Chương 2. Biến và những kiểu dữ liệu đơn giản

Nội dung giới thiệu

- Giới thiệu chung về biến (variables)
- Chuỗi (Strings)
- Số (Numbers)
- Chú thích (comments)

2.1. Biến

```
message = "Hello Python world!"
print(message)
```

```
Hello Python world!
```

Thêm một dòng ở đầu tệp và sửa dòng thứ hai:

Mở rộng chương trình với việc chỉnh sửa hello_world.py, để in ra thông điệp thứ hai. Thêm dòng trống vào tệp hello_world.py và thêm hai dòng code mới:

```
message = "Hello Python world!"
print(message)
message = "Hello Python Crash Course
world!"
print(message)
```

```
Hello Python world!
Hello Python Crash Course world!
```

2.1.1. Định danh và sử dụng các biến

Quy tắc và một số hướng dẫn:

❑Tên biến chỉ có thể chứa các chữ cái, số và dấu gạch dưới. Chúng có thể bắt đầu bằng một chữ cái hoặc một dấu gạch dưới, nhưng không phải bằng một số.

❑Không được phép sử dụng dấu cách trong tên biến, nhưng có thể sử dụng dấu gạch dưới để tách các từ trong tên biến.

✓ message_1

✗ 1_message.

✓ welcome_message

✗ welcome message

2.1.1. Định danh và sử dụng các biến

- Tránh sử dụng các từ khóa Python và tên hàm làm tên biến
- Tên biến phải ngắn nhưng mang tính mô tả.

- Cẩn thận khi sử dụng chữ cái viết thường l và chữ cái viết hoa O vì chúng có thể bị nhầm lẫn với số 1 và số 0

DOs

student_name

name_length

DON'Ts

s_n

length_of_persons_name

Tránh đặt tên lỗi khi sử dụng các biến

❑Viết mã sau, bao gồm thông báo từ sai chính tả được in đậm:

```
message = "Hello Python Crash Course reader!"
print(mesage)
```

❑Dấu vết (traceback) là một bản ghi về nơi trình thông dịch gặp sự cố khi cố gắng thực thi code đã viết.

```
Traceback (most recent call last):
① File "hello_world.py", line 2, in <module>
② print(mesage)
③ NameError: name 'mesage' is not defined
```

Tránh đặt tên lỗi khi sử dụng các biến

- Xem xét trường hợp sau:

```
mesage = "Hello Python Crash Course reader!"
print(mesage)
```

- Kết quả:

```
Hello Python Crash Course reader!
```

Biến là các nhãn

❑Các biến thường được mô tả như các hộp có thể lưu trữ các giá trị. Ý tưởng này có thể hữu ích khi mới sử dụng một vài biến, nhưng đó không phải là cách chính xác để mô tả cách các biến được biểu diễn bên trong bằng Python.

❑Cách tốt hơn là nên coi các biến dưới dạng nhãn mà chúng ta có thể gán cho các giá trị. Ta có thể cũng nói rằng một biến tham chiếu đến một giá trị nhất định.

2.2. Chuỗi (Strings)

❑Một chuỗi là một chuỗi các ký tự. Mọi thứ bên trong dấu ngoặc kép đều được xem xét một chuỗi với Python và ta có thể sử dụng dấu ngoặc đơn hoặc dấu ngoặc kép xung quanh chuỗi :

❑Sự linh hoạt này cho phép ta sử dụng dấu nháy đơn hoặc nháy kép trong chuỗi.

```
'I told my friend, "Python is my favorite language!"'
"The language 'Python' is named after Monty Python, not the snake."
"One of Python's strengths is its diverse and supportive community."
```

```
"This is a string."
'This is also a string.'
```

Sử dụng chữ hoa trong chuỗi với các phương thức

- Phương thức title() thay đổi chữ thường thành chữ hoa như tiêu đề, do đó mỗi ký tự đầu chữ sẽ thành chữ hoa.

```
name = "ada lovelace"
print(name.title())
```

```
Ada Lovelace
```

- Phương thức upper() đổi hết thành chữ hoa
- Phương thức lower() đổi hết thành chữ thường

```
name = "Ada Lovelace"
print(name.upper())
print(name.lower())
```

```
ADA LOVELACE
ada lovelace
```

Sử dụng biến trong chuỗi

- Sử dụng giá trị của một biến bên trong một chuỗi. (f-strings)
- Sử dụng f-string để tạo thông điệp hoàn chỉnh
- Sử dụng f-string để soạn thông điệp, sau đó gán toàn bộ thông điệp cho một biến

```
first_name = "ada"
last_name = "lovelace"
full_name = f"{first_name} {last_name}"
print(full_name)
```

```
ada lovelace
```

```
first_name = "ada"
last_name = "lovelace"
full_name = f"{first_name} {last_name}"
print(f"Hello, {full_name.title()}!")
```

```
Hello, Ada Lovelace!
```

```
first_name = "ada"
last_name = "lovelace"
full_name = f"{first_name} {last_name}"
message = f"Hello, {full_name.title()}!"
print(message)
```

```
Hello, Ada Lovelace!
```

Thêm khoảng trắng vào chuỗi với Tab hoặc dòng mới

❑Để thêm Tab vào chuỗi, sử dụng tổ hợp ký tự: \t

```
>>> print("Python")
Python
>>> print("\tPython")
        Python
```

❑Để thêm một dòng mới trong một chuỗi, hãy sử dụng tổ hợp ký tự \ n

```
>>> print("Languages:\nPython\nC\nJavaScript")
Languages:
Python
C
JavaScript
```

Bỏ khoảng trắng

- Khoảng trắng thừa có thể gây nhầm lẫn, do cần loại bỏ trước khi thực hiện các tác vụ.
- Python giúp loại bỏ dễ dàng khoảng trắng thừa từ dữ liệu người dùng nhập vào bằng phương thức strip()

- Lưu ý: Để xóa khoảng trắng vĩnh viễn, ta thực hiện phép gán:

```
>>> favorite_language = 'python '
>>> favorite_language
'python '
>>> favorite_language.rstrip()
'python'
>>> favorite_language
'python '

>>> favorite_language = 'python '
>>> favorite_language = favorite_language.rstrip()
>>> favorite_language
'python'
```

Bỏ khoảng trắng (t)

- Xóa khoảng trắng bên phải: rstrip()
- Xóa khoảng trắng bên trái: lstrip()

```
>>> favorite_language = ' python '
>>> favorite_language.rstrip()
' python'
>>> favorite_language.lstrip()
'python '
>>> favorite_language.strip()
'python'
```

Tránh lỗi cú pháp với String

❑Cách sử dụng chính xác dấu nháy kép và dấu nháy đơn:

```
message = "One of Python's strengths is its diverse community."
print(message)
```

```
One of Python's strengths is its diverse community.
```

❑Nếu sử dụng dấu nháy đơn, Python không thể xác định vị trí chuỗi nên kết thúc:

```
message = 'One of Python's strengths is its diverse community.'
print(message)
```

```
File "apostrophe.py", line 1
message = 'One of Python's strengths is its diverse community.'
          SyntaxError: invalid syntax
```

2.3 Số - Numbers

❑Các con số được sử dụng khá thường xuyên trong lập trình để lưu điểm số trong trò chơi, đại diện cho dữ liệu trong trực quan hóa, lưu trữ thông tin trong các ứng dụng web và các ứng dụng khác.

❑Python xử lý các số theo các cách khác nhau, dựa trên cách sử dụng các số.

Số nguyên - Integers

Có thể cộng (+), trừ (-), nhân (*) và chia (/) số nguyên trong Python

```
>>> 2 + 3
5
>>> 3 - 2
1
>>> 2 * 3
6
>>> 3 / 2
1.5
```

❑Trong một phiên đầu cuối, Python trả về kết quả của toán tử. Python sử dụng hai ký hiệu nhân để biểu diễn số mũ:

```
>>> 3 ** 2
9
>>> 3 ** 3
27
>>> 10 ** 6
1000000
```

Số nguyên – Integers (t)

❑Python cũng hỗ trợ thứ tự các phép toán, có thể sử dụng nhiều các phép toán trong một biểu thức.

```
>>> 2 + 3*4
14
```

❑Có thể sử dụng dấu ngoặc đơn để sửa đổi thứ tự của các phép toán để Python có thể đánh giá biểu thức theo thứ tự được chỉ định

```
>>> (2 + 3) * 4
20
```

Số thực - Floats

- Python gọi bất kỳ số nào có dấu thập phân là số thực (float)

- Chỉ cần nhập các số cần sử dụng và Python sẽ xử lý số đúng như mong đợi

```
>>> 0.1 + 0.1
0.2
>>> 0.2 + 0.2
0.4
>>> 2 * 0.1
0.2
>>> 2 * 0.2
0.4
```

Số nguyên và số thực

❑Khi chia hai số bất kỳ, ngay cả khi chúng là số nguyên dẫn đến kết quả đều sẽ là số thực

```
>>> 4/2
2.0
```

❑Nếu kết hợp số nguyên và số thực trong một phép toán, kết quả sẽ nhận được là số thực

```
>>> 1 + 2.0
3.0
>>> 2 * 3.0
6.0
>>> 3.0 ** 2
9.0
```

Các gạch dưới trong số

❑Khi viết các số dài, ta có thể nhóm các chữ số bằng dấu gạch dưới để làm cho các số lớn dễ đọc hơn:

```
>>> universe_age = 14_000_000_000
```

❑Khi in ra số dùng gạch dưới, Python chỉ in ra các số

```
>>> print(universe_age)
14000000000
```

Gán nhiều biến cùng lúc

❑Gán giá trị cho nhiều biến chỉ bằng một dòng duy nhất, điều này làm cho chương trình ngắn hơn và dễ đọc hơn

❑Ví dụ: đây là cách khởi tạo các biến x, y và z về 0

```
>>> x, y, z = 0, 0, 0
```

❑Cần tách các tên biến bằng dấu phẩy và thực hiện tương tự với các giá trị và Python sẽ chỉ định từng giá trị tương ứng với biến theo vị trí

Hằng số - Constants

- Một hằng số giống như một biến có giá trị không đổi trong suốt vòng đời của một chương trình.
- Python không có sẵn các loại hằng số, nhưng các lập trình viên Python sử dụng tất cả các chữ cái viết hoa để chỉ ra một biến nên được coi là không đổi và không bao giờ bị thay đổi:
- Khi muốn coi biến là một hằng số trong mã, hay viết hoa tất cả các chữ cái trong tên biến.

```
MAX_CONNECTIONS = 5000
```

2.4. Chú thích (Comments)

❑Chú thích là một tính năng cực kỳ hữu ích trong hầu hết các ngôn ngữ lập trình. Mọi thứ ta đã viết trong các chương trình cho đến nay đều là mã Python.

❑Khi các chương trình trở nên dài hơn và phức tạp hơn, ta nên thêm ghi chú trong các chương trình mô tả cách tiếp cận tổng thể đối với vấn đề bài toán đang giải quyết

Cách viết các chú thích

❑Trong Python, dấu thăng (#) chỉ ra một chú thích. Bất cứ điều gì theo sau dấu thăng trong mã bị trình thông dịch Python bỏ qua.

```
# Say hello to everyone.
print("Hello Python people!")
```

❑Python sẽ bỏ qua dòng đầu tiên vì có gặp dấu #, chỉ thực thi dòng thứ hai

```
Hello Python people!
```

Loại chú thích nên viết

- Giải thích mã nguồn để làm gì và cách làm cho mã nguồn hoạt động
- Viết rõ ràng, ngắn gọn nhận xét trong mã nguồn
- Cách tiếp cận để tìm ra mã nguồn hợp lý

Kết chương

❑Nội dung đã học:

- Cách làm việc với Biến
- Chuỗi và hiển thị chuỗi bằng chữ thường, chữ hoa
- Sử dụng và loại bỏ khoảng trắng
- Số nguyên, số thực
- Cách làm việc với dữ liệu số
- Cách viết chú thích

❑Nội dung chương tiếp theo:

- Lưu trữ các bộ thông tin trong cấu trúc dữ liệu được gọi là danh sách.
- Làm việc trên danh sách
- Sử dụng thông tin trong danh sách

Bài tập chương 2

- Bài 2-1. Simple Message: Gán thông điệp vào một biến và in ra thông điệp đó
- Bài 2-2. Simple Message: Gán thông điệp vào một biến và in thông điệp đó ra. Sau đó thay đổi giá trị của biến thành một thông điệp khác và in thông điệp mới đó ra.
- Bài 2-3. Personal Message: sử dụng một biến đại diện cho tên của một người và in thông điệp cho người đó. Thông điệp cần đơn giản như “Hello Eric, would you like to learn some Python today?”.
- Bài 2-4. Name Cases: Sử dụng một biến để biểu diễn tên của một người, sau đó in tên người đó với chữ thường, chữ hoa, và chữ kiểu tiêu đề (title).
- Bài 2-5. Famous Quote: Tìm một trích dẫn từ một người nổi tiếng, sau đó in trích dẫn và tên tác giả của trích dẫn đó. Đầu ra như dưới đây (bao gồm cả ký tự trích dẫn):
- Albert Einstein once said, “A person who never made a mistake never tried anything new.”
- Bài 2-6. Famous Quote2: Lặp lại ví dụ 2-5 nhưng lần này gán tên người nổi tiếng vào biến famous_person. Sau đó, gán thông điệp của người đó vào biến message, sau đó in ra thông điệp như trong bài tập 2-5

Bài tập chương 2 (t)

- Bài 2-7. Stripping Names: Sử dụng một biến để đại diện cho tên một người bao gồm một số ký tự trắng tại phần đầu và cuối của tên. Đảm bảo rằng mỗi tổ hợp ký tự “\t” và “\n” được sử dụng ít nhất một lần.
- In ra tên một lần để thấy các ký tự trắng quanh tên được hiển thị. Sau đó in tên có sử dụng các phương thức lstrip(), rstrip(), strip().
- Bài 2-8. Number Eight: Viết các phép tính cộng, trừ, nhân, chia các phép toán mà mỗi kết quả cho ra là số 8. Đầu ra có bốn dòng với số 8 xuất hiện một lần trên mỗi dòng.
- Bài 2-9. Favorite Number: Sử dụng một biến để đại diện cho một con số yêu thích. Sau đó, sử dụng biến đó, tạo một thông điệp tiết lộ số yêu thích đó. In ra thông điệp đó.
- Bài 2-10. Adding Comments: Chọn 2 chương trình đã viết trong các phần trước và thêm ít nhất mỗi ghi chú cho chương trình đó. Nếu không có gì đặc biệt để ghi chú thì ghi tên, và ngày viết chương trình vào đầu mỗi tệp chương trình. Sau đó ghi chú mô tả chương trình thực hiện vấn đề gì.
- Bài 2.11. Zen of Python: gõ lệnh import this ở cửa sổ terminal và nhìn kết quả hiển thị để thấy được các nguyên lý của Python.