



Lê Trọng Nhân - Nguyễn Trần Hữu Nguyên - Võ Tấn Phương
Nguyễn Thanh Hải - Phạm Văn Vinh
Lê Phương Nam - Băng Ngọc Bảo Tâm

Mục lục

I	Các câu lệnh Python cơ bản	7
Chương 1.	Hướng dẫn cài đặt Python và Pycharm	9
1	Giới thiệu	10
2	Cài đặt Python trên Windows	11
3	Giao diện lập trình Python GUI	12
4	Cài đặt PyCharm trên Window	14
5	Hướng dẫn setup và chạy chương trình với PyCharm	18
6	Chương trình đầu tiên trên PyCharm	22
7	Một số thao tác khác trên PyCharm	23
8	Câu hỏi ôn tập	25
Chương 2.	Hiển thị kết quả trên Python	27
1	Giới thiệu	28
2	Hiển thị nhiều thông tin	28
3	Hiển thị với kí tự phân cách	28
4	Hiển thị với kí tự kết thúc	29
5	Hiển thị với số thập phân	29
6	Câu hỏi ôn tập	30
Chương 3.	Nhập dữ liệu và Kiểu dữ liệu	31
1	Giới thiệu	32
2	Câu lệnh nhập dữ liệu	32
3	Kiểu số nguyên trên Python	32
3.1	Khai báo một biến kiểu số nguyên	32
3.2	Nhập số nguyên từ bàn phím	33
3.3	Các phép toán trên số nguyên	33
4	Kiểu số thực trên Python	34
4.1	Khai báo một biến kiểu số thực	34
4.2	Nhập số thực từ bàn phím	34
4.3	Các phép toán trên số thực	34
5	Xử lý lỗi nhập liệu	35
6	Câu hỏi ôn tập	36
Chương 4.	Câu lệnh điều kiện IF	39
1	Giới thiệu	40
2	Câu lệnh if	40

3	Câu lệnh if else	41
4	Câu lệnh if elif else	42
5	Câu lệnh lồng nhau	42
6	Bài tập	43
7	Câu hỏi ôn tập	46
Chương 5. Mảng một chiều - Cấu trúc lặp FOR		49
1	Giới thiệu	50
2	Khai báo và truy xuất mảng một chiều	50
3	Cấu trúc lặp for	50
4	Nhập mảng từ bàn phím	52
5	Bài tập	52
6	Câu hỏi ôn tập	55
Chương 6. Mảng nhiều chiều - FOR lồng nhau		57
1	Giới thiệu	58
2	Thêm một file Python	58
3	Khai báo và truy xuất mảng nhiều chiều	59
4	Duyệt mảng nhiều chiều	61
5	Nhập mảng nhiều chiều từ bàn phím	61
6	Bài tập	62
7	Câu hỏi ôn tập	64
Chương 7. Cấu trúc lặp while		67
1	Giới thiệu	68
2	Cú pháp vòng lặp while	68
2.1	Câu lệnh break và continue trong vòng lặp while	69
2.1.1	Câu lệnh break	69
2.1.2	Câu lệnh continue	69
2.2	Sử dụng while trên một dòng	69
3	Cú pháp vòng lặp while-else	69
4	Bài tập	70
5	Câu hỏi ôn tập	72
Chương 8. Các thao tác trên FILE		75
1	Giới thiệu	76
2	Ghi dữ liệu ra File	76
3	Đọc dữ liệu từ File	77
4	Đọc mảng 1 chiều từ File	78
5	Đọc mảng nhiều chiều từ File	79
6	Bài tập	81
7	Câu hỏi ôn tập	82
Chương 9. Hàm và lời gọi hàm		85
1	Giới thiệu	86
2	Định nghĩa hàm	86
3	Gọi Hàm	86
3.1	Đối Số Mặc Định	87
3.2	Giá Trị Trả Về	87
3.3	Câu lệnh pass	88

4	Viết hàm để tính giai thừa	89
4.1	Viết hàm sử dụng vòng lặp for	89
4.2	Viết hàm sử dụng đệ quy	89
5	Bài tập	90
6	Câu hỏi ôn tập	91
Chương 10. Cấu trúc dữ liệu nâng cao trên Python		95
1	Giới thiệu	96
2	Chuỗi (string)	96
2.1	Nối chuỗi, thay đổi hoặc xóa chuỗi	96
2.2	Phương thức dùng với biến kiểu chuỗi	97
3	Cấu trúc dữ liệu Tuple	97
3.1	Khởi tạo và truy xuất các phần tử trong Tuple	97
3.2	Các thao tác với Tuple	98
3.3	Khi nào sử dụng Tuple	98
4	Cấu trúc dữ liệu tập hợp (Set)	99
4.1	Khởi tạo và truy xuất các phần tử trong tập hợp	99
4.2	Thay đổi tập hợp	99
4.3	Các phép toán trong tập hợp	100
4.3.1	Phép hợp (Union)	100
4.3.2	Phép giao (Intersection)	100
4.3.3	Phép hiệu (Difference)	100
4.3.4	Hiệu đối xứng của hai tập hợp (Symmetric difference)	101
4.4	Khi nào sử dụng tập hợp	101
5	Cấu trúc dữ liệu từ điển (Dictionary)	101
5.1	Khởi tạo và truy xuất các phần tử trong từ điển	101
5.2	Thêm và cập nhật phần tử trong từ điển	102
5.3	Xóa phần tử khỏi từ điển	102
5.4	Một số phương thức hay dùng trong từ điển	103
5.5	Khi nào sử dụng từ điển	103
6	Bài tập	103
7	Câu hỏi ôn tập	104
II Dự án ứng dụng		107
Chương 11. Trợ lý ảo trên Python - Văn bản thành Giọng nói		109
1	Giới thiệu	110
2	Tạo mới một ứng dụng	110
3	Cài đặt thư viện	111
4	Hiện thực chương trình	112
5	Câu hỏi ôn tập	113
Chương 12. Trợ lý ảo trên Python - Nhận diện giọng nói		115
1	Giới thiệu	116
2	Cài đặt thư viện	116
3	Hiện thực chương trình	117
4	Câu hỏi ôn tập	119

Chương 13. Trờ lý ảo trên Python - Xây dựng trí thông minh	121
1 Giới thiệu	122
2 Truy xuất ngày hiện tại	122
3 Hoàn thiện chương trình	122
4 Câu hỏi ôn tập	124
Chương 14. Lập trình giao diện trên Python	125
1 Giới thiệu	126
2 Thiết kế giao diện	126
3 Thêm đối tượng giao diện vào cửa sổ	127
4 Câu hỏi ôn tập	130
Chương 15. Tạo hàm xử lý cho nút nhấn	131
1 Giới thiệu	132
2 Khai báo hàm xử lý	132
3 Hiện thực hàm xử lý cho nút nhấn	133
4 Bài tập	133
5 Câu hỏi ôn tập	134
Chương 16. Xử lý lỗi trên giao diện	135
1 Giới thiệu	136
2 Xử lý lỗi nhập liệu	136
3 Câu hỏi ôn tập	138

Phần I

Các câu lệnh Python cơ bản

CHƯƠNG 1



Hướng dẫn cài đặt Python và Pycharm

pythonTM
Package
Index

1 Giới thiệu

Python là một ngôn ngữ lập trình cấp cao và dễ tiếp cận hơn đối với những người mới bắt đầu học ngôn ngữ lập trình. Nếu so sánh Python với các ngôn ngữ truyền thống như Pascal hay C thì mức độ phức tạp của Python là thấp hơn. Một số ưu điểm của ngôn ngữ lập trình Python có thể kể ra như sau:

- Python có cú pháp rất đơn giản. Nó dễ đọc và viết hơn rất nhiều khi so sánh với những ngôn ngữ lập trình khác như Pascal hay C. Mặc dù đôi lúc sự đơn giản này có thể gây ra một số phiền phức trong việc quản lý chương trình. Tuy nhiên Python làm cho việc lập trình trở đơn giản hơn, đặc biệt là các tác vụ nhập xuất (input và output), cho phép bạn tập trung vào những giải pháp chứ không phải cú pháp của ngôn ngữ lập trình. Một cách so sánh trừu tượng, Python tạo điều kiện cho bạn "giao tiếp ngôn ngữ", hơn là việc phải "học ngôn ngữ" một cách truyền thống.
- Mã nguồn mở rộng rãi: Đây là yếu tố hết sức quan trọng dành cho người mới bắt đầu. Vì là mã nguồn mở, 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. Python có một cộng đồng rộng lớn, không ngừng cải thiện nó mỗi lần cập nhật. Rất nhiều mã nguồn mở liên quan đến trí tuệ nhân tạo hay nhận diện giọng nói đang có sẵn trên Python, để người dùng có thể phát triển những ứng dụng rất cao cấp.
- Tương thích đa nền tảng: Rất nhiều hệ thống, từ máy tính cho đến các bo mạch nhúng, thậm chí là các bo mạch có tài nguyên thấp như MicroBit, cũng hỗ trợ ngôn ngữ lập trình Python. Do đó, bạn có thể hình dung rằng, phần mềm có thể chỉ cần phát triển một lần là có thể sử dụng cho rất nhiều nền tảng khác nhau. Chỉ với những thay đổi rất nhỏ, một phần mềm đang chạy trên máy tính cũng có thể chạy trên một bo mạch mini.

Như vậy, có thể thấy rằng, ngôn ngữ lập trình Python là một khái niệm rất rộng. Tùy vào việc phát triển ứng dụng trên nền tảng nào, mà phần mềm đi kèm phù hợp sẽ được sử dụng, như minh họa ở Hình 1.1. Phần mềm ở đây không chỉ đóng vai trò là chương trình để chúng ta soạn thảo chương trình (hay còn gọi là lập trình), mà nó còn đóng vai trò biên dịch từ ngôn ngữ lập trình sang ngôn ngữ thực thi.

Như minh họa ở Hình 1.1, khi lập trình Python trên máy tính, chúng ta sẽ cần 1 chương trình dịch từ ngôn ngữ Python sang ngôn ngữ mà máy tính có thể hiểu được. Và chương trình này, thông dụng nhất hiện tại là Python3. Tuy nhiên, nếu chúng ta muốn lập trình bằng ngôn ngữ Python nhưng cho mạch MicroBit, chúng ta sẽ cần một chương trình khác, chẳng hạn như là Mu. Chương trình này sẽ dịch từ ngôn ngữ Python sang ngôn ngữ mà mạch MicroBit có thể hiểu được. Trong hình Hình 1.1, chúng tôi gọi Máy tính hoặc mạch MicroBit là nền tảng thực thi.

Cuối cùng, là phương thức dịch từ ngôn ngữ Python sang ngôn ngữ mà nền tảng thực thi có thể hiểu được, gọi là thông dịch (interpreter). Như một thông dịch viên, phần mềm sẽ dịch câu lệnh python đầu tiên, rồi thực thi nó. Sau đó, lại dịch tiếp câu lệnh thứ 2, và thực thi cho đến khi kết thúc chương trình. Đây là điểm vô cùng khác biệt so với ngôn ngữ Pascal, trình biên dịch sẽ dịch hết chương trình rồi mới



Hình 1.1: Ngôn ngữ lập trình và Trình biên dịch

thực thi. Cũng vì lý do này, mà Python tương thích với nhiều nền tảng thực thi, vì bản chất chương trình Python chỉ là một file văn bản. Chỉ khi thực thi, nó mới được dịch sang ngôn ngữ mà nền tảng phần cứng hiểu được.

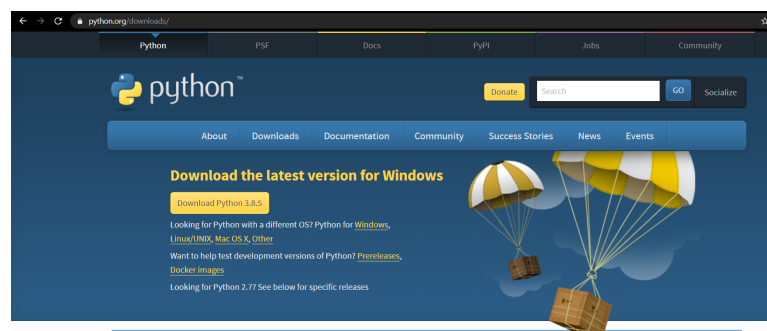
Hướng dẫn bên dưới sẽ hướng dẫn chi tiết các bước cài đặt trình thông dịch Python3. Thực ra, sau khi cài đặt trình thông dịch này, chúng ta đã có một giao diện để lập trình bằng ngôn ngữ Python. Tuy nhiên, giao diện này khá nghèo nàn, nên chúng tôi khuyến khích bạn cài thêm PyCharm, một phần mềm rất được cộng đồng Python khuyến dùng.

2 Cài đặt Python trên Windows

- Bước 1: Bạn tải Python phiên bản 3.8.5 từ đường dẫn sau đây:

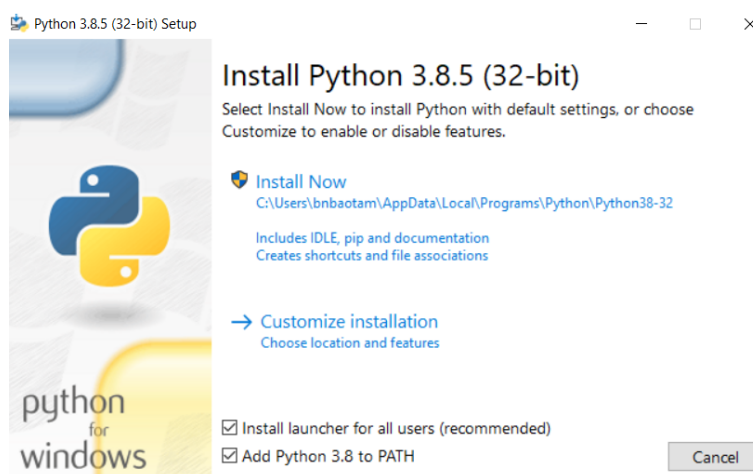
https://ubc.sgp1.cdn.digitaloceanspaces.com/DARIU/python_3_8_5.exe

Trong trường hợp muốn cập nhật phiên bản mới hơn, bạn có thể truy cập vào trang chủ của Python là <http://www.python.org/>, chọn tiếp vào mục **Downloads** để lựa chọn phiên bản Python muốn cài. **Tuy nhiên chúng tôi khuyến khích bạn xài phiên bản 3.8.5 trong hướng dẫn này.**



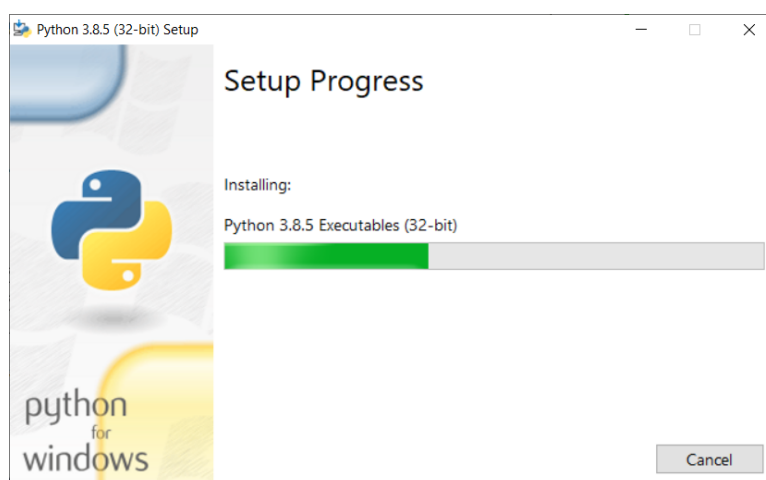
Hình 1.2: Tải phần mềm Python từ trang chủ <http://www.python.org/>

- Bước 2: Khi tải xong, bạn tiến hành chạy file cài đặt có đuôi .exe vừa tải ở bước 1. Nhấp vào Install Now để cài đặt. **Bạn cần phải chọn vào 2 tùy chọn cuối cùng** (Install for all user và Add Python to PATH), như minh họa ở hình bên dưới.



Hình 1.3: Chọn cả 2 tùy chọn, trước khi nhấn vào Install Now để cài đặt

- Bước 3: Bạn có thể nhìn thấy tiến trình cài đặt Python lên máy.



Hình 1.4: Đợi chương trình cài đặt Python

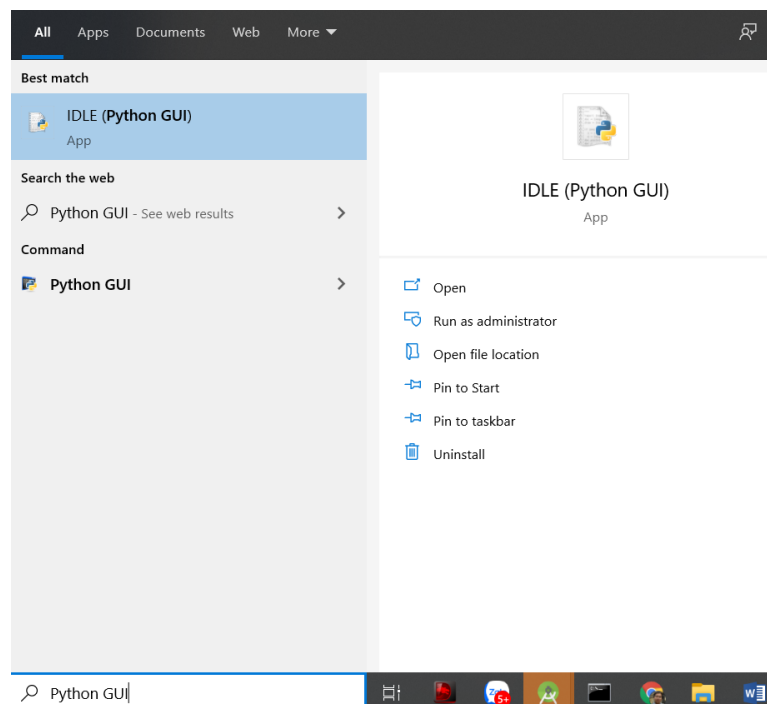
- Bước 4: Khi cài đặt hoàn tất, bạn sẽ nhìn thấy một màn hình thông báo như dưới đây. Khi đó hãy click “Close” (ở góc dưới bên phải) để hoàn tất cài đặt.

3 Giao diện lập trình Python GUI

Ngay sau khi cài đặt xong Python, bạn đã có thể bắt đầu lập trình. Từ nút khởi động Windows, bạn tìm kiếm với từ khóa Python GUI là thấy ứng dụng này, như minh họa ở Hình 14.1.

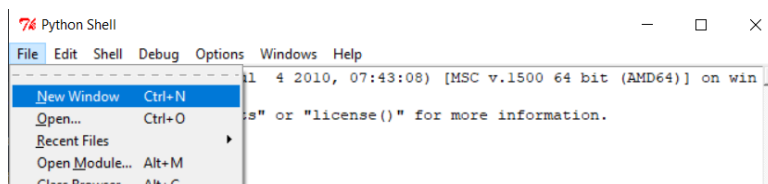


Hình 1.5: Quá trình cài đặt Python thành công



Hình 1.6: Chương trình Python GUI

Giao diện Python Shell hiện ra, từ menu File, bạn chọn tiếp New Window, như minh họa ở Hình 14.3.

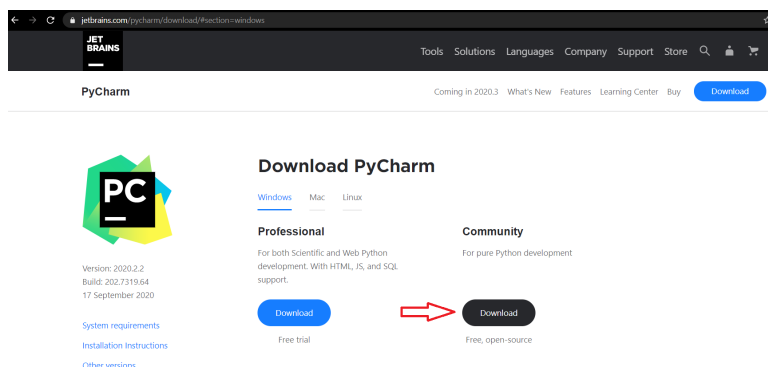


Hình 1.7: Mở một cửa sổ để lập trình trên Python IDE

Chương trình đầu tiên rất đơn giản, chỉ có 1 câu lệnh là **print("Xin chào")**. Để chạy chương trình, bạn chọn vào Run, chọn tiếp Run mode hoặc nhấn F5. Kết quả của chương trình sẽ được hiển thị trên Python Shell. Rõ ràng, giao diện lập trình này khá nghèo nàn và không có nhiều hỗ trợ khi viết câu lệnh, điều mà bạn sẽ thấy rõ ràng trên phần mềm PyCharm. Tuy nhiên phần mềm này cũng có thể dùng tạm khi bạn chưa cài đặt PyCharm thành công. Trong phần hướng dẫn tiếp theo, chúng tôi sẽ hướng dẫn bạn cách cài đặt và viết chương trình trên PyCharm.

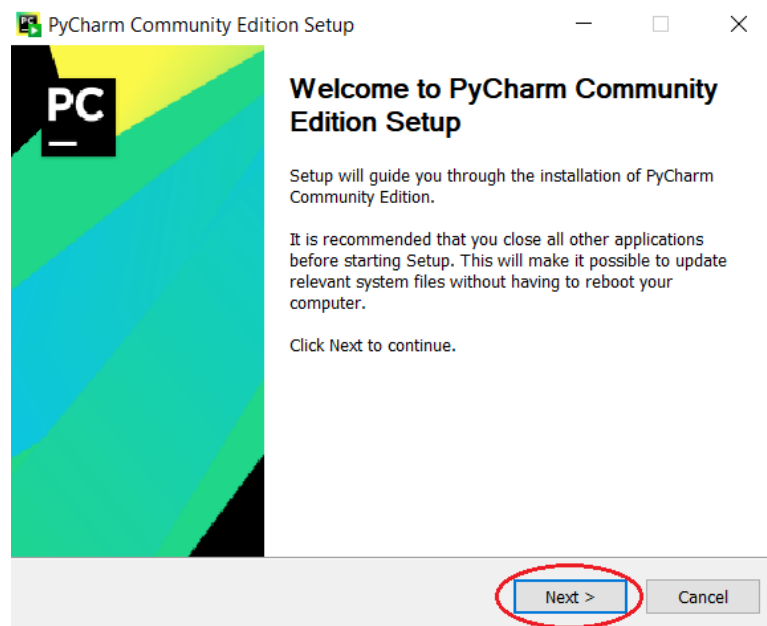
4 Cài đặt PyCharm trên Window

- Bước 1: Từ trang chủ www.jetbrains.com/pycharm/download/, nhấn vào nút “Download”, và chọn phiên bản miễn phí **Community**, như hướng dẫn ở hình bên dưới.



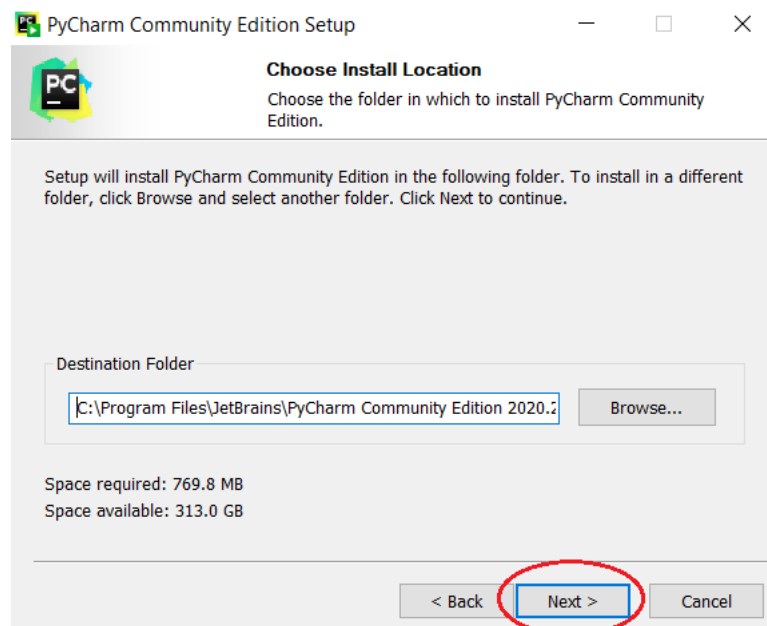
Hình 1.8: Tải Pycharm từ trang chủ

- Bước 2: Khi tải hoàn tất, tiến hành chạy để cài đặt PyCharm. Một cửa sổ cài đặt sẽ hiện ra, hãy nhấp vào “Next”.



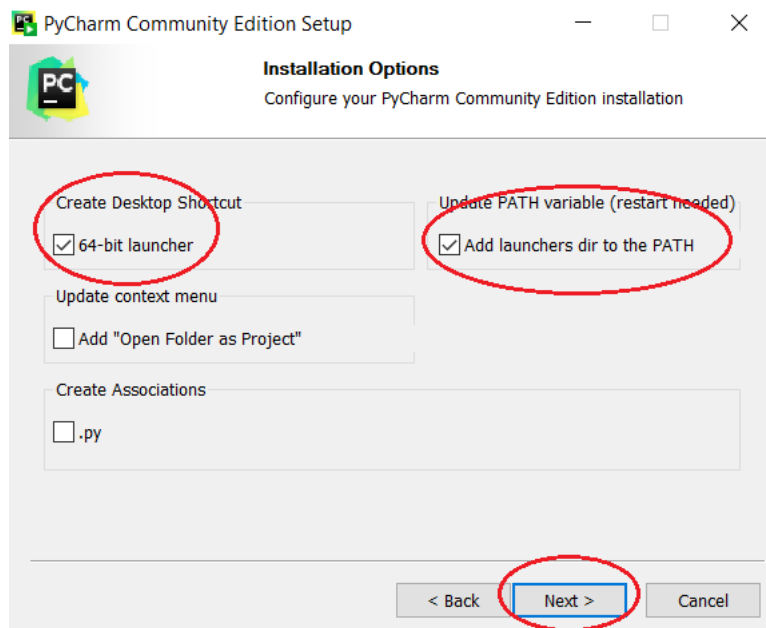
Hình 1.9: Ấn "Next" để tiếp tục

- Bước 3: Ở màn hình tiếp theo, bạn có thể thay đổi đường dẫn để cài đặt PyCharm hoặc để mặc định. Sau đó nhấp vào “Next”.



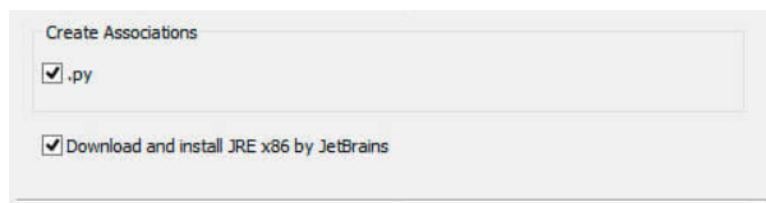
Hình 1.10: Ấn "Next" để tiếp tục

- Bước 4: Bạn sẽ thấy một màn hình có một vài lựa chọn, bạn chọn vào các ô như hình (chọn tương tự nếu như máy bạn chỉ hiện 32-bit thay vì 64-bit). Sau đó ấn vào “Next”.



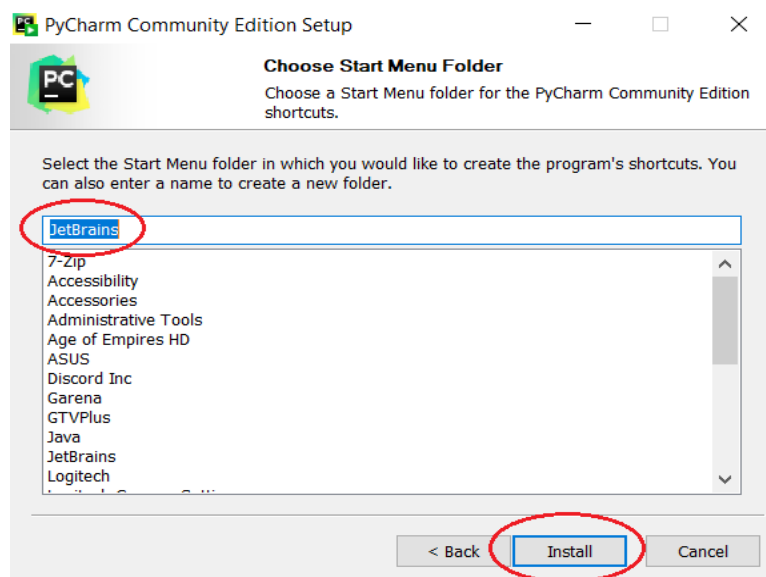
Hình 1.11: Ấn "Next" để tiếp tục

- Bên cạnh đó, nếu như trong bước này, trong khung cài đặt bạn có thêm lựa chọn như hình bên dưới. Điều đó có nghĩa là bạn chưa cài thư viện Java, bạn hãy chọn để cài đặt thư viện Java tự động nhé.



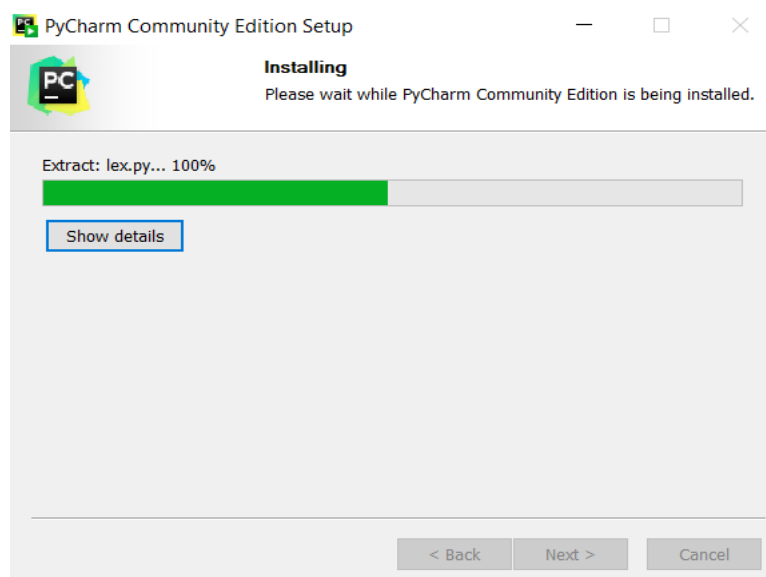
Hình 1.12: Đánh dấu cho 2 lựa chọn

- Bước 5: Chọn thư mục trong start menu của Windows. Hãy để mặc định là thư mục JetBrains vào nhấp vào “Install”.



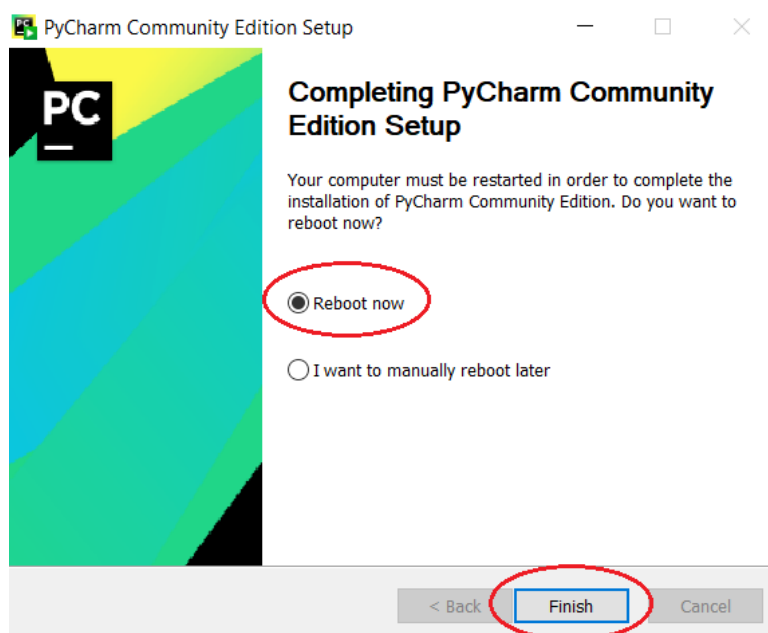
Hình 1.13: Nhấn vào "Install" để tiến hành cài đặt

- Bước 6: Chờ quá trình cài đặt hoàn tất.



Hình 1.14: Đợi chương trình cài đặt Pycharm

- Bước 7: Khi cài đặt hoàn thành, bạn sẽ nhìn thấy màn hình thông báo như dưới đây. PyCharm hỏi ta có muốn khởi động lại máy luôn hay không. Ta có thể chọn RebootNow để khởi động lại máy tính nhằm hoàn tất quá trình cài đặt.

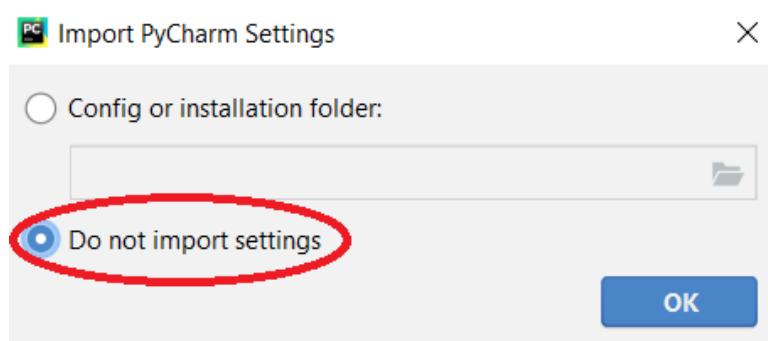


Hình 1.15: Ấn "Finish" để hoàn thành cài đặt

5 Hướng dẫn setup và chạy chương trình với PyCharm

Như vậy là về cơ bản chúng ta đã cài đặt xong PyCharm. Nhưng để biết bạn đã cài đúng chưa thì chúng ta cần phải thử xem chương trình python của bạn có chạy được với PyCharm vừa cài không.

- Bước 1: Mở chương trình PyCharm vừa cài đặt thành công trên máy tính của bạn.
- Bước 2: Ta sẽ được hỏi "Có muốn Import các thiết lập đã có từ trước hay không?". Nếu cài mới hoàn toàn, ta chọn mục Do not import settings, rồi nhấn OK, như hướng dẫn ở Hình 1.16



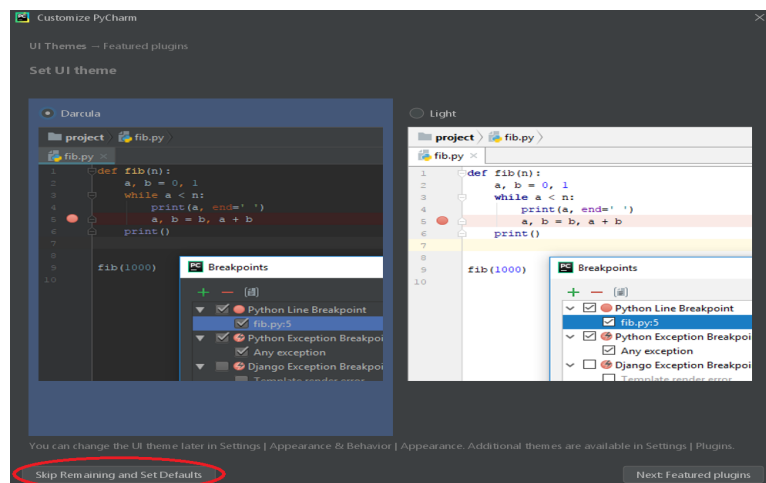
Hình 1.16: Không import những setting cũ khi cài mới

Trong phần chính sách bảo mật (nếu có), ta nhấn xác nhận và nhấn Continue để tiếp tục.



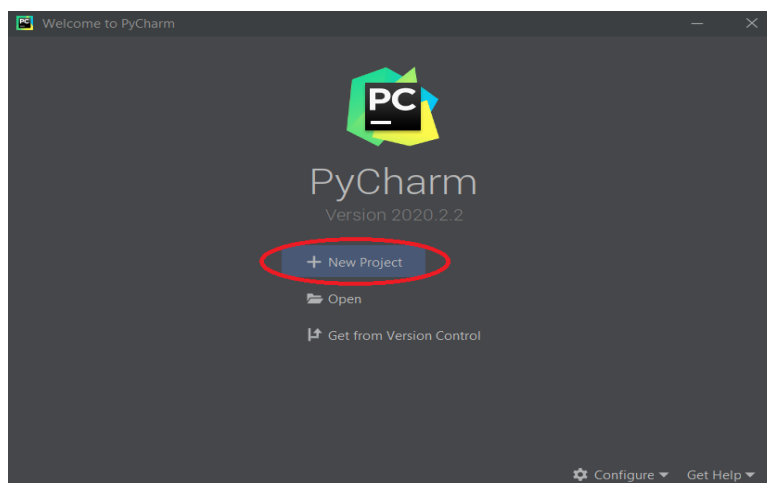
Hình 1.17: Đồng ý với cách chính sách của PyCharm

- Bước 3: Trong màn hình tùy biến PyCharm, ta chọn Skip Remaining and Set Defaults để lựa chọn các thiết lập mặc định.



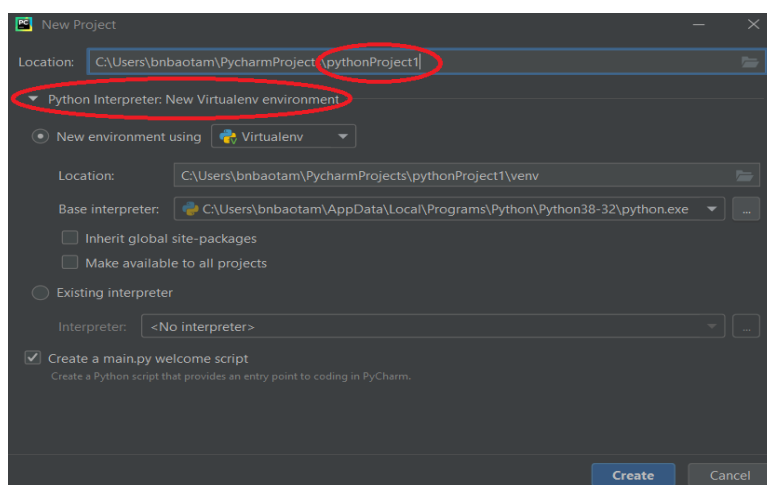
Hình 1.18: Chọn cấu hình mặc định cho phần mềm PyCharm

- Bước 4: Sau đó là màn hình chào hỏi của PyCharm, ta chọn mục Create New Project để tạo một Project mới.



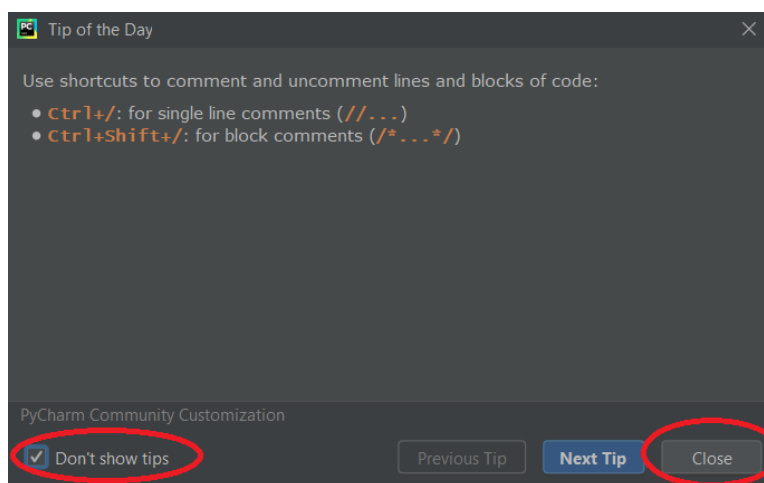
Hình 1.19: Tạo mới một dự án bằng cách nhấn vào New Project

- Bước 5: Ta chọn thư mục để lưu project tại mục **Location**. Đồng thời, bạn kiểm tra các thông tin môi trường bên dưới, xem có giống với hình hướng dẫn hay không. Thông thường các giá trị mặc định sẽ chính xác và chúng ta không phải chỉnh sửa gì cả. Cuối cùng, chúng ta nhấn vào nút **Create**.



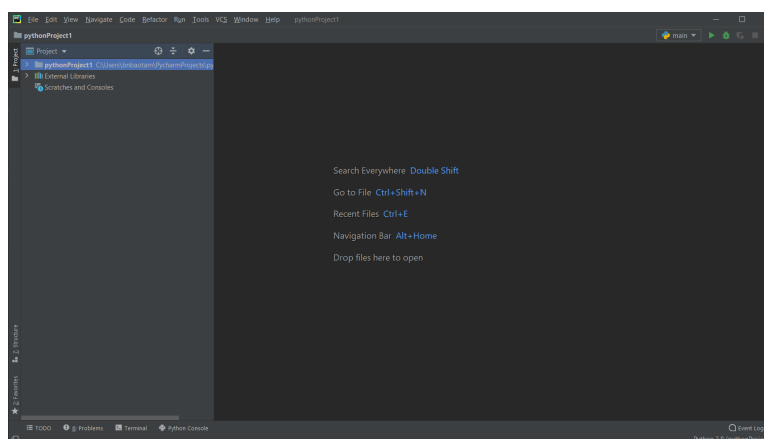
Hình 1.20: Chọn đường dẫn cho dự án và kiểm tra thông tin môi trường

- Bước 6: Sau khi quá trình trên được hoàn tất, hộp thoại hướng dẫn như hình bên dưới có thể sẽ xuất hiện, và nếu bạn không thích nó hiện ra nữa. Bạn có thể chọn vào ô không hiển thị các mẹo ở góc dưới bên trái và chọn close để thoát.



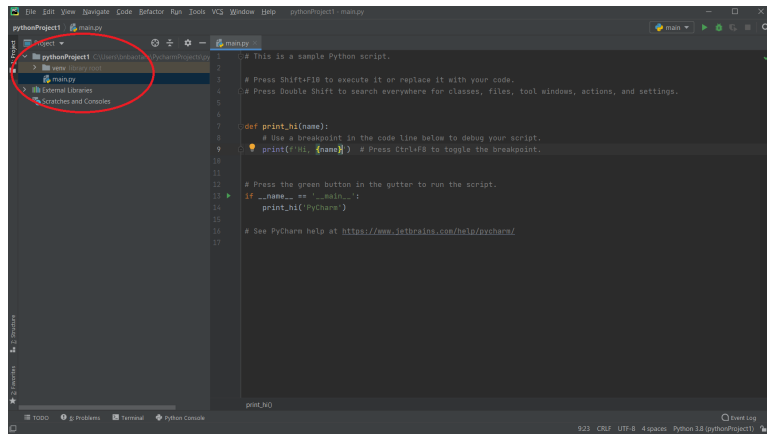
Hình 1.21: Tắt hộp thoại hướng dẫn các mẹo lập trình

Project mới sẽ được tạo ra trên PyCharm như hình bên dưới.

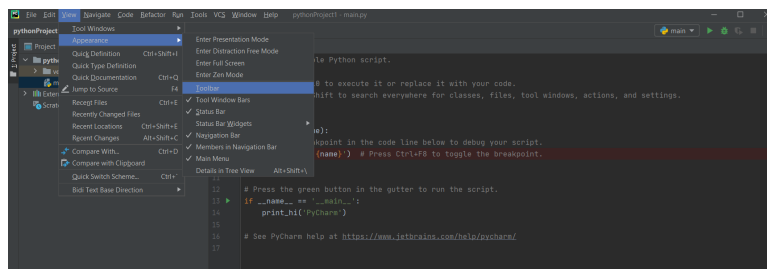


Hình 1.22: Dự án mới được tạo ra thành công trên PyCharm

- Bước 7: Để xem nội dung đoạn code được tạo sẵn khi ta tạo project ta bấm chọn vào phần project (vùng khoanh tròn màu đỏ) chúng ta vừa tạo để xem tất cả các files trong project và chọn vào file **main.py**. Sau đó chúng ta được kết quả như hình bên dưới.
- Bước 8: Để hiển thị thêm các nút bấm hữu ích như chạy chương trình, chúng ta vào phần View -> Appearance -> chọn vào Toolbar.

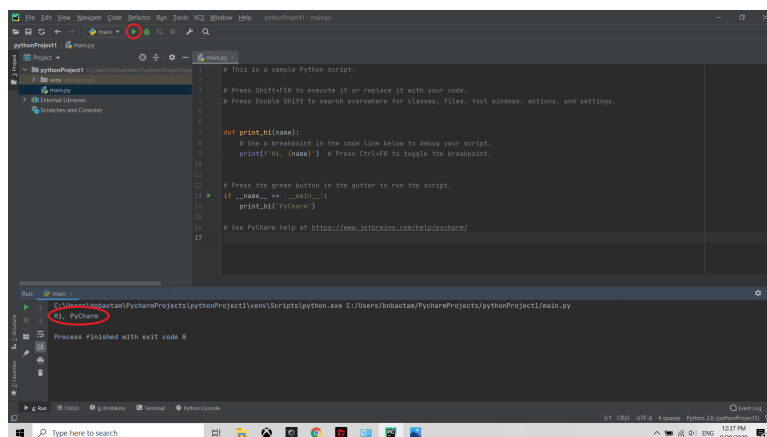


Hình 1.23: Bấm chọn phần Project trong vùng khoanh màu đỏ



Hình 1.24: Hiện thị thanh công cụ Toolbar trên PyCharm

- Bước 9: Sau khi đã hiển thị thanh công cụ, chúng ta chọn vào dấu mũi tên màu xanh. Sau đó chương trình sẽ được thực thi và kết quả sẽ được hiện thị phía bên dưới như hình.



Hình 1.25: Nhấn nút Run để chạy thử chương trình mặc định trên PyCharm

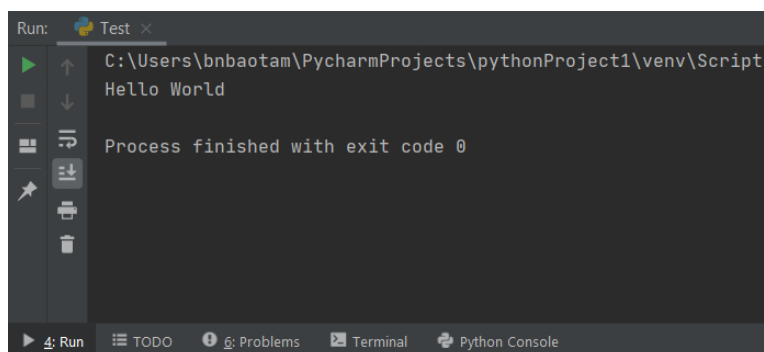
Như vậy là chúng ta đã cài đặt xong và chạy thử thành công một chương trình trên PyCharm.

6 Chương trình đầu tiên trên PyCharm

Khi tạo mới một dự án (project) trên PyCharm, phần mềm tự tạo cho chúng ta file mặc định là **main.py** cùng với một số đoạn chương trình mẫu. Chúng ta có thể

nhấp đôi vào file này, xóa hết nội dung và bắt đầu bằng chương trình của mình.

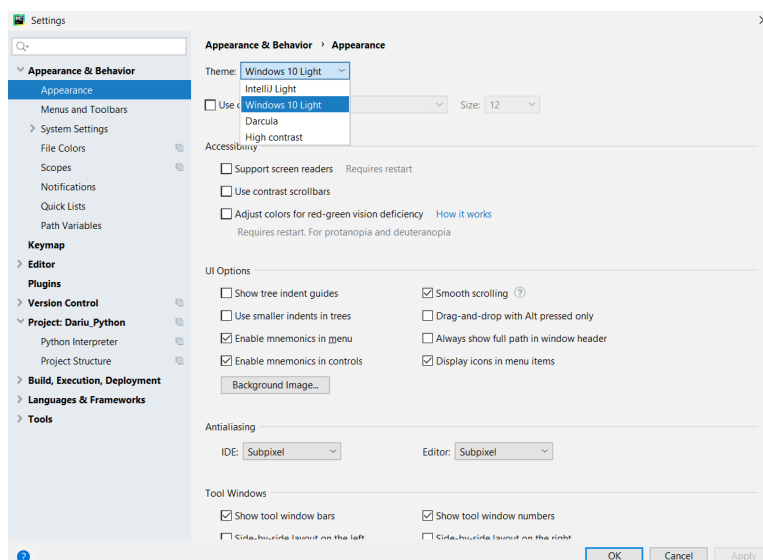
Ở chương trình đầu tiên này, chúng ta cũng chỉ đơn giản in ra màn hình một câu. Chương trình của chúng ta chỉ có một lệnh là **print("Hello World")**. Và đây là kết quả sau khi chạy chương trình:



Hình 1.26: Kết quả in ra màn hình cho chương trình đầu tiên trên PyCharm

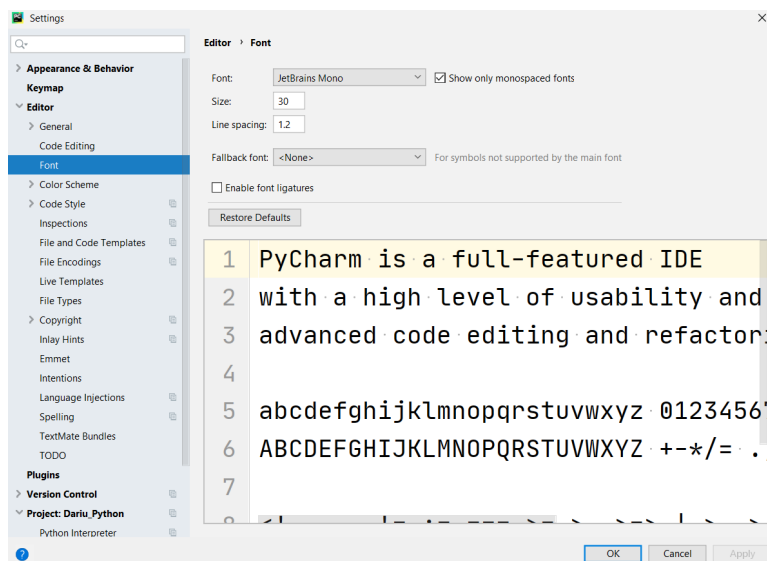
7 Một số thao tác khác trên PyCharm

Tông màu mặc định của PyCharm là màu tối (nền đen chữ trắng). Trong trường hợp bạn muốn đổi tông màu hiển thị, có thể vào File, Settings. Với cửa sổ mới hiện ra, bạn chọn tiếp vào Appearance và lựa chọn tông màu ưa thích, như hướng dẫn ở Hình 1.27



Hình 1.27: Thay đổi tông màu hiển thị trên PyCharm

Trong trường hợp muốn thay đổi kích thước của trình soạn thảo, để việc lập trình được dễ nhìn, từ giao diện Settings ở Hình 1.27, chúng ta chọn tiếp qua Editor, và chọn Font và thay đổi thông số Size của nó, như minh họa ở Hình 1.28 là 30.



Hình 1.28: Thay đổi kích thước chữ khi lập trình

8 Câu hỏi ôn tập

1. Phát biểu nào là đúng về ngôn ngữ Python?
 - A. Ngôn ngữ lập trình cấp thấp
 - B. Ngôn ngữ lập trình cấp cao
 - C. Ngôn ngữ máy
 - D. Ngôn ngữ MicroBit
2. Các lợi thế của ngôn ngữ Python so với Pascal là gì?
 - A. Đơn giản dễ tiếp cận
 - B. Tương thích với nhiều nền tảng
 - C. Mã nguồn mở với nhiều thư viện hỗ trợ
 - D. Tất cả đều đúng
3. Để chuyển từ ngôn ngữ Python sang ngôn ngữ mà máy tính hiểu được, chúng ta cần:
 - A. Trình biên dịch Python3
 - B. Trình thông dịch Python3
 - C. Cả 2 chương trình trên
 - D. Tất cả đều sai
4. Đặc điểm nào là đúng cho trình thông dịch?
 - A. Dịch từng lệnh và thực thi lệnh đó
 - B. Dịch toàn bộ chương trình và thực thi
 - C. Tùy trường hợp mà 1 trong 2 phương pháp trên sẽ được chọn lựa
 - D. Tất cả đều đúng
5. Đặc điểm nào là đúng cho trình biên dịch?
 - A. Dịch từng lệnh và thực thi lệnh đó
 - B. Dịch toàn bộ chương trình và thực thi
 - C. Tùy trường hợp mà 1 trong 2 phương pháp trên sẽ được chọn lựa
 - D. Tất cả đều đúng
6. Ngôn ngữ lập trình Python là:
 - A. Ngôn ngữ biên dịch
 - B. Ngôn ngữ thông dịch
 - C. Tất cả đều đúng
 - D. Tất cả đều sai
7. Ngôn ngữ lập trình Pascal là:
 - A. Ngôn ngữ biên dịch
 - B. Ngôn ngữ thông dịch
 - C. Tất cả đều đúng
 - D. Tất cả đều sai

Đáp án

1. B 2. D 3. B 4. A 5. B 6. B 7. A

CHƯƠNG 2



Hiển thị kết quả trên Python

pythonTM
Package
Index

1 Giới thiệu

Để tìm hiểu một ngôn ngữ lập trình nói chung và python nói riêng, thông thường chúng ta sẽ bắt đầu bằng câu lệnh hiển thị để in kết quả ra màn hình. Thực ra, trong một hệ thống máy tính với 3 thành phần đặc trưng là Dữ liệu đầu vào - Xử lý - Kết quả đầu ra, thì bước cuối cùng lại là bước đơn giản nhất, tiếp theo là phần dữ liệu đầu vào và cuối cùng sẽ là xử lý. Cũng vì lý do đó, mà chúng ta thường tìm hiểu câu lệnh hiển thị trước, câu lệnh đơn giản nhất thuộc nhóm xuất kết quả đầu ra (Output).

Ở bài trước, thực ra chúng ta đã xài câu lệnh hiển thị **print** để in chữ "Xin chào" ra màn hình. Trong bài này, chúng tôi sẽ trình bày chi tiết hơn các tùy chọn của câu lệnh print.

Các câu lệnh trong hướng dẫn bên dưới sẽ được nhập trực tiếp vào file mặc định **main.py**. Sau đó, khi chạy chương trình, kết quả sẽ được in ra ở khung cửa sổ ngay bên dưới chương trình.

2 Hiển thị nhiều thông tin

Bên cạnh việc in ra 1 chuỗi như bài trước, hàm print có thể in nhiều thông tin, mỗi thông tin cách nhau bằng **dấu phẩy** (,), như ví dụ ở chương trình sau đây:

```
1 print("Xin chào", 10, 12.1, "Ket thuc")
```

Chương trình 2.1: In nhiều thông tin

Bên cạnh chuỗi kí tự, hàm print cũng có thể in số nguyên lẫn số thập phân. Python sẽ tự động **thêm dấu khoảng trắng** vào giữa các thông tin cần in. Một biến số cũng được xem là một thông tin, và hoàn toàn có thể ghép vào câu lệnh print theo cách như trên. Một ví dụ như sau:

```
1 x = 20
2 print("Xin chào", 10, 12.1, "Ket thuc", x)
```

Chương trình 2.2: In ra giá trị của biến số

Ở ví dụ trên, *x* là một biến số. Tuy nhiên trên Python, việc khai báo một biến số đã được lược giản đi rất nhiều. Thậm chí, chúng ta không cần phải chỉ định kiểu dữ liệu cho biến số. Chi tiết của việc khai báo biến và kiểu dữ liệu sẽ được trình bày cụ thể ở những bài sau.

3 Hiển thị với kí tự phân cách

Thay vì mỗi thông tin in ra được cách nhau bằng khoảng trắng như ở trên, chúng ta cũng có thể tùy chỉnh bằng chuỗi kí tự khác, dựa vào tùy chọn **sep** như ví dụ bên dưới:

```
1 print("Xin chào", "Ket thuc", sep="---")
```

Chương trình 2.3: In kết quả với kí tự phân cách

4 Hiện thị với kí tự kết thúc

Cuối cùng, chúng ta cũng có thể thêm tùy chọn **end** để định nghĩa kí tự kết thúc của chuỗi cần in, như ví dụ sau:

```
1 print("Xin chào", "Ket thuc", sep="---", end="!!")
```

Chương trình 2.4: In kết quả với tùy chọn end

Tất nhiên, vì là tùy chọn, sep và end có thể có hoặc không, hoặc thậm chí xuất hiện cùng nhau trong câu lệnh print. Yêu cầu duy nhất là các tùy chọn này phải cách nhau bằng dấu phẩy và **không có ràng buộc về thứ tự**. Chương trình ?? cũng sẽ in ra kết quả tương tự như Chương trình ??

```
1 print("Xin chào", "Ket thuc", end="!!", sep="---")
```

Chương trình 2.5: Các tùy chọn không cần có thứ tự trước sau

5 Hiện thị với số thập phân

Một trong những thao tác quan trọng khi tính toán, kết quả có thể là số thập phân với rất nhiều kí số sau dấu phẩy. Trong một số trường hợp, chúng ta chỉ cần in 2 kí số sau dấu phẩy mà thôi. Câu lệnh print hỗ trợ trong trường hợp này sẽ như ví dụ bên dưới:

```
1 a = 3.14165
2 print("Ket qua la %.2f" %a)
```

Chương trình 2.6: Tùy chỉnh số kí số sau dấu thập phân

Ở đây, `%.2f` là định dạng của dữ liệu cần in. Kết quả in ra màn hình sẽ chỉ lấy 2 chữ số thập phân của biến `a` mà thôi. Một lưu ý quan trọng là **kết quả sẽ được làm tròn**.

6 Câu hỏi ôn tập

1. Câu lệnh để in kết quả ra màn hình trên Python là gì?
A. print
B. Log
C. Out
D. Tất cả đều đúng
2. Kết quả in ra màn hình của lệnh **print("Xin chào", "123")** là gì?
A. Xin chào
B. 123
C. Xin chào 123
D. Lệnh sai cú pháp
3. Kết quả in ra màn hình của lệnh **print("Xin chào", "123", sep="*")** là gì?
A. Xin chào 123 *
B. Xin chào * 123
C. Xin chào * 123 *
D. Lệnh sai cú pháp
4. Kết quả in ra màn hình của lệnh **print("Xin chào", "123", end="*")** là gì?
A. Xin chào 123 *
B. Xin chào * 123
C. Xin chào * 123 *
D. Lệnh sai cú pháp
5. Kết quả in ra màn hình của lệnh **print("Xin chào", "123", sep = "*", end="*")** là gì?
A. Xin chào 123 *
B. Xin chào * 123
C. Xin chào * 123 *
D. Lệnh sai cú pháp
6. Kết quả in ra màn hình của lệnh **print("Xin chào", "123", end = "*", sep="*")** là gì?
A. Xin chào 123 *
B. Xin chào * 123
C. Xin chào * 123 *
D. Lệnh sai cú pháp
7. Kết quả in ra màn hình của lệnh **print("Ví dụ %.2f", % 3.14999)** là gì?
A. Ví dụ 3.14
B. Ví dụ 3.15
C. Ví dụ 3.14999
D. Lệnh sai cú pháp

Đáp án

1. A 2. C 3. B 4. A 5. C 6. C 7. B

CHƯƠNG 3



Nhập dữ liệu và Kiểu dữ liệu

pythonTM
Package
Index

1 Giới thiệu

Trước khi bắt đầu vào việc xử lý giải thuật, chúng ta phải thực hiện qua bước nhập dữ liệu đầu vào cho bài toán. Mặc dù là quy trình đầu tiên của một hệ thống, độ phức tạp của nó lại cao hơn nhiều so với việc xuất dữ liệu. Một trong những khó khăn cho quy trình này, là người dùng có thể nhập sai, hoặc không tuân theo cú pháp dữ liệu của chương trình. Điều này có thể dẫn tới việc xử lý dữ liệu bị lỗi, nặng hơn có thể làm thoát chương trình của chúng ta. Vì vậy, khi làm các chương trình lớn, luôn luôn đi kèm với quá trình nhập liệu, là quá trình xử lý lỗi dữ liệu đầu vào.

Trong hướng dẫn này, chúng tôi tập trung hướng dẫn việc nhập dữ liệu từ bàn phím, và chỉ tập trung vào dữ liệu số, cũng như các phép toán cho dữ liệu số. Một kĩ thuật đơn giản nhưng hiệu quả để xử lý lỗi dữ liệu đầu vào cũng sẽ được giới thiệu. Các cấu trúc dữ liệu phức tạp hơn sẽ được trình bày ở những bài sau.

2 Câu lệnh nhập dữ liệu

Để khởi tạo giá trị ban đầu cho biến, ta có thể dùng lệnh gán để gán một giá trị cho biến. Những chương trình đưa dữ liệu vào cho phép đưa dữ liệu từ bàn phím hoặc từ đĩa vào gán cho các biến, làm cho chương trình trở nên linh hoạt, có thể tính toán với nhiều bộ dữ liệu đầu vào khác nhau.

Câu lệnh nhận dữ liệu từ bàn phím là **input**, có cú pháp như sau:

`<tên biến> = input(<Thông báo>)`

Lưu ý, giá trị nhận vào từ bàn phím là giá trị kiểu chuỗi, vì vậy muốn trả về giá trị kiểu gì ta phải ép kiểu cho giá trị đó. Thông thường, chúng ta cũng thêm 1 câu thông báo để hướng dẫn cho người dùng thông tin cần thiết cho việc nhập liệu, chẳng hạn như ví dụ bên dưới:

- Kiểu nguyên: `a = int(input("Nhập số nguyên a: "))`
- Kiểu thực: `b = float(input("Nhập số thực b: "))`

Trong bài hướng dẫn này, chúng ta sẽ tìm hiểu 2 kiểu dữ liệu số thông dụng bậc nhất khi mới bắt đầu với ngôn ngữ lập trình, là kiểu số nguyên và kiểu số thực, sẽ được trình bày ở 2 phần tiếp theo.

3 Kiểu số nguyên trên Python

3.1 Khai báo một biến kiểu số nguyên

Với tính mềm dẻo và linh động, Python cho phép chúng ta không cần khai báo biến lần xác định kiểu dữ liệu cho nó. Chỉ đơn giản bằng một phép gán, một biến sẽ được tạo ra và Python tự động chọn kiểu dữ liệu cho nó, phụ thuộc vào giá trị của phép gán. Đối với người lập trình bằng ngôn ngữ Python, khai báo một biến kiểu số nguyên sẽ thường sử dụng câu lệnh gán sau đây:


```

1 #Khai bao bien a, b la so nguyen
2 a = 5
3 b = int(6)
4 print(a, b)

```

Chương trình 3.1: Khai báo một biến là kiểu số nguyên

Một cách truyền thống hơn, như ví dụ ở trên, chúng ta có thể chỉ định kiểu cho biến, bằng cách thêm từ khóa **int**. Tuy nhiên điều này thực sự không được sử dụng nhiều với câu lệnh khai báo biến. Câu lệnh **b = int(6)** thực ra có 1 ý nghĩa lớn hơn, là chuyển tất cả những gì trong dấu ngoặc sang số nguyên. Nó sẽ có nhiều tác dụng hơn trong phần hướng dẫn tiếp theo, là đọc biến từ bàn phím.

3.2 Nhập số nguyên từ bàn phím

Một đoạn chương trình nhỏ, để nhận số nguyên từ việc nhập từ bàn phím, như sau:

```

1 a = int(input("Nhap so nguyen a: "))
2 print(a)

```

Chương trình 3.2: Nhận số nguyên từ bàn phím

Với chức năng này, bắt buộc chúng ta phải có từ khóa **int** chứ không thể lược bỏ như phần khai báo biến. Lý do là những gì nhập từ bàn phím, là kiểu chuỗi. Từ khóa **int** ở phía trước sẽ làm nhiệm vụ chuyển đổi từ kiểu chuỗi sang kiểu số. Do đó, bạn có thể nhập vào số nguyên 6 hoặc số thập phân 6.1, thì giá trị của biến **a** vẫn chỉ có giá trị là 6 mà thôi.

3.3 Các phép toán trên số nguyên

Trước khi thử nghiệm các phép toán trên số nguyên, chúng ta sẽ hiện thực một đoạn chương trình nhỏ, là nhập 2 số **n1** và **n2** từ bàn phím, và in ra tổng của 2 số. Chương trình ví dụ như sau:

```

1 n1 = int(input("So thu nhhat: "))
2 n2 = int(input("So thu hai: "))
3
4 print("Tong cua 2 so: ", n1+n2)

```

Chương trình 3.3: Chương trình tính toán đơn giản trên 2 số nguyên

Các phép toán phổ biến trên số nguyên, được liệt kê ra như sau:

- **+** : Phép cộng
- **-** : Phép trừ
- ***** : Phép nhân
- **//** : Chia làm tròn
- **/** : Chia số thực
- **%** : Chia lấy dư

- **: Phép mũ

Bây giờ, bạn có thể thử thay đổi phép toán trong Chương trình 3.3 để hiểu rõ hơn các phép toán được liệt kê ở trên.

4 Kiểu số thực trên Python

4.1 Khai báo một biến kiểu số thực

Cũng tương tự như khi làm việc với số nguyên, Python cho phép chúng ta không cần khai báo biến lần xác định kiểu dữ liệu cho nó. Chỉ đơn giản bằng một phép gán, một biến sẽ được tạo ra và Python tự động chọn kiểu dữ liệu cho nó, phụ thuộc vào giá trị của phép gán. Đối với người lập trình bằng ngôn ngữ Python, khai báo một biến kiểu số thực sẽ thường sử dụng câu lệnh gán sau đây:

```
1 #Khai bao bien a, b la so thuc
2 a = 5.2
3 b = float(6.7)
4 print(a, b)
```

Chương trình 3.4: Khai báo một biến là kiểu số thực

4.2 Nhập số thực từ bàn phím

Một đoạn chương trình nhỏ, để nhận số thực từ việc nhập từ bàn phím, như sau:

```
1 a = float(input("Nhap so thuc a: "))
2 print(a)
```

Chương trình 3.5: Nhận số thực từ bàn phím

Với chức năng này, bắt buộc chúng ta phải có từ khóa **float** chứ không thể lược bỏ như phần khai báo biến. Lý do là những gì nhập từ bàn phím, là kiểu chuỗi. Từ khóa float ở phía trước sẽ làm nhiệm vụ chuyển đổi từ kiểu chuỗi sang kiểu số. Do đó, bạn có thể nhập vào số nguyên 6, nó vẫn sẽ là con số hợp lệ để chuyển sang số thực.

4.3 Các phép toán trên số thực

Trước khi thử nghiệm các phép toán trên số nguyên, chúng ta sẽ hiện thực một đoạn chương trình nhỏ, là nhập 2 số n1 và n2 từ bàn phím, và in ra tổng của 2 số. Chương trình ví dụ như sau:

```
1 n1 = float(input("So thu nhat: "))
2 n2 = float(input("So thu hai: "))
3
4 print("Tong cua 2 so: ", n1+n2)
```

Chương trình 3.6: Chương trình tính toán đơn giản trên 2 số thực

Các phép toán phổ biến trên số thực, cũng gần giống với số nguyên, được liệt kê ra như sau:

- + : Phép cộng
- - : Phép trừ
- * : Phép nhân
- / : Chia số thực
- **: Phép mũ

Bây giờ, bạn có thể thử thay đổi phép toán trong Chương trình 3.7 để hiểu rõ hơn các phép toán được liệt kê ở trên.

5 Xử lý lỗi nhập liệu

Đây sẽ là phần quan trọng bậc nhất trong việc xử lý dữ liệu đầu vào. Khi đưa ra thông báo để nhập vào một số nguyên, người sử dụng hoàn toàn có thể nhập vào một chuỗi. Do đó, câu lệnh chuyển đổi kiểu có thể hoạt động sai và chương trình phải dừng. Mặc dù Python sẽ bỏ qua lỗi nhập số nguyên nhưng đầu vào lại là số thực và ngược lại, vì 2 thông tin này cũng là số.

Để xử lý cho trường hợp người dùng nhập vào chuỗi, hoặc gọi chung là **giá trị số không hợp lệ**, kĩ thuật đơn giản nhất là sử dụng câu lệnh try except, như chương trình ví dụ sau đây:

```
1 try:
2     a = int(input("Nhập số nguyên: "))
3 except:
4     a = 0
5 print(a)
```

Chương trình 3.7: Chương trình xử lý lỗi đầu vào

Trong đoạn chương trình trên, Python sẽ cố gắng chuyển đổi giá trị chuỗi nhập vào sang số nguyên và gán nó vào biến a. Trong trường hợp quá trình này thất bại (do dữ liệu không hợp lệ), phần except sẽ được thực thi và a có giá trị là 0. Ngược lại, nếu việc chuyển đổi thành công, phần except sẽ không được thực thi.

6 Câu hỏi ôn tập

1. Trong Python, cần phải khai báo một biến trước khi gán giá trị cho biến đó. Nhận định trên đúng hay sai?
A. Đúng
B. Sai
2. Cách để gán số nguyên có giá trị là 100 vào biến a là:
A. `a := 100`
B. `a <- 100`
C. `a = 100`
D. `a « 100`
3. Trong Python, một biến a được gán bằng 1 giá trị với kiểu int, sau đó vẫn có thể tiếp tục gán 1 giá trị có kiểu float cho biến a. Nhận định trên đúng hay sai?
A. Sai
B. Đúng

4. Cho đoạn code như sau:

```
1 employeeNumber = 4398
2 EmployeeNumber = 4398
3 employeeNumber = 4398
```

Nhận định nào dưới đây là đúng:

- A. Các đoạn lệnh trên đều khởi tạo giá trị cho cùng 1 biến
B. Các đoạn lệnh trên khởi tạo giá trị cho các biến khác nhau
5. Hãy cho biết sau khi người dùng nhập giá trị xong thì kiểu dữ liệu của biến N trong đoạn lệnh sau là gì :

```
1 N = input("Nhập N") # Gia su nguoi dung nhap 5
2 print(N)
```

- A. str
B. int
C. float
D. Không có kiểu dữ liệu
6. Cho đoạn code như sau:

```
1 a = 4 / 2
```

Hãy cho biết kết quả và kiểu dữ liệu của biến a là gì:

- A. `a = 2.0`, kiểu float
B. `a = 2.0`, kiểu int
C. `a = 2`, kiểu float
D. `a = 2`, kiểu int

7. Cho đoạn code như sau:

```
1 a = 5
2 b = 5.0
3 c = a + b
```

Hãy cho biết kết quả và kiểu dữ liệu của biến c là gì:

- A. c = 10.0, kiểu float
- B. c = 10, kiểu int
- C. Báo lỗi do không thể cộng hai số khác kiểu

Đáp án

1. B 2. C 3. B 4. B 5. A 6. A 7. A

CHƯƠNG 4



Câu lệnh điều kiện IF

pythonTM
Package
Index

1 Giới thiệu

Trong bất kì một ngôn ngữ lập trình nào, sẽ đều có câu lệnh điều kiện. Nhờ câu lệnh này, mà việc ánh xạ những tác vụ được thực thi khi điều kiện nào đó xảy ra, mới có thể được hiện thực trên máy tính. Khi chúng ta nói "Nếu trời mưa thì tôi sẽ mang dù", là một ví dụ cho hành vi **mang dù** chỉ được thực hiện khi có **trời mưa**. Trong ngôn ngữ lập trình, **trời mưa** được xem là một điều kiện. Khi điều kiện này là đúng, máy tính sẽ thực hiện một chức năng nào đó, như là việc **mang dù** chẳng hạn. Cũng từ bài này, chúng ta chính thức bắt đầu vào khái niệm **xử lý** trên máy tính. Những bài toán ngày một phức tạp hơn và cần phải có sự kết hợp của nhiều câu lệnh phức tạp.

Trong ngôn ngữ lập trình Python, một điều kiện đơn giản thường được thể hiện qua một phép so sánh. Và một biểu thức so sánh sẽ trả về kết quả có kiểu dữ liệu là Boolean, tức là True hay False. Các toán tử so sánh có thể được kể đến như sau:

- So sánh bằng: $a == b$
- Khác nhau: $a != b$
- Bé hơn: $a < b$
- Bé hơn hoặc bằng: $a <= b$
- Lớn hơn: $a > b$
- Lớn hơn hoặc bằng: $a >= b$

Ở đây a và b là ví dụ cho 2 biến số. Tuy nhiên chúng ta cũng có thể so sánh giữa một biến số và 1 hằng số, tương tự như những ngôn ngữ lập trình các. Ngoài ra, các toán tử luận lý như **and** và **or** cũng có thể sử dụng khi xét điều kiện kết hợp của nhiều mệnh đề luận lí.

Thực ra, câu lệnh điều kiện **if** có nhiều cách sử dụng khác nhau, thông qua nhiều **biến thể** của nó. Trong tài liệu này, chúng tôi phân ra 3 câu lệnh, bao gồm câu lệnh **if** đơn giản, câu lệnh **if else** và câu lệnh **if elif else**. Cách sử dụng cũng như ý nghĩa của từng câu lệnh trong Python lần lượt được trình bày bên dưới.

2 Câu lệnh if

Đây là câu lệnh đơn giản nhất, có cấu trúc câu lệnh điều kiện như sau:

```
1 if <Dieu_Kien>:  
2     <Cau_Lenh>
```

Trong cấu trúc này, nếu điều kiện là đúng, thì các câu lệnh mới được thực thi. Ngược lại, nếu điều kiện là sai, thì các câu lệnh sẽ không được thực thi. Bắt buộc chúng ta phải có ít nhất một câu lệnh theo sau câu lệnh **if**, nếu không chương trình sẽ báo lỗi.

Một lưu ý cực kì quan trọng khác, là sau câu lệnh **if**, các câu lệnh nằm trong nó phải được thụt vào **một kí tự tab**. Khi làm việc trên PyCharm, chỉ cần bạn gõ tới dấu hai chấm (:), rồi nhấn Enter, thì chương trình sẽ tự động thụt vào. Sau khi hiện thực xong các câu lệnh nằm bên trong **if**, bạn nhấn Delete để thoát ra khỏi tầm vực của câu lệnh điều kiện.

Đối với ngôn ngữ Pascal, việc xây dựng một tầm vực con cho câu lệnh điều kiện thường được đặt giữa 2 từ khóa **begin end**, hay như ngôn ngữ C, các câu lệnh sẽ được đặt giữa 2 kí tự mở và đóng ngoặc {}. Tuy nhiên, trên Python, một kí tự tab có nghĩa là chúng ta vừa mở ra một tầm vực mới. Do đó, việc lập trình phải hết sức cẩn thận trong việc quản lý tầm vực.

```
1 a = 100
2 b = 10
3 if b == a:
4     print("b bang a")
```

Chương trình 4.1: Câu lệnh điều kiện if

Trong Chương trình 4.1 ví dụ ở trên, chỉ khi nào *a* và *b* có giá trị bằng nhau thì chương trình mới in ra câu **b bang a**. Ngoài ra, chương trình không in ra câu nào cả.

3 Câu lệnh if else

Trong câu lệnh này, sẽ có thêm một nhóm thứ 2, mà nếu câu lệnh điều kiện không đúng, chương trình sẽ thực thi các câu lệnh ở nhóm này. Trong các ngôn ngữ khác, đây là câu lệnh **Nếu thì ... nữa ...**. Cấu trúc của câu lệnh điều kiện đầy đủ như sau:

```
1 if <Dieu_Kien>:
2     <Cau_Lenh>
3 else:
4     <Cau_Lenh>
```

Một ví dụ cho việc sử dụng câu lệnh này như sau:

```
1 a = 100
2 b = 10
3 if b == a:
4     print("b bang a")
5 else:
6     print("a khac b")
```

Chương trình 4.2: Câu lệnh điều kiện if else

Với Chương trình 4.2, một trong 2 câu thông báo sẽ được in ra màn hình. Trong ví dụ trên, câu "a khac b" sẽ được in ra. Trong trường hợp này, điều kiện **b == a** là sai, nên phần lệnh trong **else** sẽ được thực hiện.

4 Câu lệnh if elif else

Với câu lệnh này, nhiều điều kiện sẽ được xem xét **lần lượt** trước khi phần cuối cùng là else sẽ được hiện thực, nếu như tất cả các điều kiện trước là sai. Cấu trúc của câu lệnh này như sau:

```
1 if <Dieu_Kien_1>:
2     <Cau_Lenh>
3 elif <Dieu_Kien_2>:
4     <Cau_Lenh>
5 elif <Dieu_Kien_3>:
6     <Cau_Lenh>
7 else:
8     <Cau_Lenh>
```

Một nhầm lẫn mà người học sẽ dễ gặp phải khi sử dụng cấu trúc này, là từ trên xuống, khi một điều kiện là đúng, các câu lệnh trong điều kiện đó sẽ được thực thi và toàn bộ các điều kiện còn lại sẽ không được xem xét nữa. Chẳng hạn điều kiện thứ nhất và thứ 2 đều đúng, thì chương trình chỉ thực hiện các câu lệnh trong nhóm 1 mà thôi. Nói một cách khác, chỉ **điều kiện đầu tiên đúng** được thực thi trong câu lệnh **if**.

Một ví dụ cho câu lệnh này như sau:

```
1 a=200
2 b=10
3 if a<b:
4     print("a be hon b")
5 elif a==b:
6     print("a bang b")
7 else:
8     print("a lon hon b")
```

Chương trình 4.3: Ví dụ lệnh if...elif...else

Lưu ý rằng, phần **else** là một phần tùy chọn, có thể có hoặc không.

5 Câu lệnh lồng nhau

Thực ra **if**, **if else** hay thậm chí **if elif else** cũng chỉ là một câu lệnh. Do đó, nó có thể xuất hiện bên trong một điều kiện của câu lệnh **if**, tạo ra hiện tượng các câu lệnh điều kiện lồng nhau.

Phần trình bày này chỉ muốn nhấn mạnh lại với người đọc, là việc sử dụng cấu trúc lồng nhau này có thể xảy ra nhầm lẫn, nếu như bạn không nắm vững nguyên lý của lệnh **if**. Một chương trình ví dụ về việc sử dụng các câu lệnh điều kiện lồng nhau sẽ như sau:

```
1 a = 2
2 b = 3
3 if a > 1:
4     if b > 4:
```

```

5         a = a + 1
6         b = b + 1
7     else:
8         a = a + 2
9         b = b + 2
10    print(a, b, sep=" - ")

```

Chương trình 4.4: Ví dụ lệnh if lồng nhau

Nếu như không nắm vững nguyên lý của lệnh điều kiện, nhiều người sẽ nhầm rằng kết quả cuối cùng của a và b lần lượt là 4 và 5. Tuy nhiên, khi điều kiện $a > 1$ đã đúng, thì phần **else** của nó sẽ không còn được thực hiện nữa. Bên trong phần **if** đầu tiên, điều kiện $b > 4$ sai và không có đoạn chương trình nào được thực hiện nữa. Cuối cùng, giá trị của a và b không thay đổi, vẫn là 2 và 3.

6 Bài tập

1. Nhập một số từ bàn phím, in ra kết quả đó là số chẵn hay số lẻ.

```

1 num = float(input("Nhap mot so "))
2 if num % 2 == 0:
3     print("So chan")
4 else:
5     print("So le")

```

Chương trình 4.5: Đáp án gợi ý cho Bài 1

2. Nhập một số từ bàn phím, in ra kết quả đó là số dương, số âm hay số không (0).

```

1 num = float(input("Nhap mot so "))
2 if num >= 0:
3     if num == 0:
4         print("So khong")
5     else:
6         print("So duong")
7 else:
8     print("So am")

```

Chương trình 4.6: Đáp án gợi ý cho Bài 2

3. Viết chương trình mô phỏng việc giải phương trình bậc nhất 1 ẩn số: Cho phép người dùng nhập số a và b , sau đó in ra kết quả của phương trình.

```

1 print("Phuong trinh bac nhat: ax + b = 0")
2 a = int(input("a = "))
3 b = int(input("b = "))
4 if (a == 0):
5     print("Phuong trinh vo nghiem")
6 else:
7     x = -b/a

```

```
print("nghiem cua phuong trinh: x = ", x)
```

Chương trình 4.7: Đáp án gợi ý cho Bài 3

4. Viết phương trình mô phỏng việc giải phương trình bậc hai 1 ẩn số: Cho phép người dùng nhập vào a, b và c, sau đó in ra kết quả của chương trình. Trong bài này, chúng ta cần phải thêm thư viện sqrt để tính căn bậc 2 bằng câu lệnh **from math import sqrt**.

```
1 from math import sqrt
2
3 print("Phuong trinh bac hai : ax^2 + bx + c = 0")
4 a = float(input("a = "))
5 b = float(input("b = "))
6 c = float(input("c = "))
7
8 if a == 0:
9     if b == 0:
10         if c == 0:
11             print("Phuong trinh vo so nghiem!")
12         else:
13             print("Phuong trinh vo nghiem!")
14     else:
15         if c == 0:
16             print("Phuong trinh co 1 nghiem x = 0")
17         else:
18             x = -c / b
19             print("Phuong trinh co 1 nghiem x = ", x)
20 else:
21     delta = b ** 2 - 4 * a * c
22     if delta < 0:
23         print("Phuong trinh vo nghiem!")
24     elif delta == 0:
25         x = -b / (2 * a)
26         print("Phuong trinh co 1 nghiem x = ", x)
27     else:
28         print("Phuong trinh co 2 nghiem phan biet!")
29         x1 = float((-b - sqrt(delta)) / (2 * a))
30         x2 = float((-b + sqrt(delta)) / (2 * a))
31         print("x1 = ", x1)
32         print("x2 = ", x2)
```

Chương trình 4.8: Đáp án gợi ý cho Bài 4

7 Câu hỏi ôn tập

1. Câu lệnh if nào là đúng cú pháp trong Python?

- A. if a>=2:
- B. if (a >= 2)
- C. if a >=2
- D. Tất cả đều đúng

2. Kết quả của đoạn chương trình sau là gì?

```
1 x = True
2 y = False
3 z = False
4 if not x or y:
5     print (1)
6 elif not x or not y and z:
7     print (2)
8 elif not x or y or not y and x:
9     print (3)
10 else:
11     print (4)
```

- A. 1
- B. 2
- C. 3
- D. 4

3. Đoạn lệnh nào sau đây sẽ thực hiện không thành công?

A.

```
1 if (1, 2): print('a')
```

B.

```
1 if (1, 2):
2     print('a')
```

C.

```
1     if (1, 2):
2     print('a')
```

D.

```
1 if (1, 2):
2
3     print('a')
```

4. Để kết thúc một khối các câu lệnh trong phần thân của câu lệnh if, chúng ta dùng:

- A. Dấu chấm phẩy
- B. Dấu đóng ngoặc nhọn
- C. end
- D. Một câu lệnh được thực thi ít hơn so với câu lệnh trước

5. Đoạn code sau có lỗi hay không?

```
1 d = {'a': 0, 'b': 1, 'c': 0}
2 if d['a'] > 0:
3     print('ok')
4 elif d['b'] > 0:
5     print('ok')
6 elif d['c'] > 0:
7     print('ok')
8 elif d['d'] > 0:
9     print('ok')
10 else:
11     print('not ok')
```

- A. Không có lỗi
- B. Có lỗi

6. Đoạn lệnh nào dưới đây sẽ thực thi thành công, giả sử x và y đã được khởi tạo từ trước?

A.

```
1 if x < y: if x > 10: print('foo')
```

B.

```
1 if x < y: print('foo') else: print('bar')
```

C.

```
1 if x < y: print('foo'); print('a'); print('b')
```

D.

```
1 if x < y: print('foo')
2 elif y < x: print('bar')
3 else: print('baz')
```

7. Đoạn lệnh sau dùng để làm gì:

```
1 a = int(input("a = "))
2 if a % 2 == 0:
3     print(a)
4 else:
5     print(0)
```

- A. In ra giá trị của a
- B. Nếu là số chẵn thì in giá trị vừa nhập, ngược lại in ra 0
- C. Vừa in ra giá trị của a vừa in ra 0
- D. Nếu là số lẻ thì in giá trị vừa nhập, ngược lại in ra 0

Đáp án

1. A 2. C 3. C 4. D 5. A 6. C 6. D 7. B

CHƯƠNG 5



Mảng một chiều - Cấu trúc lặp FOR

pythonTM
Package
Index

1 Giới thiệu

Mảng là một cấu trúc dữ liệu cơ bản của tất cả các ngôn ngữ lập trình. Khác với biến đơn, chỉ dùng để lưu giá trị riêng lẻ, mảng là tập hợp của nhiều phần tử thuộc cùng một kiểu dữ liệu duy nhất, ví dụ như mảng số nguyên hay mảng chuỗi kí tự. Tuy nhiên, trên Python, khái niệm mảng thuần túy như các ngôn ngữ lập trình khác như Pascal, C hay C++ khá phức tạp và không thân thiện với người mới bắt đầu. Do đó, trong hướng dẫn này, chúng tôi sẽ dùng khái niệm danh sách (list) để thay cho mảng. Thoạt nhìn, chúng ta sẽ thấy nó không khác gì là mảng trong các ngôn ngữ lập trình khác. Chúng tôi vẫn dùng thuật ngữ "mảng" (array) trong bài hướng dẫn này, cho gần gũi với người đọc.

2 Khai báo và truy xuất mảng một chiều

Mảng được khai báo trên Python bằng chuỗi kí tự [], ở giữa là các phần tử của mảng, cách nhau bằng dấu phẩy (,), như ví dụ bên dưới:

```
1 a = [1, 4, 5, 7]
2 print("So thu nhât", a[0])
3 print("So thu hai", a[1])
```

Chương trình 5.1: Khai báo và truy xuất một mảng

Như ví dụ ở Chương trình 5.1, mảng a có tất cả **4 phần tử**. Việc truy xuất từng phần tử trong a được thực hiện qua tác vụ a[0] hay a[1]. Điều này có nghĩa là, phần tử đầu tiên sẽ có chỉ số là 0, và phần tử cuối cùng sẽ có chỉ số là 3. Vì lý do nào đó, mà bạn truy xuất phần tử a[4], chương trình sẽ báo lỗi "**list index out of range**", tức là truy xuất vượt tầm giới hạn của mảng.

Sự mềm dẻo trong ngôn ngữ Python cho phép bạn truy cập các phần tử với chỉ số âm, ví dụ như a[-1] cho phần tử cuối cùng và lùi dần tới phần tử đầu tiên, là a[-4]. Lỗi tràn giới hạn tương tự như trên cũng xuất hiện khi bạn truy xuất tới a[-5]. Tuy nhiên chúng tôi thực sự không khuyến khích bạn làm theo quy luật này.

3 Cấu trúc lặp for

Khi làm việc với mảng, tác vụ thường sử dụng nhất gọi là **duyệt mảng**. Chúng ta cần 1 vòng lặp để xử lý qua tất cả các phần tử của mảng. Trong bài hướng dẫn này, chúng ta sẽ sử dụng cấu trúc lặp FOR.

Trong cấu trúc này, chúng ta cần một biến số, đặt tên là **i**, để giữ chỉ số phần tử hiện tại của mảng. Biến này sẽ tự động tăng lên 1 sau mỗi lần lặp. Trong Python, cách đơn giản nhất để hiện thực chức năng này là xài câu lệnh **range(Số cận dưới, Số cận trên)**, được trình bày như chương trình bên dưới:

```
1 a = [1, 4, 5, 7]
2 N = len(a)
3 print("Kich thuoc mang", N)
```

```

4 for i in range(0, N):
5     print(i)

```

Chương trình 5.2: Câu lệnh range trên Python

Trong ví dụ ở Chương trình 5.2, chúng ta lấy kích thước của mảng thông qua toán tử **len** rồi gán kết quả vào biến **N**. Biến **N** sau đó sẽ có giá trị là 4, vì mảng **a** đang có 4 phần tử. Tuy nhiên trong vòng lặp **for**, các giá trị được in ra từ 0 đến 3 mà thôi. Đó là lý do tại sao, câu lệnh **range** sẽ vô vùng phù hợp với tác vụ duyệt mảng trên Python, bởi chỉ số của mảng được đánh từ 0 cho tới **N - 1**, với **N** là kích thước của mảng.

Chúng ta sẽ cải tiến chương trình ở trên để in ra từng giá trị của mảng, như sau:

```

1 a = [1, 4, 5, 7]
2 N = len(a)
3 print("Kích thước mảng", N)
4 for i in range(0, N):
5     print("Số thứ", i + 1, a[i])

```

Chương trình 5.3: In các giá trị trong mảng

Một lần nữa, trong ngôn ngữ tự nhiên, chúng ta thường định vị số đầu tiên là số thứ nhất (số 1), còn trong ngôn ngữ lập trình Python, vị trí đầu tiên là vị trí 0. Do đó, trong câu lệnh in ra màn hình, để thông tin thân thiện hơn với người dùng, chúng ta sẽ cộng biến **i** thêm 1 đơn vị.

Cấu trúc lặp **FOR** trình bày trong hướng dẫn này còn gọi là cấu trúc lặp hữu hạn. Điều này có nghĩa là trước khi bắt đầu vòng lặp, các giá trị của biến **i** đã được thiết lập dựa vào giá trị của **N**. Từ đó, vòng lặp được thực thi thông qua biến **i** và hoàn toàn độc lập với **N**. Do đó, nếu chúng ta có thay đổi giá trị **N** bên trong vòng lặp, kết quả vẫn là không thay đổi.

```

1 a = [1, 4, 5, 7]
2 N = len(a)
3 print("Kích thước mảng", N)
4 for i in range(0, N):
5     print("Số thứ", i + 1, a[i])
6     N = N - 1

```

Chương trình 5.4: Vòng lặp hữu hạn for không phụ thuộc vào giá trị N

Chương trình 5.4 là ví dụ để minh họa cho số vòng lặp cố định của cấu trúc **FOR**. Tại thời điểm thiết lập ban đầu, **N = 4** và chương trình sẽ được lặp 4 lần, bất chấp việc chúng ta thay đổi giá trị **N** bên trong vòng lặp. Do đó, cấu trúc này thường được sử dụng cho những trường hợp có số vòng lặp cố định.

Đến đây, người học có thể tạm dừng để chuyển sang phần bài tập (từ 1 đến 5) để luyện tập với phần mảng trên Python. Phần tiếp theo của hướng dẫn này sẽ trình bày cách nhập một mảng từ bàn phím, thay vì gán một mảng cố định như các ví dụ ở trên.

4 Nhập mảng từ bàn phím

Theo tư tưởng trình bày trong hướng dẫn này, chúng tôi đặt biệt ưu tiên việc xuất kết quả trước, còn việc xử lý đầu vào sẽ thực hiện sau. Lý do là việc xử lý dữ liệu đầu vào thường là phức tạp hơn. Ví dụ như trong yêu cầu này, chúng ta mong đợi người dùng sẽ nhập vào một con số. Tuy nhiên vì lý do nào đó, người dùng lại nhập vào một chuỗi kí tự. Việc xử lý những lỗi này cũng không hề đơn giản, chúng ta sẽ yêu cầu người dùng nhập lại, hay là sẽ mặc định cho các giá trị nhập sai là 0.

Trong hướng dẫn này, chúng tôi sẽ chỉ tập trung vào một ứng dụng đơn giản: Chương trình yêu cầu nhập vào kích thước của mảng, một vòng lặp sẽ hỏi người dùng nhập vào từng phần tử cho mảng.

Với yêu cầu như trên, chúng ta sẽ khởi động mảng `a = []`, chưa có phần tử nào cả. Sau khi hỏi người dùng nhập số `N`, vòng lặp `for` sẽ hỏi để nhập từng phần tử. Chương trình gợi ý như sau:

```
1 a = []
2 N = int(input("Nhập kích thước mảng: "))
3 for i in range(0, N):
4     temp = int(input("Nhập phần tử thứ " + str(i + 1)))
5     a.append(temp)
6 print(a)
```

Chương trình 5.5: Nhập các giá trị của mảng từ bàn phím

Để cho câu lệnh `input` trở nên thân thiện hơn, chúng ta sẽ thêm thông tin về số thứ tự của mảng khi yêu cầu người dùng nhập vào. Do `i + 1` là kiểu số, nên nó phải được chuyển sang kiểu chuỗi trước khi ghép với một chuỗi khác. Cuối cùng, toán tử **`append`** là để thêm phần tử vừa nhập vào mảng `a`. Kết thúc chương trình, chúng ta sẽ dùng lệnh **`print(a)`** để kiểm tra các giá trị vừa nhập vào của mảng.

Cần lưu ý là, đối với các ngôn ngữ lập trình, sẽ không có toán tử thêm 1 phần tử vào mảng như Chương trình 9.2. Ở đây, bản chất là chúng ta đang dùng danh sách để thay thế cho chức năng của một mảng, nên mới có toán tử này.

5 Bài tập

1. Viết chương trình tính tổng các phần tử trong mảng. Mảng được nhập cố định ở đầu chương trình.

```
1 a = [1, 2, 3, 4, 5]
2 N = len(a)
3 Tong = 0
4 for i in range(0, N):
5     Tong = Tong + a[i]
6 print("Tổng các phần tử trong mảng là:", Tong)
```

Chương trình 5.6: Đáp án gợi ý cho Bài 1

2. Viết chương trình đếm số lượng số chẵn trong mảng. Mảng được nhập cố định ở đầu chương trình.

```

1 a = [1,2,3,4,5]
2 N = len(a)
3 chan = 0
4 for i in range(0,N):
5     if a[i] % 2 == 0:
6         chan = chan + 1
7 print("So phan tu chan trong mang la:", chan)

```

Chương trình 5.7: Đáp án gợi ý cho Bài 2

3. Viết chương trình đếm số lượng số lẻ trong mảng. Mảng được nhập cố định ở đầu chương trình.

```

1 a = [1,2,3,4,5]
2 N = len(a)
3 le = 0
4 for i in range(0,N):
5     if a[i] % 2 != 0:
6         le = le + 1
7 print("So phan tu le trong mang la:", le)

```

Chương trình 5.8: Đáp án gợi ý cho Bài 3

4. Viết chương trình tìm số lớn nhất trong mảng. Mảng được nhập cố định ở đầu chương trình.

```

1 a = [1,2,3,4,5]
2 N = len(a)
3 max = a[0]
4 for i in range(1,N):
5     if max < a[i]:
6         max = a[i]
7 print("Phan tu lon nhat trong mang la:", max)

```

Chương trình 5.9: Đáp án gợi ý cho Bài 4

5. Viết chương trình tìm số bé nhất trong mảng. Mảng được nhập cố định ở đầu chương trình.

```

1 a = [1,2,3,4,5]
2 N = len(a)
3 min = a[0]
4 for i in range(1,N):
5     if min > a[i]:
6         min = a[i]
7 print("Phan tu nho nhat trong mang la:", min)

```

Chương trình 5.10: Đáp án gợi ý cho Bài 5

6. Hiện thực lại tất cả các yêu cầu trên, với mảng được nhập vào từ bàn phím.

```

1 a = []
2 N = int(input("Nhap kích thước mảng: "))
3 for i in range(0, N):
4     temp = int(input("Nhap phần tử thứ " + str(i+1)))
5     a.append(temp)
6

```

```

7 #Tinh tong, tim so chan, le
8 tong = chan = le = 0
9 for i in range(0,N):
10     tong = tong + a[i]
11     if a[i] % 2 == 0:
12         chan = chan + 1
13     else:
14         le = le + 1
15
16 #Tim min, max
17 max1 = min1 = a[0]
18 for i in range(1,N):
19     if a[i] < min1:
20         min1 = a[i]
21     if a[i] > max1:
22         max1 = a[i]
23
24 print(tong, chan, le, max1, min1)

```

Chương trình 5.11: Đáp án gợi ý cho Bài 6

6 Câu hỏi ôn tập

1. Kết quả của đoạn lệnh sau là gì?

```
1 numbers = [1, 2, 3, 4]
2 numbers.append([5,6,7,8])
3 print(len(numbers))
```

- A. 4
- B. 5
- C. 8
- D. Báo lỗi

2. Kết quả của đoạn lệnh sau là gì?

```
1 list1 = [1, 2, 3, 4]
2 list2 = [5, 6, 7, 8]
3 print(len(list1 + list2))
```

- A. 2
- B. 4
- C. 8
- D. Báo lỗi

3. Kết quả của đoạn lệnh sau là gì?

```
1 list1 = [1, 2, 3, 4]
2 print(list1[4])
```

- A. 3
- B. 4
- C. 1
- D. Báo lỗi

4. Kết quả của đoạn lệnh sau là gì?

```
1 x = sum(range(5))
2 print(x)
```

- A. 4
- B. 5
- C. 10
- D. 15

5. Đoạn lệnh nào sau đây được dùng để in các phần tử của mảng list ra màn hình:

```
1 list1 = [1,2,3,4]
```

A.

```
1 for i in list1:  
2     print(i)
```

B.

```
1 for i in range(len(list1))  
2     print(a[i])
```

- C. Cả hai đoạn lệnh trên đều đúng
D. Cả hai đoạn lệnh trên đều sai

6. Kết quả của đoạn code dưới đây là gì, biết rằng lệnh break dùng để thoát khỏi vòng lặp đang chứa nó:

```
1 for i in range(10):  
2     if i == 4:  
3         break  
4     else:  
5         print(i)  
6 else:  
7     print("Python")
```

- A. 0 1 2 3 4
B. 0 1 2 3 Python
C. 0 1 2 3
D. 0 1 2 3 4 Python

7. Giá trị của a sau khi kết thúc vòng lặp là bao nhiêu :

```
1 a = 0  
2 for i in range(10, 14):  
3     a = a + 3
```

- A. 9
B. 12
C. 15
D. 18

Đáp án

1. C 2. C 3. D 4. C 5. C 6. C 7. C

CHƯƠNG 6



Mảng nhiều chiều - FOR lồng nhau

pythonTM
Package
Index

1 Giới thiệu

Như đã giới thiệu ở bài trước, mảng một chiều chứa tập hợp nhiều phần tử có cùng kiểu dữ liệu với nhau. Trong một số trường hợp, chúng ta sẽ có mỗi phần tử là một mảng. Điều này tạo nên một khái niệm mới, gọi là mảng nhiều chiều. Một trong những ví dụ điển hình về mảng nhiều chiều là ma trận, cấu trúc dữ liệu thường thấy cho việc xử lý ảnh hoặc xử lý trong toán học.

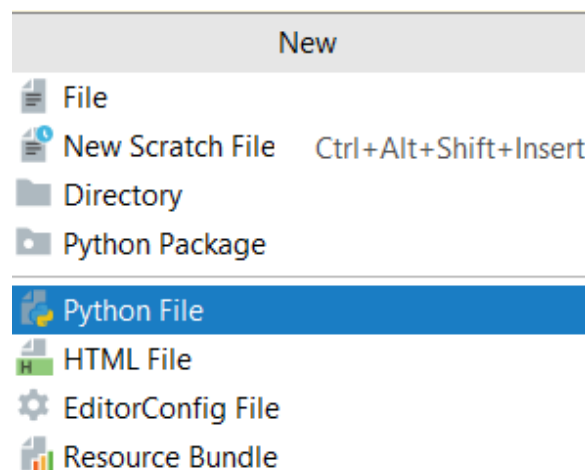
Cũng như ở bài trước, chúng ta có thể sử dụng danh sách (list) để hiện thực mảng nhiều chiều: Mỗi một phần tử của danh sách là một danh sách khác. Khi sử dụng danh sách để hiện thực thay cho mảng, các thao tác làm việc sẽ rất giống với cấu trúc mảng truyền thống trên các ngôn ngữ khác như Pascal hay C. Việc truy xuất 1 phần tử trong mảng sẽ cần 2 chỉ số cho mảng 2 chiều, tương tự như khái niệm hàng và cột trong các ngôn ngữ khác.

Tuy nhiên, trước khi đi vào chi tiết của mảng nhiều chiều, chúng ta sẽ tạo mới thêm một file python trong dự án. Dự án đầu tiên này, chúng ta có thể lưu nó như một dạng thư viện chương trình mẫu, và sử dụng nó cho các dự án sau này.

2 Thêm một file Python

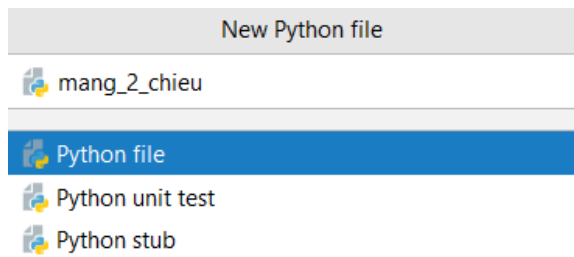
Mặc định, khi mới tạo ra dự án (project), chúng ta chỉ có 1 file python, là **main.py**. Khi bấm nút **Run** trên thanh công cụ (hoặc nhấn phím nóng **Shift F10**), Pycharm sẽ mặc định chạy chương trình trong file main.py. Trong hướng dẫn này, chúng ta sẽ thêm một file mới, đặt tên là **mang_2_chieu.py**, và cấu hình lại PyCharm để nó có thể chạy chương trình trong file này. Các bước hướng dẫn được trình bày chi tiết bên dưới.

Bước 1: Từ menu File, chọn vào **New...** (tránh chọn nhầm vào New Project...). Một giao diện hiện ra và chúng ta chọn tiếp vào Python File, như minh họa ở Hình 6.1.



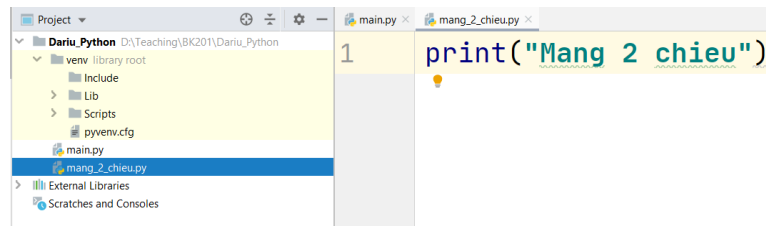
Hình 6.1: Thêm mới một file Python

Bước 2: Đặt tên cho file là **mang_2_chieu** rồi nhấn **Enter**, như minh họa ở Hình 6.2. Phần đuôi mở rộng **.py** cho file sẽ tự động thêm vào.



Hình 6.2: Đặt tên cho file Python

Sau khi thêm vào thành công, chúng ta sẽ thấy file mới thêm vào xuất hiện ở cửa sổ Project. Chúng ta mở file này bằng cách nhấp đôi vào nó, và cho in ra màn hình một câu đơn giản, như minh họa ở Hình 6.3.



Hình 6.3: File mới thêm vào trên PyCharm

Bước 3: Từ cửa sổ Project, bạn nhấp chuột phải, và chọn **Run mang_2_chieu**, như minh họa ở Hình 6.4.

Bây giờ nếu để ý ở tùy chọn trước nút Run trên thanh công cụ, bạn sẽ thấy PyCharm tự động thêm vào một tùy chọn nữa (xem Hình 6.5). Chúng ta đang có 2 tùy chọn là **main** và **mang_2_chieu**. Chúng ta có thể đổi việc thực thi file python bằng cách lựa chọn ở đây. Cũng bởi vì trong dự án của chúng ta đang có nhiều file, hãy in ra một dòng gì đó trước khi thực thi, như là 1 cách để quản lý chương trình được dễ hơn.

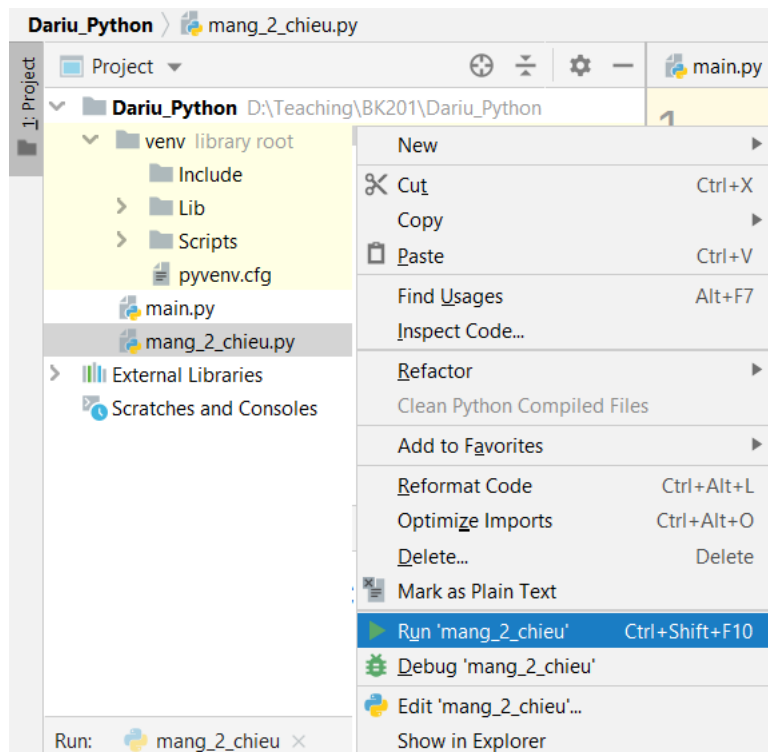
3 Khai báo và truy xuất mảng nhiều chiều

Mỗi chiều của mảng sẽ được truy xuất thông qua một chỉ số. Trong chương trình bên dưới, mảng **a** được khởi tạo gồm có 2 chiều: Chiều thứ nhất có 2 phần tử, chiều thứ 2, mỗi chiều có 3 phần tử.

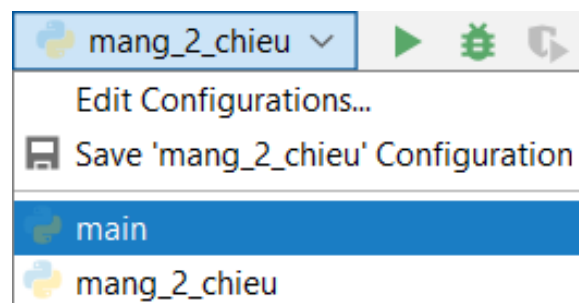
```
1 a = [[1, 2, 3], [4, 5, 6]]
2 print(a[0][0])
3 print(a[0][1])
4 print(a[1][1])
```

Chương trình 6.1: Ví dụ cho mảng 2 chiều

Cũng như mảng một chiều, **phần tử đầu tiên trong mỗi chiều đánh được chỉ số là 0**. Đây là điều hết sức quan trọng khi làm việc với mảng trên Python. Qua ví dụ ở trên thì chúng ta cũng có thể ánh xạ mảng 2 chiều như 1 dạng ma trận: chiều thứ nhất là hàng và chiều thứ 2 là cột. Chúng ta cũng có thể phát biểu ma trận **a** ở trên có 2 hàng và 3 cột.



Hình 6.4: Chuyển chương trình sang file mới thêm vào



Hình 6.5: Tùy chọn thay đổi chương trình thực thi trên thanh công cụ

4 Duyệt mảng nhiều chiều

Qua việc khai báo mảng nhiều chiều, chúng ta có thể nhận ra rằng nó là 2 cấu trúc mảng một chiều lồng nhau (từ chuyên ngành là nested list). Do đó, cách duyệt mảng nhiều chiều phổ biến nhất là dùng 2 vòng lặp for lồng nhau. Chương trình sau đây, sẽ in ra thông tin về vị trí hàng, cột đi kèm với giá trị của phần tử tại đó, như ví dụ ở Chương trình 6.2.

```
1 a = [[1, 2, 3], [4, 5, 6]]
2 ROW = len(a)
3 COL = len(a[0])
4 for i in range(0, ROW):
5     for j in range(0, COL):
6         print("Hang", i + 1, "Cot", j + 1, ":", a[i][j])
```

Chương trình 6.2: Duyệt mảng 2 chiều

Thực ra, cách duyệt mảng ở trên chỉ đúng khi giả định rằng, tất cả các hàng đều có cùng số lượng cột. Do đó, chương trình mở trên chỉ cần tính số lượng cột của hàng đầu tiên mà thôi (câu lệnh **COL = len(a[0])**). Đây cũng là nguyên tắc của mảng trên các ngôn ngữ lập trình có cấu trúc chặt chẽ như Pascal hay C. Tuy nhiên, trên Python, chúng ta được phép **thay đổi số lượng cột của mỗi hàng** trong ma trận nhiều chiều. Chương trình 6.2 cần phải thay đổi vòng lặp bên trong để có thể hoạt động đúng chức năng, như ví dụ ở Chương trình 6.3.

```
1 a = [[1, 2, 3], [4, 5, 6], [7, 8]]
2 ROW = len(a)
3 for i in range(0, ROW):
4     COL = len(a[i])
5     for j in range(0, COL):
6         print("Hang", i + 1, "Cot", j + 1, ":", a[i][j])
```

Chương trình 6.3: Duyệt mảng 2 chiều

Trong ví dụ ở Chương trình 6.3, hàng thứ 3 của mảng **a** chỉ có 2 cột mà thôi. Do đó, trước khi vào vòng lặp của cột, chúng ta tính lại số lượng cột bằng câu lệnh **COL = len(a[i])**.

5 Nhập mảng nhiều chiều từ bàn phím

Thực ra, nhu cầu nhập mảng nhiều chiều từ bàn phím không nhiều do số lượng phần tử phải nhập là khá nhiều. Đối với chức năng đọc dữ liệu từ mảng nhiều chiều, mà chủ yếu là mảng 2 chiều, chúng ta sẽ đọc dữ liệu lên từ 1 file có sẵn. Tuy nhiên trong hướng dẫn này, chúng tôi cũng gợi ý cho người học những bước xử lý căn bản nhất khi hiện thực việc đọc dữ liệu từ bàn phím cho mảng 2 chiều. Ý tưởng chính ở đây là nhập vào từng hàng (mảng 1 chiều), rồi thêm từng hàng đó vào mảng 2 chiều. Chương trình gợi ý như sau:

```
1 a = []
2 ROW = int(input("Nhap so Hang"))
```

```

3 COL = int(input("Nhap so Cot"))
4 for i in range (0, ROW):
5     temp_row = []
6     for j in range (0, COL):
7         temp = int(input("Phan tu "+str(i)+"-"+str(j)))
8         temp_row.append(temp)
9     a.append(temp_row)
10 print(a)

```

Chương trình 6.4: Nhập mảng 2 chiều từ bàn phím

Trong Chương trình 6.4, biến **a** dùng để lưu thông tin của mảng 2 chiều, biến **temp_row** dùng để lưu từng hàng của mảng 2 chiều. Vòng lặp với biến **j** thực chất là vòng lặp để nhập mảng 1 chiều từ bàn phím ở bài trước. Cuối cùng, biến **a** được in ra để kiểm tra kết quả nhập vào. Chương trình này đang cho phép nhập vào số nguyên mà thôi. Trường hợp nhập vào số thực sẽ xem như là một bài tập nâng cao. Chương trình 6.4 là sự luyện tập rất tốt cho việc phân biệt tầm vực trong Python. Chúng ta có 2 vòng for lồng nhau và việc gán lại các giá trị cho biến số cần phải thực hiện đúng tầm vực thì mới có được kết quả mong muốn.

Trong chương trình này, chúng ta ngầm quy định rằng các hàng đều có số lượng cột bằng nhau. Nếu muốn nhập vào một ma trận với số lượng không biết trước, ví dụ như nhập cho đến khi gặp số 0, chúng ta sẽ phải dùng kĩ thuật khác, sẽ được trình bày trong bài hướng dẫn sau.

6 Bài tập

1. Viết chương trình tính tổng các phần tử trong mảng 2 chiều. Mảng được nhập cố định ở đầu chương trình.

```

1 a = [[1, 2, 3], [4, 5, 6]]
2 ROW = len(a)
3 COL = len(a[0])
4 tong = 0
5 for i in range(0, ROW):
6     for j in range(0, COL):
7         tong = tong + a[i][j]
8 print("Tong cua cac phan tu la: ", tong)

```

Chương trình 6.5: Đáp án gợi ý cho bài 1

2. Viết chương trình đếm số lượng số chẵn trong mảng 2 chiều. Mảng được nhập cố định ở đầu chương trình.

```

1 a = [[1, 2, 3], [4, 5, 6]]
2 ROW = len(a)
3 COL = len(a[0])
4 chan = 0
5 for i in range(0, ROW):
6     for j in range(0, COL):
7         if a[i][j] % 2 == 0:
8             chan = chan + 1

```

```
9 print("So phan tu chan la: ", chan)
```

Chương trình 6.6: Đáp án gợi ý cho bài 2

3. Viết chương trình đếm số lượng số lẻ trong mảng 2 chiều. Mảng được nhập cố định ở đầu chương trình.

```
1 a = [[1, 2, 3], [4, 5, 6]]
2 ROW = len(a)
3 COL = len(a[0])
4 le = 0
5 for i in range(0, ROW):
6     for j in range(0, COL):
7         if a[i][j] % 2 != 0:
8             le = le + 1
9 print("So phan tu chan la: ", le)
```

Chương trình 6.7: Đáp án gợi ý cho bài 3

4. Viết chương trình tìm số lớn nhất trong mảng 2 chiều. Mảng được nhập cố định ở đầu chương trình.

```
1 a = [[1, 2, 3], [4, 5, 6]]
2 ROW = len(a)
3 COL = len(a[0])
4 max1 = a[0][0]
5 for i in range(0, ROW):
6     for j in range(0, COL):
7         if a[i][j] > max1:
8             max1 = a[i][j]
9 print("Phan tu lon nhat la: ", max1)
```

Chương trình 6.8: Đáp án gợi ý cho bài 4

5. Viết chương trình tìm số bé nhất trong mảng 2 chiều. Mảng được nhập cố định ở đầu chương trình.

```
1 a = [[1, 2, 3], [4, 5, 6]]
2 ROW = len(a)
3 COL = len(a[0])
4 min1 = a[0][0]
5 for i in range(0, ROW):
6     for j in range(0, COL):
7         if a[i][j] < min1:
8             min1 = a[i][j]
9 print("Phan tu nho nhat la: ", min1)
```

Chương trình 6.9: Đáp án gợi ý cho bài 5

6. Hiện thực lại tất cả các yêu cầu trên, với mảng được nhập vào từ bàn phím.

```
1 a=[]
2 ROW = int(input("Nhap so Hang "))
3 COL = int(input("Nhap so Cot "))
4 for i in range (0, ROW):
5     temp_row = []
6     for j in range(0, COL):
```

```

7         temp = int(input("Phan tu "+str(i)+ "-" + str(j) +
8         ")))
9         temp_row.append(temp)
10        a.append(temp_row)
11
12    for i in range(0, ROW):
13        for j in range(0, COL):
14            print(a[i][j])
15
16    # Tinh tong, tim so chan, le
17    tong = chan = le = 0
18    for i in range(0, ROW):
19        for j in range(0, COL):
20            tong = tong + a[i][j]
21            if a[i][j] % 2 == 0:
22                chan = chan + 1
23            else:
24                le = le + 1
25
26    # Tim min, max
27    max1 = min1 = a[0][0]
28    for i in range(0, ROW):
29        for j in range(0, COL):
30            if a[i][j] < min1:
31                min1 = a[i][j]
32            if a[i][j] > max1:
33                max1 = a[i][j]
34
35    print(tong, chan, le, max1, min1)

```

Chương trình 6.10: Đáp án gợi ý cho bài 6

7 Câu hỏi ôn tập

- Cách nào sau đây dùng để khai báo mảng hai chiều với 2 hàng và 3 cột:
 - A. `a = [[1,2,3], [4,5,6]]`
 - B. `a = [1,2,3,4,5,6]`
 - C. `a = [1,2,3];[4,5,6]`
- Cách nào sau đây dùng để gán giá trị của phần tử đầu tiên ở hàng thứ 2 của mảng hai chiều a cho biến x:
 - A. `x = a[0,1]`
 - B. `x = a[2][1]`
 - C. `x = a[1,0]`
 - D. `x = a[1][0]`
- Cho đoạn code sau:

```

1 for i in range (0,2):
2     for j in range (0,2):

```



```
3 print(1)
```

Lệnh print được thực hiện bao nhiêu lần:

- A. 2
- B. 3
- C. 4
- D. 7

4. Cho đoạn code sau:

```
1 a = [[1,2,3],[4,5,6],[7,8,9]]
2 for i in range(0,2):
3     for j in range(0,2):
4         print(a[i][j])
```

Đoạn lệnh trên thực hiện công việc gì:

- A. In tất cả các giá trị có trong mảng hai chiều a
- B. In tất cả các giá trị của 2 hàng đầu trong mảng hai chiều a
- C. In 2 giá trị đầu của 2 hàng đầu trong mảng hai chiều a

5. Cho đoạn code sau:

```
1 x = 0
2 a = [[1,2,3],[4,5,6]]
3 for i in range(0,2):
4     for j in range(0,3):
5         if a[i][j] % 2 == 0:
6             x = x + a[i][j]
```

Giá trị của biến x là:

- A. 9
- B. 12
- C. 21

6. Cho đoạn code sau:

```
1 x = 0
2 for i in range(0,1):
3     for j in range(0,2):
4         for k in range(0,3):
5             x = x + 1
```

Giá trị của biến x là:

- A. 6
- B. 8
- C. 9
- D. 12

7. Cho đoạn code sau:

```
1 x = 0
2 a = [[1,2,3],[4,5,6]]
3 for i in range (0,2):
4     for j in range (0,3):
5         if a[i][j] % 2 == 0:
6             break;
7         else:
8             x = x + a[i][j]
```

Giá trị của biến x là bao nhiêu, biết rằng lệnh break dùng để thoát khỏi vòng lặp đang chứa nó:

- A. 0
- B. 1
- C. 4
- D. 9

Đáp án

1. A 2. D 3. C 4. C 5. B 6. A 7. B

CHƯƠNG 7



Cấu trúc lặp while

pythonTM
Package
Index

1 Giới thiệu

Vòng lặp là gì? Vòng lặp cho phép bạn lặp lại việc thực thi đoạn code cho tới khi một điều kiện nhất định được thỏa mãn. Vòng lặp được sử dụng khá phổ biến trong lập trình. Không giống các ngôn ngữ lập trình khác với các vòng lặp khác nhau như for, while, do...while..., Python chỉ hỗ trợ vòng lặp for và vòng lặp while.

Trong bài này chúng ta sẽ tìm hiểu về vòng lặp while trong Python, đây là vòng lặp được sử dụng khá nhiều khi bạn làm các ứng dụng trong thực tế, không chỉ riêng ở Python mà ở các ngôn ngữ khác đều vậy.

2 Cú pháp vòng lặp while

Python là ngôn ngữ đơn giản nên cú pháp của nó cũng đơn giản. Sau đây là cú pháp chung của vòng lặp while.

```
1 while expression:
2     statement(s)
```

Trong đó:

- **statement(s)**: là một lệnh đơn hoặc một tập lệnh gồm nhiều lệnh đơn. Nếu chỉ có một lệnh thì bạn có thể không cần khoảng trắng, tuy nhiên lời khuyên là bạn nên sử dụng khoảng trắng để chương trình được rõ ràng và dễ bảo trì hơn.
- **expression**: có thể là một biến hoặc một biểu thức, nhưng bắt buộc giá trị của nó phải là TRUE hoặc FALSE.

Ví dụ:

```
1 count = 0
2 while (count <= 6):
3     print("Luot dem:", count)
4     count = count + 1
5
6 print("Good bye!")
```

Kết quả in ra:

```
1 Luot dem: 0
2 Luot dem: 1
3 Luot dem: 2
4 Luot dem: 3
5 Luot dem: 4
6 Luot dem: 5
7 Luot dem: 6
8 Good bye!
```

Dòng Good bye! không bị lặp bởi vì nó nằm ngoài vòng lặp, còn lượt đếm sẽ bị lặp 7 lần biến count có giá trị ban đầu là 0 (count = 0), sau mỗi vòng lặp nó tăng lên 1 đơn vị (count = count + 1) và điều kiện dừng vòng lặp là count bé hơn hoặc bằng 6 (count <= 6).

2.1 Câu lệnh break và continue trong vòng lặp while

2.1.1 Câu lệnh break

Dùng để kết thúc vòng lặp. Cứ nó nằm trong block của vòng lặp nào thì vòng lặp đó sẽ kết thúc khi chạy câu lệnh này.

Trong trường hợp vòng lặp a chứa vòng lặp b. Trong vòng lặp b chạy câu lệnh break thì chỉ vòng lặp b kết thúc, còn vòng lặp a thì không.

2.1.2 Câu lệnh continue

Câu lệnh này dùng để chạy tiếp vòng lặp. Giả sử một vòng lặp có cấu trúc như sau:

```
1 while expression:
2     #while-block-1
3     continue
4     #while-block-2
```

Khi thực hiện xong while-block-1, câu lệnh continue sẽ tiếp tục vòng lặp, không quan tâm những câu lệnh ở dưới continue và như vậy nó đã bỏ qua while-block-2.

2.2 Sử dụng while trên một dòng

Nếu khối lệnh của bạn chỉ có một dòng code, bạn có thể viết vòng while của bạn chỉ trên một dòng:

```
1 while True: print("hello")
```

Phía trên là một vòng lặp vô hạn, với điều kiện luôn luôn là True nên vòng lặp sẽ chạy mãi cho tới khi bạn kết thúc chương trình.

3 Cú pháp vòng lặp while-else

Có một sự mới mẻ đối với vòng lặp while trong Python đó là bạn có thể kết hợp thêm từ khóa ELSE để xử lý cho lần lặp không được thực hiện khi điều kiện lặp false.

```
1 while expression:
2     statement(s)
3 else:
4     statement(s)
```

Ví dụ:

```
1 count = 0
2 while count < 5:
3     print(count, " is less than 5")
4     count = count + 1
5 else:
6     print(count, " is not less than 5")
```

Chạy ví dụ này kết quả sẽ như sau:

```
1 0 is less than 5
2 1 is less than 5
3 2 is less than 5
4 3 is less than 5
5 4 is less than 5
6 5 is not less than 5
```

Lần lặp thứ 6 sẽ không xảy ra nên code trong lệnh else sẽ được thực hiện.

4 Bài tập

1. Viết chương trình dùng vòng lặp while để tính tổng các phần tử trong mảng. Mảng được nhập cố định ở đầu chương trình.

```
1 a = [1,2,3,4,5]
2 N = len(a)
3 Tong = 0
4 i = 0
5 while (i < N):
6     Tong = Tong + a[i]
7     i = i + 1;
8 print("Tong cac phan tu trong mang la:", Tong)
```

Chương trình 7.1: Đáp án gợi ý cho Bài 1

2. Viết chương trình dùng vòng lặp while để đếm số lượng số chẵn trong mảng. Mảng được nhập cố định ở đầu chương trình.

```
1 a = [1,2,3,4,5]
2 N = len(a)
3 chan = 0
4 i = 0
5 while (i < N):
6     if a[i] % 2 == 0:
7         chan = chan + 1
8     i = i + 1
9 print("So phan tu chan trong mang la:", chan)
```

Chương trình 7.2: Đáp án gợi ý cho Bài 2

3. Viết chương trình dùng vòng lặp while để đếm số lượng số lẻ trong mảng. Mảng được nhập cố định ở đầu chương trình.

```
1 a = [1,2,3,4,5]
2 N = len(a)
3 le = 0
4 i = 0
5 while (i < N):
6     if a[i] % 2 != 0:
7         le = le + 1
8     i = i + 1
9 print("So phan tu le trong mang la:", le)
```

Chương trình 7.3: Đáp án gợi ý cho Bài 3

4. Viết chương trình dùng vòng lặp while để tìm số lớn nhất trong mảng. Mảng được nhập cố định ở đầu chương trình.

```
1 a = [1,2,3,4,5]
2 N = len(a)
3 max = a[0]
4 i = 0
5 while (i < N):
6     if max < a[i]:
7         max = a[i]
8     i = i + 1
9 print("Phan tu lon nhat trong mang la:", max)
```

Chương trình 7.4: Đáp án gợi ý cho Bài 4

5. Viết chương trình dùng vòng lặp while để tìm số bé nhất trong mảng. Mảng được nhập cố định ở đầu chương trình.

```
1 a = [1,2,3,4,5]
2 N = len(a)
3 min = a[0]
4 i = 0
5 while (i < N):
6     if min > a[i]:
7         min = a[i]
8     i = i + 1
9 print("Phan tu nho nhat trong mang la:", min)
```

Chương trình 7.5: Đáp án gợi ý cho Bài 5

6. Hiện thực lại tất cả các yêu cầu trên bằng vòng lặp while, với mảng được nhập vào từ bàn phím.

```
1 a = []
2 N = int(input("Nhap kich thuoc mang: "))
3 i = 0
4 while (i < N):
5     temp = int(input("Nhap phan tu thu " + str(i+1)))
6     a.append(temp)
7     i = i + 1
8
9 #Tinh tong, tim so chan, le
10 tong = chan = le = 0
11 i = 0
12 while (i < N):
13     tong = tong + a[i]
14     if a[i] % 2 == 0:
15         chan = chan + 1
16     else:
17         le = le + 1
18     i = i + 1
19 #Tim min, max
20 max1 = min1 = a[0]
21 i = 0
```

```

22 while (i < N):
23     if a[i] < min1:
24         min1 = a[i]
25     if a[i] > max1:
26         max1 = a[i]
27     i = i + 1
28 print(tong, chan, le, max1, min1)

```

Chương trình 7.6: Đáp án gợi ý cho Bài 6

5 Câu hỏi ôn tập

1. Giá trị của x là gì

```

1 x = 0
2 while (x > 10):
3     x+=2
4 print(x)

```

- A. 0
- B. 10
- C. 12
- D. Không biết giá trị của x vì vòng lặp vô tận

2. Giá trị của x là gì

```

1 x = 0
2 while (x < 10):
3     x+=2
4 print(x)

```

- A. 8
- B. 100
- C. 101
- D. Không biết giá trị của x vì vòng lặp vô tận

3. Giá trị của x là gì

```

1 x = 0
2 while (x < 50):
3     x-=2
4 print(x)

```

- A. 49
- B. 50
- C. 51
- D. Không biết giá trị của x vì vòng lặp vô tận

4. Trong các lệnh while bên dưới, lệnh while nào sẽ tạo vòng lặp vô tận:

A.

```

1 while True:
2     ...

```


B.

```
1 a = [1,2,3]
2 while a:
3     ...
```

C.

```
1 while "0":
2     ...
```

D. Còn phụ thuộc vào đoạn lệnh bên trong lệnh while.

5. Đoạn lệnh bên dưới in ra màn hình bao nhiêu lần ký tự "a" và ký tự "b" ?

```
1 x = 0
2 while x < 5:
3     print("a")
4     x = x + 2
5 else:
6     print("b")
```

- A. 1 lần "a" và 1 lần "b"
- B. 2 lần "a" và 1 lần "b"
- C. 3 lần "a" và 2 lần "b"
- D. 3 lần "a" và 2 lần "b"

6. Kết quả của s là gì?

```
1 s = ""
2 n = 5
3 while n > 0:
4     n -= 1
5     if (n % 2) == 0:
6         continue
7     a = ['foo', 'bar', 'baz']
8     while a:
9         s += str(n) + a.pop(0)
10        if len(a) < 2:
11            break
```

- A. 3bar3bar1for1for
- B. 3foo3bar1foo1bar
- C. 3foo3bar1bar1for
- D. Tất cả đều sai

Đáp án

1. A 2. A 3. D 4. D 5. B 6. B

CHƯƠNG 8



Các thao tác trên FILE

pythonTM
Package
Index

1 Giới thiệu

Trong một số trường hợp, việc nhập dữ liệu từ bàn phím là không khả thi và mất khá nhiều thời gian, chẳng hạn như đọc dữ liệu từ mảng nhiều chiều ở bài trước. Hay như một ví dụ khác, chúng ta cần lưu lại kết quả của việc thực thi chương trình để xem lại. Trong những trường hợp như vậy, chúng ta sẽ đọc nội dung từ file hoặc lưu kết quả thực thi ra file.

Quy trình làm việc với file thường có 3 bước cơ bản như sau: Mở file, đọc hoặc ghi dữ liệu, cuối cùng là đóng file. Nếu file quên mở, hiển nhiên bạn sẽ không thao tác được gì. Ngược lại, nếu file quên đóng, lần mở tiếp theo có thể chương trình sẽ báo lỗi, tùy thuộc vào mức độ hỗ trợ của chương trình soạn thảo. Tuy nhiên để cho an toàn, hay luôn nhớ đóng lại file ở cuối chương trình.

2 Ghi dữ liệu ra File

Chúng tôi sẽ bắt đầu với hướng dẫn ghi dữ liệu ra file trước. Có thể xem đây là một quá trình xuất dữ liệu (output). Và thông thường, quá trình này sẽ đơn giản hơn so với việc đọc dữ liệu từ file. Một chương trình ví dụ cho việc mở 1 file có tên là **test.txt**, ghi 1 số nội dung và đóng file đó lại như sau:

```
1 file = open("test.txt", "w")
2 file.write("Test ghi file 1")
3 file.write("Test ghi file 2")
4 file.close()
```

Chương trình 8.1: Chương trình mở file ghi nội dung và đóng file

Trong Chương trình 8.1, **file** là một biến số, tham chiếu tới file cần làm việc. Biến file sẽ có hiệu lực sau câu lệnh đầu tiên, là **open**. Câu lệnh open này có 2 tham số quan trọng:

- Tên file: Đặt tên cho file cần lưu dữ liệu. File này sẽ được đặt cùng thư mục với file Python đang làm việc. Trong trường hợp bạn muốn lưu ở một đường dẫn nào đó trong bộ nhớ, chúng ta có thể chỉ định đường dẫn tuyệt đối, ví dụ như là **"D:/test.txt"**. Bạn hãy lưu ý ký tự phân cách trong đường dẫn, là ký tự hướng từ phải sang trái, nó ngược lại với ký tự đường dẫn bình thường.
- Chế độ mở file: Có rất nhiều chế độ mở file trên Python, ở đây chúng tôi chỉ trình bày những chế độ thường dùng nhất, như trình bày bên dưới:
 - "r": Chỉ mở file để đọc.
 - "w": Chỉ mở file để ghi. Nếu file chưa tồn tại thì tạo file mới. Nếu file đã có thì xóa nội dung file cũ và ghi nội dung mới.
 - "a": Chỉ mở file để ghi thêm. Nếu file chưa tồn tại thì tạo file mới. Nếu file đã có thì ghi thêm nội dung vào file.

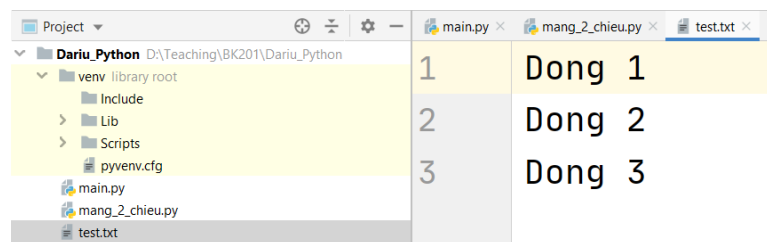
Trong ví dụ ở Chương trình 8.1, chúng ta mở file ở chế độ **"w"**, tức là mở file để ghi. Mới ban đầu, file **test.txt** này không tồn tại, nên hệ thống sẽ tự tạo ra file này và lưu ở cùng thư mục với file python mà chúng ta đang làm việc. Cứ mỗi lần chạy chương trình, nội dung cũ của file này sẽ bị xóa và ghi lại nội dung mới. Trong trường hợp

chúng ta muốn xuống dòng, ký tự "\n" có thể được thêm vào, như ví dụ sau đây:

```
1 file = open("test.txt", "w")
2 file.write("Dong 1\n")
3 file.write("Dong 2" + "\n")
4 file.write("Dong 3")
5 file.close()
```

Chương trình 8.2: Chương trình mở file ghi nội dung và đóng file

Kết quả của file **test.txt** sẽ như sau:



1	Dong 1
2	Dong 2
3	Dong 3

Hình 8.1: Ghi dữ liệu vào file

3 Đọc dữ liệu từ File

Để đọc dữ liệu từ file, chúng ta sẽ mở nó lên với chế độ "r". Một chương trình đọc dữ liệu từ file sẽ được minh họa như bên dưới:

```
1 file = open("test.txt", "r")
2 a = file.read(3)
3 b = file.readline()
4 print(a, b, sep="--")
5 file.close()
```

Chương trình 8.3: Chương trình mở file ghi nội dung và đóng file

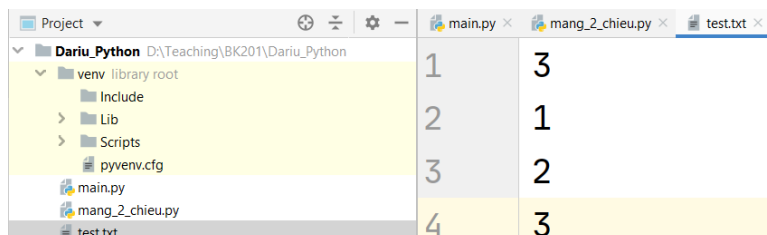
Hai câu lệnh mà chúng ta sẽ thường dùng để đọc giá trị từ file, được thống kê lại như sau:

- `read(n)`: Từ vị trí hiện tại, đọc **n** ký tự tiếp theo, vị trí mới tăng thêm **n** ký tự.
- `readline()`: Từ vị trí hiện tại, đọc cho đến khi xuống dòng, vị trí mới nằm ở dòng tiếp theo.

Như vậy, ta có thể thấy, việc đọc dữ liệu từ file phụ thuộc rất nhiều vào vị trí hiện tại của nó. Sau khi thực hiện một tác vụ đọc dữ liệu, vị trí mới trong file sẽ thay đổi. Trong Chương trình 8.3, biến `a` chỉ có 3 ký tự là "Don", còn biến `b`, sẽ là những ký tự còn lại của dòng 1, tức là "g 1".

4 Đọc mảng 1 chiều từ File

Đây là một yêu cầu thường xuyên sử dụng ở các ngôn ngữ như Pascal hay C. Để có thể đọc 1 mảng 1 chiều từ file, thường chúng ta sẽ phải quy định cấu trúc của file đó. Giả sử chúng ta sẽ quy định dòng đầu tiên của file là số phần tử trong mảng, các dòng tiếp theo là các phần tử của mảng, mỗi dòng chỉ chứa một phần tử. Nội dung của 1 file như vậy sẽ như sau:



1	3
2	1
3	2
4	3

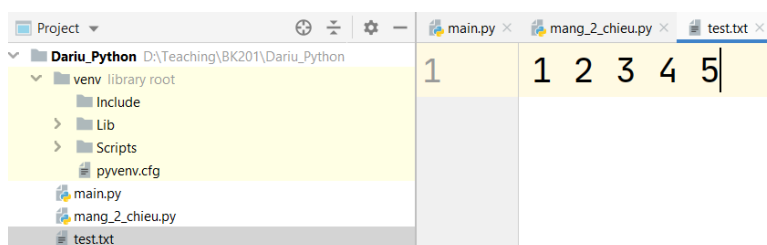
Hình 8.2: File chứa dữ liệu của 1 mảng 1 chiều 3 phần tử

Rõ ràng, việc đọc dữ liệu từ file này và lưu vào một mảng khá đơn giản: đọc số lượng phần tử và dùng 1 vòng for để đọc tiếp. Câu lệnh chính sẽ sử dụng là `readline`, như minh họa sau đây:

```
1 file = open("test.txt", "r")
2 a = []
3 N = int(file.readline())
4 for i in range(0,N):
5     temp = int(file.readline())
6     a.append(temp)
7 print(a)
8 file.close()
```

Chương trình 8.4: Đọc mảng một chiều từ file

Một bài toán phức tạp hơn, đó là trong file không có thông tin về kích thước. Các phần tử của mảng cách nhau bằng khoảng trắng và viết liền nhau trên một hàng như minh họa ở Hình 8.3.



1	1	2	3	4	5
---	---	---	---	---	---

Hình 8.3: Mảng 1 chiều không có thông tin về số phần tử

Với yêu cầu như trên, việc đọc dữ liệu vào với các ngôn ngữ khác có thể sẽ phức tạp. Tuy nhiên với sự hỗ trợ của Python, công việc này cực kì đơn giản. Chúng ta chỉ việc đọc hết dòng dữ liệu, và cắt nó ra bằng câu lệnh **`split()`**. Chương trình gợi ý cho yêu cầu này như sau:

```

1 file = open("test.txt", "r")
2 a = []
3 data = file.readline().split()
4 for i in data:
5     temp = int(i)
6     a.append(temp)
7 print(a)
8 file.close()

```

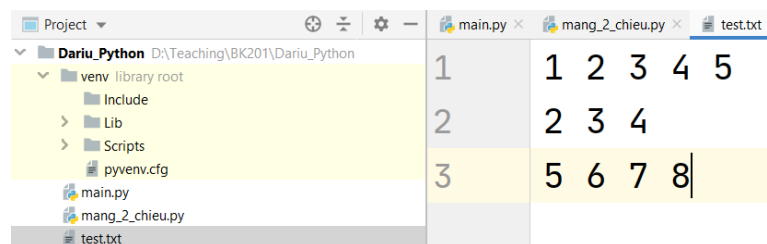
Chương trình 8.5: Đọc mảng không có thông tin về số phần tử

Sau khi dùng toán tử `split()`, bản thân biến `data` đã là mảng 1 chiều. Tuy nhiên, các phần tử của nó đang là chuỗi ký tự. Vòng lặp `for` tiếp theo chỉ đơn giản là chuyển từ kiểu ký tự sang kiểu số nguyên, rồi nối vào mảng `a` của chúng ta. Một lỗi quan trọng mà người dùng có thể gặp phải, là tự nhấn thêm ký tự Enter vào file text. Điều này làm cho việc ép kiểu sang số nguyên sẽ xảy ra lỗi.

Thông qua ví dụ này, chúng tôi cũng muốn trình bày một cách khác để dùng vòng lặp `for`. Với cách dùng này, chúng ta **không quan tâm tới chỉ số của phần tử trong mảng**, mà chỉ quan tâm tới việc duyệt hết mọi phần tử trong mảng mà thôi. Ở cách dùng này, biến `data` được xem là một tập hợp, và `i` sẽ lần lượt là các giá trị trong tập hợp đó.

5 Đọc mảng nhiều chiều từ File

Trong yêu cầu này, chúng ta sẽ đọc một mảng 2 chiều từ file: mỗi hàng của file là một hàng của ma trận và không có ràng buộc là số lượng phần tử của mỗi hàng phải bằng nhau. Dữ liệu của file `test.txt` như sau:



1	1	2	3	4	5
2	2	3	4		
3	5	6	7	8	

Hình 8.4: Dữ liệu mảng nhiều chiều trong file

Chương trình gợi ý cho yêu cầu này sẽ như sau:

```

1 file = open("test.txt", "r")
2 a = []
3
4 for i in file:
5     data = i.split()
6     temp_row = []
7     for j in data:
8         temp = int(j)
9         temp_row.append(temp)
10    a.append(temp_row)

```

```
11 print(a)
12 file.close()
```

Chương trình 8.6: Đọc mảng nhiều chiều từ file

Một lần nữa, cấu trúc mới của vòng for còn được thể hiện cho cả đối tượng file. Chính nhờ sự mềm dẻo trong Python, câu lệnh của chương trình đã được rút ngắn rất nhiều. Những công cụ này được chúng tôi trình bày vì muốn rút ngắn thời gian cho việc xử lý dữ liệu đầu vào, để người học có thể tập trung vào việc xử lý giải thuật tính toán phức tạp hơn. Thực ra, file cũng là một cấu trúc thuộc dạng tập hợp, với mỗi phần tử của nó là một hàng. Do đó, chúng ta mới có thể viết được vòng **for i in file** như trên. Khi đã truy xuất tới từng hàng, việc tách từng phần tử và ép kiểu về số nguyên là tương tự với việc đọc mảng một chiều.

6 Bài tập

1. Đọc giá trị của mảng 1 chiều từ file "input.txt" và xuất kết quả ra màn hình. Giả sử, các giá trị trong file đều nằm trên một dòng, cách nhau bởi 1 khoảng trắng.

```
1 # file input.txt duoc dat chung voi file chua code
2 f = open("input.txt", encoding = 'utf-8')
3 x = f.readline()
4 print(x)
5 f.close()
```

Chương trình 8.7: Đáp án gợi ý cho Bài 1

2. Ghi giá trị từng phần tử của một cho trước trên từng dòng vào file "output.txt"

```
1 a = [1,2,3,4,5]
2 f = open("output.txt","w")
3 for i in range(0,len(a)):
4     f.write(str(a[i]) + "\n")
5 f.close()
```

Chương trình 8.8: Đáp án gợi ý cho Bài 2

3. Đọc giá trị của mảng 1 chiều từ file "input.txt", tính giá trị trung bình của mảng đó và xuất kết quả ra màn hình. Giả sử, các giá trị trong file đều nằm trên một dòng, cách nhau bởi 1 khoảng trắng.

```
1 # file input.txt duoc dat chung voi file chua code
2 f = open("input.txt", encoding = 'utf-8')
3 x = f.readline()
4 a = x.split()
5 tb = 0
6 for i in range(0,len(a)):
7     tb = tb + int(a[i])
8 tb = tb / len(a)
9 print(tb)
10 f.close()
```

Chương trình 8.9: Đáp án gợi ý cho Bài 3

4. Tương tự bài 3 nhưng giá trị cuối cùng được ghi vào file "output.txt"

```
1 # file input.txt duoc dat chung voi file chua code
2 f = open("input.txt", encoding = 'utf-8')
3 x = f.readline()
4 a = x.split()
5 tb = 0
6 for i in range(0,len(a)):
7     tb = tb + int(a[i])
8 tb = tb / len(a)
9 fw = open("output.txt","w")
10 fw.write("Gia tri trung binh la: " + str(tb))
11 f.close()
12 fw.close()
```

Chương trình 8.10: Đáp án gợi ý cho Bài 4

7 Câu hỏi ôn tập

1. Mở file với chế độ mode 'wb' có ý nghĩa gì?
 - A. Mở file để ghi.
 - B. Mở file để đọc và ghi.
 - C. Mở file để ghi cho dạng nhị phân.
 - D. Mở file để đọc và ghi cho dạng nhị phân.

2. Đoạn code dưới đây có ý nghĩa gì?

```
1 f = open("sample.txt")
```

- A. Mở file sample.txt được phép đọc và ghi vào file.
- B. Mở file sample.txt và chỉ được phép đọc file.
- C. Mở file sample.txt và được phép ghi đè vào file
- D. Mở file sample.txt và được phép ghi tiếp vào file.

3. Đoạn code dưới đây có ý nghĩa gì?

```
1 f = open("sample.txt", "a")
```

- A. Mở file sample.txt được phép đọc và ghi vào file.
- B. Mở file sample.txt và chỉ được phép đọc file.
- C. Mở file sample.txt và được phép ghi đè vào file
- D. Mở file sample.txt và được phép ghi tiếp vào file.

4. Điều gì sẽ xảy ra khi mở một file không tồn tại?
 - A. Python tự động tạo một file mới dưới tên bạn đang gọi ra.
 - B. Không có gì xảy ra vì file không tồn tại.
 - C. Báo lỗi
 - D. Không có đáp án nào đúng

5. Cho cây thư mục như sau:

```
University/  
|  
├─ ClassA/  
|   ├─ student1.gif  
|   └─ student2.gif  
|  
├─ ClassB/  
|   └─ student3.gif  
|  
└─ University.csv
```

Hãy cho biết đường dẫn tuyệt đối đến tệp student1.gif là đường dẫn nào sau đây? Giả sử đường dẫn chỉ bắt đầu từ University.

- A. University/student1.gif
- B. University/ClassA/student1.gif.
- C. /student1.gif
- D. University/

6. Câu lệnh nào sau đây sẽ đọc file từ vị trí hiện tại cho đến khi xuống dòng, vị trí mới nằm ở dòng tiếp theo?
- A. `read()`
 - B. `readline()`
 - C. `read(n)`
 - D. Tất cả các đáp án trên đều sai.
7. Câu lệnh nào sau đây sẽ đọc n ký tự tiếp theo từ vị trí hiện tại, vị trí mới tăng thêm n ký tự?
- A. `read()`
 - B. `readline(n)`
 - C. `read(n)`
 - D. Tất cả các đáp án trên đều sai.

Đáp án

1. C 2. B 3. D 4. C 5. B 6. B 7. C

CHƯƠNG 9



Hàm và lời gọi hàm

pythonTM
Package
Index

1 Giới thiệu

Trong các bài học trước chúng ta đã từng sử dụng một số hàm có trong Python như `print()`, `open()`, `write()`... Vậy hàm là gì? Trong lập trình, hàm là một nhóm bao gồm một hoặc nhiều câu lệnh và được dùng để thực hiện một số tác vụ nhất định.

Hàm được chia thành hai nhóm:

- Hàm có sẵn (built-in function): là những hàm được cung cấp sẵn bởi ngôn ngữ Python. Ví dụ như các hàm `print()`, `range()`, `max()`... Chúng ta không sửa đổi logic bên trong các hàm này.
- Hàm tự định nghĩa (user-defined function): là những hàm được định nghĩa bởi các lập trình viên. Lập trình viên sẽ tự khai báo logic bên trong các hàm tự định nghĩa.

2 Định nghĩa hàm

Trong Python các hàm (tự định nghĩa) được định nghĩa sử dụng cú pháp sau:

```
1 def <function_name>(<parameters>):  
2     <statements>
```

Trong đó:

- `<function_name>`: là tên của hàm được định nghĩa.
- `<parameters>`: (tham số) là danh sách các biến số đặc biệt được sử dụng bên trong hàm. Giá trị của từng tham số được xác định khi chúng ta sử dụng hàm (hay **gọi hàm**). Một hàm được định nghĩa với một hoặc nhiều hoặc không có tham số nào.
- `<statement>`: bao gồm các câu lệnh sẽ được thực thi khi gọi hàm.

Ví dụ một hàm tự định nghĩa `helloPython()` như sau:

```
1 def helloPython():  
2     print("Hello, Python!")  
3  
4 helloPython() # Hiện thị: Hello, Python!
```

Ví dụ hàm `hello()` dưới đây được định nghĩa với một **tham số** đầu vào với tên là `language`:

```
1 def hello(language):  
2     print("Hello, %s!" % (language))
```

3 Gọi Hàm

Sau khi được định nghĩa thì để sử dụng hàm chúng ta sẽ thực hiện việc gọi hàm. Gọi hàm được thực hiện đơn giản thông qua cú pháp sau:

```
1 <function_name>(<arguments>)
```

Trong đó:

- `< function_name >` là tên hàm được gọi.
- `< arguments >` (đối số) là các giá trị truyền vào tương ứng với các tham số được định nghĩa bởi hàm được gọi.

Ví dụ sau sẽ gọi hàm `hello()` được định nghĩa ở ví dụ trên với đối số truyền vào là Python:

```
1 hello("Python") # Hien thi: Hello , Python!
```

Số lượng đối số truyền vào khi gọi hàm cần phải tương ứng với số lượng tham số sử dụng khi định nghĩa hàm. Nếu như số lượng tham số và đối số không khớp nhau thì Python sẽ báo lỗi khi chạy.

Ví dụ nếu bạn gọi hàm `hello()` được định nghĩa ở trên như sau:

```
1 hello()
```

Bạn sẽ thấy Python báo về lỗi:

```
1 Traceback (most recent call last):
2   File "function.py", line 4, in <module>
3     hello()
4 TypeError: hello() takes exactly 1 argument (0 given)
```

3.1 Đối Số Mặc Định

Khi định nghĩa một hàm có sử dụng tham số chúng ta có thể gán giá trị mặc định (hay đối số mặc định) cho các tham số.

Trong ví dụ dưới đây thì hàm `hello()` được định nghĩa với một tham số `language` có giá trị mặc định là Python:

```
1 def hello(language = "Python"):
2     print("Hello, %s!" % (language))
```

Khi gọi hàm `hello()` ở ví dụ trên, nếu như bạn không truyền vào đối số nào thì đối số mặc định sẽ được sử dụng:

```
1 hello() # Hien thi: Hello , Python!
```

3.2 Giá Trị Trả Về

Ở các ví dụ phía trước thì hàm `hello()` được định nghĩa để thực hiện tác vụ hiển thị thông báo ra màn hình. Tuy nhiên trong nhiều trường hợp chúng ta lại không muốn hiển thị bất kỳ thông báo nào khi gọi hàm, thay vào đó chúng ta cần lấy ra giá trị trả về từ hàm. Để thực hiện việc này chúng ta sẽ sử dụng từ khoá `return`.

Ví dụ:

```
1 def sum(a, b):
2     return a + b;
```

Ở ví dụ trên hàm `sum()` sẽ trả về giá trị tổng của hai tham số `a` và `b`. Chúng ta có thể sử dụng giá trị trả về này để hiển thị một thông báo hoặc gán cho một biến khác...

```
1 number_1 = 3
2 number_2 = 4
3
4 def sum(a, b):
5     return a + b
6
7 total = sum(number_1, number_2)
8 print("Tổng của %s và %s là %s" %(number_1, number_2, total
9     ))
```

Giá trị trả về khi gọi hàm `sum()` ở trên được gán vào biến `total`. Và như vậy khi chạy đoạn code trên khi chạy sẽ hiển thị kết quả: *Tổng của 3 và 4 là 7*.

Các hàm không sử dụng `return` hoặc sử dụng `return` (không có giá trị nào tiếp theo sau từ khoá) sẽ trả về giá trị là `None`:

```
1 def hello_1():
2     print("Hello 1!")
3
4 def hello_2():
5     print("Hello 2!")
6     return
7
8 returnValue = hello_1()
9 print(returnValue) # Hien thi: None
10 print(type(returnValue)) # Hien thi: <type 'NoneType'>
11
12 return_value = hello_2()
13 print(returnValue) # Hien thi: None
14 print(type(returnValue)) # Hien thi: <type 'NoneType'>
```

Một lưu ý khác khi sử dụng `return` trong hàm đó là một khi hàm đã `return` (trả về giá trị) thì Python sẽ chấm dừng quá trình chạy hàm. Ví dụ:

```
1 def hello():
2     print("Hello!")
3     return
4     print("Nice to meet you!")
5
6 hello() # Hien thi: Hello!
```

Trong đoạn code trên khi gọi hàm `hello()` thì chỉ có câu lệnh `print("Hello!")` đầu tiên được chạy.

3.3 Câu lệnh `pass`

Hàm tự định nghĩa không thể bị bỏ trống (tức là không thể không có hàm), tuy nhiên vì một số lý do nào đó mà hàm tự định nghĩa không có nội dung. Thì câu lệnh `pass` sẽ giúp cho việc định nghĩa hàm không bị lỗi. Ví dụ:


```

1 def myfunction():
2     pass

```

4 Viết hàm để tính giai thừa

Dựa vào những gì được biết về hàm, thì chúng ta sẽ tiến hành một ví dụ về cách định nghĩa hàm. Ví dụ: Hàm tính giai thừa. Và chúng ta sẽ thực hiện việc định nghĩa hàm tính giai thừa theo 2 cách là vòng lặp for và đệ quy.

Trong toán học, giai thừa là một toán tử một ngôi trên tập hợp các số tự nhiên. Cho n là một số tự nhiên dương, " n giai thừa", ký hiệu $n!$ là tích của n số tự nhiên dương đầu tiên.

$$n! = 1 \times 2 \times 3 \times \dots \times n$$

Ví dụ: $4! = 1.2.3.4 = 24$

$8! = 1.2.3.....7.8 = 40320$

Đặc biệt, với $n = 0$, người ta quy ước $0! = 1$.

4.1 Viết hàm sử dụng vòng lặp for

```

1 def tinhgiaithua(n):
2     giai_thua = 1;
3     if (n == 0 or n == 1):
4         return giai_thua;
5     else:
6         for i in range(2, n + 1):
7             giai_thua = giai_thua * i;
8         return giai_thua;
9
10 n=10
11 print("Giai thua cua", n, "la", tinhgiaithua(n));

```

Kết quả là: *Giai thua của 10 là 3628800*

4.2 Viết hàm sử dụng đệ quy

```

1 def tinhgiaithua(n):
2     if n == 0:
3         return 1
4     return n * tinhgiaithua(n - 1)
5
6 n=9
7 print("Giai thua cua", n, "la", tinhgiaithua(n));

```

Kết quả là: *Giai thua của 9 là 362880*

5 Bài tập

1. Viết hàm printMax, nhận hai giá trị đầu vào a và b. Sau đó in ra màn hình số lớn hơn. Nếu hai số bằng nhau thì chỉ cần in ra màn hình 1 số.

```
1 def printMax(a, b):
2     if a > b:
3         print(a)
4     elif a == b:
5         print(b)
6     else:
7         print(b)
8 a = int(input("Nhap so thu nhat: "))
9 b = int(input("Nhap so thu hai: "))
10 printMax(a, b)
```

Chương trình 9.1: Đáp án gợi ý cho Bài 1

2. Cho Dãy Fibonacci được tính dựa trên công thức sau:

$f(n) = 0$ nếu $n = 0$

$f(n) = 1$ nếu $n = 1$

$f(n) = f(n-1) + f(n-2)$ nếu $n > 1$

Hãy viết hàm đệ quy tính giá trị của $f(n)$ với n là số được người dùng nhập và trả về giá trị tương ứng. Ví dụ: Nếu n được nhập vào là 7 thì chương trình sẽ trả về 13.

```
1 def f(n):
2     if n == 0: return 0
3     elif n == 1: return 1
4     else: return f(n-1)+f(n-2)
5
6 n = int(input("Nhap n: "))
7 a = f(n)
8 print(a)
```

Chương trình 9.2: Đáp án gợi ý cho Bài 1

3. Viết hàm tính lũy thừa x mũ y (không được dùng toán tử $**$), với x, y nhập từ bàn phím và $x, y > 0$. Nếu x hoặc y nhỏ hơn 0 thì trả về 0.

```
1 def power(x,y):
2     ketqua = 1
3     if x < 0 or y < 0:
4         ketqua = 0
5     else:
6         for i in range (0,y):
7             ketqua = ketqua * x
8     return ketqua
9
10 x = int(input("Nhap co so: "))
11 y = int(input("Nhap so mu: "))
12 a = power(x,y)
13 print(a)
```

6 Câu hỏi ôn tập

1. Chọn đáp án đúng: Phát biểu nào chính xác khi nói về Hàm trong Python?
 - A. Hàm có thể được tái sử dụng trong chương trình.
 - B. Sử dụng hàm không có tác động tích cực gì đến các module trong chương trình.
 - C. Không thể tự tạo các hàm của riêng người viết chương trình.
 - D. Tất cả các đáp án trên đều đúng.
2. Từ khóa nào được sử dụng để bắt đầu hàm?
 - A. Fun
 - B. Define
 - C. Def
 - D. Function

3. Output của chương trình dưới đây là gì?

```
1 def sayHello():  
2     print('Hello World!')  
3 sayHello()  
4 sayHello()
```

A.
Hello World!
Hello World!

B.
'Hello World!'
'Hello World!'

C.
Hello
Hello

D. Không có đáp án đúng

4. Output của chương trình dưới đây là gì?

```
1 def printMax(a, b):  
2     if a > b:  
3         print(a, 'is maximum')  
4     elif a == b:  
5         print(a, 'is equal to', b)  
6     else:  
7         print(b, 'is maximum')  
8 printMax(3, 4)
```

A. 3
B. 4
C. 4 is maximum
D. Không có đáp án đúng

5. Output của chương trình dưới đây là gì?

```
1 x = 50
2     def func(x):
3         print('Gia tri cua x la', x)
4         x = 2
5         print('Gia tri cua x duoc thay doi thanh', x)
6 func(x)
7 print('Gia tri hien tai cua x la', x)
```

- A. Gia tri hien tai cua x la 50
- B. Gia tri hien tai cua x la 100
- C. Gia tri hien tai cua x la 2
- D. Không có đáp án đúng

6. Output của chương trình dưới đây là gì?

```
1 x = 50
2     def func():
3         global x
4         print('Gia tri cua x la', x)
5         x = 2
6         print('Gia tri cua x duoc thay doi thanh', x)
7 func()
8 print('Gia tri hien tai cua x la', x)
```

- A.
Gia tri cua x la 50
Gia tri cua x duoc thay doi thanh 2
Gia tri hien tai cua x la 50
- B.
Gia tri cua x la 50
Gia tri cua x duoc thay doi thanh 2
Gia tri hien tai cua x la 2
- C.
Gia tri cua x la 50
Gia tri cua x duoc thay doi thanh 50
Gia tri hien tai cua x la 50
- D. Không có đáp án đúng

7. Output của chương trình dưới đây là gì?

```
1 def maximum(x, y):
2     if x > y:
3         return x
4     elif x == y:
5         return 'Cac so bang nhau'
6     else:
7         return y
8 print(maximum(2, 3))
```

- A. 2
- B. 3
- C. Các số bằng nhau
- D. Không có đáp án đúng

8. Đây là kết quả của chương trình dưới đây?

```
1 def outerFunction():
2     global a
3     a = 20
4     def innerFunction():
5         global a
6         a = 30
7         print('a =', a)
8 a = 10
9 outerFunction()
10 print('a =', a)
```

- A. a = 10 a = 30
- B. a = 10
- C. a = 20
- D. a = 30

9. Output của chương trình dưới đây là gì?

```
1 def cube(x):
2     return x * x * x
3 x = cube(3)
4 print x
```

- A. 9
- B. 3
- C. 27
- D. 30

10. Output của chương trình dưới đây là gì?

```
1 def C2F(c):
2     return c * 9/5 + 32
3 print C2F(100)
4 print C2F(0)
```

- A.
212
32
- B.
314
24
- C.
567

D. Không có đáp án đúng

Đáp án

1. A 2. C 3. A 4. C 5. A 6. B 7. B 8. C 9. C 10. A

CHƯƠNG 10



Cấu trúc dữ liệu nâng cao trên Python

pythonTM
Package
Index

1 Giới thiệu

Ở các phần trước, chúng ta đã tìm hiểu qua các kiểu dữ liệu cơ bản như kiểu số nguyên, số thập phân hay kể cả list. Đối với các kiểu dữ liệu cơ bản này, chúng ta đã hoàn toàn có thể viết các ứng dụng đơn giản với Python. Tuy nhiên, đối với các ứng dụng có số lượng các biến nhiều, dữ liệu lớn và phức tạp, chúng ta cần tới các cấu trúc dữ liệu nâng cao hơn để giúp cho việc viết code ngắn gọn và hiệu quả.

2 Chuỗi (string)

Chuỗi ký tự (string) là một kiểu dữ liệu rất hay dùng trong Python, một từ, một đoạn văn bản đều là kiểu chuỗi. Chuỗi trong Python được đánh dấu bằng dấu nháy đơn ' hoặc nháy kép ", tuy nhiên nếu bắt đầu bằng dấu nào thì phải kết thúc bằng dấu đấy. Ví dụ:

```
1 a = "Xin chào"
2 b = 'Hello\n' # \n nghĩa là dòng mới
3
4 # c = 'Hi", cách khai báo này không dùng
```

Chương trình 10.1: Ví dụ cho cách khởi tạo string

Python hỗ trợ kiểu dữ liệu chuỗi có thể chứa đoạn văn với nhiều dòng bằng cách bắt đầu và kết thúc chuỗi bằng 3 dấu nháy kép """. Ví dụ:

```
1 print("""\
2     Xin chào
3     Hello
4 """)
```

Chương trình 10.2: Viết chuỗi trên nhiều dòng

Chuỗi có thể được xem như một list các ký tự, nên cách duyệt và truy xuất vào các phần tử của chuỗi tương tự như cách chúng ta làm với list (dùng index). Cụ thể như sau:

```
1 a = "Xin chào"
2 for i in range(0, len(a)):
3     print(a[i])
```

Chương trình 10.3: Duyệt và truy xuất phần tử trong chuỗi

2.1 Nối chuỗi, thay đổi hoặc xóa chuỗi

Các chuỗi có thể được nối với nhau bằng toán tử + và thay thế bằng *:

```
1 a = "Xin chào"
2 b = "Việt Nam"
3 c = a + b # c = "Xin chào Việt Nam"
```

Chương trình 10.4: Phép nối hai chuỗi

Các chuỗi Python không thể thay đổi - chúng là cố định. Vì vậy, nếu cứ cố tình gán một ký tự nào đó cho vị trí đã được lập chỉ mục thì sẽ nhận được thông báo lỗi. Nên nếu cần một chuỗi mới, cách tốt nhất là tạo mới.

```
1 a = "Hello"
2 # a[1] = a, lệnh gan nay se bao loi
```

Chương trình 10.5: Không thể thay đổi giá trị trong chuỗi

2.2 Phương thức dùng với biến kiểu chuỗi

Có rất nhiều phương thức được tích hợp sẵn trong Python để làm việc với string như:

- `lower()`: Chuyển tất cả các chữ cái trong chuỗi thành chữ thường
- `upper()`: Chuyển tất cả các chữ cái trong chuỗi thành chữ hoa
- `split(x)`: Trả về một list bằng cách chia các phần tử bằng ký tự "x", nếu để trống sẽ mặc định là ký tự khoảng trắng.
- `find(x)`: Trả về số lần chuỗi x được tìm thấy trong chuỗi gốc, trả về -1 nếu không tìm thấy.
- `replace(x,y)`: Thay thế chuỗi x thành chuỗi y trong chuỗi gốc.
- ...

3 Cấu trúc dữ liệu Tuple

Python hỗ trợ một cấu trúc dữ liệu tương tự với List, có tên là Tuple. Tuy nhiên, khác với list, Tuple là một danh sách bất biến. Nghĩa là ngay sau khi khởi tạo Tuple, chúng ta không thể thay đổi nó.

3.1 Khởi tạo và truy xuất các phần tử trong Tuple

Tuple được khai báo trên Python bằng chuỗi ký tự (), ở giữa là các phần tử của mảng, cách nhau bằng dấu phẩy (.). Chúng ta có thể bỏ dấu ngoặc đơn nếu muốn, nhưng nên thêm nó vào cho code rõ ràng hơn. Ngoài ra, tuple không bị giới hạn số lượng phần tử và có thể có nhiều kiểu dữ liệu khác nhau như số nguyên, số thập phân, list, string,...

Tuy nhiên, nếu tạo tuple theo cách thông thường là cho phần tử đó vào trong cặp dấu () là chưa đủ, cần phải thêm dấu phẩy để chỉ ra rằng, đây là tuple. Hãy xem ví dụ sau đây

```
1 a = (1, 2, 3, 4)
2 #tuple co the khoi tao ma khong can dau ()
3 b = 5,6,7,8
4 #luu y khi khoi tao tuple
5 c = ("Xin chao") # kieu du lieu cua c la str
```

```
6 d = ("Xin chào",) # kiểu dữ liệu của d mới là tuple
```

Chương trình 10.6: Cách để khởi tạo tuple

Tương tự như list, để truy xuất các phần tử bên trong tuple, chúng ta sử dụng toán tử index [] với index bắt đầu bằng 0.

```
1 a = (1, 2, 3, 4)
2 print("Phần tử thứ nhất: ", a[0])
3 print("Phần tử thứ hai: ", a[1])
```

Chương trình 10.7: Truy xuất các phần tử trong tuple

3.2 Các thao tác với Tuple

Chúng ta có thể sử dụng tất cả các kỹ thuật, các hàm tương tự với cách chúng ta sử dụng với list. Tuy nhiên, tuple là một danh sách bất biến, không thể thay đổi khi đã tạo ra nên chúng ta sẽ loại trừ những hàm tác động thay đổi nội dung.

Ví dụ: chúng ta có thể sử dụng hàm len() để duyệt các phần tử trong tuple, hoặc index(x) để trả về giá trị index của phần tử x đầu tiên mà nó gặp trong tuple:

```
1 a = (1, 2, 3, 4)
2 print(a.index(1)) # kết quả là 0
3 N = len(a)
4 for i in range(0, N):
5     print(a[i])
```

Chương trình 10.8: Duyệt các phần tử trong tuple

Còn lại, tất cả các phương thức như: append(), insert(), pop(), extend(), remove(), sort(), reverse(),... không sử dụng được với tuple.

Tuy nhiên, nếu bản thân các phần tử đó là một kiểu dữ liệu có thể thay đổi (như list chẳng hạn) thì các phần tử lồng nhau có thể được thay đổi. Cụ thể, hãy xem ví dụ minh họa bên dưới:

```
1 a = (1, 2, 3, [4,5])
2 # Nếu thay đổi giá trị của tuple bằng cách gán a[0] = 7 thì
   # sẽ báo lỗi
3 # Chỉ có thể thay đổi giá trị của các phần tử trong list
   # [4,5], vì list có thể thay đổi
4 a[3][0] = 5;
5 print(a)
```

Chương trình 10.9: Thay đổi các giá trị bên trong tuple

3.3 Khi nào sử dụng Tuple

Tuple có những hạn chế nhất định như khi tạo ra thì không thể thay đổi được, nhưng cũng có những ưu điểm đáng kể đến như sau:

- Thứ nhất, tuple có tốc độ xử lý nhanh hơn list. Ngoài ra, khi chúng ta muốn định nghĩa một tập các giá trị là hằng số và sau đó duyệt qua tập hợp này thì nên chọn tuple.

- Sử dụng Tuple giúp code an toàn hơn, bởi vì đặc tính của tuple giúp cho dữ liệu không thể thay đổi. Do vậy nên lựa chọn tuple cho những dữ liệu dạng hằng số, dữ liệu không thay đổi theo thời gian.

4 Cấu trúc dữ liệu tập hợp (Set)

Set (hay còn gọi là tập hợp) là tập hợp có thể chứa nhiều các phần tử và các phần tử này không có thứ tự, vị trí của nó hỗn loạn trong tập hợp..

Chúng ta có thể duyệt qua các phần tử trong tập hợp, có thể thêm hoặc xóa đi các phần tử và thực hiện các phép toán tập hợp như phép hợp (union), phép giao (intersection), phép hiệu (difference)... Bên cạnh đó, các phần tử của tập hợp phải là các dữ liệu không thể thay đổi như một số (int), một chuỗi (string), hoặc một Tuple.

4.1 Khởi tạo và truy xuất các phần tử trong tập hợp

Chúng ta cần chú ý các đặc tính của tập hợp như sau:

- Các phần tử trong tập hợp không có thứ tự.
- Các phần tử này là duy nhất, không cho phép lặp lại.
- tập hợp có thể thay đổi (thêm bớt phần tử) nhưng các phần tử của tập hợp phải ở dạng không thể thay đổi.

Các phần tử trong tập hợp phân cách nhau bằng dấu phẩy và nằm trong dấu ngoặc nhọn . Và do tính chất của tập hợp nên chúng ta không thể dùng toán tử index [] để truy xuất các phần tử mà phải truy xuất trực tiếp các phần tử đó. Cụ thể như ví dụ bên dưới:

```

1 a = {1, 2, 3, 4}
2 b = {{7.0, "Xin chào", (4, 5, 6)}}
3 c = set() # cách tạo một tập hợp rỗng
4
5 #duyet va in tat ca phan tu trong tap hop
6 for x in b:
7     print(x)

```

Chương trình 10.10: Khởi tạo và duyệt phần tử trong tập hợp

4.2 Thay đổi tập hợp

Python hỗ trợ rất nhiều các phương thức để thực hiện thao tác thay đổi tập hợp và chú ý là kết quả có thể khác đi do tập hợp không sắp xếp các phần tử theo một trật tự nào cả.

- add(): Phương thức sử dụng để thêm một phần tử vào tập hợp.
- remove(): Loại bỏ một phần tử trong tập hợp và báo lỗi nếu phần tử đó không tồn tại.

- `discard()`: Tương tự như `remove()` nhưng nếu phần tử đó không tồn tại sẽ không báo lỗi gì cả.
- `pop()`: Loại bỏ một phần tử ngẫu nhiên khỏi tập hợp.
- `clear()`: Loại bỏ tất cả các phần tử trong tập hợp.
- `update()`: Dùng để thêm nhiều phần tử vào tập hợp.

4.3 Các phép toán trong tập hợp

Các tập hợp có lợi thế hơn các cấu trúc dữ liệu khác ở chỗ nó thực hiện được các phép toán tập hợp như hợp, hiệu, giao... Và để hiểu rõ hơn về các phép toán được dùng phổ biến trong các tập hợp, chúng ta xem xét các ví dụ sau:

4.3.1 Phép hợp (Union)

Hợp của hai tập hợp cho kết quả là tất cả các phần tử trong hai tập hợp, chú ý phần tử nào lặp lại sẽ chỉ xuất hiện 1 lần trong tập kết quả.

```
1 a = {1, 2, 3}
2 b = {3, 4}
3 c = a.union(b)
4 print(c) # ket qua la {1,2,3,4}
```

Chương trình 10.11: Phép hợp trong tập hợp

4.3.2 Phép giao (Intersection)

Phép giao hai tập hợp cho kết quả là các phần tử đồng thời thuộc cả hai tập hợp.

```
1 a = {1, 2, 3}
2 b = {3, 4}
3 c = a.intersection(b)
4 print(c) # ket qua la {3}
```

Chương trình 10.12: Phép giao trong tập hợp

4.3.3 Phép hiệu (Difference)

Hiệu của một tập A trừ đi một tập B cho kết quả là tất cả các phần tử thuộc A nhưng không thuộc B.

```
1 a = {1, 2, 3}
2 b = {3, 4}
3 c = a.difference(b)
4 print(c) # ket qua la {1, 2}
```

Chương trình 10.13: Phép hiệu trong tập hợp

4.3.4 Hiệu đối xứng của hai tập hợp (Symmetric difference)

Hiệu đối xứng của hai tập A và B được kết quả là tập hợp các phần tử thuộc cả A và B nhưng không đồng thời thuộc cả tập A và B.

```
1 a = {1, 2, 3}
2 b = {3, 4}
3 c = a.symmetric_difference(b)
4 print(c) # kết quả là {1, 2, 4}
```

Chương trình 10.14: Phép hiệu đối xứng của hai tập hợp

4.4 Khi nào sử dụng tập hợp

Tập hợp cũng là một cấu trúc dữ liệu giống như với list và tuple nhưng nó lại được xây dựng sẵn các phép toán tập hợp, do vậy nếu chương trình của chúng ta cần thực hiện các phép toán như: union(), intersection(), ... thì chúng ta có thể cân nhắc dùng đến tập hợp. Ngoài ra cũng cần phải lưu ý các đặc tính của tập hợp để áp dụng vào các bài toán sao cho hợp lý.

5 Cấu trúc dữ liệu từ điển (Dictionary)

Dictionary (hay còn gọi là từ điển) bao gồm rất nhiều các phần tử mà mỗi phần tử đi theo cặp khóa - giá trị (key-value). Từ điển thường được sử dụng khi chúng ta có một số lượng lớn dữ liệu và muốn tối ưu hóa chúng để trích xuất dữ liệu với điều kiện phải biết được khóa để lấy giá trị.

5.1 Khởi tạo và truy xuất các phần tử trong từ điển

Trong python, từ điển được định nghĩa trong dấu ngoặc nhọn tương tự như tập hợp. Nhưng bên trong là các phần tử theo cặp (khóa và giá trị) được phân tách bởi dấu phẩy, phân tách giữa khóa và giá trị của từng phần tử bởi dấu hai chấm. Cụ thể:

```
1 a = dict({"A": 24, "B": 30, "C": 27})
2 b = dict([("D", 24), ("E", 30), ("F", 27)])
3 # duyệt lan lượt các khóa trong từ điển a
4 for key in a:
5     print(key)
```

Chương trình 10.15: Cách để khởi tạo từ điển

Lưu ý:

- Khóa của từ điển có thể là chuỗi hoặc số, khóa này là duy nhất ở từng cấp.
- Giá trị của từ điển có thể số, chuỗi hoặc các cấu trúc list, tuple, tập hợp hoặc thậm chí là một từ điển khác.

Chúng ta có thể truy xuất đến các phần tử trong từ điển thông qua các khóa, cụ thể như sau:

```

1 a = dict({"A": 24, "B": 30, "C": {"D": 21}})
2 print(a[A]) # ket qua la 24
3 #truy xuất phần tử của các tu diện lồng nhau
4 print(a["C"]["D"]) #ket qua la 21

```

Chương trình 10.16: Truy xuất các phần tử trong từ điển

5.2 Thêm và cập nhật phần tử trong từ điển

Từ điển có thể thay đổi, nên chúng ta có thể thêm mới hoặc thay đổi giá trị của các phần tử hiện có bằng cách sử dụng toán tử gán. Và bởi vì từ điển truy xuất các phần tử thông qua khóa, nên khi gán giá trị cho một phần tử thì:

- Khóa tồn tại thì giá trị phần tử được cập nhật.
- Khóa không tồn tại thì từ điển được thêm một phần tử có khóa và giá trị trong câu lệnh.

```

1 dict = {"a": 1, "b": 2}
2 # thêm một phần tử vào từ điển dict
3 dict["c"] = 2 # ket qua: {'a': 1, 'b': 2, 'c': 2}
4 # cập nhật giá trị của một phần tử
5 dict["c"] = 3 # ket qua: {'a': 1, 'b': 2, 'c': 3}

```

Chương trình 10.17: Thêm và cập nhật phần tử trong từ điển

5.3 Xóa phần tử khỏi từ điển

Giống như list, có nhiều phương thức có sẵn dùng để xóa một phần tử khỏi từ điển như pop(), popitem(), clear(), del. Chức năng của các phương thức trên như sau:

- pop(): Xóa phần tử có key đã cho và trả về giá trị của phần tử đó.
- popitem(): Xóa và trả về một phần tử ngẫu nhiên dưới dạng (key, value)
- clear(): Xóa tất cả các phần tử trong từ điển
- del: Xóa một phần tử hoặc xóa hẳn biến chứa từ điển.

```

1 dict = {"a": 1, "b": 2, "c": 3}
2 a = dict.pop("a") # a = 1, dict = {"b": 2, "c": 3}
3 del dict["c"] #dict = {"b": 2}
4 dict.clear() # dict = {}
5 del dict
6 print(dict) #dict không tồn tại

```

Chương trình 10.18: Các phương pháp xóa phần tử khỏi từ điển

5.4 Một số phương thức hay dùng trong từ điển

- Phương thức `get()`: Trả về giá trị một phần tử trong từ điển với một khóa cho trước.
- Phương thức `key()` và `value`: Trả về danh sách khóa và danh sách các giá trị của từ điển

```
1 dict = {"a": 1, "b": 2, "c": 3}
2 a = dict.get("a") # a = 1
3 b = dict.keys() # b = ['a', 'b', 'c']
4 c = dict.values() # c = [1, 2, 3]
```

Chương trình 10.19: Phép hiệu đối xứng của hai tập hợp

5.5 Khi nào sử dụng từ điển

Chúng ta có thể cân nhắc sử dụng từ điển khi dữ liệu thay đổi liên tục, có sự liên kết giữa khóa và giá trị hay quan trọng là muốn tăng hiệu quả khi truy xuất dữ liệu thông qua khóa.

6 Bài tập

1. Nhập 2 chuỗi bất kỳ từ bàn phím, nối 2 chuỗi đó lại với nhau. In chuỗi kết quả và cho biết chuỗi đó có bao nhiêu ký tự.

```
1 a = input("nhap chuoi thu nhat: ")
2 b = input("nhap chuoi thu hai: ")
3 ketqua = a + b
4 print(ketqua)
5 print("So ky tu cua chuoi tren la: ", len(ketqua))
```

Chương trình 10.20: Đáp án gợi ý bài 1

2. Viết một chương trình để tạo một tuple khác, chứa các giá trị là số chẵn trong 1 tuple cho trước.

Gợi ý: Sử dụng `tuple()` để tạo tuple từ list.

```
1 a = (1,2,3,4,5,6,7,8)
2 tp = list()
3 for i in range(0, len(a)):
4     if a[i]%2==0:
5         tp.append(a[i])
6 tp1 = tuple(tp)
7 print(tp1)
```

Chương trình 10.21: Đáp án gợi ý bài 2

3. Viết một chương trình xóa các phần tử trùng nhau giữa 2 tập hợp cho trước:

```
1 a = {1,2,3,4,5,6,7,8}
2 b = {2,4,6,8}
3 tp = a.difference(b)
```

```
4 print(tp)
```

Chương trình 10.22: Đáp án gợi ý bài 3

4. Nhập số nguyên N từ bàn phím, hãy viết chương trình để tạo ra một từ điển chứa (i, i*i) như là số nguyên từ 1 đến N (bao gồm cả 1 và n) sau đó in ra từ điển này.

Ví dụ: Giả sử số n là 8 thì đầu ra sẽ là: 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64.

```
1 n=int(input("Nhap vao so nguyen N:"))
2
3 d=dict()
4 for i in range(1,n+1):
5     d[i]=i*i
6 print(d)
```

Chương trình 10.23: Đáp án gợi ý bài 4

7 Câu hỏi ôn tập

1. Cho đoạn code sau:

```
1 a = "Xin chao Viet Nam"
2 print(a.find("i"))
```

Kết quả của đoạn lệnh trên là gì?

- A. 1
- B. 2
- C. 3
- D. i

2. Cách nào sau đây dùng để khai báo một tuple:

- A. a = [1,2,3,4]
- B. a = (1,)
- C. a = (1,2,3,4)
- D. a = (1)

3. Cho đoạn code sau:

```
1 a = (1, 2, 3, 4)
2 a[1] = 3
3 print(a)
```

Kết quả đoạn lệnh trên sẽ là:

- A. (1,3,3,4)
- B. (3,2,3,4)
- C. (1,2,3,4)
- D. Báo lỗi

4. Có thể truy xuất các phần tử của một tập hợp thông qua toán tử index []. Nhận định trên đúng hay sai?

- A. Đúng
- B. Sai


```
5. 1 a = {1, 2, 3}
    2 b = {3, 4}
    3 a = a.union(b)
```

Giá trị của a sau khi thực hiện đoạn lệnh trên là:

- A. a = 1,2,3
- B. a = 1,2,3,4
- C. a = 4
- D. Báo lỗi do tập hợp không thể thay đổi

6. Trong từ điển, các khóa có thể giống nhau. Nhận định trên đúng hay sai?

- A. Đúng
- B. Sai

7. Cho đoạn code sau:

```
1 dict = {"a": 1, "b": 2}
2 print(dict["a"])
```

Đoạn lệnh trên dùng để làm gì:

- A. Để truy xuất phần tử có khóa là "a"
- B. Để in ra màn hình phần tử có khóa là "a"
- C. Để in ký tự "a" ra màn hình.

Đáp án

1. A 2. B 2. C 3. D 5. B 6. B 7. B

Phần II

Dự án ứng dụng

CHƯƠNG 11



Trợ lý ảo trên Python - Văn bản thành Giọng nói

pythonTM
Package
Index

1 Giới thiệu

Trong bài hướng dẫn này, chúng tôi sẽ đưa ra những bước cơ bản nhất để làm một ứng dụng mô phỏng cho Trợ lý ảo. Trong ứng dụng này sẽ có 3 phần chính: Nhận dạng giọng nói từ người dùng, Trí tuệ của máy tính, và cuối cùng là Trả lời lại cho người dùng. Trong 3 phần này, chúng ta có thể xem Nhận dạng giọng nói là **đầu vào (Input)**, **Trí tuệ của máy tính là phần Xử lý**, và cuối cùng, là **đầu ra (Output)**.

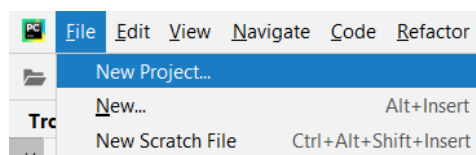
Với một dự án, việc phân tích đầu là dữ liệu đầu vào, dữ liệu đầu ra là một bước rất quan trọng, trước khi bắt đầu vào việc lập trình để xử lý. Thông thường, khi so sánh độ phức tạp của dữ liệu đầu vào và đầu ra, thì đầu ra sẽ đơn giản hơn, và nên được hiện thực trước tiên.

Trong bài hướng dẫn này, một đoạn chương trình đơn giản sẽ được hiện thực để máy tính trả lời kết quả lại cho chúng ta dưới dạng giọng nói. Kỹ thuật này gọi là "Text To Speech". Tức là, bằng việc lập trình, chúng ta đưa cho máy tính một chuỗi dữ liệu, nó sẽ nói ra thành lời thông qua loa của máy tính. Các bước hướng dẫn bên dưới sẽ trình bày chi tiết cách hiện thực chức năng "Text To Speech", phần đầu ra (Output) cho dự án Trợ lý ảo.

2 Tạo mới một ứng dụng

Đối với các bài hướng dẫn trước, chúng ta đã quen thuộc với việc thêm mới 1 file python vào chương trình và chạy được nó. Tuy nhiên, khi hiện thực một dự án mới, chúng ta nên tạo lại ứng dụng từ đầu để dễ quản lý. Trình tự để tạo mới một ứng dụng được trình bày như bên dưới.

Bước 1: Từ menu File, chúng ta chọn **New Project...**, như hướng dẫn ở Hình 11.1.

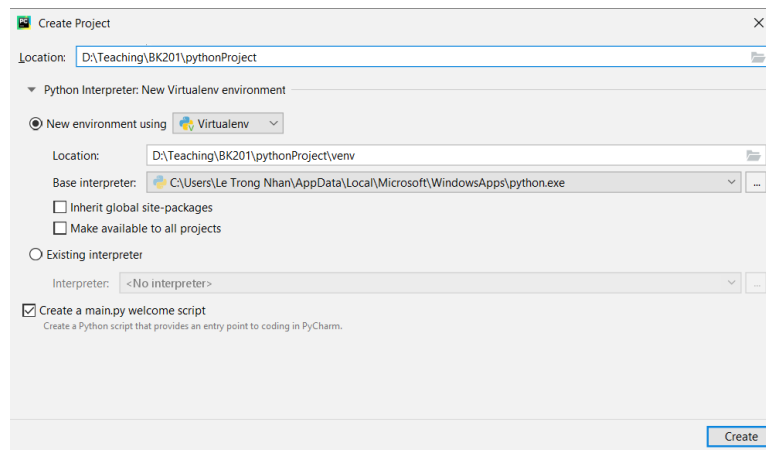


Hình 11.1: Tạo mới một dự án

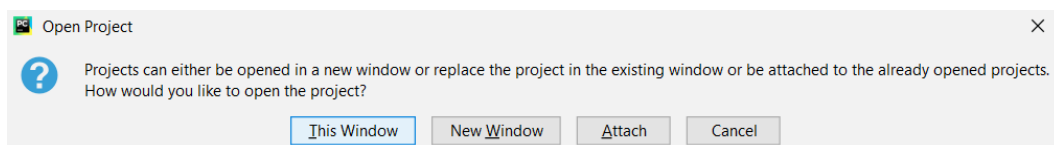
Bước 2: Tiếp theo đó, chúng ta cần chọn đường dẫn để lưu mới dự án, tại mục **Location**, như hướng dẫn ở Hình 11.2. Cuối cùng bạn nhấn vào nút **Create**.

Bước 3: Cuối cùng, hệ thống sẽ xuất hiện một thông báo, với ý nghĩa rằng bạn sẽ đóng dự án hiện tại, và mở dự án mới tại cửa sổ này, hay là vẫn giữ dự án cũ và mở một cửa sổ mới cho dự án vừa tạo. Ở đây, chúng tôi khuyên bạn là nên chọn vào **This Window** để đóng lại dự án cũ và chỉ mở dự án vừa tạo mà thôi, như minh họa ở Hình 11.3.

Tuy nhiên, chúng ta vẫn chưa thể bắt đầu lập trình được. Đối với các dự án dần lớn hơn, việc đầu tiên là cài đặt các phần mềm hỗ trợ cho việc hiện thực chương trình, sẽ được trình bày ở phần tiếp theo của hướng dẫn này.



Hình 11.2: Chọn đường dẫn để lưu dự án



Hình 11.3: Chọn This Window để đóng dự án cũ và mở dự án mới

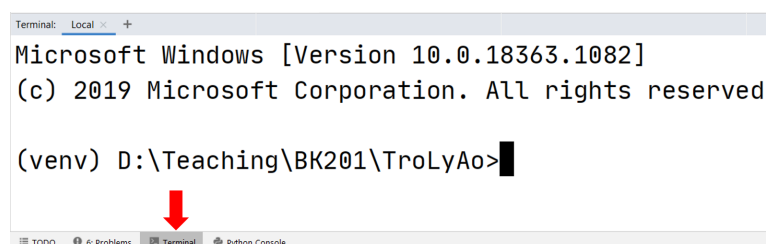
3 Cài đặt thư viện

Thực ra, khi hiện thực các dự án phức tạp, chúng ta sẽ khó có thể bắt đầu mọi thứ từ đầu. Thông thường, chúng ta sẽ cài đặt thêm thư viện bên ngoài, để hỗ trợ cho việc lập trình của chúng ta. Đây cũng chính là sức mạnh của cộng đồng lập trình mã nguồn mở, và Python là một ví dụ. Khi tham khảo các dự án lớn, bạn sẽ dễ dàng nhận ra, việc cài đặt thư viện sẽ là những bước đi đầu tiên.

Quá trình cài đặt thư viện sẽ phức tạp ở chỗ, là thư viện bạn cần dùng, chẳng hạn là thư viện A, nó lại phụ thuộc vào một thư viện khác, chẳng hạn là thư viện B. Vấn đề này sẽ xuất hiện khi bạn đang cài thư viện A, và hệ thống sẽ báo lỗi là thiếu thư viện B. Bạn cần phải cài đặt thư viện B trước, sau đó mới cài đặt lại thư viện A.

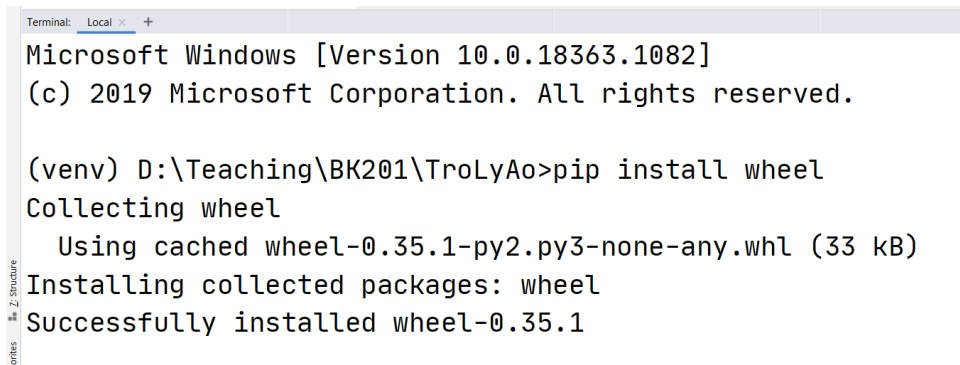
Trong bài này, chúng ta sẽ lần lượt cài các thư viện sau đây: **wheel**, **pipwin**, **pyaudio** và **pyttsx3**. Đây là thứ tự cài đặt tốt nhất mà chúng tôi tìm hiểu để việc cài đặt sẽ không bị báo lỗi.

Bước 1: Để bắt đầu việc cài đặt, chúng ta cần phải chuyển qua giao diện **Terminal**, như minh họa ở Hình 11.4



Hình 11.4: Giao diện Terminal để cài đặt thư viện.

Từ đây, bạn gõ vào dòng lệnh **pip install wheel** rồi nhấn Enter. Bạn cũng cần phải kết nối mạng để chương trình có thể tải thư viện về trước khi cài đặt. Trong trường hợp mạng quá kém, quá thời gian cho phép, hệ thống cũng sẽ tự động dừng việc cài đặt và bạn cần phải làm lại, cho đến khi nào thấy được dòng chữ **Successfully installed wheel**, như minh họa ở Hình 12.1. Ở đây, pip là một công cụ để cài đặt thư viện wheel.



```
Terminal: Local x +
Microsoft Windows [Version 10.0.18363.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

(venv) D:\Teaching\BK201\TroLyAo>pip install wheel
Collecting wheel
  Using cached wheel-0.35.1-py2.py3-none-any.whl (33 kB)
Installing collected packages: wheel
Successfully installed wheel-0.35.1
```

Hình 11.5: Cài đặt thành công thư viện wheel.

Bước 2: Tương tự, chúng ta sẽ cài đặt tiếp thư viện pipwin bằng công cụ pip, với câu lệnh **pip install pipwin**.

Bước 3: Để cài đặt thư viện **pyaudio**, chúng ta không thể cài đặt bằng công cụ pip được. Lý do là nó không tương thích với một số driver về âm thanh trên máy tính xài hệ điều hành Windows. Chúng ta cần phải cài thư viện này bằng công cụ pipwin, tức là gõ **pipwin install pyaudio** trên cửa sổ Terminal, rồi nhấn Enter.

Bước 4: Cuối cùng, chúng ta cài thư viện pytttsx3 bằng công cụ pip, tức là gõ **pip install pytttsx3** rồi nhấn Enter.

4 Hiện thực chương trình

Trở lại với file main.py, chúng ta bắt đầu hiện thực chương trình bên dưới.

```
1 import pytttsx3
2
3 engine = pytttsx3.init()
4 engine.say("I am AI python")
5 engine.runAndWait()
```

Chương trình 11.1: Chương trình chuyển từ chữ sang giọng nói

Với thư viện hỗ trợ là pytttsx3 (được lấy vào chương trình bằng câu lệnh import ở dòng 1), công việc của chúng ta khá đơn giản: Khởi tạo (dòng 3), truyền tham số (dòng 4) và cuối cùng, cho phép chương trình nói ra loa. Bạn hãy lưu ý rằng, hiện tại thư viện pytttsx3 chỉ hỗ trợ phát âm tiếng Anh mà thôi.

5 Câu hỏi ôn tập

1. Cửa sổ dùng để cài đặt thư viện trên PyCharm gọi là gì?
A. Terminal
B. Log
C. Console
D. Tất cả đều đúng
2. Câu lệnh nào sau đây được dùng để cài đặt thư viện **wheel**?
A. install wheel
B. pip install wheel
C. wheel install pin
D. Tất cả đều đúng
3. Để cài đặt thư viện pipwin, câu lệnh nào sau đây được sử dụng?
A. install pipwin
B. pipwin install pip
C. pip install pipwin
D. Tất cả đều sai
4. Để cài đặt thư viện pyaudio, câu lệnh nào sau đây được sử dụng?
A. pipwin install pyaudio
B. pip install pyaudio
C. pip install pipwin pyaudio
D. Tất cả đều sai
5. Thư viện chính nào dùng để chuyển từ văn bản sang giọng nói?
A. pipwin
B. pyaudio
C. pip
D. pyttsx3
6. Để có thể viết chương trình chuyển từ văn bản sang giọng nói, thư viện nào sẽ được thêm vào đầu chương trình? sang giọng nói?
A. import pipwin
B. import pyaudio
C. import pyttsx3
D. Tất cả đều đúng
7. Câu lệnh nào dùng để **khởi tạo** cho việc phát ra giọng nói từ văn bản?
A. engine = pyttsx3.init()
B. engine = pyttsx3.say("Init")
C. engine.runAndWait()
D. Tất cả đều đúng
8. Câu lệnh nào dùng để **truyền thông số dạng chữ** cho việc phát ra giọng nói?
A. engine = pyttsx3.init()
B. engine = pyttsx3.say("Init")
C. engine.runAndWait()
D. Tất cả đều đúng

9. Câu lệnh nào dùng để kích hoạt việc **phát ra âm thanh** khi viết chương trình chuyển từ văn bản sang giọng nói?
- A. `engine = pyttsx3.init()`
 - B. `engine = pyttsx3.say("Init")`
 - C. `engine.runAndWait()`
 - D. Tất cả đều đúng
10. Kỹ thuật chuyển từ văn bản sang giọng nói còn gọi là gì?
- A. Saying Text
 - B. Speech To Text
 - C. Text To Speech
 - D. Tất cả đều sai
11. Chuyển đổi giọng nói thành văn bản đóng vai trò nào trong hệ thống trợ lý ảo?
- A. Đầu vào Input
 - B. Xử lý AI
 - C. Đầu ra Output
 - D. Tất cả đều đúng
12. Ngôn ngữ hỗ trợ cho thư viện pyttsx3 là gì?
- A. Tiếng Việt
 - B. Tiếng Anh
 - C. Tiếng Việt và Tiếng Anh
 - D. Tất cả các thứ tiếng

Đáp án

1. A 2. B 3. B 4. A 5. D 6. C 7. A 8. B 9. C 10. C 11. C 12. B

CHƯƠNG 12



Trợ lý ảo trên Python - Nhận diện giọng nói

pythonTM
Package
Index

1 Giới thiệu

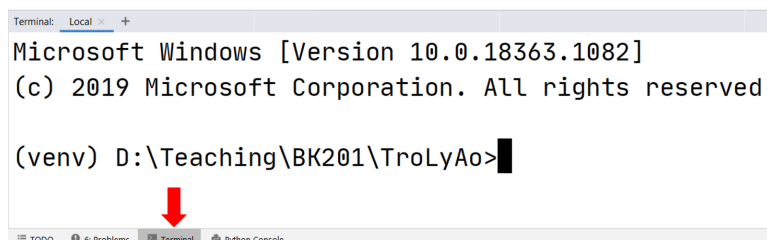
Trong phần tiếp theo của dự án trợ lý ảo, chúng ta sẽ thực hiện chức năng nhận diện giọng nói từ người dùng. Chức năng này được xem như là phần dữ liệu đầu vào của dự án, còn được gọi là Speech To Text. Cũng giống như chức năng phát ra giọng nói ở bài trước, đây hoàn toàn là một tác vụ rất phức tạp. Tuy nhiên với sự hỗ trợ từ các thư viện có sẵn, công việc lập trình của chúng ta trở nên đơn giản hơn.

Tuy nhiên, đối với chức năng nhập dữ liệu, bao giờ chúng ta cũng phải đi kèm với việc xử lý lỗi, để cho ứng dụng có thể hoạt động chính xác hơn. Ví dụ chúng ta mong đợi người dùng nói một câu, nhưng người dùng lại không nói câu nào cả, hoặc chẳng hạn vì các lý do khác, như mất mạng, và hệ thống không nhận được kết quả. Tất cả những chức năng này, làm cho việc xử lý dữ liệu đầu vào bao giờ cũng phức tạp hơn so với dữ liệu đầu ra.

2 Cài đặt thư viện

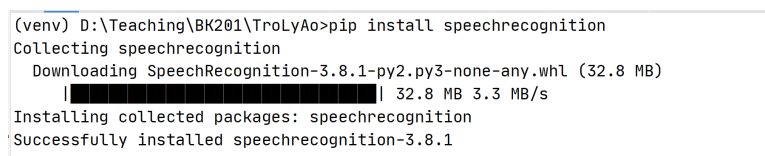
Để thực hiện chức năng nhận diện giọng nói, chúng ta cần cài đặt thêm một thư viện là `speechrecognition` để thực hiện chức năng nhận dạng giọng nói. Trình tự để làm gồm 2 bước như sau:

Bước 1: Để bắt đầu việc cài đặt, chúng ta cần phải chuyển qua giao diện **Terminal**, như minh họa ở Hình 12.1.



Hình 12.1: Giao diện Terminal để cài đặt thư viện.

Bước 2: Gõ dòng lệnh **`pip install wheel`** rồi nhấn Enter. Bạn cũng cần phải kết nối mạng để chương trình có thể tải thư viện về trước khi cài đặt. Trong trường hợp mạng quá kém, quá thời gian cho phép, hệ thống cũng sẽ tự động dừng việc cài đặt và bạn cần phải làm lại, cho đến khi nào thấy được dòng chữ **Successfully installed speechrecognition**, như minh họa ở Hình 12.2.



Hình 12.2: Cài đặt thành công thư viện SpeechRecognition

Lưu ý: Nếu bạn muốn hiện thực một chương trình chỉ có chức năng nhận diện giọng nói, bạn cần phải cài đặt `wheel`, `pipwin`, `pyaudio` và `speechrecognition`.

3 Hiện thực chương trình

Để hiện thực chức năng nhận diện giọng nói, chúng ta cần những dòng lệnh căn bản sau đây:

```
1 import speech_recognition
2
3 voice = speech_recognition.Recognizer()
4 with speech_recognition.Microphone() as mic:
5     print("I am listenning...")
6     audio = voice.listen(mic)
7 input_text = voice.recognize_google(audio)
8 print(input_text)
```

Chương trình 12.1: Chương trình chuyển từ chữ sang giọng nói

Ở dòng đầu tiên, chúng ta đang thêm thư viện vào chương trình. Bạn hãy lưu ý tên thư viện lúc import, có thêm dấu gạch dưới(import speech_recognition), khác với tên đánh liền lúc cài đặt thư viện (pip install speechrecognition). Dòng tiếp theo là để khởi tạo một đối tượng (đặt tên là voice) làm nhiệm vụ chính là nhận diện giọng nói. Câu lệnh **with** dùng để khởi tạo microphone của máy tính chúng ta, và bắt đầu ghi âm lại, lưu vào biến audio. Khi việc ghi âm kết thúc (chúng ta ngừng nói), dữ liệu âm thanh sẽ được nhận dạng bằng phương thức recognize_google. Với kết quả lưu trong biến **input_text**, chúng ta có thể in ra màn hình để theo dõi.

Trong trường hợp người dùng không nói gì, hoặc mạng bị lỗi, câu lệnh **text = voice.recognize_google(audio)** có thể xảy ra lỗi. Chúng ta sẽ khắc phục sự cố này bằng câu lệnh **try except**. Khéo léo kết hợp với chương trình đã hiện thực ở bài trước, chúng ta có chương trình như sau:

```
1 import speech_recognition
2 import pyttsx3
3
4 voice = speech_recognition.Recognizer()
5 with speech_recognition.Microphone() as mic:
6     print("I am listenning...")
7     audio = voice.listen(mic)
8
9 try:
10     input_text = voice.recognize_google(audio)
11 except:
12     input_text = "I do not understand"
13 print(input_text)
14
15 engine = pyttsx3.init()
16 engine.say("I hear " + input_text)
17 engine.runAndWait()
```

Chương trình 12.2: Nhận diện giọng nói và phát âm lại

Trong chương trình ở trên, chúng ta sẽ cho máy tính nói lại những gì mà chúng ta vừa nói. Đây là một cách rất hiệu quả để kiểm tra quá trình xử lý đầu vào (input) và

đầu ra (output), trước khi chúng ta bắt tay vào hiện thực cho phần xử lý, tạm gọi là "trí thông minh" của trợ lý ảo. Lưu ý là, hệ thống nhận dạng giọng nói vẫn chỉ hỗ trợ cho tiếng Anh mà thôi. Để có thể làm trợ lý ảo hoàn toàn bằng tiếng Việt, bạn phải xài những thư viện cao cấp hơn. Trong giới hạn của hướng dẫn này, chúng tôi chỉ sử dụng những thư viện đơn giản nhất.

4 Câu hỏi ôn tập

1. Quá trình nhận dạng giọng nói, đóng vai trò gì trong hệ thống trợ lý ảo?
A. Xử lý đầu vào
B. Xử lý đầu ra
C. Xử lý trí thông minh
D. Tất cả đều đúng
2. Để xử lý lỗi trong việc nhận dạng giọng nói, câu lệnh nào được sử dụng?
A. `Recognizer()`
B. `try except`
C. `if else`
D. Tất cả đều đúng
3. Thư viện chính để nhận dạng giọng nói tiếng Anh là gì?
A. `pyaudio`
B. `pyttsx3`
C. `speech recognition`
D. Tất cả đều sai
4. Câu lệnh nào dùng để khởi động chương trình nhận diện giọng nói?
A. `pyttsx3.init()`
B. `engine.say()`
C. `speech_recognition.Recognizer()`
D. Tất cả câu lệnh trên
5. Ngôn ngữ hỗ trợ cho thư viện `speechrecognition` là gì?
A. Tiếng Việt
B. Tiếng Anh
C. Tiếng Việt và Tiếng Anh
D. Tất cả các thứ tiếng
6. Câu lệnh để thêm thư viện nhận dạng giọng nói vào chương trình Python là gì?
A. `import pyttsx3`
B. `import speechrecognition`
C. `import speech_recognition`
D. Tất cả đều đúng
7. Các lỗi có thể xảy ra khi nhận diện giọng nói là gì?
A. Người sử dụng không nói gì
B. Mất kết nối mạng
C. Nói bằng ngôn ngữ hệ thống không nhận dạng được
D. Tất cả các lỗi trên đều có thể xảy ra

Đáp án

1. A 2. B 3. C 4. C 5. B 6. C 7. D

CHƯƠNG 13



Trợ lý ảo trên Python - Xây dựng trí thông minh

pythonTM
Package
Index

1 Giới thiệu

Bài hướng dẫn này là bước thứ 3, bước cuối cùng khi hiện thực 1 dự án: Xử lý dữ liệu đầu vào, và xuất kết quả ra đầu ra. Đây phải là bước hiện thực sau cùng, vì chỉ khi việc xử lý đầu vào và đầu ra ổn định, chúng ta mới có cơ sở để hiện thực giải thuật ngày một hoàn thiện hơn và ổn định hơn.

Trong bài hướng dẫn này, chúng tôi chỉ đưa ra những ý tưởng căn bản nhất để người đọc có thể hiện thực một chương trình trợ lý ảo đơn giản, để nó ngày trở nên thông minh hơn, người học có thể tự phát triển.

Trong hướng dẫn này, trợ lý ảo của chúng tôi sẽ trả lời lại 2 câu hỏi từ người dùng là "hello" (xin chào) và "what is today" (hôm nay là thứ mấy)

2 Truy xuất ngày hiện tại

Có nhiều phương pháp để lấy thông tin ngày hiện tại của hệ thống, phần chương trình hướng dẫn bên dưới là một ví dụ.

```
1 from datetime import date
2
3 today = date.today()
4 strTextToday = today.strftime("%B %d, %Y")
5 print(strTextToday)
```

Chương trình 13.1: Lấy ngày hiện tại của hệ thống

Trong chương trình trên, chúng ta cần phải thêm thư viện date, sau đó định dạng cho chuỗi dữ liệu mô tả ngày hiện tại. Tất nhiên, kết quả sẽ là tiếng Anh, bởi chúng ta đang hiện thực trợ lý ảo có thể nhận dạng và nói bằng tiếng Anh.

3 Hoàn thiện chương trình

Đơn giản nhất trong việc hiện thực chức năng trả lời tự động cho trợ lý ảo là sử dụng những câu lệnh **if**. Với mỗi lệnh từ người dùng nhận vào, chúng ta sẽ có những câu trả lời tương ứng. Chương trình gợi ý cho phần này như sau:

```
1 import speech_recognition
2 import pyttsx3
3 from datetime import date
4
5 today = date.today()
6 strTextToday = today.strftime("%B %d, %Y")
7
8 voice = speech_recognition.Recognizer()
9 with speech_recognition.Microphone() as mic:
10     print("I am listenning...")
11     audio = voice.listen(mic)
```

```

12
13 try:
14     input_text = voice.recognize_google(audio)
15 except:
16     input_text = "I do not understand"
17 print(input_text)
18
19 output_text = ""
20 if input_text == "hello":
21     output = "Hello, I am AI Python"
22 elif input_text == "what is today":
23     output_text = "Today is " + strTextToday
24 else:
25     output_text = "Sorry I do not understand"
26
27 engine = pyttsx3.init()
28 engine.say(output_text)
29 engine.runAndWait()

```

Chương trình 13.2: Hoàn thiện phần xử lý cho trợ lý ảo

Thực ra, chương trình hiện tại vẫn còn phải phát triển thêm để nó có thể gần giống như một trợ lý ảo. Các hướng mở rộng dành cho người học được liệt kê như sau:

- Cải thiện việc so trùng: Hiện tại, trong câu lệnh **if**, việc so sánh trùng sẽ rất khó xảy ra trong thực tế. Ví dụ người dùng nói "hello hello", thì trợ lý của chúng ta sẽ không chào lại. Chúng ta sẽ so sánh theo kiểu từ khóa: Trong câu nói của người dùng có chữ hello, thì chúng ta có thể chào lại. Câu lệnh gợi ý cho chức năng này là **if "hello" in input_text**.
- Hiện tại, chương trình cũng chỉ chạy một lần rồi dừng. Đây là cơ hội để người học vận dụng câu lệnh **while**: Chương trình sẽ liên tục hoạt động cho đến khi người dùng nói **bye bye**.
- Chương trình hiện tại chỉ hỗ trợ 2 câu lệnh, người dùng có thể tự định nghĩa thêm nhiều câu lệnh khác, để trợ lý ảo trở nên thông minh hơn!!!

Các hướng mở rộng trình bày ở trên, chúng tôi không hướng dẫn bằng chương trình cụ thể, mà xem nó như là ý tưởng để người học tự sáng tạo. Chúc các bạn thành công với trợ lý ảo của mình.

4 Câu hỏi ôn tập

1. Quá trình hiện thực trí thông minh cho trợ lý ảo, thuộc phần nào sau đây?
 - A. Xử lý đầu vào
 - B. Xử lý đầu ra
 - C. Xử lý giải thuật
 - D. Tất cả đều đúng
2. Để trợ lý ảo có thể hoạt động liên tục cho đến khi người dùng nói "bye bye", câu lệnh nào là tốt nhất để hiện thực?
 - A. if
 - B. while
 - C. for
 - D. Tất cả đều sai
3. Để hiện thực trí thông minh cho trợ lý ảo, câu lệnh nào có thể được sử dụng một cách hiệu quả nhất?
 - A. if
 - B. while
 - C. for
 - D. Tất cả đều sai
4. Để có thể truy xuất ngày hiện tại của hệ thống, thư viện nào cần được thêm vào chương trình?
 - A. pyttsx3
 - B. date
 - C. speech_recognition.Recognizer()
 - D. Tất cả câu lệnh trên
5. Hạn chế của câu lệnh so sánh bằng giữa 2 chuỗi khi hiện thực trí thông minh cho trợ lý ảo là gì?
 - A. Xác suất nhận lệnh đúng từ người dùng sẽ thấp
 - B. Khi người dùng nói "hello hello", hệ thống sẽ không hiểu nếu chỉ so sánh bằng với "hello"
 - C. Hạn chế rất nhiều khi triển khai thực tế
 - D. Tất cả đều đúng

Đáp án

1. C 2. B 3. A 4. B 5. D

CHƯƠNG 14



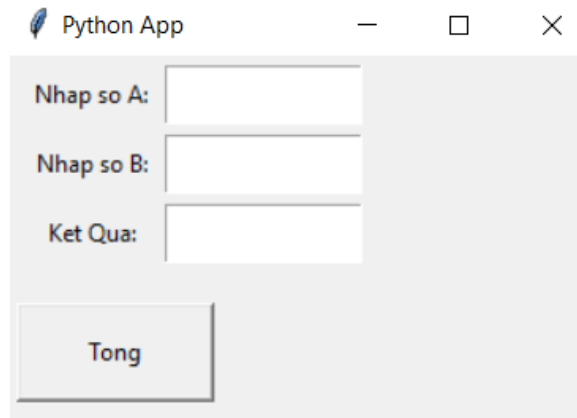
Lập trình giao diện trên Python

pythonTM
Package
Index

1 Giới thiệu

Cũng như các ngôn ngữ lập trình cấp cao khác, Python cho chúng ta cơ hội để hiện thực các ứng dụng với giao diện thân thiện. Các ứng dụng này được gọi là ứng dụng Winform. Sở dĩ có tên như vậy, là vì giao diện của ứng dụng thực sự giống với các ứng dụng mà chúng ta đang sử dụng trên hệ điều hành Windows.

Trong bài hướng dẫn này, chúng ta sẽ thiết kế một ứng dụng đơn giản: Người dùng nhập vào 2 số và nhấn nút Cộng. Kết quả sẽ được in ra trên chương trình, như minh họa ở hình bên dưới:



Hình 14.1: Một ứng dụng giao diện trên Python

Rõ ràng, một chương trình như minh họa ở Hình 14.1 sẽ thân thiện hơn với rất nhiều người dùng, thay vì chỉ tương tác từ màn hình console như các ứng dụng truyền thống như Pascal hoặc phần hướng dẫn cơ bản của Python. Tuy nhiên, với ứng dụng như vậy, chúng ta sẽ cần phải tỉ mỉ ở phần thiết kế giao diện, trước khi bắt đầu lập trình. Điều đặc biệt là trên Python, chúng ta sẽ lập trình để tạo ra giao diện, thay vì kéo thả để thiết kế giao diện như các ngôn ngữ cao cấp khác. Trong bài hướng dẫn này, chúng tôi sẽ trình bày những câu lệnh đơn giản nhất để thiết kế ra ứng dụng như trình bày ở Hình 14.1.

2 Thiết kế giao diện

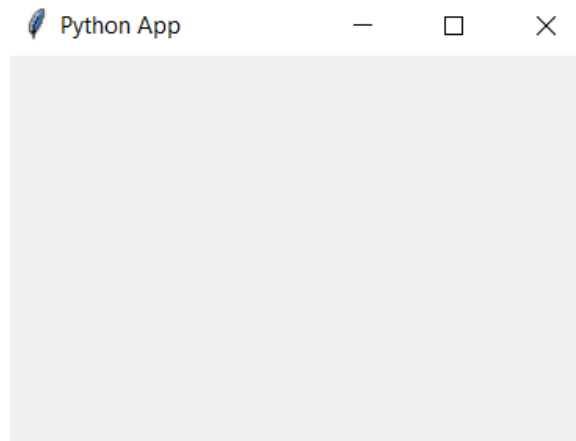
Trên Python, đã có hỗ trợ sẵn thư viện thiết kế giao diện là **tkinter**. Nguyên lý của việc lập trình giao diện là chúng ta sẽ tạo ra một cửa sổ (Windows), sau đó thêm các đối tượng giao diện vào (khung nhập liệu, nút nhấn, v.v...). Chúng ta sẽ bắt đầu bước đầu tiên, là tạo ra cửa sổ giao diện, với chương trình sau đây:

```
1 import tkinter as tk
2
3 window = tk.Tk()
4
5 window.title("Python App")
6 window.geometry("300x200")
7
```

```
8 window.mainloop()
```

Chương trình 14.1: Tạo ra cửa sổ (Windows) cho ứng dụng

Trong Chương trình 14.1, câu lệnh đầu tiên dùng để thêm thư viện thiết kế giao diện **tkinter**. Vì sau này, chúng ta sẽ phải gõ rất nhiều từ khóa **tkinter**, chúng ta đặt cho nó một tên viết tắt, là **tk**. Tiếp theo đó, chúng ta tạo ra một cửa sổ bằng câu lệnh **tk.Tk()**, thay đổi các thông số của cửa sổ như tiêu đề, kích thước của cửa sổ (ngang 300 và cao 200 pixel). Cuối cùng, chúng ta sẽ kích hoạt cửa sổ đó lên bằng câu lệnh **window.mainloop()**. Khi chạy chương trình này, chúng ta sẽ có kết quả như sau:



Hình 14.2: Cửa sổ ứng dụng trên Windows

Kết quả chương trình ở Hình 14.2 là lý do tại sao ứng dụng được gọi là ứng dụng **winform**, bởi nó có cấu trúc mặc định của một cửa sổ trên hệ điều hành Windows: nút thu nhỏ, phóng to, tắt cửa sổ cũng như thanh tiêu đề trên ứng dụng.

3 Thêm đối tượng giao diện vào cửa sổ

Sau khi đã có khung cửa sổ tiêu chuẩn, chúng ta bắt đầu thêm các đối tượng giao diện vào. Để có thể hiển thị hướng dẫn nhập số A, chúng ta cần 2 đối tượng giao diện: **Lable** và **Text**. Đối tượng **Label** chỉ dùng để hiển thị, còn **Text** là đối tượng dùng để nhập liệu. Đối với mỗi đối tượng, chúng ta phải dùng 2 câu lệnh: Tạo ra đối tượng và định vị trí của nó trong khung cửa sổ. Chương trình cho phần này sẽ như sau:

```
1 import tkinter as tk
2
3 window = tk.Tk()
4
5 window.title("Python App")
6 window.geometry("300x200")
7
8 labelA = tk.Label(text="Nhập số A: ")
9 labelA.place(x=5, y=5, width = 80, height = 30)
10
11 txtA = tk.Text()
```

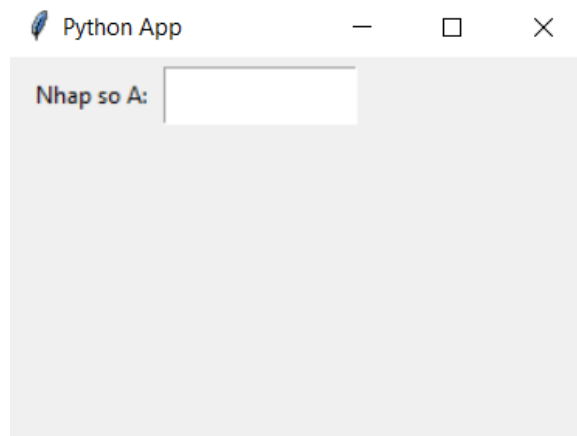
```

12 txtA.place(x = 80, y =5, width = 100, height = 30)
13
14 window.mainloop()

```

Chương trình 14.2: Tạo ra cửa sổ (Windows) cho ứng dụng

Trong Chương trình 14.3, các thông số về x, y là vị trí của đối tượng trong giao diện, width và height là kích thước chiều rộng và cao của đối tượng. Bạn hãy thử thay đổi nó để thấy hiệu ứng thay đổi trên giao diện. Với đoạn chương trình trên, kết quả của chúng ta sẽ như sau:



Hình 14.3: Thêm đối tượng giao diện lên ứng dụng

Tiếp tục làm tương tự như vậy để thêm phần hướng dẫn nhập số B, và phần hiển thị kết quả. Hãy lưu ý rằng chiều cao của dòng đầu tiên đang là 30, và tọa độ y đang là 5. Nên dòng thứ 2 tối thiểu tọa độ y phải là 35. Chúng tôi cho tọa độ y của dòng 2 là 40 để có được một khoảng trống vừa phải. Tương tự như vậy, dòng 3 tọa độ y sẽ là 80. Nút nhấn sẽ được thêm vào cuối cùng. Chương trình hoàn thiện của chúng ta như sau:

```

1 import tkinter as tk
2
3 window = tk.Tk()
4
5 window.title("Python App")
6 window.geometry("300x200")
7
8 labelA = tk.Label(text="Nhập số A: ")
9 labelA.place(x=5, y=5, width = 80, height = 30)
10
11 txtA = tk.Text()
12 txtA.place(x = 80, y =5, width = 100, height = 30)
13
14 labelB = tk.Label(text="Nhập số B: ")
15 labelB.place(x=5, y= 40, width = 80, height = 30)
16
17 txtB = tk.Text()
18 txtB.place(x = 80, y =40, width = 100, height = 30)

```



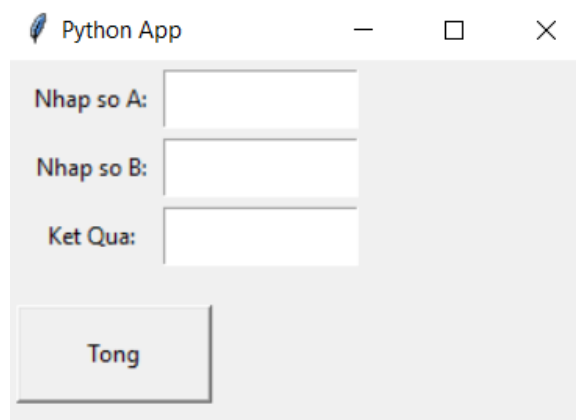
```

19
20 labelKQ = tk.Label(text="Ket Qua: ")
21 labelKQ.place(x=5, y= 75, width = 80, height = 30)
22
23 txtKQ = tk.Text()
24 txtKQ.place(x = 80, y =75, width = 100, height = 30)
25
26 button = tk.Button(text = "Tong")
27 button.place(x = 5, y = 125, width =100, height = 50)
28
29 window.mainloop()

```

Chương trình 14.3: Thiết kế giao diện hoàn chỉnh cho ứng dụng

Cuối cùng, giao diện của ứng dụng của chúng ta sẽ như minh họa ở Hình 14.4.



Hình 14.4: Thiết kế giao diện hoàn chỉnh

4 Câu hỏi ôn tập

1. Ứng dụng chạy trên Windows gọi là gì?
 - A. Winform
 - B. Console
 - C. Log
 - D. Tất cả đều đúng
2. Để lập trình giao diện bằng Python trên PyCharm cần phải:
 - A. Kéo và thả đối tượng giao diện vào chương trình
 - B. Viết chương trình tạo ra giao diện
 - C. Python không hỗ trợ lập trình giao diện
 - D. Tất cả đều sai
3. Thông thường, để in thông tin hướng dẫn trên ứng dụng, đối tượng giao diện nào sẽ được dùng?
 - A. Label
 - B. Text
 - C. Button
 - D. Windows
4. Thông thường, để người dùng nhập thông tin vào ứng dụng, đối tượng giao diện nào sẽ được dùng?
 - A. Label
 - B. Text
 - C. Button
 - D. Windows
5. Thông thường, để nhận tương tác từ người dùng, chẳng hạn như nhấp chuột, đối tượng giao diện nào sẽ được dùng?
 - A. Label
 - B. Text
 - C. Button
 - D. Windows
6. Để chỉnh vị trí của đối tượng giao diện trên ứng dụng, chúng ta sẽ can thiệp vào thuộc tính nào?
 - A. Tọa độ x và tọa độ y
 - B. Thông số width và height
 - C. Cả 2 câu trên đều đúng
 - D. Tất cả đều sai
7. Để chỉnh kích thước của đối tượng giao diện trên ứng dụng, chúng ta sẽ can thiệp vào thuộc tính nào?
 - A. Tọa độ x và tọa độ y
 - B. Thông số width và height
 - C. Cả 2 câu trên đều đúng
 - D. Tất cả đều sai

Đáp án

1. A 2. B 3. A 4. B 5. C 6. A 7. B

CHƯƠNG 15



Tạo hàm xử lý cho nút nhấn

pythonTM
Package
Index

1 Giới thiệu

Sau khi thiết kế xong giao diện, bước tiếp theo chúng ta cần phải tạo hàm xử lý khi người dùng nhấn vào nút Tong. Nút nhấn gần như là đối tượng tương tác chính của người dùng lên ứng dụng chạy trên máy tính. Hàm mà chúng ta chuẩn bị tạo sẽ được tự động gọi khi người dùng nhấp vào nút nhấn.

2 Khai báo hàm xử lý

Trước tiên, chúng ta sẽ khai báo một hàm, đặt tên là `onClick` trên Python. Hàm này trước mắt chỉ in ra 1 thông báo ngắn, như ví dụ bên dưới:

```
1 def onClick():
2     print("Nhấn nút")
```

Chương trình 15.1: Khai báo hàm `onClick`

Tiếp theo, chúng ta cần liên kết sự kiện nhấn nút Tong với hàm `onClick` này. Trong câu lệnh khởi tạo nút nhấn ở bài trước, chúng ta hiệu chỉnh lại như sau:

```
1 button = tk.Button(text = "Tong", command = onClick)
```

Chương trình 15.2: Khai báo hàm `onClick`

Lưu ý quan trọng là chúng ta phải khai báo hàm `onClick` trước khi gán nó cho sự kiện nút nhấn. Chương trình hoàn chỉnh của chúng ta sẽ như sau

```
1 import tkinter as tk
2
3 def onClick():
4     print("Nhấn nút")
5
6 window = tk.Tk()
7 window.title("Python App")
8
9 labelA = tk.Label(text="Nhập số A: ")
10 labelA.place(x=5, y=5, width = 80, height = 30)
11
12 txtA = tk.Text()
13 txtA.place(x = 80, y = 5, width = 100, height = 30)
14
15 labelB = tk.Label(text="Nhập số B: ")
16 labelB.place(x=5, y= 40, width = 80, height = 30)
17
18 txtB = tk.Text()
19 txtB.place(x = 80, y =40, width = 100, height = 30)
20
21 labelKQ = tk.Label(text="Kết Quả: ")
22 labelKQ.place(x=5, y= 75, width = 80, height = 30)
23
24 txtKQ = tk.Text()
```

```

25 txtKQ.place(x = 80, y = 75, width = 100, height = 30)
26
27 button = tk.Button(text = "Tong", command = onClick)
28 button.place(x = 5, y = 125, width = 100, height = 50)
29
30 window.geometry("300x200")
31 window.mainloop()

```

Chương trình 15.3: Hoàn thiện chương trình với sự kiện nút nhấn

3 Hiện thực hàm xử lý cho nút nhấn

Để hiện thực cho hàm này, chúng ta cần phải lấy thông tin từ 2 đối tượng là txtA và txtB. Dữ liệu từ 2 đối tượng này đều là kiểu chuỗi, nên chúng ta phải đổi sang kiểu dữ liệu mong muốn, trong trường hợp này là dữ liệu số. Đến bước này thì mọi việc đã trở nên đơn giản hơn nhiều, chúng ta tính tổng của 2 s. Và cuối cùng là xuất kết quả ra txtKQ. Chương trình hướng dẫn cho hàm xử lý sự kiện này như sau:

```

1 def onClick():
2     print("Nhan nut")
3     a = int(txtA.get("1.0"))
4     b = int(txtB.get("1.0"))
5
6     kq = a + b
7
8     txtKQ.delete(1.0)
9     txtKQ.insert(1.0, kq)

```

Chương trình 15.4: Hoàn thiện chương trình với sự kiện nút nhấn

Chương trình 15.4 là ví dụ cho những thao tác xử lý cơ bản từ việc lấy dữ liệu từ Text, xử lý và xuất kết quả ra lại trên một đối tượng Text. Trước khi xuất dữ liệu mới, chúng ta cần phải xóa đi dữ liệu cũ. Con số "1.0" trong chương trình có ý nghĩa là hàng 1, vị trí đầu tiên (vị trí 0). Đối với câu lệnh get, thì nó có nghĩa là đọc dữ liệu tại hàng 1, vị trí đầu tiên đến hết. Đối với câu lệnh delete, nó có nghĩa là xóa toàn bộ hàng 1. Câu lệnh insert tức là thêm dữ liệu mới ở hàng 1, từ vị trí đầu tiên.

Bây giờ, bạn đã có thể chạy chương trình, nó là mô phỏng cho việc tính tổng của 2 số. Chương trình sẽ vẫn chạy cho đến khi bạn nhấn vào biểu tượng đóng cửa sổ ở góc trên bên phải.

4 Bài tập

1. Hãy viết chương trình tính nghiệm của phương trình bậc nhất, với các hệ số a và b được nhập vào từ giao diện. 2. Hãy viết chương trình giải phương trình bậc 2, với các hệ số a, b và c được nhập từ giao diện.

5 Câu hỏi ôn tập

1. Để tạo hàm xử lý sự kiện cho một nút nhấn, chúng ta cần phải làm gì?
 - A. Định nghĩa một hàm, ví dụ là `def onClick()`
 - B. Khai báo nó trong lệnh khởi tạo nút nhấn bằng lệnh `command = onClick`
 - C. Thực hiện cả 2 bước trên
 - D. Tất cả đều sai
2. Câu phát biểu nào là đúng về hàm xử lý sự kiện cho nút nhấn:
 - A. Nó phải được khai báo trước nút nhấn
 - B. Được khai báo ở bất kì vị trí nào trong chương trình
 - C. Không cần phải khai báo, Python mặc định gọi
 - D. Tất cả đều sai
3. Các bước để hiện thực hàm xử lý sự kiện nút nhấn trên Python là gì?
 - A. Lấy dữ liệu từ giao diện vào
 - B. Xử lý dữ liệu
 - C. Xuất dữ liệu ra giao diện
 - D. Tất cả đều đúng
4. Ý nghĩa của câu lệnh `txtA.get("1.0")` là gì?
 - A. Đọc toàn bộ dữ liệu của `txtA`
 - B. Xuất dữ liệu ra `txtA` từ vị trí đầu tiên
 - C. Python không hỗ trợ lệnh này
 - D. Tất cả đều sai
5. Ý nghĩa của câu lệnh `txtA.insert("1.0")` là gì?
 - A. Đọc toàn bộ dữ liệu của `txtA`
 - B. Xuất thêm dữ liệu ra `txtA` từ vị trí đầu tiên
 - C. Python không hỗ trợ lệnh này
 - D. Tất cả đều sai
6. Ý nghĩa của câu lệnh `txtA.delete("1.0")` là gì?
 - A. Xóa toàn bộ dữ liệu của `txtA`
 - B. Xóa dữ liệu của `txtA` từ hàng 1, vị trí đầu tiên cho đến hết
 - C. Cả 2 phát biểu trên đều đúng
 - D. Tất cả đều sai

Đáp án

1. A 2. A 3. D 4. A 5. B 6. C

CHƯƠNG 16



Xử lý lỗi trên giao diện

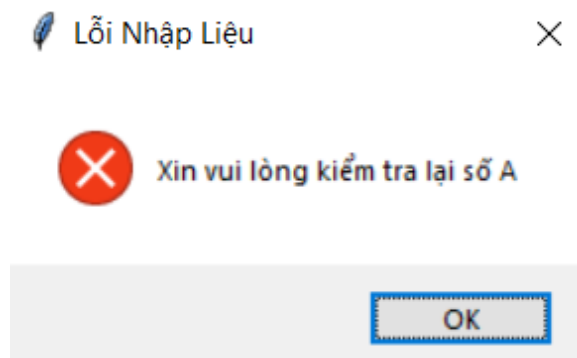
pythonTM
Package
Index

1 Giới thiệu

Thông qua dự án hiện thực một ứng dụng có giao diện trên Windows bằng ngôn ngữ Python, bạn có thể thấy rằng, việc xử lý của chúng ta chỉ có đúng 1 dòng lệnh, là tính tổng 2 số. Còn lại, tất cả các dòng lệnh chủ yếu là dùng cho việc thiết kế giao diện, lấy dữ liệu từ các Text trên giao diện và xuất kết quả lại ra giao diện.

Tuy nhiên, cũng giống như bao ứng dụng khác, việc xử lý lỗi cho phần dữ liệu đầu vào luôn luôn phải được chú ý, để giúp cho ứng dụng trở nên thực tế hơn và tương tác dễ hơn với người sử dụng. Trong ứng dụng này, chúng ta đang mặc định là phải nhập vào một số nguyên. Tuy nhiên **người dùng có thể nhập vào một chuỗi** và chương trình của chúng ta sẽ bị báo lỗi.

Bên cạnh việc dùng câu lệnh **try except**, trong bài này sẽ hướng dẫn chúng ta cách hiển thị một hộp thông báo lỗi (message box), như minh họa ở hình bên dưới:



Hình 16.1: Thông báo báo lỗi từ chương trình

Những thông báo như trên mới sẽ làm cho chương trình chúng ta trở nên thân thiện hơn người dùng rất nhiều, thay vì chỉ in ra những dòng chữ trên giao diện Console.

2 Xử lý lỗi nhập liệu

Về nguyên tắc, bất kì quy trình nhận dữ liệu nào từ phía người dùng cũng cần phải được kiểm tra lỗi. Ở đây, chúng ta mong đợi người dùng sẽ nhập vào một số, nên viết câu lệnh chuyển nó qua số nguyên. Một câu lệnh vô cùng mạnh mẽ hỗ trợ cho chúng ta trong việc xử lý lỗi là **try except**. Bây giờ, trong phần **except**, chúng ta sẽ hiển thị thêm thông báo lỗi.

Đầu tiên, chúng ta cần phải thêm thư viện cho việc hiển thị thông báo lỗi, bằng câu lệnh import ở đầu chương trình, như sau:

```
1 import tkinter as tk
2 from tkinter import messagebox
```

Chương trình 16.1: Khai báo hàm onClick

Chương trình cho hàm xử lý sự kiện nhấn nút của chúng ta sẽ như sau:

```
1
2 def onClick():
3
4     print("Nhấn nút")
5     try:
6         a = int(txtA.get("1.0"))
7     except:
8         messagebox.showerror("Lỗi Nhập Liệu", "Xin vui lòng
kiểm tra lại số A")
9         return
10    try:
11        b = int(txtB.get("1.0"))
12    except:
13        messagebox.showerror("Lỗi Nhập Liệu", "Xin vui lòng
kiểm tra lại số B")
14        return
15
16    c = a + b
17
18    txtKQ.delete(1.0)
19    txtKQ.insert(1.0, c)
```

Chương trình 16.2: Bắt lỗi và thông báo khi nhập dữ liệu từ người dùng

Bạn hãy lưu ý cách sử dụng câu lệnh **return**: Khi xảy ra lỗi nhập liệu, chúng ta thông báo cho người dùng và đồng thời phải dừng luồng xử lý bên dưới. Câu lệnh **return** dùng để thoát khỏi hàm xử lý sự kiện hiện tại, bởi đơn giản, dữ liệu đầu vào chưa hợp lệ thì chúng ta chưa xử lý.

Các thao tác khôi phục lại giao diện của chương trình, chẳng hạn như khi người dùng nhập sai dữ liệu, chúng ta thông báo nhưng đồng thời cũng xóa dữ liệu nhập sai, sẽ xem như là phần nâng cao cho người đọc tự hoàn thiện. Bạn có thể sử dụng tiếng Việt trong câu thông báo. Trong giáo trình này, do font chữ cho ngôn ngữ Python, chúng tôi không thể dùng tiếng Việt có dấu.

Đến đây, bạn đã có thể tự xây dựng những ứng dụng lớn hơn, chẳng hạn như phần mềm Calculator mà chúng ta đang sử dụng trên máy tính, hoặc các phần mềm hỗ trợ học Toán cho học sinh cấp 2 cấp 3.

3 Câu hỏi ôn tập

1. Tại sao chúng ta phải xử lý lỗi nhập liệu?
 - A. Chương trình sẽ chạy sai nếu dữ liệu không hợp lệ
 - B. Python không thực hiện được việc xử lý nếu dữ liệu đầu vào không hợp lệ
 - C. Chương trình sẽ bị thoát nếu dữ liệu không hợp lệ
 - D. Tất cả đều đúng
2. Một công cụ hiệu quả để thông báo lỗi cho người dùng là:
 - A. Hiện chữ trên màn hình console
 - B. Hiện hộp thông báo trên ứng dụng
 - C. Không cần phải làm gì, Python tự xử lý
 - D. Tất cả đều đúng
3. Các thêm thư viện để sử dụng hộp thông báo (messageBox) là gì?
 - A. from tkinter import messagebox
 - B. from messagebox import tkinter
 - C. import messagebox
 - D. Tất cả đều đúng
4. Phát biểu nào sau đây là đúng cho messageBox?
 - A. Message box có thể hiển thị được tiếng Việt
 - B. Đóng message box bằng cách nhấn vào nút OK
 - C. Có thể thay đổi tiêu đề của message box
 - D. Tất cả đều đúng
5. Ý nghĩa của câu lệnh return là gì?
 - A. Đóng ứng dụng hiện tại
 - B. Thoát khỏi luồng xử lý của hàm hiện tại
 - C. Tất cả đều đúng
 - D. Tất cả đều sai

Đáp án

1. D 2. B 3. A 3. D 4. D 5. B