5. Kiểu dữ liệu và biến

Ngay cả khi bạn nghĩ rằng bạn biết rất nhiều về các kiểu dữ liệu và các biến, bởi vì bạn đã lập trình các ngôn ngữ bậc thấp như C, C ++ hoặc các ngôn ngữ lập trình tương tự khác, chúng tôi vẫn khuyên bạn nên đọc chương này. Các kiểu dữ liệu và các biến trong Python khác nhau ở một số khía cạnh so với các ngôn ngữ lập trình khác. Có số nguyên, số thực, chuỗi, và nhiều hơn nữa, nhưng mọi thứ không giống như trong C hoặc C++. Nếu bạn muốn sử dụng các danh sách liên kết trong C, bạn sẽ phải tự xây dựng lại, ví dụ: cách cấp phát bộ nhớ, tổ chức danh sách liên kết. Bạn cũng sẽ phải thực hiện các chức năng cơ bản như tìm kiếm và truy cập. Trong khi đó, Python đã cung cấp các kiểu dữ liệu đó như là một phần lõi của ngôn ngữ, ví dụ list, dictionary,set, .vv.

Biến

Như tên của nó, một biến là cái gì đó có thể thay đổi. Một biến là một cách để đề cập đến một vị trí bộ nhớ được sử dụng bởi một chương trình máy tính. Biến là một tên tượng trưng cho vị trí thực tế này. Vị trí bộ nhớ này chứa các giá trị, như số, văn bản hoặc các kiểu phức tạp hơn.

Một biến có thể được xem như là một container để lưu trữ giá trị. Khi chương trình đang chạy, các biến được truy cập và đôi khi được thay đổi, nghĩa là một giá trị mới sẽ được gán cho biến. **Một trong những khác biệt chính giữa Python và các ngôn ngữ khác như C, C ++ hay Java là cách mà nó tự nhận biết kiểu dữ liệu của biến. Trong các ngôn ngữ đó (C,C++) mọi biến phải có một kiểu dữ liệu xác định**. Ví dụ. Nếu một biến là của kiểu số nguyên, chỉ có số nguyên mới có thể được lưu trong biến. Trong Java hoặc C, mọi biến phải được khai báo, tức là liên kết nó với một kiểu dữ liệu, trước khi sử dụng nó. **Việc khai báo các biến không bắt buộc trong Python** Nếu có nhu cầu sử dụng một biến, bạn nghĩ về một cái tên và bắt đầu sử dụng nó như là một biến.

Một khía cạnh đáng chú ý của Python: Không chỉ giá trị của một biến có thể thay đổi trong quá trình thực hiện chương trình mà thậm chí cả kiểu dữ liệu của biến cũng có thể thay đổi. Bạn có thể gán một giá trị số nguyên cho một biến, sử dụng nó như một số nguyên trong một thời gian và sau đó có thể gán một chuỗi cho biến đó.

Biến và định danh

Các biến và định danh thường bị nhầm là các từ đồng nghĩa. Trong thuật ngữ đơn giản: Tên của một biến là một định danh, nhưng một biến "không chỉ là một cái tên". Một biến ngoài tên, nó còn có kiểu dữ liệu, phạm vi, và cả giá trị. Bên cạnh đó, một định danh không chỉ được sử dụng cho các biến. Một định danh có thể biểu thị các thực thể khác nhau như nhãn, chương trình con hoặc các hàm, các gói, vv.

Cách đặt tên biến trong Python

Mỗi ngôn ngữ có các quy tắc để đặt tên định danh. Các quy tắc trong Python như sau:

Một định danh hợp lệ là một chuỗi các ký tự có độ dài lớn hơn 0 và thỏa mãn:

Ký tự bắt đầu có thể là dấu gạch dưới "_" hoặc chữ viết hoa hoặc chữ thường.

Các ký tự sau ký tự bắt đầu có thể là bất cứ thứ gì được cho phép như một ký tự bắt đầu theo sau là chữ số.

Định danh được phân biệt chữ hoa, chữ thường!

Tên định danh của biến không được phép giống các từ mặc định khóa thuộc Python.

Từ khóa trong Python

Ví dụ các từ khóa trong Python:*and, as, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while, with, yield*

Thay đổi các loại dữ liệu và vị trí lưu dữ liệu

Như chúng ta đã nói ở trên, loại biến có thể thay đổi trong quá trình thực thi chương trình. Chúng tôi minh họa điều này:

```
>>> i = 42  #i is an integer
>>> print i, type(i)
42 <type 'int'>
>>> i = "Python"   # i is assigned an string
>>> print i, type(i)
Python <type 'str'>
```

python tự động biểu diễn vật lý cho các loại dữ liệu khác nhau, tức là một giá trị số nguyên sẽ được lưu trữ ở vị trí bộ nhớ khác với một số thực hoặc một chuỗi.

Điều gì đang xảy ra, khi chúng ta thực hiện đoạn mã sau?

```
>>> x = 3
>>> y = x
>>> y = 2
```

Phép toán đầu tiên không phải là vấn đề: Python chọn vị trí bộ nhớ cho x và lưu lại giá trị số nguyên 3. Phép gán thứ hai có giá trị hơn: Trực giác, bạn có thể giả sử rằng Python sẽ tìm vị trí khác cho biến y và sẽ sao chép giá trị của 3 tại đây. Nhưng Python đi theo cách riêng của mình, khác với cách của C và C++. Vì cả hai biến sẽ có cùng một giá trị sau khi gán, Python cho phép y trỏ đến vị trí bộ nhớ của x. Câu hỏi quan trọng xuất hiện trong dòng tiếp theo của mã. Y sẽ được đặt thành giá trị số nguyên 2. Điều gì sẽ xảy ra với giá trị của x? Các lập trình viên dùng C sẽ cho rằng x cũng sẽ được thay đổi thành 2, bởi vì chúng ta đã nói trước đó biến y đang "trỏ" đến vị trí của x. Nhưng đây không phải là một con trỏ trong C. Bởi vì x và y sẽ không chia sẻ cùng một giá trị nữa, y có vị trí bộ nhớ riêng của mình để chứa 2 và x chỉ đến 3, như có thể nhìn thấy trong hình trên.

Nhưng những gì chúng tôi đã nói trước đó không thể được xác định bằng cách gõ vào ba dòng mã. Nhưng làm thế nào chúng ta có thể chứng minh nó? Mỗi biến có một id riêng chúng ta có thể sử dụng nó cho mục đích này. Mỗi cá thể (đối tượng hoặc biến) có một nhận dạng (id), tức là một số nguyên duy nhất trong chương trình hoặc tập lệnh, nghĩa là các đối tượng khác nhau có các id khác nhau.

Vì vậy, chúng ta hãy nhìn vào ví dụ trước của chúng ta và làm thế nào các đặc tính sẽ thay đổi:

```
>>> x = 3
>>> id(x)
6580376L
>>> id(3)
6580376L
>>> y = x
>>> id(y)
6580376L
>>> y = 2
>>> id(y)
6580400L
>>> id(2)
6580400L
```

Số

Có bốn kiểu dữ liệu số được xây dựng sẵn trong python là:

Số nguyên: dạng thông thường, e.g. 123; dạng Octal (cơ số 8), e.g. 010; dạng Hex (cơ số 16), e.g. 0xA0F.

```
>>> 123
123
>>> 010
8
>>> 0xA0F
2575
```

Số nguyên kiểu Long: Không có giới hạn về size

```
>>> a = 999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999
>>> type(a)
<type 'long'>
>>>
```

Số thực: 0.133131134535353

Số phức: 3 + 5j

```
>>> type(3+4j)
<type 'complex'>
```

Chuỗi

Chuỗi được đánh dấu bởi các dấu ngoặc đơn hoặc kép:

+ Tạo từ dấu nháy đơn: ‘dung mot dau ngoac’. (ví dụ: str1)

+ Tạo từ dấu nháy kép: “it’s me” bạn có thể thấy phía trong có thể chứa kí tự (‘). (ví dujL str2)

+ Tạo từ dấu 3 nháy đơn ''': phía trong có thể chứa kí tự (‘) hoặc (“) hoặc xuống dòng tuy ý trong terminal. (ví dụ str3 và str4)

```
>>> srt1 = 'dung mot dau ngoac'
>>> str2 = "it's me"
>>> str3 = "'A String in triple quotes can extend
... over multiple lines like this one, and can contain
... 'single' and "double" quotes.'"
>>> str4 = "' A string " hai vai "'
```

Một chuỗi trong Python bao gồm một chuỗi các ký tự - chữ cái, số và ký tự đặc biệt. Tương tự như C, ký tự đầu tiên của một chuỗi có chỉ mục (index) là 0.

```
>>> str = "'A string " hai vai "'
>>> str[0]
```

Điểm nhấn trong Python đó là ta có thể dùng chỉ mục là một số âm. Nếu chỉ mục dương cho ta đánh chỉ mục từ TRÁI qua PHẢI bắt đầu là 0 thì với chỉ mục âm ta sẽ tính từ PHẢI qua TRÁI và bắt đầu của chỉ mục tại kí tự cuối cùng là -1

```
>>> str = "'A string " hai vai'"
>>> str[-1]
'i'
>>> str[len(str)-1]
'i'
```

Bảng sau sẽ mô tả lại những gì tôi nói phía trên:

string	P	Y	T	H	O	N
index	0	1	2	3	4	5
	-6	-5	-4	-3	-2	-1

Một vài functions và operators trong string:

Toán tử '+' để ghép 2 strings:

```
>>> str = "Look" + " for"
>>> str
'Look for'
```

Toán tử "*" để lặp lại chuỗi:

```
>>> str = " it's me "
>>> str * 4
" it's me  it's me  it's me  it's me "
>>>
```

Toán tử "[]" để truy cập đến kí tự của chuỗi: str[-1]

Toán tử ":" hay slicing trong string. Slicing là một kĩ thuật hay trong Python giúp ta có thể có được thông tin của string bắt đầu từ kí tự có chỉ mục là 'i' đến kí tự có chỉ mục là 'j'.

```
>>> str = "***it's me****"
>>> str[5:-2]
"'s me**"
```

Tìm size: len("string")

Giống như các chuỗi trong Java nhưng không giống như C hoặc C++, các chuỗi Python không thể thay đổi. Cố gắng thay đổi một vị trí lập chỉ mục sẽ gây ra một lỗi:

```
>>> str = "it's me"
>>> str[1] = "r"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

Trong xử lý chuỗi chúng ta quan tâm đến dấu gạch chéo ngược "\" để chỉ định kí tự hoặc nhóm kí tự phía sau nó là kí tự đặc biệt không còn mang ý nghĩa kí tự thông thường (escape characters)

	Ý nghĩa

\newline	Bỏ qua
\\	\
\'	‘
\"	“
\a	ASCII Bell (BEL)
\b	ASCII Backspace (BS)
\f	ASCII Formfeed (FF)
\n	ASCII Linefeed (LF)
\N{name}	name sẽ ở mã Unicode (Unicode only)
\r	ASCII Carriage Return (CR)
\t	ASCII Horizontal Tab (TAB)
\uxxxx	xxxx dạng 16-bit hex (Unicode only)
\Uxxxxxxxx	xxxxxxxx dạng 32-bit hex (Unicode only)
\v	ASCII Vertical Tab (VT)
\ooo	ooo dạng cơ số 8
\xhh	hh dạng cơ số 16

```
>>> '\''
"'"
>>> '\a'
'\x07'
>>> '\010'  # 010( co so 8) = 0x08(co so 16) = 8 (co so 10)
'\x08'
```

Kết Luận

Qua bài này, các bạn đã có một cái nhìn chi tiết hơn về biến, định danh cũng như các kiểu dữ liệu (kiểu số và kiểu chuỗi) và vị trí lưu trữ của chúng trong bộ nhớ tương ứng. Việc khai báo biến trong Python cũng thuận tiện và linh hoạt hơn hẳn một số ngôn ngữ lập trình khác như C/C++ hay Java. Tuy nhiên các bạn cần chú ý trong cách đặt tên biến cần tuân theo các nguyên tắc chúng tôi đã đề cập và tránh các từ khóa trong Python. Cách đánh số chỉ mục đặc biệt (có thể dùng số âm) và các phép toán thực hiện trên chuỗi.

Trong bài tiếp theo chúng ta sẽ tìm hiểu về các toán tử cơ bản trong Python.