Python Data type - Kiểu dữ liệu trong Python

Các kiểu dữ liệu trong Python

Python bao gồm các kiểu dữ liệu cơ bản sau:

- Numbers
- String
- List
- Tuple
- Dictionary

1. Python Number

Number được sử dụng để lưu trữ các giá trị số. Một đối tượng Number được tạo khi khai báo và gán giá trị cho chúng.

```
var1 = 1
var2 = 10
```

Bạn có thể delete tham chiếu tới 1 đối tượng Number bằng cách sử dụng cú pháp `del`:

```
del var1[,var2[,var3[....,varN]]]
```

Các kiểu Number trong Python bao gồm:

- int (signed integers)
- long (long integers, they can also be represented in octal and hexadecimal)
- float (floating point real values)
- complex (complex numbers)

int	long	float	complex
10	51924361L	0.0	3.14j
100	-0x19323L	15.20	45.j

-786	0122L	-21.9	9.322e-36j
080	0xDEFABCECBDAECBFBAE I	32.3+e18	.876j
-0490	535633629843L	-90.	-.6545+0J
-0x260	-052318172735L	-32.54e100	3e+26J
0x69	-4721885298529L	70.2-E12	4.53e-7j

2. Python String

String trong Python được định nghĩa như một tập ký tự được thể hiện trong cặp dấu nháy (nháy đơn hoặc đôi):

- Để truy cập từng ký tự trong String, sử dụng mảng [] (hoặc [:]) với vị trí bắt đầu từ 0.
- Toán tử (+) String là lệnh thực hiện ghép chuỗi (concat) và toán tử (*) là lệnh lặp chuỗi.

```
str = 'Hello World!'

print str           # Prints complete string
print str[0]        # Prints first character of the string
print str[2:5]      # Prints characters starting from 3rd to 5th
print str[2:]       # Prints string starting from 3rd character
print str * 2       # Prints string two times
print str + "TEST"  # Prints concatenated string
```

Output:

```
Hello World!
H
llo
llo World!
Hello World!Hello World!
Hello World!TEST
```

3. Python List

List là một kiểu dữ liệu dãy (sequence) các phần tử (element), nó cho phép loại bỏ, hoặc thêm các phần tử vào danh sách, đồng thời cho phép cắt lát (slice) các phần tử.

- Để truy cập từng phần tử trong Python List, sử dụng mảng [] (hoặc [:]) với vị trí bắt đầu từ 0.
- Toán tử (+) Python List là lệnh thực hiện ghép 2 List thành một và toán tử (*) là lệnh lặp List để thành một List có độ dài gấp đôi.

```
list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tinylist = [123, 'john']

print list          # Prints complete list
print list[0]       # Prints first element of the list
print list[1:3]     # Prints elements starting from 2nd till 3rd
print list[2:]      # Prints elements starting from 3rd element
print tinylist * 2  # Prints list two times
print list + tinylist # Prints concatenated lists
```

Output:

```
['abcd', 786, 2.23, 'john', 70.2]
abcd
[786, 2.23]
[2.23, 'john', 70.2]
[123, 'john', 123, 'john']
['abcd', 786, 2.23, 'john', 70.2, 123, 'john']
```

4. Python Tuple

Tuple là một kiểu dữ liệu chuỗi khác tương tự như List. Tuple bao gồm một số giá trị được phân tách bằng dấu phẩy. Tuy nhiên, không giống như List, các phần tử trong Tuple được đặt trong dấu ngoặc đơn.

Tuple có thể được coi là danh sách chỉ đọc (read-only), các phần tử trong Tuple là cố định và không thể thay đổi.

```
tuple = ( 'abcd', 786 , 2.23, 'john', 70.2  )
tinytuple = (123, 'john')

print tuple           # Prints complete list
print tuple[0]        # Prints first element of the list
print tuple[1:3]      # Prints elements starting from 2nd till 3rd
print tuple[2:]       # Prints elements starting from 3rd element
print tinytuple * 2   # Prints list two times
print tuple + tinytuple # Prints concatenated lists
```

Output:

```
('abcd', 786, 2.23, 'john', 70.2)
abcd
(786, 2.23)
(2.23, 'john', 70.2)
(123, 'john', 123, 'john')
('abcd', 786, 2.23, 'john', 70.2, 123, 'john')
```

- Đoạn lệnh sau là không hợp lệ vì thực hiện update giá trị phần tử của Tuple:

```
tuple = ( 'abcd', 786 , 2.23, 'john', 70.2  )
list = [ 'abcd', 786 , 2.23, 'john', 70.2  ]
tuple[2] = 1000    # Invalid syntax with tuple
list[2] = 1000     # Valid syntax with list
```

5. Python Dictionary

Dictionary của Python là loại bảng băm (hash table). Chúng hoạt động như các mảng (array) hoặc băm (hash) được tìm thấy trong Perl và bao gồm các cặp key-value. Dictionary key có thể là hầu hết mọi loại Python, nhưng thường là Number hoặc String. Mặt khác, các giá trị có thể là bất kỳ đối tượng Python tùy ý.

Từ điển được bao quanh bởi dấu ngoặc nhọn {} và các giá trị có thể được chỉ định và truy cập bằng dấu ngoặc vuông ([]).

```
ict = {}
dict['one'] = "This is one"
dict[2]     = "This is two"

tinydict = {'name': 'john','code':6734, 'dept': 'sales'}



print dict['one']       # Prints value for 'one' key
print dict[2]           # Prints value for 2 key
print tinydict          # Prints complete dictionary
print tinydict.keys()   # Prints all the keys
print tinydict.values() # Prints all the values
```

Output:

```
This is one
This is two
{'dept': 'sales', 'code': 6734, 'name': 'john'}
['dept', 'code', 'name']
['sales', 6734, 'john']
```

6. Chuyển đổi kiểu dữ liệu

Có một số hàm dựng sẵn để thực hiện chuyển đổi từ loại dữ liệu này sang loại dữ liệu khác. Các hàm này trả về một đối tượng mới biểu thị giá trị được chuyển đổi.

#	Function	Mô tả
1	int(x [,base])	Chuyển đổi x sang kiểu Integer
2	long(x [,base])	Chuyển đổi x sang kiểu Long
3	float(x)	Chuyển đổi x sang kiểu Float
4	complex(real [,imag])	Tạo một số phức
5	str(x)	Chuyển đổi x sang kiểu String
6	repr(x)	Chuyển đổi x sang kiểu expression string
7	eval(str)	Đánh giá một chuỗi và trả về một đối tượng
8	tuple(s)	Chuyển đổi sang kiểu Tuple
9	list(s)	Chuyển đổi sang kiểu List
10	set(s)	Chuyển đổi sang kiểu Set
11	dict(d)	Tạo một Dictionary, d phải là một sequence (key, value) tuple
12	frozenset(s)	Chuyển đổi sang kiểu Frozen Set
13	chr(x)	Chuyển đổi một Integer thành Character
14	unichr(x)	Chuyển đổi Integer thành Unicode Character

15	ord(x)	Chuyển đổi Character thành kiểu Integer
16	hex(x)	Chuyển đổi Integer thành Hex String
17	oct(x)	Chuyển đổi Integer thành Octal String

Revision #6
Created 8 November 2019 15:23:32 by Laptrinh.vn
Updated 5 June 2021 15:01:36 by Laptrinh.vn