung bình lớn hơn hoặc bằng 8 và nhỏ hơn 9 thì xếp loại giỏi.
- Nếu điểm trung bình lớn hơn hoặc bằng 7 và nhỏ hơn 8 thì xếp loại khá.
- Nếu điểm trung bình lớn hơn hoặc bằng 5 và nhỏ hơn 7 thì xếp loại trung bình.
- Nếu điểm trung bình nhỏ hơn 5 thì xếp loại yếu.

Phân tích: Chúng có thể phân chia công việc của chương trình quản lý học sinh ở trên thành các công việc nhỏ hơn, mỗi công việc nhỏ tương ứng với một chương trình con (hàm) sau:

- Nhập thông tin một học sinh.
- Nhập danh sách học sinh.
- Tính điểm trung bình của học sinh.
- Xếp loại học sinh.
- Hiển thị thông tin học sinh ra màn hình.
- Chương trình chính thực hiện gọi các hàm.

Vidu6_9.py

```
'''
    Chương trình Quản lý học sinh
'''
#Hàm nhập thông tin học sinh từ bàn phím
def nhapHocSinh():
        # Nhập mã học sinh thoả điều kiện  khác xâu rỗng
```

```
    while(True):
        mahs = input("Nhập mã học sinh:")
        if len(mahs.strip())>0: break;
    # Nhập họ tên học sinh thoả điều kiện  khác xâu rỗng
    while(True):
        hoten = input("Nhập họ tên học sinh:")
        if len(hoten.strip()) > 0: break;
    # Nhập điểm toán thoả điều kiện nằm trong đoạn từ 0 đến 10
    while (True):
        dtoan = float(input("Nhập điểm toán = "))
        if(dtoan >=0 and dtoan <=10): break
 #Nhập điểm tiếng việt thoả điều kiện nằm trong đoạn từ 0 đến 10
    while (True):
        dtiengViet = float(input("Nhập điểm tiếng việt = "))
        if (dtiengViet >= 0 and dtiengViet <= 10): break
    # Trả về một từ điển lưu thông tin của học sinh
    return {'MaHS': mahs,'TenHS':hoten, 'DToan': dtoan, 'DTV':dtiengViet}
# Hàm nhập danh sách học sinh
def NhapDSHocSinh():
    lstHS=[]
    while (True):
        n = int(input("Nhập số lượng học sinh:"))
        if n >= 0: break
    for i in range (1, n+1):
        print('Nhập thông tin cho học sinh thứ ', i)
        lstHS.append(nhapHocSinh())
    return lstHS # Trả về danh sách học sinh vừa nhập
#Hàm tính điểm trung bình của học sinh
def diemTB(diemToan, diemTV):
    return (diemToan + diemTV)/2
#Hàm xếp loại học sinh với dtb là tham số đầu vào,
def xepLoai(dtb):
        xl=""
        if (9.0<=dtb and dtb<=10):
                xl="Xuất sắc"
        elif (8.0<=dtb and dtb<9):
                xl = "Giỏi"
        elif (7.0 <= dtb and dtb < 8):
```

```
                xl = "Khá"
        elif (5.0 <= dtb and dtb < 7):
                xl = "Trung bình"
        else:
            xl = "Yếu"
        return xl
'''

Hàm hiển thị thông tin học sinh, tham số đầu vào là danh sách học sinh
'''

def hienThi(dshs):
       print("KẾT QUẢ XẾP LOẠI SINH")
       print('STT Mã học sinh  Tên học sinh  Điểm toán Điểm tiếng việt
Điểm trung bình Xếp loại')
       for i, hs in enumerate(dshs):
           dtb = diemTB(hs['DToan'], hs['DTV'])
           xeploai = xepLoai(dtb)
           print(i+1,hs['MaHS'], hs['TenHS'],hs['DToan'], hs['DTV'], dtb,
xeploai, sep='\t\t')

#Chương trình chính
lstHS = NhapDSHocSinh()
hienThi(lstHS)
```

Kết quả chạy chương trình ***Vidu6_9.py***

```
Nhập số lượng học sinh:-2
Nhập số lượng học sinh:3
Nhập thông tin cho học sinh thứ  1
Nhập mã học sinh:hs01
Nhập họ tên học sinh:Nguyễn Thương
Nhập điểm toán = 9
Nhập điểm tiếng việt = 5
Nhập thông tin cho học sinh thứ  2
Nhập mã học sinh:hs02
Nhập họ tên học sinh:Trần Minh
```

```
Nhập điểm toán = 7
Nhập điểm tiếng việt = 6
Nhập thông tin cho học sinh thứ  3
Nhập mã học sinh:hs03
Nhập họ tên học sinh:Đỗ Anh
Nhập điểm toán = 9
Nhập điểm tiếng việt = 10
KẾT QUẢ XẾP LOẠI SINH
STT Mã học sinh  Tên học sinh  Điểm toán Điểm tiếng việt Điểm trung bình
Xếp loại
1       hs01    Nguyễn Thương   9.0     5.0     7.0     Khá
2       hs02    Trần Minh       7.0     6.0     6.5     Trung bình
3       hs03    Đỗ Anh          9.0     10.0    9.5     Xuất sắc
```

6.7. TỔNG KẾT

Trong bài học này, chúng ta đã tìm hiểu về cách xây dựng hàm do người dùng định nghĩa cũng như các phương pháp sử dụng tham số truyền vào cho hàm (tham số bắt buộc, tham số mặc định, tham số từ khóa). Với kỹ thuật sử dụng hàm, các kỹ sư lập trình có thể xây dựng thư viện (nhiều danh sách hàm) sau đó đóng gói lại thành các tệp mã chương trình riêng rẽ cung cấp cho người dùng như nhà cung cấp thứ ba. Từ đó người dùng có thể giảm được thời gian, công sức lao động thông qua việc sử dụng các thư viện, hàm có sẵn hoặc do một bên thứ ba cung cấp mà vẫn đảm bảo được chất lượng công việc. Bài học sau chúng ta sẽ tìm hiểu về các thao tác vào ra với tệp trong Python.

6.8. BÀI TẬP

A. Câu hỏi ôn tập

Câu 6.1: Hãy chọn các phát biểu đúng về ưu điểm của việc có sử dụng hàm (chương trình con) so với việc không sử dụng hàm trong chương trình?

A. Có thể tái sử dụng lại mã
B. Dễ cho việc quản lý chương trình và kiểm soát lỗi

C. Chương trình ngắn gọn hơn

D. Chương trình dễ bị lỗi hơn

Câu 6.2: Chọn các phát biểu đúng về hàm trong chương trình Python?

A. Hàm gồm một tập hợp các câu lệnh

B. Hàm có thể được gọi thực thi ở bất kỳ vị trí nào trong chương trình

C. Hàm chỉ có thể trả về duy nhất một giá trị

D. Hàm có thể trả về nhiều giá trị thuộc các kiểu dữ liệu khác nhau

Câu 6.3: Cho biết kết quả hiển thị ra màn hình sau khi máy chạy đoạn chương trình Python sau?

```
def func():
  print("Function–Hàm")
a = func ()
```

A. function–Hàm

B. function–hàm

C. Function–Hàm

D. Chương trình báo lỗi

Câu 6.4: Cho biết kết quả hiển thị ra màn hình sau khi máy chạy đoạn chương trình Python sau?

```
def  f(p, q):
   return p%q
print(f(3, 2))
```

A. 0

B. 1

C. 2B

D. 3

Câu 6.5: Cho biết kết quả hiển thị ra màn hình sau khi máy chạy đoạn chương trình Python sau?

```
def a(x):
        x = x - 1
        print(x)
a(100+1)
```

A. 99

B. 100

C. 101

D. 102

Câu 6.6: Cho biết kết quả hiển thị ra màn hình sau khi máy chạy đoạn chương trình Python sau?

```
def f1(*args):
    s = 0
    for arg in args: s = s + arg;
    return s
print(f1(2, 3, 4))
```

A. 2

B. 4

C. 9

D. 7

Câu 6.7: Cho biết kết quả hiển thị ra màn hình sau khi máy chạy đoạn chương trình Python sau?

```
def timMax(*args):
        if (len(args) == 0):
                return;
        else:
                max = args[0]
        for arg in args:
                if (max < arg):
                        max = arg
        return max;
print(timMax(2,3,4), timMax())
```

A. None

B. 4

C. 4 None

D. 4 4

Câu 6.8: Cho biết đâu là khai báo đúng cho hàm **Tong** hiển thị ra màn hình tích của 2 số nguyên 6 và 8

A. **def** Tong:
    return 6*8

B. **def** Tong(6,8):
    return 6*8

C. **def** Tong():
    print (6*8)

D. **def** Tong(6,8):
    return 6*8

Câu 6.9: Cho biết đâu là khai báo đúng cho hàm BinhPhuong(a) có tham số truyền vào là số nguyên a và trả về bình phương của nó a^2. Chọn 2 đáp án

A.
```
def TongTich(a):
    return a*2
```

B.
```
def TongTich(a):
    return a*b
```

C.
```
def TongTich(a):
    return a**2
```

D.
```
def TongTich(a):
    return a*a
```

Câu 6.10: Cho biết đâu là khai báo đúng cho hàm tongtich(a,b) có tham số truyền vào là 2 số a, b và giá trị trả về là tổng và tích của 2 số a và b đó?

A.
```
def TongTich(a,b):
    return a+b;a*b
```

B.
```
def TongTich(int a,int b):
    return a+b,a*b
```

C.
```
def TongTich(a,b):
    return a+b,a*b
```

D.
```
def TongTich(a,b):
    return a+b
    return a*b
```

B. Lập trình

Bài tập 6.1: Viết hàm kiểm tra một số có phải là số nguyên tố hay không.

Bài tập 6.2: Xây dựng chương trình hiển thị ra màn hình các số nguyên tố có giá trị nhỏ hơn 100.

Bài tập 6.3: Xây dựng hàm thực hiện chuẩn hoá xâu (giữa các từ chỉ phân tách nhau bởi một dấu cách, viết hoa ký tự đầu tiên của các từ, các ký tự không phải là ký từ đầu của từ phải là chữ in thường.

Bài tập 6.4: Xây dựng chương trình nhập vào một chuỗi ký tự từ bàn phím, hiển thị ra màn hình chuỗi ký tự đó sau khi được chuẩn hoá bằng hàm viết ở Bài tập 6.3.

Bài tập 6.5: Xây dựng hàm có tên ht_skcp(n) hiển thị ra màn hình các số không là số chính phương trong khoảng từ 2 đến n.

Ví dụ: Khi gọi hàm ht_skcp (12), ta thu được kết quả trên màn hình

2 3 5 6 7 8 10 11 12

Bài tập 6.6: Xây dựng hàm thực hiện tính tổng chuỗi S_n với tham số đầu vào hàm là số n, giá trị trả về của hàm là tổng tính được

$$S_n = 1 + \frac{1}{3} + \frac{1}{5} + ... + \frac{1}{2n+1}$$

Bài tập 6.7: Xây dựng chương trình quản lý kỳ thi tuyển sinh vào lớp 10 của một trường THPT gồm các thông tin:

Số báo danh là một chuỗi chỉ gồm các ký số và có độ dài bằng 5.

Họ và tên thí sinh (là một xâu ký tự có độ dài không quá 25).

Điểm Toán là một số thực có giá trị từ [0 đến 10].

Điểm tiếng Việt là một số thực có giá trị từ [0 đến 10].

Thực hiện các chức năng sau:

– Nhập danh sách thí sinh từ bàn phím.

– Hiển thị danh sách thí sinh ra màn hình.
– Hiển thị danh sách thí sinh có tổng điểm lớn hơn 10 ra màn hình.
– Hiển thị danh sách thí sinh có điểm liệt ra màn hình (các thí sinh có ít nhất một điểm thi là điểm 0).

Yêu cầu: Phân chia chương trình thành các chức năng nhỏ, sử dụng hàm để thực hiện các chức năng đó.

Bài 7

THAO TÁC VỚI TỆP DỮ LIỆU

Sau bài học này, người học có thể:

- *Thao tác với tệp: mở, ghi, đọc dữ liệu từ chương trình Python*
- *Sử dụng thư viện pandas*
- *Tạo ứng dụng quản lý một lớp sinh viên*

7.1. ĐẶT VẤN ĐỀ

Bài toán 7.1: Xây dựng chương trình nhập vào một danh sách sinh viên của một lớp, hiển thị ra màn hình danh sách sinh viên vừa nhập. Biết rằng mỗi sinh viên gồm các thông tin: Mã sinh viên, họ tên, lớp, quê quán.

Thuật toán

- Bước 1: Tạo danh sách trống để lưu thông tin của các sinh viên.
- Bước 2: Nhập mã sinh viên.
- Bước 3: Nếu mã sinh viên trống thì kết thúc việc nhập và chuyển đến Bước 6.
- Bước 4: Nhập họ tên, lớp, quê quán của sinh viên.
- Bước 5: Thêm sinh viên vào danh sách và quay lại Bước 2.
- Bước 6: Hiển thị danh sách sinh viên vừa nhập ra màn hình.
- Bước 7: Kết thúc.

Chương trình 7.1: *Nhập và hiển thị danh sách sinh viên của một lớp*

```
""" Chương trình nhập vào một danh sách sinh viên của một lớp, hiển thị ra
màn hình danh sách sinh viên vừa nhập. Biết rằng mỗi sinh viên gồm các
thông tin: Mã sinh viên, Họ tên, Lớp, Quê quán.
"""
```

```
DSSV=[] #B1
#Nhập danh sách sinh viên từ bàn phím
while(True):
    maSV = input("Mã sinh viên: ") #B2
    if maSV=="":  #B3
        break
    tenSV = input("Tên sinh viên: ") #B4
    Lop = input("Lớp: ")  # B5
    QueQuan = input("Quên quán: ")  # B6
    DSSV.append([maSV, tenSV,Lop, QueQuan]) #B7
#B8 Hiển thị danh sách sinh viên ra màn hình
print('Danh sách sinh viên vừa nhập')
print('\t'.join(['Mã SV', 'Họ tên SV', ' Lớp ', 'Quê quán']))
for sv in DSSV:
    print('\t'.join(sv))
```

Kết quả thực hiện ***chương trình 7.1***

```
Mã sinh viên: 101151
Tên sinh viên: Nguyễn Hoàng Anh
Lớp: TK15.1
Quên quán: Hưng yên
Mã sinh viên: 101152
Tên sinh viên: Đặng Vân Anh
Lớp: TK15.1
Quên quán: Hưng Yên
Mã sinh viên:
Danh sách sinh viên vừa nhập
Mã SV   Họ tên SV       Lớp     Quê quán
101151  Nguyễn Hoàng Anh        TK15.1  Hưng yên
101152  Đặng Vân Anh    TK15.1  Hưng Yên
Process finished with exit code 0
```

Nhận xét:

Chương trình trên cho phép chúng ta nhập danh sách sinh viên và hiển thị danh sách đó ra màn hình. Tuy nhiên, mỗi khi chúng ta chạy lại chương

trình thì danh sách sinh viên mất đi, và ta phải nhập lại từ đầu. Vậy, làm thế nào để chúng ta có thể lưu thông tin vừa nhập để mỗi lần chạy lại chương trình dữ liệu nhập vào không bị mất đi.

Giải pháp cho vấn đề này là:

- Thay vì lưu dữ liệu vào danh sách, chúng ta sẽ thực hiện lưu vào tệp có tên là 'SinhVien.txt'.
- Để hiển thị danh sách sinh viên vừa nhập ra màn hình chúng ta chỉ cần đọc dữ liệu từ tệp và hiển thị kết quả lên màn hình.

Chương trình 7.2: *Đọc danh sách sinh viên của một lớp từ tệp*

```
"""
Chương trình nhập vào một danh sách sinh viên của một lớp, hiển thị ra màn
hình danh sách sinh viên vừa nhập. Thông tin của sinh viên được lưu vào file
Biết rằng mỗi sinh viên gồm các thông tin: Mã sinh viên, Họ tên, Lớp, Quê quán.
"""
file = open('SinhVien.txt','a', encoding='utf-8')
#Nhập danh sách sinh viên từ bàn phím
while(True):
    maSV = input("Mã sinh viên: ") #B2
    if maSV=="":  #B3
        break
    tenSV = input("Tên sinh viên: ") #B4
    Lop = input("Lớp: ")  # B5
    QueQuan = input("Quê quán: ")  # B6
# Ghi dữ liệu vào file
file.write('\t'.join([maSV, tenSV,Lop, QueQuan])+'\n')
file.close()
#B8 Hiển thị danh sách sinh viên ra màn hình
print('Danh sách sinh viên vừa nhập')
print('\t'.join(['Mã SV', 'Họ tên SV', ' Lớp ', 'Quê quán']))
file = open('SinhVien.txt','r', encoding='utf-8')
for sv in file.readlines():
    print(sv)
```

Kết quả chạy ***chương trình 7.2*** lần 1

```
Mã sinh viên: 101151
Tên sinh viên: Trần Thanh Bình
Lớp: TK15.1
Quên quán: Hưng yên
Mã sinh viên: 101152
Tên sinh viên: Nguyễn Yên Vui
Lớp: TK15.1
Quên quán: Hưng Yên
Mã sinh viên:
Danh sách sinh viên vừa nhập
Mã SV   Họ tên SV        Lớp    Quê quán
101151  Trần Thanh Bình TK15.1  Hưng yên
101152  Nguyễn Yên Vui  TK15.1  Hưng Yên
```

Kết quả chạy ***chương trình 7.2*** lần 2

```
Mã sinh viên:
Danh sách sinh viên vừa nhập
Mã SV   Họ tên SV        Lớp    Quê quán
101151  Trần Thanh Bình TK15.1  Hưng yên
101152  Nguyễn Yên Vui  TK15.1  Hưng Yên
```

Nhận xét: Như vậy ở chương trình 7.2 nhờ ghi dữ liệu vào tệp mà dữ liệu của chúng ta không bị mất khi chạy lại chương trình, điều này giúp chúng ta lưu trữ được dữ liệu lâu dài; tiết kiệm thời gian, công sức nhập liệu.

7.2. TỆP TRONG PYTHON

Tệp (File) là một vùng nhớ trên ổ đĩa của máy tính được quản lý bởi tên; tệp dùng để lưu trữ dữ liệu, dữ liệu lưu trong tệp sẽ không bị mất khi chương trình kết thúc hoặc tắt máy tính. Tên tệp bao gồm hai phần: Phần tên và

phần mở rộng, hai phần này ngăn cách nhau bởi dấu chấm. Phần tên do người sử dụng đặt theo quy tắc đặt tên tệp của hệ điều hành và trong một thư mục không chứa hai tệp có cùng tên và cùng phần mở rộng; phần mở rộng cho ta biết kiểu của tệp. Ví dụ tệp có đuôi *.doc hay *. docx là một tệp văn bản được tạo ra bởi phần mềm Microsoft Word; các tệp lưu trữ ảnh *.bmp, *. gif, v.v...; tệp bảng tính *.xls; tệp có phần mở rộng *.txt lưu trữ dữ liệu dạng văn bản; .v.v..

Trong một chương trình máy tính, để làm việc với tệp dữ liệu chúng ta thực hiện các bước sau:

- Bước 1: Mở (tạo) tệp dữ liệu.
- Bước 2: Đọc (ghi) dữ liệu từ (vào) tệp.
- Bước 3: Đóng tệp để các tài nguyên gắn với tệp được giải phóng.

7.3. MỞ TỆP

Python cung cấp hàm open() để thực hiện mở tệp với tên và đường dẫn (tạo tệp nếu tệp chưa có), hàm này trả về một đối tượng File, được sử dụng để đọc hoặc ghi dữ liệu vào tệp tương ứng với chế độ được thiết lập khi gọi hàm open().

Cú pháp:

<Tên biến tệp> = open(file_name, mode, encoding)

Trong đó:

- *<Tên biến tệp>*: Là tên biến của đối tượng tệp do người dùng đặt theo quy tắc đặt tên của Python.
- *file_name*: Là đường dẫn trỏ đến tệp cần mở (tạo mới nếu chưa có).
- *mode*: Là chế độ mở tệp được mô tả trong Bảng 7.1.
- *encoding*: Kiểu mã hoá khi đọc (hoặc ghi) tệp, nếu muốn dùng tiếng việt thì nên đặt encoding = "utf–8".

Bảng 7.1: Các chế độ mở tệp

Chế độ	Mô tả
'r'	Mở tệp để đọc dữ liệu (chế độ mặc định)
'w'	Mở tệp để ghi dữ liệu vào tệp; nếu tệp chưa tồn tại thì sẽ tạo tệp mới, nếu tệp đã tồn tại thì tất cả dữ liệu cũ trong tệp sẽ bị xoá.
'x'	Mở một tệp đã tồn tại trên ổ đĩa, nếu tệp chưa tồn tại sẽ phát sinh lỗi.
'a'	Mở tệp để ghi dữ liệu vào tệp; nếu tệp chưa tồn tại thì sẽ tạo tệp, nếu tệp đã tồn tại thì sẽ ghi dữ liệu vào cuối tệp (ghi nối đuôi).
't'	Mở tệp ở chế độ văn bản.

Ví dụ 7.1: Mở tệp

Vidu7_1.py

```
# Mở tệp có tên test.txt để đọc
f = open("test.txt")
# Mở tệp có tên test.txt ở chế độ ghi văn bản
f = open("test.txt",'w')
# Mở tệp có tên test.txt để đọc với mã hoá utf-8
f = open("test.txt",mode = 'r',encoding = 'utf-8')
```

7.4. GHI DỮ LIỆU VÀO TỆP

Để ghi dữ liệu vào tệp ta thực hiện các bước sau đây:

- Bước 1: Mở tệp ở chế độ ‘w’ hoặc ‘x’ hoặc ‘+’ hoặc ‘a’
- Bước 2: Sử dụng phương thức write() hoặc writelines() để ghi dữ liệu vào tệp theo cú pháp sau:

  <Tên biến tệp>.write(s)

  < Tên biến tệp>.writelines(lines)

Trong đó:

- *s* : Là một chuỗi ký tự chứa dữ liệu cần ghi vào tệp
- *lines*: Là một danh sách, mỗi phần tử của danh sách là một chuỗi ký tự

- Bước 3: Đóng tệp bằng câu lệnh *<Tên biến tệp>.close()*
- Bước 4: Kết thúc.

Ví dụ 7.2: Ghi dữ liệu vào tệp văn bản

Vidu7_2.py

```
'''
    Ghi dữ liệu vào tệp văn bản test.txt
'''
f = open('test.txt', mode='w', encoding='utf-8')
f .writelines(['Đường vô xứ Huế loanh quanh\n', 'Non xanh nước biếc như tranh hoạ đồ.'])
f .close()
```

Kết quả chạy chương trình ***Vidu7_2.py***

```
Ta sẽ thu được tệp test.txt có nội dung như sau:
Đường vô xứ Huế loanh quanh
Non xanh nước biếc như tranh hoạ đồ.
```

Chú ý: Để đảm bảo tệp có thể được truy cập bởi một đối tượng khác mà không phát sinh lỗi sau khi đọc/ghi dữ liệu vào tệp ta nên sử dụng phương thức close() để giải phóng tài nguyên tệp, đóng tệp.

7.5. ĐỌC DỮ LIỆU TỪ TỆP

Để đọc dữ liệu từ tệp, ta thực hiện các bước sau đây:

- Bước 1: Mở tệp ở chế độ đọc .
- Bước 2: Sử dụng phương thức *read*() hoặc *readlines*() hoặc *readline*() của đối tượng tệp để đọc dữ liệu từ tệp:

– *<Tên biến tệp>.read(n):* Đọc n ký tự từ tệp, nếu n bằng không hoặc n lớn hơn kích thước của tệp thì sẽ đọc toàn bộ tệp.

– *<Tên biến tệp>.readline():* Đọc và trả về một dòng dữ liệu từ tệp.

– *<Tên biến tệp>.readlines():* Đọc và trả về một danh sách, mỗi phần tử của danh sách là một dòng dữ liệu từ tệp.

- Bước 3: Đóng tệp bằng câu lệnh *<Tên biến tệp>.close()*
- Bước 4: Kết thúc.

Ví dụ 7.3: Đọc dữ liệu từ tệp test.txt

Vidu7_3.py

```
'''
    Đọc dữ liệu từ tệp văn bản test.txt, hiển thị nội dung tệp ra màn
hình
'''
file = open('test.txt', mode='r', encoding='utf-8')
lines = file.readlines()
for line in lines: print(line)
file .close()
```

Ta sẽ thấy trên màn hình có kết quả như sau:

Kết quả chạy chương trình ***Vidu7_3.py***

```
Đường vô xứ Huế loanh quanh
Non xanh nước biếc như tranh hoạ đồ.
```

Ví dụ 7.4: Viết chương trình Python thực hiện các công việc sau đây:

- Nhập thông tin về một danh sách sinh viên, mỗi sinh viên gồm: Mã sinh viên, họ tên, lớp, quê quán.
- Lưu thông tin về danh sách sinh viên này vào tệp sinhvien.txt.
- Đọc và hiển thị thông tin về sinh viên lưu trong tệp sinhvien.txt ra màn hình.

Vidu7_4.py

```
'''
    Nhập, ghi thông tin về danh sách sinh viên vào tệp văn bản
sinhvien.txt
'''
file = open('sinhvien.txt','a', encoding='utf-8')
#Nhập danh sách sinh viên từ bàn phím
```

```
while (True):
        maSV = input("Mã sinh viên: ")  #Nhập vào mã sinh viên
        if (maSV == ""):  # Nếu mã sinh viên là xâu rỗng
                break #Thì kết thúc việc nhập
        tenSV = input("Tên sinh viên: ")# Nhập tên sinh viên
        Lop = input("Lớp: ")  # Nhập lớp sinh viên
        QueQuan = input("Quê quán: ")    # Nhập quê quán
        file.write('\t'.join([maSV, tenSV,Lop, QueQuan])+'\n')       #
Ghi dữ liệu vào file
file.close()
#Hiển thị danh sách sinh viên ra màn hình
print('Danh sách sinh viên vừa nhập')
print('\t'.join(['Mã SV', 'Họ tên SV', ' Lớp ', 'Quê quán']))
file = open('sinhvien.txt','r', encoding='utf-8')
for sv in file.readlines():
        print(sv.replace('\n',''))
file.close()
```

Ta sẽ thu được tệp sinhvien.txt có nội dung như sau:

Kết quả chạy chương trình ***Vidu7_4.py***

```
Mã sinh viên: sv001
Tên sinh viên: Nguyễn Thanh Bình
Lớp: TK15.1
Quê quán: Hưng yên
Mã sinh viên: sv002
Tên sinh viên: Trần Văn Vui
Lớp: TK15.1
Quê quán: Hải Dương
Mã sinh viên: sv03
Tên sinh viên: Nguyễn Hồng Nhung
Lớp: TK15.2
Quê quán: Thái Bình
Mã sinh viên: sv004
```

```
Tên sinh viên: Trần Văn Đường
Lớp: TK15.3
Quên quán: Hà Nội
Mã sinh viên: sv05
Tên sinh viên: Nguyễn Xuân Anh
Lớp: TK15.3
Quên quán: Hải Dương
Mã sinh viên:
Danh sách sinh viên vừa nhập
Mã SV   Họ tên SV        Lớp    Quê quán
sv001   Nguyễn Thanh Bình       TK15.1  Hưng yên
sv002   Trần Văn Vui    TK15.1  Hải Dương
sv03    Nguyễn Hồng Nhung       TK15.2  Thái Bình
sv004   Trần Văn Đường  TK15.3  Hà Nội
sv05    Nguyễn Xuân Anh TK15.3  Hải Dương
```

7.6. GIỚI THIỆU THƯ VIỆN PANDAS

Pandas là một gói thư viện mã nguồn mở của Python, nó cung cấp các thành phần để hỗ trợ người dùng lưu trữ, phân tích và xử lý dữ liệu dưới dạng bảng (theo hàng và cột, mỗi hàng biểu diễn cho một đối tượng, mỗi cột biểu diễn một thuộc tính của dối tượng, gọi là "dataframe"). Với Pandas, chúng ta có thể thực hiện được các thao tác sau:

- Đọc, ghi dữ liệu giữa bộ nhớ và nhiều định dạng tệp dữ liệu khác nhau như: csv, text, excel, sql database,...
- Thêm, xóa các cột dữ liệu trong bảng dữ liệu
- Thay đổi bố cục của dữ liệu
- Sắp xếp dữ liệu trong bảng
- Lưu dữ liệu trong bảng vào tệp *.csv
- Tính tổng theo nhóm dữ liệu trong bảng,...

Pandas được sử dụng rộng rãi trong các ngành khoa học, tài chính, thống kê, kinh tế xã hội và nhiều lĩnh vực khoa học kỹ thuật khác.

7.6.1. Cài đặt thư viện Pandas

Để cài đặt thư viện Pandas trên windows ta thực hiện:

- Bước 1: Vào run, thực thi lệnh cmd.
- Bước 2: Nhập dòng lệnh pip install pandas.

Chú ý: Để cài đặt được Pandas theo cách này, ta cần thực hiện thiết lập biến môi trường trỏ đến thư mục chứa tệp pip.exe và máy tính có kết nối Internet.

7.6.2. Khai báo sử dụng thư viện Pandas

Để sử dụng được các phương thức trong thư viện Pandas, đầu tiên ta thực hiện các bước sau:

- Bước 1: Khai báo sử dụng thư viện Pandas trong chương trình bằng câu lệnh:

  import pandas [**as** pd]

- Bước 2: Gọi các phương thức của thư viện Pandas theo cú pháp sau:

  pandas.<*Tên phương thức*>*(đối số)*

7.6.3. Đọc dữ liệu từ tệp

Để dọc dữ liệu từ tệp văn bản (text), tệp dữ liệu bảng (csv), chúng ta sử dụng phương thức read_csv() trong thư viện Pandas.

Cú pháp:

<*df*> = pd.read_csv(*file,* sep= *sep1*, header= None, names=*columns*)

Trong đó:

- <*df*>: Là đối tượng dữ liệu bảng được dùng để chứa dữ liệu đọc được từ tệp.
- *file*: Là đường dẫn, tên tệp chứa dữ liệu mà chúng ta cần đọc.
- *sep1*: Là ký tự được sử dụng để phân tách các cột dữ liệu trong tệp (mặc định nó là dấu phẩy).
- *columns*: Là một danh sách nhãn đại diện cho tên các cột chứa dữ liệu cần đọc.

Ví dụ 7.5: Viết chương trình đọc dữ liệu từ tệp *sinhvien.txt* và hiển thị ra màn hình danh sách sinh viên đã được sắp xếp theo lớp.

Vidu7_5.py

```
'''
    Đọc dữ liệu từ tệp văn bản sinhvien.txt sử dụng thư viện pandas
'''
#Khai báo sử dụng thư viện pandas với bí danh là pd
import pandas  as  pd
file ="SinhVien.txt"
#Đọc dữ liệu từ file SinhVien.txt bằng thư viện pandas
SV = pd.read_csv(file, sep= '\t', header= None, names=['Mã_SV',
'Họ_tên_SV', 'Lớp', 'Quê_quán'])
# Sắp xếp danh sách sinh viên theo lớp
DSSV =SV.sort(['Lớp'])
# Hiển thị danh sách sinh viên ra màn hình
print(DSSV)
```

Kết quả chạy chương trình ***Vidu7_5.py***

```
Mã_SV          Họ_tên_SV     Lớp   Quê_quán
0  sv001  Nguyễn Thanh Bình  TK15.1   Hưng yên
1  sv002       Trần Văn Vui  TK15.1  Hải Dương
2   sv03  Nguyễn Hồng Nhung  TK15.2  Thái Bình
3  sv004     Trần Văn Đường  TK15.3     Hà Nội
4   sv05    Nguyễn Xuân Anh  TK15.3  Hải Dương
```

7.6.4. Trích rút, chọn dữ liệu từ DataFrame

Trong nhiều bài toán, chúng ta không lấy hết tất cả dữ liệu trong bảng hoặc tệp dữ liệu mà chúng ta chỉ muốn trích rút, chọn một phần dữ liệu trong tập dữ liệu thoả mãn điều kiện cho trước, khi đó ta sẽ thực hiện lệnh theo cú pháp:

<df1> = df.query([điều kiện trích chọn])

Trong đó:

- *df*: Là một đối tượng, chứa toàn bộ dữ liệu bảng.
- *df1*: Là một đối tượng dữ liệu bảng chứa toàn bộ hoặc một phần dữ liệu trong đối tượng *df* thoả mãn điều kiện trích chọn.
- *[điều kiện trích chọn]:* Là một xâu mô tả điều kiện chúng ta muốn lấy dữ liệu, nó tương tự với biểu thức điều kiện trong mệnh đề where của một câu truy vấn SQL.

Ví dụ 7.6: Giả sử chúng ta có một danh sách sinh viên được nhập từ Chương trình Vidu_7.5 và được lưu trong Dataframe có tên *SV* gồm các cột dữ liệu: *Mã_SV; Họ_tên_SV; Lớp; Quê_quán.*

Khi đó chúng ta có thể sử dụng các câu lệnh truy vấn để thực hiện trích rút, chọn một phần dữ liệu như hiển thị danh sách sinh viên thuộc lớp TK15.1 hoặc quê ở Hải Dương bằng chương trình sau:

Vidu7_6.py

```
'''
    Đọc dữ liệu từ tệp văn bản sinhvien.txt sử dụng thư viện pandas
'''
#Khai báo sử dụng thư viện pandas với bí danh là pd
import pandas  as  pd
file ="SinhVien.txt"
#Đọc dữ liệu từ file SinhVien.txt bằng thư viện pandas
SV = pd.read_csv(file, sep= '\t', header= None, names=['Mã_SV',
'Họ_tên_SV', 'Lớp', 'Quê_quán'])
# Trích rút dữ liệu theo điều kiện truy vấn
sv152=SV.query('Quê_quán =="Hải Dương" or Lớp =="TK15.1"')
# Hiển thị kết quả ra màn hình
print('-----------\nDanh sách sinh viên \n--------\n', sv152)
```

Kết quả chạy chương trình ***Vidu7_6.py***

```
--------
Danh sách sinh viên
--------
    Mã_SV         Họ_tên_SV     Lớp   Quê_quán
0  sv001  Nguyễn Thanh Bình  TK15.1   Hưng yên
1  sv002       Trần Văn Vui  TK15.1  Hải Dương
4   sv05    Nguyễn Xuân Anh  TK15.3  Hải Dương
```

Ví dụ 7.7: Tương tự như ví dụ 7.6, chúng ta sử dụng câu truy vấn để thực hiện trích rút, chọn một phần dữ liệu như hiển thị danh sách sinh viên thuộc lớp TK15.1 hoặc TK15.2 bằng chương trình sau:

Vidu7_7.py

```
'''
    Đọc dữ liệu từ tệp văn bản sinhvien.txt sử dụng thư viện pandas
'''
#Khai báo sử dụng thư viện pandas với bí danh là pd
import pandas  as  pd
file ="SinhVien.txt"
#Đọc dữ liệu từ file SinhVien.txt bằng thư viện pandas
SV = pd.read_csv(file, sep= '\t', header= None, names=['Mã_SV',
'Họ_tên_SV', 'Lớp', 'Quê_quán'])
# Trích rút dữ liệu theo điều kiện truy vấn
Lop =['TK15.1', 'TK15.2']
sv152=SV.query('Lớp in @Lop')
# Hiển thị kết quả ra màn hình
print('-----------\nDanh sách sinh viên \n--------\n', sv152)
```

Kết quả chạy chương trình ***Vidu7_7.py***

```
------------
Danh sách sinh viên
--------
    Mã_SV          Họ_tên_SV    Lớp   Quê_quán
0  sv001  Nguyễn Thanh Bình  TK15.1   Hưng yên
1  sv002       Trần Văn Vui  TK15.1  Hải Dương
2   sv03  Nguyễn Hồng Nhung  TK15.2  Thái Bình
```

7.7. TẠO ỨNG DỤNG VỚI TỆP DỮ LIỆU

Ví dụ 7.8: Xây dựng chương trình Python quản lý một lớp (sinh viên) với các chức năng chính sau đây:

- Nhập vào thông tin của một danh sách sinh viên gồm: Mã sinh viên, họ tên, lớp, quê quán từ bàn phím.
- Lưu thông tin của các sinh viên này vào tệp svchung.txt.
- Hiển thị ra màn hình danh sách sinh viên của lớp *X*.
- Hiển thị ra màn hình các sinh viên có tên *X*.

- Đọc dữ liệu từ tệp svchung.txt, sau đó ghi danh sách sinh viên này sang các tệp khác nhau sao cho sinh viên của mỗi lớp được lưu trong một tệp riêng với tên tệp chính là tên lớp.

Phân tích:

Chúng ta có thể sử dụng một cấu trúc dữ liệu danh sách để lưu trữ thông tin về sinh viên, mỗi sinh viên này lại được lưu trong một danh sách con khác. Thông tin của tất cả các sinh viên cần được lưu vào tệp bằng mã hoá theo chuẩn định dạng chuyển đổi Unicode 8–bít, do đó chúng ta sẽ sử dụng tham số encoding='utf–8' trong câu lệnh mở tệp để đọc, ghi. Để tách sinh viên của từng lớp trong dữ liệu bảng, chúng ta thực hiện theo các bước sau:

- Bước 1: Lấy về danh sách các lớp

  DSLop = DSSV['Lớp'].unique()

- Bước 2: Duyệt qua từng lớp

  for lop **in** DSLop:

  - Bước 2.1: Lấy về danh sách sinh viên thuộc từng lớp theo biến lop

    sv_lop = DSSV.query('Lớp ==@lop')

  - Bước 2.2: Ghi danh sách sinh viên của lop vào tệp lop.txt

    sv_lop.to_csv (lop+'.txt', index=None, encoding='utf–8', sep='\t')

- Bước 3: Đóng tệp
- Bước 4: Kết thúc

Vidu7_8.py

```
'''
    Chương trình xử lý thông tin tệp sinh viên
'''
#Khai báo sử dụng thư viện pandas với bí danh là pd
import pandas  as  pd
import os
# Khai báo thư viện os để thao tác với tệp thư mục
#Nhập danh sách sinh viên từ bàn phím
def NhapSV():
        file = open("svchung.txt", 'a', encoding='utf-8')
```

```
        while(True):
            maSV = input("Mã sinh viên: ")
            if (maSV==""):
                        break
            tenSV = input("Tên sinh viên: ")
            Lop = input("Lớp: ")
            QueQuan = input("Quê quán: ")
            # Ghi dữ liệu vừa nhập vào tệp
            file.write('\t'.join([maSV, tenSV,Lop, QueQuan])+'\n')
        file.close()
#Đọc dữ liệu từ tệp svchung.txt
def Doc_DSSV_Tu_File(file):
   SV = pd.read_csv(file, sep='\t', header=None, names=['Mã_SV',
'Họ_tên_SV', 'Lớp', 'Quê_quán'])
   return SV
# Hiển thị danh sách sinh viên cua lớp x
def HienThi_DSSV_1_Lop(Lop, DSSV):
        DSSV_Lop  = DSSV.query('Lớp ==@Lop')
        sosv = DSSV_Lop.shape[0]
        if (sosv > 0):
           print(' Có :', sosv, ' sinh viên trong lớp: ', Lop)
           print(DSSV_Lop)
        else:
           print('Không có sinh viên nào trong lớp: ', Lop)
# Hiển thị danh sách sinh viên có tên là X
def HienThi_DSSV_ten(TenSV, DSSV):
        DSSV_Ten  = DSSV.query('Họ_tên_SV == @TenSV')
        sosv =DSSV_Ten.shape[0]
        if (sosv>0):
                print(' Có :',sosv,' sinh viên có tên ', TenSV)
                print(DSSV_Ten)
        else:
                print('Không có sinh viên nào có tên ',TenSV )
#Ghi dữ liệu vào tệp bảng tính Excel
def Ghi_DSSV_vao_File( DSSV):
        # Lấy về tất cả tên lớp có trong DataFrame DSSV
    DSLop = DSSV['Lớp'].unique()
    for lop in DSLop:
           # Lấy về danh sách sinh viên thuộc lớp @lop
        sv_lop = DSSV.query('Lớp ==@lop')
```

```
        # ghi danh sách sinh viên thuộc lớp @lop vào file lop.txt
        sv_lop.to_csv (lop + '.txt', index = None, encoding='utf-8',
sep='\t')
#Nội dung chương trình chính
file =" svchung.txt"
NhapSV()                        #Gọi hàm nhập dữ liệu
SV = Doc_DSSV_Tu_File(file)     #Gọi hàm đọc danh sách sinh viên
#Gọi hàm hiển thị danh sách sinh viên của một lớp
HienThi_DSSV_1_Lop('TK15.1',SV)
#Gọi hàm hiển thị thông tin của một sinh viên
HienThi_DSSV_ten('Nguyễn Thanh Bình',SV)
#Gọi hàm ghi thông tin sinh viên vào các tệp tương ứng với lớp
Ghi_DSSV_vao_File( SV)
```

Kết quả chạy chương trình ***Vidu7_8.py***

```
Mã sinh viên: sv001
Tên sinh viên: Nguyễn Thanh Bình
Lớp: TK15.1
Quên quán: Hưng yên
Mã sinh viên: sv002
Tên sinh viên: Trần Văn Vui
Lớp: TK15.1
Quên quán: Hải Dương
Mã sinh viên: sv03
Tên sinh viên: Nguyễn Hồng Nhung
Lớp: TK15.2
Quên quán: Thái Bình
Mã sinh viên: sv004
Tên sinh viên: Trần Văn Đường
Lớp: TK15.3
Quên quán: Hà Nội
Mã sinh viên: sv05
Tên sinh viên: Nguyễn Xuân Anh
Lớp: TK15.3
Quên quán: Hải Dương
Mã sinh viên:
Có : 2  sinh viên trong lớp:  TK15.1
   Mã_SV          Họ_tên_SV     Lớp   Quê_quán
0  sv001  Nguyễn Thanh Bình  TK15.1   Hưng yên
```

```
1  sv002       Trần Văn Vui  TK15.1  Hải Dương
 Có : 1  sinh viên có tên  Nguyễn Thanh Bình
   Mã_SV          Họ_tên_SV     Lớp  Quê_quán
0  sv001  Nguyễn Thanh Bình  TK15.1  Hưng yên
Và 3 file có tên TK15.1.txt, TK15.2.txt, TK15.3.txt chứa các sinh viên
thuộc lớp đó
```

7.8. TỔNG KẾT BÀI HỌC

Sau khi học xong bài này, chúng ta thấy Python đã cung cấp cho người dùng các phương thức để thực hiện các thao tác trên tệp dữ liệu như: Mở, đọc, ghi, các thao tác đó rất phù hợp với công việc trong thực tế cuộc sống của chúng ta. Hơn nữa, Python còn cung cấp cho người dùng thư viện Pandas để giúp người dùng dễ dàng đưa ra các truy vấn để trích rút, chọn, lọc,... trên các tệp dữ liệu. Ưu điểm của việc sử dụng thư viện Pandas trong Python là người dùng có thể gọi trực tiếp các hàm, phương thức được xây dựng sẵn; xử lý trên nhiều kiểu dữ liệu khác nhau; tốc độ thực hiện câu lệnh truy vấn nhanh, lưu trữ dữ liệu trên tệp văn bản tốn ít bộ nhớ hơn các hệ quản trị cơ sở dữ liệu chuyên dụng khác như MySQL, SQL Server.

7.9. BÀI TẬP

A. Câu hỏi ôn tập

Câu 7.1: Chọn câu lệnh thực hiện mở tệp văn bản data.txt để đọc dữ liệu?

A. file = open('data.txt', 'a', encoding='utf–8')
B. file = open('data.txt', 'w', encoding='utf–8')
C. file = open('data.txt', 'r', encoding='utf–8')
D. file = open('data1.txt', 'r', encoding='utf–8')

Câu 7.2: Chọn câu lệnh thực hiện mở tệp văn bản data.txt để ghi nối vào cuối tệp?

A. file = open('data.txt', 'a', encoding='utf–8')
B. file = open('data.txt', 'w', encoding='utf–8')
C. file = open('data.txt', 'r', encoding='utf–8')
D. file = open('data.txt', 'w', encoding='utf–8')

Câu 7.3: Giả sử chúng ta cần mở tệp data.txt để ghi dữ liệu vào tệp sao cho dữ liệu mới được ghi đè lên dữ liệu đã có trong tệp; khi đó câu lệnh nào sau đây là đúng.

A. file = open('data.txt', 'a', encoding='utf–8')

B. file = open('data.txt', 'w', encoding='utf–8')

C. file = open('data.txt', 'r', encoding='utf–8')

D. file = open('data.txt', 't', encoding='utf–8')

Câu 7.4: Giả sử chúng ta có tệp *data.txt* chứa hai câu thơ được ghi trên hai dòng như sau:

Đường vô xứ Huế loanh quanh

Non xanh nước biếc như tranh hoạ đồ

Cho biết giá trị của biến s sau khi máy thực thi đoạn lệnh Python dưới đây:

```
file = open('data.txt', 'r', encoding='utf-8')
s = file.readline()
```

A. 'Đường vô xứ Huế loanh quanh'

B. 'Đường vô xứ Huế loanh quanh \nNon xanh nước biếc như tranh hoạ đồ'

C. 'Đường vô xứ Huế loanh quanh

Non xanh nước biếc như tranh hoạ đồ'

D. 'Non xanh nước biếc như tranh hoạ đồ'

Câu 7.5: Giả sử chúng ta có tệp data.txt chứa hai câu thơ:

Đường vô xứ Huế loanh quanh

Non xanh nước biếc như tranh hoạ đồ

Cho biết giá trị của biến s sau khi máy thực thi đoạn lệnh Python dưới đây:

```
file = open('data.txt', 'r', encoding='utf-8')
s = file.readlines()
```

A. [‘Đường vô xứ Huế loanh quanh’, ‘Non xanh nước biếc như tranh hoạ đồ’]

B. ‘Đường vô xứ Huế loanh quanh \nNon xanh nước biếc như tranh hoạ đồ’

C. [‘Đường vô xứ Huế loanh quanh Non xanh nước biếc như tranh hoạ đồ’]

D. [‘Non xanh nước biếc như tranh hoạ đồ’]

Câu 7.6: Chọn câu lệnh thực hiện đọc dữ liệu từ tệp data.txt

import pandas **as** pd

A. Df = pd.read_csv(‘data.txt’, header = None)

B. Df = pd.read_text(‘data.txt’, header = None)

C. Df = pd.open(‘data.txt’, header = None)

D. Df = pd.open(‘data.txt’)

Câu 7.7: Chọn câu lệnh cho phép chúng ta ghi dữ liệu được chứa trong bảng Df vào tệp data.txt, mỗi trường dữ liệu phân tách nhau bởi ký tự ‘#’

import pandas **as** pd

A. Df.to_csv(‘data.txt’, header = None)

B. Df.to_csv(‘data.txt’, header = None, sep=’#’)

C. Df.write_csv(‘data.txt’, header = None, sep=’#’)

D. Df.write(‘data.txt’, header = None, sep=’#’)

Câu 7.8: Cho biết mệnh đề nào chưa đúng trong các mệnh đề sau?

A. Pandas là 1 thư viện trong python

B. Pandas cho phép thao tác với nhiều loại dữ liệu khác nhau

C. Pandas đọc dữ liệu là loại dataframe

D. Pandas chỉ cho phép thao tác với tệp thuộc 1 trong 3 kiểu csv, txt và xls

Câu 7.9: Cho các câu lệnh khai báo thư viện Pandas để thao tác với tệp trong Python. Cho biết khai báo nào sau đây chưa đúng

A. **import** pandas as pd

B. **import** pandas

C. **import** pandas **as** pan

D. **import** pd

Câu 7.10: Phát biểu nào trong các phát biểu sau đây là đúng?

A. Pandas có hỗ trợ truy vấn để trích rút 1 phần dữ liệu.

B. Pandas không hỗ trợ truy vấn để trích rút dữ liệu.

C. Dataframe trong truy vấn df1=df.query(*điều kiện trích chọn*) là một đối tượng chứa toàn bộ dữ liệu.

D. Dataframe trong truy vấn df1=df.query(*điều kiện trích chọn*) là một đối tượng chứa một phần dữ liệu.

B. Lập trình

Bài tập 7.1: Viết chương trình Python ghi các số nguyên có giá trị từ 1 đến 10 vào tệp văn bản data71.txt, mỗi số trên một dòng.

Bài tập 7.2: Viết chương trình Python đọc dữ liệu từ tệp văn bản data71.txt và hiển thị kết quả đọc được ra màn hình.

Bài tập 7.3: Viết chương trình Python đọc dữ liệu từ tệp văn bản data71.txt sau đó ghi nối vào dòng cuối cùng của tệp văn bản data71.txt giá trị tổng của các số có trong tệp data71.txt, sao cho dữ liệu cũ của tệp data71.txt không bị mất.

Bài tập 7.4: Viết chương trình Python cho người dùng nhập thông tin gồm họ và tên, tuổi của người dùng sau đó ghi thông tin vừa nhập vào tệp văn bản data72.txt.

Bài tập 7.5: Xây dựng chương trình quản lý tiêu thụ điện của các hộ dân tại chung cư Phúc Hưng gồm các công việc chính sau:

a) Nhập thông tin các hộ gia đình sử dụng điện tại khu chung cư Phúc Hưng, biết rằng mỗi hộ gia đình cần nhập các thông tin:

Số hiệu phòng

Số hiệu tòa nhà

Tên chủ hộ

Số kW điện tiêu thụ trong tháng

Các thông tin của người dùng nhập vào cần thoả mãn điều kiện: Số hiệu phòng, số hiệu tòa nhà, tên chủ hộ phải khác xâu rỗng (xâu rỗng là xâu không chứa ký tự nào); số điện tiêu thụ trong tháng phải lớn hơn hoặc bằng không; nếu người dùng nhập thông tin không đúng thì yêu cầu nhập lại cho đến khi dữ liệu nhập vào thoả mãn điều kiện trên.

b) Xây dựng hàm tính tiền điện tiêu thụ trong tháng của một hộ gia đình; hàm này nhận tham số đầu vào là số kW điện tiêu thụ trong tháng, trả về giá trị là tiền điện tiêu thụ của hộ gia đình theo giá bậc thang:

- 100 kW điện đầu tiên có giá 1450 đồng/kW.
- Từ 101 kW điện đến 150 kw điện tiếp theo có giá 1750 đồng/kW.
- Từ 151 kW đến 250 kW tiếp theo có giá 2000 đồng/kW.
- Từ 250 kW trở lên có giá 2500 đồng/kW.
- Thuế VAT tính bằng 10% giá tiền phải trả.

c) Hiển thị tất cả các hộ gia đình đang sử dụng điện tại chung cư Phúc Hưng theo định dạng:

Toà nhà	Phòng	Số điện tiêu thụ	Tiền điện phải trả
A1	305	100	159500
C1	204	150	255750
C1	304	350	750750

d) Ghi nối vào cuối thông tin của các hộ gia đình vào tệp văn bản sao cho các hộ gia đình cùng toà nhà được lưu vào cùng một tệp văn bản có tên tệp là tên của từng toà nhà.

Bài 8

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG VỚI PYTHON

Python là ngôn ngữ lập trình hướng đối tượng (Object Oriented Programming – OOP), nghĩa là người lập trình có thể viết chương trình tuân theo các nguyên lý cơ bản của lập trình hướng đối tượng như: Tính đóng gói, tính kế thừa và tính đa hình. OOP giúp nâng cao năng suất, đơn giản hóa độ phức tạp khi bảo trì cũng như nâng cao khả năng mở rộng phần mềm bằng cách cho phép lập trình viên tập trung vào các lớp (class), đối tượng (object) và các thành phần. Lớp cho phép người lập trình xây dựng chương trình theo một cách riêng biệt, nhờ đó mà người lập trình có thể bổ sung tính năng của phần mềm một cách nhất quán để chúng được sử dụng theo cách thức rõ ràng.

Sau bài học này, người học có thể:

- *Giải thích được các khái niệm cơ bản về lớp, đối tượng*
- *Xây dựng được các lớp trong Python*
- *Khai báo và sử dụng các đối tượng*
- *Tạo ứng dụng quản lý môn học (sử dụng lớp, đối tượng)*

8.1. LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Chúng ta cùng xem một tình huống: Khi một học viên muốn đăng kí một khóa học tại một trung tâm tiếng Anh, việc đầu tiên là nhận một tờ khai thông tin có định dạng để điền thông tin cá nhân. Tại sao lại như vậy? Tại sao trung tâm không đưa cho học viên một tờ giấy trắng và yêu cầu "Hãy cung cấp thông tin của bạn"? Không khó để có câu trả lời, trung tâm muốn thông tin được tổ chức, định dạng theo cùng một mẫu. Mẫu gồm những

thông tin được chuẩn hóa để giúp mọi việc dễ dàng hơn cho cả trung tâm và sinh viên khi cung cấp và lưu trữ.

Trong Python, lớp (class) chính là một mẫu, chúng giúp lập trình viên tổ chức và chuẩn hóa thông tin. Dòng lệnh Python tạo ra một lớp (class) là:

```
class  HocVien():
```

… tức là bạn đang bảo Python hãy tạo một lớp (class) tên là HocVien. Dùng lớp này như một mẫu để lưu thông tin của các học viên tới đăng kí học. Mỗi học viên sẽ có thông tin riêng của chính họ cho mẫu thông tin này, nhưng những thông tin đều có chung một định dạng giống nhau (mẫu).

Một lớp giống như một tờ khai thông tin có định dạng. Trước khi nó được điền, tờ thông tin đó là giống nhau cho mọi học viên. Khi một học viên điền thông tin vào một bản sao của tờ thông tin đó, nó sẽ chứa thông tin cụ thể (duy nhất) của học viên đó, mặc dù cấu trúc thông tin vẫn giống với những học viên khác.

8.2. ĐỊNH NGHĨA LỚP TRONG PYTHON

Để định nghĩa một lớp trong Python chúng ta sử dụng từ khoá class theo cú pháp sau:

class *<tên_lớp>([tên_lớp_cha]):*

[Chuỗi mô tả về lớp]

[Khai báo các thành phần dữ liệu chung của lớp]

Định nghĩa phương thức khởi tạo của lớp

Định nghĩa các hàm thành phần của lớp

Trong đó:

- *<tên_lớp>:* Là một chuỗi biểu diễn tên của một lớp do người dùng đặt theo quy tắc đặt tên trong Python.
- *[tên_lớp_cha]:* Chỉ định tên lớp đã tồn tại để lớp hiện tại có thể kế thừa (sử dụng lại các dữ liệu thành phần và hàm thành phần), nếu tên_lớp_cha không được truyền vào thì lớp này sẽ được kế thừa từ lớp object.

8.2.1. Thuộc tính của lớp, thuộc tính của đối tượng

Thuộc tính trong Python được chia ra thành hai loại là thuộc tính của lớp và thuộc tính của đối tượng.

Thuộc tính của lớp là các thuộc tính chung cho tất cả các đối tượng thuộc lớp đó, nó được khai báo ở bên ngoài tất cả các phương thức của lớp theo cú pháp:

<thuộc tính_lớp> = giá_trị

Để truy cập vào thuộc tính của lớp chúng ta truy cập theo một trong hai cách dưới đây:

<Tên lớp>.<thuộc tính_lớp>

hoặc

<Tên đối tượng>.__class__.<thuộc tính_lớp>

Thuộc tính lớp được chia sẻ chung cho mọi đối tượng của lớp đó; nghĩa là khi ta thay đổi giá trị của thuộc tính chia sẻ trong một đối tượng thì giá trị của thuộc tính chia sẻ đó trong các đối tượng khác cũng thay đổi theo.

Trái với thuộc tính lớp, thuộc tính đối tượng chỉ dành riêng cho đối tượng. Thuộc tính đối tượng được khai báo bên trong phương thức khởi tạo hoặc phương thức khác của lớp theo cú pháp:

self.<thuộc tính_đối tượng> = giá_trị

Tên của thuộc tính_đối tượng thể hiện mức độ truy cập của thuộc tính đó là private (riêng tư) hay public (toàn cục).

Để tránh việc truy cập thuộc tính đối tượng từ bên ngoài lớp, ta sẽ đặt tên thuộc tính_đối tượng bắt đầu bằng hai ký tự gạch dưới; các thuộc tính này sẽ có mức truy cập là private và chỉ có thể được truy cập bên trong lớp. Ngược lại, các thuộc tính có mức truy cập là public, chúng có thể được truy cập từ bên ngoài thông qua đối tượng của lớp theo cú pháp:

<Tên đối tượng>.<thuộc tính_đối tượng>

Ví dụ 8.1:

Thuộc tính có tên: __bankinh: Thuộc tính này có mức truy cập là private và chỉ có thể được truy cập bởi các phương thức bên trong lớp.

Thuộc tính có tên bankinh: Thuộc tính này có mức truy cập là public và có thể truy cập được ở bên ngoài lớp thông qua các đối tượng theo cú pháp: <tên_đối_tượng>.bankinh

8.2.2. Xây dựng phương thức khởi tạo của lớp

Phương thức khởi tạo sẽ chứa các câu lệnh nhằm thiết lập giá trị ban đầu cho các thuộc tính của đối tượng, phương thức này được thực thi khi có một đối tượng của lớp được tạo ra.

Cú pháp:

```
def __init__(self, tham_số_1, tham_số_2, ....):
    self.thuộc_tính_1 = tham_số_1
    self.thuộc_tính_2 = tham_số_2
    .....
```

Các thuộc tính của đối tượng ta dùng cú pháp:

```
self.thuộc_tính_i = tham_số_i
```

Trong đó:

- ***self***: Từ khoá self được dùng để chỉ đây là một thuộc tính của đối tượng, và phân biệt thuộc tính đối tượng với các biến cục bộ hay thuộc tính của lớp (thuộc tính chia sẻ).
- *thuộc_tính_i*: Là tên của thuộc tính đối tượng của lớp.

Ví dụ 8.2: Xây dựng lớp học viên và định nghĩa phương thức khởi tạo cho lớp học viên gồm các dữ liệu thành phần như: Số thẻ căn cước, tên học viên, năm sinh và danh sách các môn học học viên đăng ký học tại trung tâm.

Vidu8_2.py

```
'''
  Ví dụ về việc xây dựng lớp và hàm khởi tạo
'''
#Sử dụng lệnh class để xây dựng lớp
class HocVien(object):
    #Định nghĩa hàm khởi tạo cho lớp HocVien
```

```
    def __init__(self, soTheCanCuoc, tenHV, NamSinh, dsMonHoc):
        self. soTheCanCuoc = soTheCanCuoc   #Gán dữ liệu ban đầu cho biến
        self. tenHV = tenHV
        self. NamSinh = NamSinh
        self. dsMonHoc = dsMonHoc
```

Chú ý: Python chỉ cho phép mỗi lớp có nhiều nhất một phương thức khởi tạo *__init__(),* do vậy để có thể tạo đối tượng của lớp theo nhiều cách khác nhau chúng ta có thể sử dụng phương thức khởi tạo với tham số mặc định.

Ví dụ 8.3: Định nghĩa phương thức khởi tạo cho lớp học viên gồm các dữ liệu thành phần như: Số thẻ căn cước, tên học viên, năm sinh và danh sách các môn học học viên đăng ký học tại trung tâm; trong đó số thẻ căn cước và tên học viên là bắt buộc, năm sinh có giá trị mặc định là 1900 và danh sách các môn học có giá trị mặc định là một danh sách rỗng ([]).

Vidu8_3.py

```
'''
  Ví dụ về việc xây dựng lớp và hàm khởi tạo
'''
#Sử dụng lệnh class để xây dựng lớp
class HocVien(object):
    #Định nghĩa hàm khởi tạo cho lớp HocVien với nhiều kiểu tham số khác nhau
    def __init__(self, soTheCanCuoc, tenHV, NamSinh =2005, dsMonHoc=[]):
        self. soTheCanCuoc = soTheCanCuoc
        self. tenHV = tenHV
        self. NamSinh = NamSinh
        self. dsMonHoc = dsMonHoc
```

8.2.3. Xây dựng các phương thức của lớp

Phương thức của lớp cũng tương tự như một hàm của Python, phương thức này luôn nhận một tham số mặc định là self.

Để định nghĩa một phương thức của lớp ta thực hiện theo cú pháp:

def *<tên_hàm>(*self *[,<danh sách tham số>]):*
<khối lệnh>
[**return** *<giá trị tv>]*

Trong đó:

- *<tên hàm>:* Tên của hàm do người dùng đặt theo quy tắc đặt tên, chúng ta nên đặt tên hàm để gợi nhớ đến nhiệm vụ hàm.
- self: Là từ khóa chỉ định hàm thuộc lớp hiện tại chúng ta đang xét.
- *<danh sách tham số>:* Là các tham số truyền vào làm giá trị đầu vào cho hàm giải quyết công việc, nếu có nhiều tham số thì các tham số cách nhau bởi dấu phẩy dưới.
- *<khối lệnh>:* Gồm hữu hạn các lệnh trong thân hàm thực hiện một công việc nào đó, nó được viết lùi sang bên phải một khoảng trống so với từ khóa def.
- *<giá trị tv>:* Là giá trị trả về của hàm sau khi hoàn thành công việc nếu có lệnh returrn.

Ví dụ 8.4: Định nghĩa phương thức thêm một môn học vào danh sách các môn học mà học viên đăng ký:

Vidu8_4.py

```
'''
  Ví dụ về việc xây dựng lớp và hàm thành phần trong lớp
'''
#Sử dụng lệnh class để xây dựng lớp
class HocVien(object):
    #Định nghĩa hàm thành phần thêm môn học vào danh sách môn học
    def themMonHoc(self, tenmon):
        #Sử dụng phương thức append của lớp danh sách để thêm
        self. dsMonHoc.append(tenmon)
```

Ví dụ 8.5: Định nghĩa hàm thành phần hiển thị các thông tin của học viên ra màn hình.

Vidu8_5.py

```
'''
    Ví dụ về việc xây dựng lớp và hàm thành phần trong lớp
'''
#Sử dụng lệnh class để xây dựng lớp
class HocVien(object):
    '''Định nghĩa hàm thành phần hiển thị thông tin của học viên ra màn hình'''
def hienThi(self):
        print(self. soTheCanCuoc, self. tenHV, self. NamSinh, len(self.
dsMonHoc))
```

8.3. KHAI BÁO VÀ SỬ DỤNG BIẾN ĐỐI TƯỢNG TRONG PYTHON

Sau khi xây dựng lớp, chúng ta không thể chạy các lớp ngay được mà cần phải tạo các đối tượng của lớp và được chương trình dịch cấp phát bộ nhớ cho các đối tượng để lưu trữ dữ liệu. Cú pháp tạo đối tượng của lớp như sau:

<tên_biến_đối_tượng> = <tên_lớp>([danh_sách_đối_số])

Khi gặp câu lệnh trên, đầu tiên máy tính sẽ tìm một vùng nhớ đủ lớn (đủ để lưu trữ dữ liệu thành phần và hàm thành phần của tên lớp đã tạo) để cấp phát cho tên biến đối tượng, sau đó máy tính gọi phương thức khởi tạo của lớp để thực thi (thông thường là gán dữ liệu từ các đối số truyền vào cho các thuộc tính trong đối tượng). Nếu đối số truyền vào không phù hợp với các tham số của phương thức khởi tạo, chương trình sẽ phát sinh lỗi.

Sau khi tạo đối tượng là thể hiện của một lớp, chúng ta có thể truy xuất đến các thành phần của đối tượng đó các thuộc tính và các phương thức thông qua tên đối tượng theo cú pháp sau:

- Gọi phương thức của lớp ra để thực thi

  <tên_biến_đối_tượng>.<tên phương thức>([danh sách tham số])

- Truy cập đến thuộc tính đối tượng (public)

  <tên_biến_đối_tượng>.<tên thuộc tính>

– Truy cập đến thuộc tính lớp (thuộc tính chia sẻ)

<tên_biến_đối_tượng>.__class__<tên thuộc tính>

Ví dụ 8.6: Tạo đối tượng của lớp học viên sau đó gọi các thành phần của đối tượng có trong lớp học viên.

Vidu8_6.py

```
#Tạo đối tượng hocvien là thể hiện của lớp HocVien
hocvien= HocVien ('111883389', 'Nguyễn Xuân Anh', 2000)
'''Gọi phương thức hiển thị thông tin của lớp HocjVien thông qua đối
tượng hocvien'''
hocvien.hienThi()
```

Kết quả chạy chương trình ***Vidu8_6.py***

```
111883389       Nguyễn Xuân Anh       2000       0
```

8.4. TẠO ỨNG DỤNG

Ví dụ 8.7: Xây dựng lớp hình chữ nhật và tạo đối tượng kiểu lớp hình chữ nhật sau đó gọi các thành phần của đối tượng có trong lớp hình chữ nhật.

Phân tích: Quan sát hình chữ nhật ta thấy hình chữ nhật có dữ liệu thành phần gồm chiều dài, chiều rộng và mầu sắc; hàm thành phần gồm hàm tính diện tích, hàm tính chu vi và hàm hiển thị thông tin về hình chữ nhật ra màn hình. Từ đó ta có thể biểu diễn lớp hình chữ nhật thông qua ngôn ngữ lập trình như sau:

Vidu8_7.py

```
'''
Biểu diễn lớp hình chữ nhật trong máy tính
'''
# Một lớp mô phỏng một hình chữ nhật.
class HinhChuNhat():
    # Một phương thức được sử dụng để tạo đối tượng (Contructor).
```

```
        def __init__(self, chieudai, chieurong, mau):
                self.__cdai = chieudai
                self.__crong = chieurong
                self.__mau = mau
        # Thuộc tính lấy về chiều dài của hình chữ nhật
        def getDai(self):
                return self.__cdai
        #Thuộc tính lấy về chiều rộng của hình chữ nhật
        def getRong(self):
                return self.__crong
        # Phương thức tính diện tích.
        def getDienTich(self):
                dt = self.__cdai * self.__crong
                return dt
        def getMau(self):
                return self.__mau
        # Phương thức tính chu vi.
        def getChuVi(self):
                cv = (self.__cdai + self.__crong)*2
                return cv
        # Phương thức hiển thị thông tin hình chữ nhật
        def hienThi(self):
                print("------------ Thông tin về hình chữ nhật ----------
------")
                print("Chiều dai = ",self.getDai())
                print("Chiều rộng = ", self.getRong())
                print("Mầu = ", self.getMau())
                print("Diện tích hình chữ nhật = ", self.getDienTich())
                print("Chu vi hình chữ nhật = ", self.getChuVi())
#Chương trình chính
#Tạo đối tượng hình chữ nhật 1
hcn1 = HinhChuNhat(40,21,"Đỏ")
hcn1.hienThi()  #Gọi phương thức hiển thị của lớp hình chữ nhật
print("Nhập thông tin hình chữ nhật 2:")
cd = float(input('Chiều dài = '))
cr = float(input('Chiều rộng = '))
mau = input('Mầu sắc = ')
#Tạo đối tượng hình chữ nhật 2
hcn1=HinhChuNhat(cd,cr,mau)
hcn1.hienThi()
```

Kết quả chạy chương trình ***Vidu8_7.py***

```
---------- Thông tin về hình chữ nhật ----------
Chiều dài =  40
Chiều rộng =  21
Mầu =  Đỏ
Diện tích hình chữ nhật =  840
Chu vi hình chữ nhật =  122
Nhập thông tin hình chữ nhật 2:
Chiều dài = 17
Chiều rộng = 9
Mầu sắc = Vàng
---------- Thông tin về hình chữ nhật ----------
Chiều dài =  17.0
Chiều rộng =  9.0
Mầu =  Vàng
Diện tích hình chữ nhật =  153.0
Chu vi hình chữ nhật =  52.0
```

Ví dụ 8.8: Xây dựng lớp môn học để quản lý môn học của các học viên gồm các thông tin: Mã môn học, tên môn học, số tín chỉ và các hàm thành phần giải quyết các công việc chính sau:

- Thêm một môn học.
- Xoá một môn học có mã bằng X được nhập vào từ bàn phím.
- Hiển thị thông tin môn học.
- Tìm kiếm môn học: Theo tên môn học, mã môn học, số tín chỉ.
- Ghi thông tin của các môn học vào tệp monhoc.txt.
- Đọc dữ liệu từ tệp monhoc.txt.

Phân tích:

Để giải quyết các công việc của bài toán đặt ra, đầu tiên chúng ta có thể xây dựng lớp môn học với các dữ liệu thành phần như: Mã môn học, tên môn học và số tín chỉ; hàm thành phần gồm: Hàm khởi tạo, hàm nhập thông tin

môn học, hàm hiển thị thông tin môn học. Sau đó, chúng ta xây dựng tiếp lớp quản lý môn học gồm dữ liệu thành phần là một danh sách các môn học, tên tệp để ghi, đọc thông tin môn học; các hàm thành phần như hàm thêm một môn học vào danh sách, hàm tìm kiếm môn học đã có, hàm ghi, đọc thông tin môn học vào tệp văn bản. Chi tiết các lớp như sau:

Vidu8_8.py

```
'''
Xây dựng lớp môn học để quản lý môn học của các học viên
'''
#Tạo đối tượng hocvien là thể hiện của lớp HocVien
class MonHoc(object):
        #Phương thức khởi tạo với các tham số có giá trị mặc định
        def __init__(self, maMH ='', tenMH='', soTC=0):
                self.maMH = maMH
                self.tenMH = tenMH
                self.soTC = soTC
        #Hiển thị thông tin của môn học ra màn hình
        def hienThiMH(self):
                print(self.maMH, self.tenMH, self.soTC, self.diem)
        #Nhập thông tin của môn học từ bàn phím
        def NhapMH(self):
                while(True):
                        self.maMH=input('Nhập mã môn học: ')
                        if (self.maMH!=""): break;
                while (True):
                        self.tenMH = input('Nhập tên môn học: ')
                        if (self.tenMH != ""): break;
                while (True):
                        self.soTC = int(input('Số tín chỉ: '))
                        if (self.soTC >0 and self.soTC <=4): break;
        #Chuyển dữ liệu từ đối tượng thành xâu
        def toString(self):
```

```
                return '#*'.join([self.maMH, self.tenMH,str(self.soTC)])
class QlMonHoc():
        #Phương thức khởi tạo
    def __init__(self, file_path):
            self.Danh_Sach_Mon_Hoc = list(); #Tạo danh sách trống
            self.file_path =file_path
     #Đọ thông tin các môn học từ file
    def read_MonHoc_from_File(self):
        file = open(self.file_path,'r', encoding='utf-8')
        Lines_monhoc = file.readlines()
        file.close()
        self.Danh_Sach_Mon_Hoc = list();
        for mh in Lines_monhoc:
            tmp=mh.split('#*')
            if (len(tmp)==3):
                mon_hoc = MonHoc(tmp[0],tmp[1],int(tmp[2]))
                self.Danh_Sach_Mon_Hoc.append(mon_hoc)
    #Ghi thông tin các môn học vào file
    def write2File(self):
            file = open(self.file_path, 'w', encoding='utf-8')
            for i in range(len(self.Danh_Sach_Mon_Hoc)-1):
                    file.write(self.Danh_Sach_Mon_Hoc[i].toString())
                    file. write ('\n')
            if (len(self.Danh_Sach_Mon_Hoc)>=1):
               ile.write(self.Danh_Sach_Mon_Hoc[-1].toString())
            file.close()
    #Thêm một môn học mới
    def themMotMonHocMoi(self):
            mh = MonHoc()
            mh.NhapMH()
            self.Danh_Sach_Mon_Hoc.append(mh);
            file = open(self.file_path, 'a', encoding='utf-8')
            file.write('\n'+mh.toString()) #ghi vào cuối file
```

```
            file.close()
    #Hiển thị danh sách môn học ra màn hình
    def hienThiMonHoc(self):
      print('STT    Mã môn học  Tên Môn học    Số tín chỉ  ')
      for (i, mh) in enumerate(self.Danh_Sach_Mon_Hoc):
        print(' '.join([str(i+1),mh.maMH, mh.tenMH, str(mh.soTC)]))
    #Xoá môn học có mã là maMH
    def xoaMonHoc(self,maMH):
            xoa=0;
            for (i, mh) in enumerate(self.Danh_Sach_Mon_Hoc):
                    if (mh.maMH == maMH):
                            self.Danh_Sach_Mon_Hoc.remove(mh)
                            xoa =1
            #Nếu có hành động xoá môn học thì update file
            if (xoa==1):self.write2File()

    #Hiển thị ra màn hình các môn học thoả mãn thông tin tìm kiếm
    def TimKiem_MonHoc(self, ma='', ten='', soTC=0):
        print('STT    Mã môn học  Tên Môn học    Số tín chỉ  ')
        sl=0;
        for (i, mh) in enumerate(self.Danh_Sach_Mon_Hoc):
          if (mh.maMH.lower().__contains__(ma.lower())\
              and mh.tenMH.lower().__contains__(ten.lower()) \
              and (soTC==0 or (soTC >0 and mh.soTC==soTC))):
                 sl=sl+1;
                 print('  '.join([str(sl), mh.maMH, mh.tenMH,
str(mh.soTC)]))
          if (sl==0):
            print("Không có môn học nào thoả mãn yêu cầu tìm kiếm")

#Chương trình chính, tạo các đối tượng và gọi các phương thức đã có
qlmonhoc = QlMonHoc('Monhoc.txt');
qlmonhoc.read_MonHoc_from_File()
```

```
qlmonhoc.themMotMonHocMoi()
print('Danh sách môn học')
qlmonhoc.hienThiMonHoc()
print('Danh sách môn học có mã MH01')
qlmonhoc.TimKiem_MonHoc(ma='mh01')
print('Danh sách môn học có tên Toán')
qlmonhoc.TimKiem_MonHoc(ten ='toán')
print('Danh sách môn học có số tín chỉ =2')
qlmonhoc.TimKiem_MonHoc(soTC=2)
print('Danh sách môn học có số tín chỉ =2 và tên môn là Toán')
qlmonhoc.TimKiem_MonHoc(soTC=2, ten='Toán')
qlmonhoc.xoaMonHoc('mh04')
print('Danh sách môn học sau khi xoá môn học có mã mh04')
qlmonhoc.hienThiMonHoc()
```

Kết quả chạy chương trình ***Vidu8_7.py***

```
Nhập mã môn học: mh04
Nhập tên môn học: Toán cao cấp 4
Số tín chỉ: 2
Danh sách môn học
STT    Mã môn học  Tên Môn học    Số tín chỉ
1  mh03  Toán cao cấp 3  3
2  mh04  Toán cao cấp 4  2
Danh sách môn học có mã MH01
STT    Mã môn học  Tên Môn học    Số tín chỉ
Không có môn học nào thoả mãn yêu cầu tìm kiếm
Danh sách môn học có tên Toán
STT    Mã môn học  Tên Môn học    Số tín chỉ
1  mh03  Toán cao cấp 3  3
2  mh04  Toán cao cấp 4  2
Danh sách môn học có số tín chỉ =2
```

```
STT    Mã môn học  Tên Môn học    Số tín chỉ
1  mh04  Toán cao cấp 4  2
Danh sách môn học có số tín chỉ =2 và tên môn là Toán
STT    Mã môn học  Tên Môn học    Số tín chỉ
1  mh04  Toán cao cấp 4  2
Danh sách môn học sau khi xoá môn học có mã mh04
STT    Mã môn học  Tên Môn học    Số tín chỉ
1  mh03  Toán cao cấp 3  3
```

8.5. TỔNG KẾT BÀI HỌC

Với Python, chúng ta có thể xây dựng chương trình theo hướng thủ tục (Procedural Oriented) hoặc hướng đối tượng (OOP). Hướng thủ tục biểu hiện ở việc người lập trình có thể chia một chương trình lớn thành nhiều chương trình con (các hàm) nhỏ hơn sau đó gọi các chương trình con này mỗi khi cần thiết. Ưu điểm của kỹ thuật này là chương trình ngắn gọn, dễ đọc, các hàm có thể được sử dụng lại trong các chương trình Python khác. Tuy nhiên nhược điểm của kỹ thuật này là hiệu quả sử dụng lại mã chưa cao, khó cho việc bảo trì và nâng cấp phần mềm.

Kỹ thuật lập trình hướng đối tượng ra đời đã phần nào khắc phục được những hạn chế nêu trên, nó giúp cho người lập trình tiến đến gần hơn tới các bài toán trong cuộc sống con người việc quan sát, phân tích đối tượng để từ đó đưa nó vào lập trình.

Bài học này đã cung cấp cho bạn đọc các khái niệm cơ bản về phương pháp lập trình hướng đối tượng, các bước định nghĩa một lớp, tạo các đối tượng và sử dụng đối tượng. Bài học cũng đã cung cấp một số minh họa đơn giản từ đó giúp bạn đọc dần làm quen với phương pháp lập trình này, tuy nhiên, để hiểu sâu hơn nữa và có thể vận dụng thành thạo OOP vào công việc, các bạn cần tìm hiểu và rèn luyện thêm kỹ năng lập trình nhiều hơn nữa.

8.6. BÀI TẬP

A. Câu hỏi ôn tập

Câu 8.1: Cho đoạn mã sau:

```
class MonHoc():
def __init__(self, maMH ='', tenMH='', soTC=0):
        self.maMH = maMH
        self.tenMH = tenMH
        self.soTC = soTC
def hienThiMH(self):
        print(self.maMH, self.tenMH, self.soTC, self.diem)
```

Câu lệnh nào dưới đây sẽ tạo một đối tượng của lớp MonHoc

A. MH = MonHoc()

B. MH = MonHoc(**'mh01'**)

C. MH = MonHoc(**'mh01'**,**'Toán'**,3)

D. Mh = MonHoc(maMH =**'MH01'**, soTC=2, tenMH=**'Môn Tiếng Việt'**)

Câu 8.2: Cho biết phát biểu nào là đúng trong các phát biểu sau

A. Lập trình hướng đối tượng là một kỹ thuật lập trình giải quyết công việc bằng cách biểu diễn bài toán thành các đối tượng ảo

B. Python là một ngôn ngữ lập trình cho phép đa kế thừa

C. Python là 1 ngôn ngữ lập trình chỉ cho phép đơn kế thừa

D. Lớp định nghĩa các thuộc tính và các phương thức chung cho tất cả các đối tượng của cùng một loại nào đó.

Câu 8.3: Cho biết những phát biểu nào đúng trong các phát biểu sau

A. Một đối tượng là một thể hiện (instance) cụ thể của một lớp.

B. Một lớp là một thiết kế hay mẫu cho các đối tượng cùng kiểu.

C. Lớp bao gồm một tập hợp dữ liệu thành phần và các hàm thành phần

D. Lớp có thể được xem như một kiểu dữ liệu để tạo ra các đối tượng

Câu 8.4: Cho biết ý nghĩa của đoạn chương trình sau:

```
def printHello():
    print("Hello! Chào mừng đến với PYTHON")
name = printHello()
```

A. printHello () là một hàm và biến name là một biến. Cả printHello và không là đối tượng.

B. Cả printHello () và biến name tham chiếu đến cùng một đối tượng.

C. printHello () và biến name tham chiếu đến các đối tượng khác nhau.

D. Lỗi cú pháp! Vì không thể gán hàm cho một biến trong Python.

Câu 8.5: Cho biết kết quả hiển thị ra màn hình sau khi máy chạy đoạn chương trình sau:

```
class Obj:
    def printLine(self, line='Python'):
        print(line)

o1 = Obj()
o1.printLine('Java')
```

A. Python

B. Line

C. Java

D. Python

Câu 8.6: Cho biết đâu là định nghĩa đúng cho phương thức khởi tạo đối tượng sinh viên gồm: Tên sinh viên, điểm trung bình, năm sinh. Trong đó, tên mặc định là xâu rỗng, năm sinh có giá trị mặc định là 1900, điểm trung bình mặc định là 0.

A.
```
class SinhVien(object):
        def __init__(self, tenSV, DiemTB, NamSinh):
                self. tenSV = tenSV
                self. DiemTB = DiemTB
                self. NamSinh = NamSinh
```

B.
```
class SinhVien(object):
        def __init__(self, tenSV="", DiemTB=0,
NamSinh=1990):
                self. tenSV = tenSV
                self. DiemTB = DiemTB
                self. NamSinh = NamSinh
```

C.
```
class SinhVien(object):
        def __init__(self, tenSV; DiemTB; NamSinh):
                self. tenSV = tenSV
                self. DiemTB = DiemTB
                self. NamSinh = NamSinh
```

D.
```
class SinhVien(object)
        def __init__(self, tenSV, DiemTB, NamSinh)
                self. tenSV = tenSV
                self. DiemTB = DiemTB
                self. NamSinh = NamSinh
```

Câu 8.7: Cho biết đâu là định nghĩa đúng cho phương thức khởi tạo đối tượng môn học gồm: Mã môn, tên môn, số tín chỉ với giá trị mặc định bằng 2

A.
```
class MonHoc(object)
        def __init__(self, Ma, Ten, soTC=2)
                self. Ma = Ma
                self. Ten = Ten
                    self. soTC = soTC
```

B.
```
class MonHoc(object):
        def __init__(self, Ma, Ten, soTC=2):
                self. Ma = Ma
                self. Ten = Ten
                    self. soTC = soTC
```

C.
```
class MonHoc(object):
        def __init__(self; Ma; Ten; soTC=2):
                self. Ma = Ma
                self. Ten = Ten
                self. soTC = soTC
```

D.
```
class MonHoc(object):
        def __init__(self, Ma, Ten, soTC):
                self. Ma = Ma
                self. Ten = Ten
                    self. soTC = soTC
```

Câu 8.8: Cho biết kết quả hiển thị ra màn hình sau khi máy chạy đoạn chương trình sau:

```
class Point:
    def __init__(self, x=0, y=0):
        self.x = x + 1
        self.y = y + 1

p1 = Point()
print(p1.x, p1.y)
```

A. 0 0

B. 1 1

C. None None

D. x y

Câu 8.9: Cho biết kết quả hiển thị ra màn hình sau khi máy chạy đoạn chương trình sau:

```
class Point:

    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

    def __sub__(self, other):
        x = self.x + other.x
        y = self.y + other.y
        return PoD.int(x, y)
```

```
p1 = Point(3, 4)
p2 = Point(1, 2)
result = p1 - p2
print(result.x, result.y)
```

A. 2 2

B. 4 6

C. 0 0

D. 1 1

Câu 8.10: Thiết kế và định nghĩa lớp cho bài toán sau:

A. Xây dựng lớp PhongHoc gồm các thành phần dữ liệu: *Mã Phòng học, Tên Phòng học, sức chứa.*

B. Xây dựng lớp Linh kiện gồm các thành phần dữ liệu: *Mã Linh kiện, Tên linh kiện, số lượng, đơn giá.*

C. Xây dựng lớp Đánh giá học sinh theo từng năm học gồm: *Điểm trung bình, Hạnh kiểm, Học lực, Năm học.*

D. Xây dựng lớp Hoc_Sinh gồm: Mã học sinh, Họ tên, Đánh giá.

B. Lập trình

Bài tập 8.1: Xây dựng lớp tam giác với dữ liệu thành phần là chiều dài ba cạnh của tam giác, mầu sắc; các hàm thành phần gồm hàm tính chu vi, diện tích tam giác, hàm hiển thị thông tin về tam giác ra màn hình, hàm hiển thị loại của tam giác như: Tam giác cân, vuông, vuông cân, đều hay tam giác thường.

Bài tập 8.2: Xây dựng lớp hình tròn với dữ liệu thành phần là bán kính của hình tròn và các hàm thành phần gồm: Hàm tính chu vi; hàm tính diện tích của hình tròn; hàm hiển thị thông tin về đường tròn đó.

Bài tập 8.3: Xây dựng lớp điểm để biểu diễn các điểm trong không gian hai chiều với dữ liệu thành phần gồm: Tọa độ x, tọa độ y và mầu sắc của điểm; các hàm thành phần gồm:

– Hàm khởi tạo dữ liệu điểm.

– Hàm hiển thị thông tin điểm .

– Hàm *tinhTien*(int x): Tịnh tiến điểm đó theo trục hoành.

– Hàm *tinhTien*(int x, int y): Tịnh tiến trục đó theo cả hai hướng Ox, Oy.

– Hàm *khoangCach*(): Tính khoảng cách của điểm đó so với gốc tọa độ O(0,0).

Hướng dẫn:

Để tịnh tiến theo trục X, ta dịch chuyển toạ độ theo chiều x, chiều y và giữ nguyên toạ độ.

Để tính khoảng cách từ điểm M(x, y) đến tọa độ O(0, 0), ta có thể sử dụng công thức tính khoảng cách giữa hai điểm:

D = math.sqrt(x*x + y*y)

Bài tập 8.4: Xây dựng lớp toán học với dữ liệu thành phần gồm n số; các hàm thành phần gồm:

– *tinhTong*(*nso): Hàm tính tổng n số.

– *tinhTrungBinh*(*nso): Hàm tính trung bình cộng của n số.

– *timMax*(*nso): Hàm tìm số lớn nhất của n số.

– *timMin*(*nso): Hàm tìm số nhỏ nhất của n số.

– *hienThi*(): Hàm hiển thị thông tin về n số ra màn hình.

Bài tập 8.5: Viết chương trình quản lý học viên của một trung tâm AI sử dụng lớp gồm các chức năng sau đây:

– Nhập thông tin của các học viên đăng ký học tại trung tâm AI và lưu vào tệp *DSHV.txt*, biết rằng: Mỗi học viên khi đến trung tâm đăng ký cần điền các thông tin như:

• Số chứng minh nhân dân/ hoặc số thẻ căn cước/hoặc mã số giấy khai sinh.

- Tên học viên.
- Năm sinh học viên.
- Danh sách các môn học học viên đăng ký học tại trung tâm, mỗi môn học gồm các thông tin: Mã môn học, tên môn học, số tiết.

– Hiển thị thông tin của các học viên đã đăng ký tại trung tâm ra màn hình.

– Hiển thị thông tin của các học viên đăng ký ít nhất hai môn học tại trung tâm ra màn hình.

Hướng dẫn:

Định nghĩa lớp môn học gồm các dữ liệu thành phần: Mã môn học, tên môn học, số tiết; và các hàm thành phần: Nhap, hienthi.

Định nghĩa lớp học viên gồm các dữ liệu thành phần: Thẻ căn cước, tên học viên, năm sinh, danh sách môn học; và các hàm thành phần: Hàm nhập thông tin, hàm hiển thị thông tin học viên, chúng ta có thể sử dụng một danh sách để lưu thông tin của các học viên.

TÀI LIỆU THAM KHẢO

1. Rob Miles: *Begin to Code with Python*. Paperback: 528 pages, Publisher: Microsoft Press; 1 edition (December 18, 2017), ISBN-10: 1509304525.
2. Zed A. Shaw: *Learn Python 3 the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code*. Paperback: 320 pages, Publisher: Addison-Wesley Professional; 1 edition (July 7, 2017), ISBN-10: 0134692888.
3. Mark Myers: *A Smarter Way to Learn Python: Learn it faster. Remember it longer*. Paperback: 236 pages, Publisher: CreateSpace Independent Publishing Platform (August 9, 2017), ISBN-10: 1974431479.
4. Mark Lutz: *Learning Python*. Paperback: 1648 pages, Publisher: O'Reilly Media; 5th edition (July 6, 2013), ISBN-10: 1449355730.
5. Dan Bader: *Python Tricks: A Buffet of Awesome Python Features*. Paperback: 302 pages, Publisher: Dan Bader; 1 edition (October 25, 2017), ISBN-10: 1775093301.

Xi nhê

"Success is the ability to go from failure to failure without losing your enthusiasm".

Winston Churchill

ISBN: 978-604-62-7914-3

9 786046 279143

Giá 97.000 VNĐ