



CÁC KIỂU DỮ LIỆU CƠ BẢN

Kỹ Thuật Lập Trình Python

Giảng viên: ThS. Nghi Hoàng Khoa | Email: khoanh@uit.edu.vn



1. Kiểu số (Number)
2. Kiểu chuỗi (String)
3. Kiểu danh sách (List)
4. Kiểu tuple /'tu:.pəl/, /'tʌ.pəl/
5. Kiểu từ điển (Dictionary)
6. Kiểu tập hợp (Set)
7. Kiểu Boolean
8. Giá trị None

- Phân loại kiểu dữ liệu số trong Python 3
 - **int**
 - **float**
 - **complex**: dạng **a + bj**
- => Có thể chuyển từ **float** ↔ **int** hoặc ngược lại. Hoặc, có thể chuyển đổi lẫn nhau giữa 03 kiểu dữ liệu.
- SV tìm hiểu thêm về giá trị giới hạn của các kiểu.
- Kiểu complex được sử dụng khi nào trong thực tế?

- Cách chuyển kiểu dữ liệu

```
1  var1 = 10
2  print("Giá trị var1 = ", var1, "\n\tKiểu: ", type(var1))
3
4  var2 = float(var1)
5  print("Giá trị var2 = ", var2, "\n\tKiểu: ", type(var2))
6
7  var3 = 3 + 4j
8  print("Giá trị var3 = ", var3, "\n\tKiểu: ", type(var3))
9
10 var4 = complex(var2)
11 print("Giá trị var4 = ", var4, "\n\tKiểu: ", type(var4))
```

- Cách chuyển kiểu dữ liệu
 - Kết quả:

```
> python test.py
Giá trị var1 = 10
        Kiểu: <class 'int'>
Giá trị var2 = 10.0
        Kiểu: <class 'float'>
Giá trị var3 = (3+4j)
        Kiểu: <class 'complex'>
Giá trị var4 = (10+0j)
        Kiểu: <class 'complex'>
```

- Giá trị giới hạn (Bài tập về nhà)

- Các toán tử
 - Cộng: $+$
 - Trừ: $-$
 - Nhân: $*$ (Vậy còn toán tử $**$ thì sao?)
 - Chia: $/$
 - Chia lấy dư: $\%$
 - Chia lấy nguyên: $//$

- Minh họa

```
1  a = 10
2  b = 3
3
4  print("a + b = ", a + b)
5  print("a - b = ", a - b)
6  print("a * b = ", a * b)
7  print("a / b = ", a / b)
8  print("a % b = ", a % b)
9  print("a // b = ", a // b)
10
11 x = 3
12 print("a ^ x = ", a ** x)
```


- Minh họa
 - Kết quả

```
> python test.py
a + b = 13
a - b = 7
a * b = 30
a / b = 3.3333333333333335
a % b = 1
a // b = 3
a ^ x = 1000
```

- Các hàm cơ bản

- `int(x)`
- `float(x)`
- `complex(x)`
- `complex(x, y)`
- `abs(x)`: lấy tuyệt đối $|-9| = 9$
- `pow(x, y)`
- `round(x, n)`: trả về số thực có n số sau dấu thập phân
- `max(x1, x2, ...)`
- `min(x1, x2, ...)`
- `sqrt(x)` => trong thư viện `math`

- Thư viện math trong Python
 - Thư viện **math** cung cấp các hàm toán học đã xây dựng sẵn. Bao gồm các hàm lượng giác, hàm logarit...
- Cú pháp:

```
import math
```

Hoặc,

```
import math as m
```

- Các hàm cơ bản:
 - $\exp(x)$: trả về e^x
 - $\text{fabs}(x)$: trả về giá trị tuyệt đối của x .
 - $\text{ceil}(x)$: trả về số nguyên nhỏ nhất mà không nhỏ hơn x .
 - $\text{floor}(x)$: trả về số nguyên lớn nhất mà không lớn hơn x .
 - $\log(x)$
 - $\log_{10}(x)$
 - $\text{modf}(x)$: Hàm này có tác dụng chuyển đổi một số về một tuple. Tuple này chứa phần thập phân và phần nguyên của số đó, cả hai đều ở dạng float.
 - $\text{fmod}(x, y)$: giống như $x \% y$, áp dụng cho float

Toán tử **is** và **==**

- Minh họa dùng toán tử **is** và **==** so sánh 2 số.
(**is**: so sánh tính đồng nhất, **True** nếu 2 biến trỏ tới cùng một object)

```
1 a = 3
2 b = 3.0
```

```
1 a == b
```

True

```
1 a is b
```

False

Nâng cao

- Mô-đun Decimal: from decimal import Decimal
- Mô-đun Fraction: from fractions import Fraction

⇒Thực hiện phép toán cho số thập phân và phân số.

⇒Sẽ được giới thiệu ở phần module

Tạo số ngẫu nhiên

Hàm có sẵn range()

- range(n) # => trả về obj, range (miền giá trị) chứa từ 0 đến n-1
- range(start, stop)
- range(start, stop, step)

Tạo số ngẫu nhiên

Mô đun *random*

- Chuyên hỗ trợ tạo các giá trị ngẫu nhiên theo mọi tình huống đã định nghĩa sẵn thông qua các hàm.
- Trước khi dùng, phải **import random**

Hàm cơ bản trong mô đun *random*

- **random.random()** #=> random **float**
- **random.randrange(6)** # random **int** chosen from **range(6)**: [0; 5]
- **randrange(start, stop, step)**

Tạo số ngẫu nhiên

*Hàm cơ bản trong mô đun **random** (tt)*

- `choice(seq): random.choice(['apple', 'pear', 'banana'])`
- `random(): random.random() # random float`
- `random.randrange(6) # random integer chosen from range(6): [0, 5]`
- `randrange (start, stop, step):`


Tạo số ngẫu nhiên

Ví dụ

```
import math
import random

print(math.sqrt(9))
print(math.fabs(-9))
print(math.modf(5))
print(math.fmod(5, 7))

print(random.random())
print(random.randrange(3))
print(random.randrange(1, 4, 3))
```



C:\Python\python.exe
3.0
9.0
(0.0, 5.0)
5.0
0.7269710182940858
0
1

Process finished with

Tạo số ngẫu nhiên

*Hàm cơ bản trong mô đun **random** (tt)*

- `seed([x]): random.seed(98)` sau đó `random.random()`

Cho sv minh họa `seed()`

- **`shuffle(list)`**: sắp xếp các item trong list một cách ngẫu nhiên.
- **`uniform(a, b)`**: Get a random number in the range `[a, b)` or `[a, b]` depending on rounding. `A <= float <= B`

Tạo số ngẫu nhiên

Tạo số ngẫu nhiên theo yêu cầu sau:

- Một số ngẫu nhiên từ dãy (start, stop, step).
- Một số thực ngẫu nhiên r trong đoạn $[x, y]$

Bài tập

1. Viết chương trình nhập vào 4 số nguyên a, b, c, d. Tính trung bình cộng của 4 số trên và in kết quả ra màn hình.
2. Viết chương trình nhập vào 2 số nguyên a, b. Tính tổng, hiệu, tích, thương, chia lấy dư, chia lấy nguyên của 2 số trên và in kết quả ra màn hình. Kết quả phép chia làm tròn 2, 3 chữ số sau dấu chấm (ví dụ kết quả 3.333333 thì làm tròn 3.333).
3. Viết chương trình cho phép nhập vào số nguyên dương N có 2 chữ số. Xuất ra màn hình tổng các chữ số của N. Ví dụ: Nhập N=48, kết quả xuất ra màn hình là $4 + 8 = 12$
4. Viết chương trình cho phép nhập vào giờ, phút và giây (ví dụ 8 39 50). Hãy đổi ra giây và in kết quả ra màn hình.
5. Viết chương trình nhập vào năm sinh, in ra tuổi. Ví dụ nhập 1988 in ra (giả sử năm hiện tại là 2023): Ban sinh năm 1988 nay bạn 35 tuổi.

Bài tập

6. Nhập vào 3 số a, b, c. Sau đó in ra phương trình bậc 2 dạng

$$aX^2 + bX + C = 0$$

Ví dụ nhập a = 2, b = 5, c = 4 thì in ra $2X^2 + 5X + 4 = 0$

7. Viết chương trình in ra menu lựa chọn sau:

===== MENU =====

1. Hu tieu
2. Chao long
3. Banh canh
4. Bun rieu
5. Pho bo

=====

Moi nhap lua chon:

=====

Bài tập

8. Nhập bán kính của đường tròn, tính và in ra chu vi, diện tích của hình tròn.
9. Viết chương trình tính số đo kiểm tra sức khỏe BMI.
Công thức BMI = $Cân_nặng\ (kg) / Chiều_cao^2\ (m)$
10. Nhập vào số xe (gồm 5 chữ số). Cho biết số xe có mấy nút.
11. Cho 1 ký tự chữ thường. In ra ký tự chữ hoa tương ứng.
12. Viết chương trình xuất ra số (nguyên, thực) ngẫu nhiên theo yêu cầu sau:
 - 0 đến 100
 - 50 đến 99
 - -39 đến 79
 - -79 đến -39

Cài đặt phép tính biểu thức A như sau (giá trị được làm tròn 3 số thập phân):

Code it!

$$A = (32)^{0,2} - \left(\frac{1}{64}\right)^{-0,25} + \left(\frac{8}{27}\right)^{\frac{1}{3}}$$

Kết quả: -0.1617

Viết chương trình nhập giá trị cho hai số thực a , b . Tính giá trị biểu thức sau:

$$\frac{\sqrt{a}-\sqrt{b}}{\sqrt[4]{a}-\sqrt[4]{b}} - \frac{\sqrt{a}+\sqrt[4]{ab}}{\sqrt[4]{a}+\sqrt[4]{b}}$$

Code it!

- Kết quả: $\sqrt[4]{b}$

Viết chương trình thực hiện công việc nhập dữ liệu từ bàn phím các số a , b . Sau đó, thực hiện tính toán và in ra các kết quả của biểu thức sau:

Code it!

$$\left(\frac{a+b}{\sqrt[3]{a} + \sqrt[3]{b}} - \sqrt[3]{ab} \right) : (\sqrt[3]{a} - \sqrt[3]{b})^2$$

Kết quả = 1.0

Cài đặt tính giá trị biểu thức sau (kết quả được làm tròn 2 số thập phân):

Code it!

$$\left(\frac{2}{5}\right)(\sqrt{3}+2) \cdot \sqrt[4]{(\sqrt{3}-2)^4} - \left(\frac{25}{4}\right)^{-1\frac{1}{2}} + \left(\frac{2}{3}\right)^{\sqrt[5]{32}}$$

- Kết quả = $\frac{4}{9} = 0.44444$

Kiểu chuỗi (string)

String?

- String là một kiểu dữ liệu để lưu trữ các **ký tự** hay một **chuỗi các ký tự**.
- Cú pháp:

```
str_name = "chuỗi khởi tạo"
```

```
str_name = 'chuỗi khởi tạo'
```

- Dùng dấu **" "** hoặc **' '** đều được.

Sequence:



("term", "element" or "member" mean the same thing)

string

[start : stop : step]

- Để lấy chuỗi con trong chuỗi: sử dụng dấu ngoặc vuông “[]” và kèm theo thứ tự của các ký tự (ký tự đầu tiên là **0**) trong chuỗi => các lấy bằng chiều dương.
- Hoặc, có thể lấy bằng chiều âm, vị trí cuối str là **-1**
- Cú pháp như sau:

`str_name[start:stop:step] (< stop)`

Hoặc, `str_name[point]`

- Minh họa để trống start, stop, step

=> Hướng dẫn học viên minh họa 10 phút cách lấy ký tự, chuỗi con.

string

Minh họa

```
var1 = 'Hello World!'  
var2 = "Python Programming"  
  
print ("Kiểm tra giá trị var1[0]: ", var1[0])  
print ("Kiểm tra giá trị var2[1:5]: ", var2[1:5])
```

Kết quả:

```
C:\Python\python.exe C:/PythonProjects/PyPB/fc_string.py  
Kiểm tra giá trị var1[0]:  H  
Kiểm tra giá trị var2[1:5]:  ytho
```

string

Minh họa

```
var = "Hello World!"  
print ("Updated String :- ", var[:6] + "Python")
```

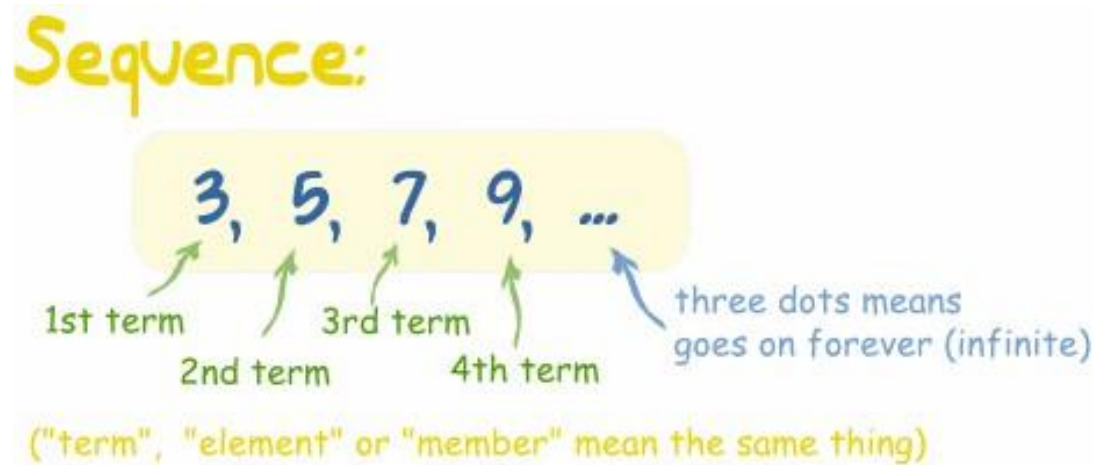
Kết quả:

```
C:\Python\python.exe C:/PythonProjects/PyPB/fc_string.py  
Updated String :-  Hello Python
```

string

Xử lý chuỗi

1. Ký hiệu định dạng chuỗi xuất ra
2. Toán tử liên quan đến chuỗi
3. Các hàm có sẵn liên quan đến chuỗi



string

Ký hiệu định dạng chuỗi xuất ra

Ký hiệu	Mô tả	Ví dụ	Kết quả
\	Escape Sequence	<code>print('I\'m Peter')</code> <code>print("C:\\Users\\HP\\nature")</code>	I'm Peter C:\Users\HP\nature
r hoặc R		<code>print(r"C:\Users\HP\nature")</code> <code>print(r'\n\t')</code> <code>print(R"\n\t")</code>	C:\Users\HP\nature \n\t \n\t
\n		<code>print('Line 1\nLine 2\nLine 3')</code>	Line 1 Line 2 Line 3
\t	Tab	<code>print('One\tTwo\tThree')</code>	One Two Three

string

Toán tử liên quan đến chuỗi

Toán tử	Mô tả
+	Cộng hai chuỗi, kết quả trả về là một chuỗi.
*	"Python"*2 ==> "PythonPython"
in	Kiểm tra một hoặc nhiều ký tự có nằm trong một chuỗi cho trước. Kết quả trả về là "True" nếu có xuất hiện
not in	Kiểm tra một hoặc nhiều ký tự không có nằm trong một chuỗi cho trước.
[index]	Lấy ký tự trong chuỗi
[start:stop:step]	Lấy chuỗi con

string

Hàm xử lý chuỗi

- `len(string)`: Trả về độ dài của chuỗi
- `split(str)`: Phân rã chuỗi dựa vào `str`
- `replace(old_str, new_str)`: thay đổi `old_str` thành `new_str`
- `count(str, beg=0, end=len(string))`: Đếm số lần `str` xuất hiện trong `string`, hoặc sub-string với chỉ định phạm vi tìm bằng `beg` và `end`.
- `find(str, beg=0, end=len(string))`: Xác định xem `str` xuất hiện trong `string` hay trong sub-string (xác định bằng `beg`, `end`). Kết quả trả về **chỉ số** nếu tìm thấy **đầu tiên** và **-1** nếu không tìm thấy.
- `index(str, beg=0, end=len(string))`: tương tự `find()`, nhưng bắt exception không tìm thấy.

string

Hàm xử lý chuỗi

- **isalnum()**: Trả về **true** nếu string có ít nhất 1 character và tất cả các character đều là chữ và số (alphanumeric: a, b... 1, 2, 3...), ngược lại là **false**.
- **isalpha()**: Trả về **true** nếu string có ít nhất 1 ký tự và tất cả các ký tự đều là chữ cái (alphabetic) và ngược lại là **false**.
- **isdigit()**: Trả về **true** nếu string chỉ chứa các chữ số (digits) và ngược lại là **false**.

Hàm xử lý chuỗi

- `islower()`: Returns **true** if string has at least 1 cased character and all cased characters are in lowercase and **false** otherwise.
- `isnumeric()`: Returns **true** if a unicode string contains only numeric characters and **false** otherwise.
- `isspace()`: Returns **true** if string contains only whitespace characters and **false** otherwise.

string

Hàm xử lý chuỗi

- `istitle()`: Returns **true** if string is properly "titlecased" and false otherwise.
- `isupper()`: Returns **true** if string has at least one cased character and all cased characters are in uppercase and false otherwise.

string

Hàm xử lý chuỗi

- `capitalize()`: Capitalizes ký tự đầu tiên của chuỗi
- `upper()`: Converts all **UPPERCASE** letters in string to lowercase.
- `lower()`: Converts all **lower** letters.
- `max(str)`: Returns the max alphabetical character from the string str.
- `min(str)`: Returns the min alphabetical character from the string str.

Giới thiệu

- Là một chuỗi các ký tự tạo thành một mẫu dùng để tìm kiếm.
- RegEx có thể được sử dụng để kiểm tra xem một chuỗi có chứa mẫu tìm kiếm đã chỉ định hay không.
- Sử dụng gói “re”

```
import re
```

```
txt = "Trường Đại học Công nghệ Thông tin"  
x = re.search("^Trường.*tin$", txt)
```

```
if x:
```

```
    print("Trùng khớp")
```

```
else:
```

```
    print("Không trùng khớp")
```


Biểu thức chính quy

Hàm

Hàm	Giải thích
findall	Trả về một danh sách trùng khớp tất cả
search	Trả về đối tượng Match nếu trùng khớp
split	Phân rã chuỗi
sub	Thay thế chuỗi

Biểu thức chính quy

Ký tự	Giải thích	Example
[]	Tập ký tự	"[a-m]"
\	Theo sau \ là ý hiệu đặc biệt quy định riêng	"\d"
.	Đại diện cho bất kỳ ký tự nào	"he..o"
^	Bắt đầu với	"^hello"
\$	Kết thúc với	"planet\$"
*	Không hoặc nhiều lần xuất hiện	"he.*o"
+	Một hoặc nhiều lần xuất hiện	"he.+o"
?	Không hoặc một xuất hiện	"he.?o"
{ }	Chỉ định chính xác số lần xuất hiện	"he.{2}o"
	Hoặc	"falls stays"
()		

BÀI #1

Cho chuỗi:

Đại học Quốc gia, Khu phố 6, P. Linh Trung, Q. Thủ Đức, Tp. HCM

Code it!

Viết các lệnh theo yêu cầu sau:

1. Tách thành các sub-string:

- Đại học Quốc gia
- Khu phố 6
- P. Linh Trung
- Q. Thủ Đức
- Tp. HCM

2. Tách thành các sub-string:

- Đại học Quốc gia
- Khu phố 6
- Linh Trung
- Thủ Đức
- HCM

BÀI TẬP KIỂU CHUỖI

BÀI #2

Viết các lệnh chuyển chuỗi sau:

“công nghệ thông tin”

thành:

- “Công Nghệ Thông Tin”
- “CÔNG NGHỆ THÔNG TIN”
- “cÔNG nGHỆ tHÔNG tIN”
- “Công nghệ thông tin”

KIỂU DANH SÁCH (list)

Giới thiệu list

- List là kiểu dữ liệu dạng dãy nối tiếp (sequence).
- Gồm nhiều phần tử có cùng kiểu dữ liệu hoặc khác kiểu dữ liệu.
- Mỗi phần tử của dãy được gán cho một số thứ tự (hay còn gọi là vị trí chỉ mục), vị trí đầu tiên là 0, tiếp theo là 1, 2...
- Trong list vẫn qui định chiều **dương** và **âm**.
- Một số tài liệu gọi **list** là kiểu dữ liệu mảng.

list

Cách khởi tạo/khai báo?

- Dùng dấu cặp dấu [] để mở và đóng list.
- Mỗi phần tử cách nhau bởi dấu “,”

Cú pháp:

`list_name = [e1, e2, ..., en]`

list

Ví dụ

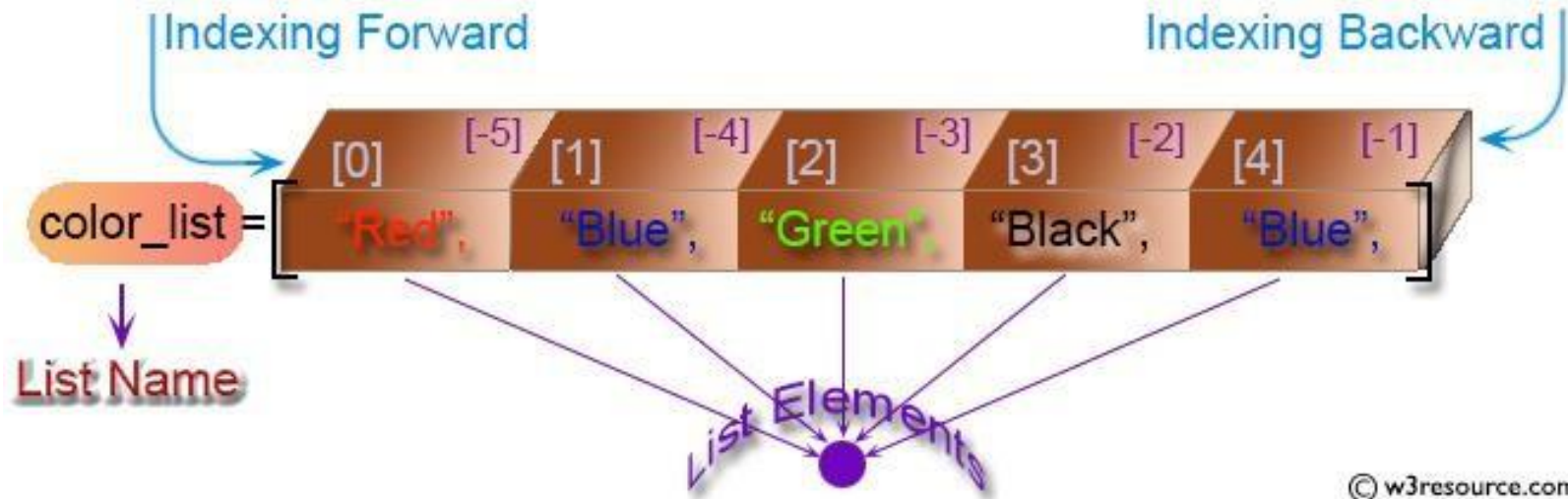
```
ds = [ 'Java', 12, 'C', 'R', 5.9, 'PyQt' ]
```

Chiều +	0	1	2	3	4	5
---------	---	---	---	---	---	---

Chiều -	-6	-5	-4	-3	-2	-1
---------	----	----	----	----	----	----

list

Ví dụ



list

Các thao tác cho phần tử trong list?

- index
- slice => sử dụng `[start : stop : step]`
- add
- multiply (*) nhân
- check
 - ⇒ loại, kiểu dữ liệu từng phần tử
 - ⇒ kiểu dữ liệu của phần tử trong list không nhất thiết phải cùng loại.

list

Index => lấy phần tử

- Chỉ số âm
- Chỉ số dương

Minh họa

- Viết các lệnh minh họa lấy giá trị phần tử bằng chỉ số dương và chỉ số âm.
- Thử trường hợp truy xuất các chỉ số không nằm trong list, bình luận lỗi.

list

Cập nhật và thêm mới giá trị

- Cập nhật giá trị các phần tử:

```
list_name[i] = new_value
```

- Dùng `append()` để thêm mới phần tử vào list (thêm vào cuối danh sách).

```
list_name.append(<phần tử cần thêm>)
```

- Dùng `extend()` thêm các phần tử của một list vào list ban đầu

```
list_name.extend([danh_sách])
```

list

Xóa

- Xóa toàn bộ danh sách:

```
del list_name
```

- Xóa phần tử ngay tại vị trí *i*:

```
del list_name[i]
```

- Xóa phần tử **đầu tiên** có giá trị là “value” trong danh sách:

```
list_name.remove("value")
```

Minh họa ba cách xóa phần tử trong danh sách.

list

Hàm xử lý cơ bản

- `len(list)`: Tính độ dài danh sách.
- `max(list)`: Trả về phần tử có giá trị lớn nhất.
- `min(list)`: Trả về phần tử có giá trị nhỏ nhất.
- `list(seq)`: Chuyển một đối tượng `seq` nào đó thành danh sách.

list

Phương thức

- `list.count(obj)`: Trả về số lần xuất hiện của `obj` trong `list`.
- `list.index(obj)`: Trả về chỉ số đầu tiên (vị trí) của `obj` trong `list`.
- `list.insert(index, obj)`: Chèn một `obj` vào vị trí `index` trong `list`, các phần tử còn lại dịch qua phải.
- `list.reverse()`: Đảo ngược các thứ tự các phần tử trong `list`.
- `list2 = list1.copy()` => phân biệt với `list2 = list1`
=> hàm `copy()` được sử dụng khá phổ biến.

list

• *Phương thức*

- `list.sort(key=None, reverse=False)`: sắp xếp các phần tử trong một danh sách theo thứ tự tăng dần (mặc định).
- Mặc định tham số `reverse=False`. Nếu `reverse=True` thì sắp xếp giảm dần.
- Tham số `key` nhận một hàm. Hàm này áp dụng cho từng phần tử trong danh sách để sắp xếp.

```
ds = [1, 9, 4, 8, 3, 6]  
ds.sort()  
  
print(ds)
```

```
ds = [1, 9, 4, 8, 3, 6]  
ds.sort(reverse=True)  
  
print(ds)
```

```
ds = ['r', 'a', '8', 'z', 'b', '6']  
ds.sort()  
  
print(ds)
```

list

Phương thức

```
ds = ['Python', 'C', 'C++', 'Java', 'CShape', 'R']  
ds.sort(key=len, reverse=True)
```

```
ds = [1, -9, 4, 8, -3, 6]  
ds.sort(key=abs)
```

```
import math
```

```
numbers = [100, 36, 64, 81, 16]  
numbers.sort(key=math.sqrt)
```


Khởi tạo một list bất kỳ gồm kiểu dữ liệu chuỗi và số. Sau đó, thực hiện các thao tác sau:

1. Index các phần tử đầu, cuối và giữa list
2. Slice từ vị trí begin đến end, begin và end nhập từ bàn phím.
3. Thao thác thêm, xóa, sửa dữ liệu tại vị trí bất kỳ.

Code it!

tuple

tuple là gì?

- Là một dãy có trật tự, giống như list.
- Tuple sử dụng cặp dấu ngoặc đơn “()”, còn danh sách thì dùng dấu ngoặc vuông “[]”.
- Tạo một tuple cũng đơn giản bằng cách đặt dấu phẩy “,” ngăn cách các phần tử với nhau.
- Sự khác biệt giữa tuple và list là các phần tử trong tuple **KHÔNG** thể thay đổi, còn list có thể thay đổi.

tuple

Cú pháp khai báo một tuple

`Tuple_name = (member1, member2, ..., memberN)`

Ví dụ:

- `tup1 = ('AI', 'DS', 2020, 2019)`
- `tup2 = (1, 2, 3, 4, 5)`
- Hoặc, `tup3 = "a", "b", "c", "d"`

tuple

- Tuple trống rỗng được viết bằng hai ngoặc đơn.

Ví dụ:

```
tup1 = ()
```

- Tuple chỉ có một phần tử, phải viết thêm dấu phẩy phía sau phần tử.

Ví dụ:

```
tup2 = (56,)
```

- Chỉ số của tuple giống như chỉ số trong chuỗi, danh sách, bắt đầu từ 0. Và, cũng có quy tắc chiều **âm-dương**.

tuple

Các thao tác trong tuple

- Xuất
- ~~Cập nhật, thêm~~
- ~~Xóa phần tử~~
- Giải phóng toàn bộ tuple
- Cộng và nhân

tuple

Xuất

Cách truy xuất giá trị giống như string và list

- tuple_name[index]
- tuple_name[begin:end:step]
- Minh họa

```
tup = ('h', 9, 'c', 3, 'm', 6)
```

```
tup[2]
```

```
tup[1:3]
```

tuple

Cập nhật

- Tuple là bất biến, cố định, có nghĩa là chúng ta không thể cập nhật hoặc thay đổi giá trị của các phần tử trong tuple.
- Có thể lấy một phần tử của tuple hiện tại để tạo các tuple mới.

tuple

Xóa

- Việc loại bỏ các phần tử riêng lẻ là không thể.
- Để loại bỏ rõ ràng toàn bộ tuple, chỉ cần sử dụng câu lệnh `del tên_tuple`.
- Khi sử dụng câu lệnh `del`, tuple bị giải phóng hoàn toàn (không phải tuple rỗng).

tuple

Xóa tuple

```
tuple_ = ("Happy", "New", "Year", 2018)

print("In tuple:", tuple_)
del tuple_
print("Tuple sau khi xóa hết:", tuple_)
```

Kết quả:

```
In tuple: ('Happy', 'New', 'Year', 2018)
Traceback (most recent call last):
  File "C:/PythonProjects/PyPB/test.py", line 6, in
    <module>
      print("Tuple sau khi xóa hết:", tuple_)
NameError: name 'tuple_' is not defined
```

tuple

Phép toán

- **Cộng (+) và Nhân (*) trong tuple:**

Nguyên tắc thực hiện giống như trong danh sách.

```
tup1 = (1, 3, 5)
tup2 = (2, 4, 6)

tup3 = tup1 + tup2
```

```
tup1 = (1, 3, 5)
tup2 = (2, 4, 6)

tup3 = 2 * tup1
tup4 = tup2 * 3
```

tuple

Hàm

- `len(tuple)`: Tính độ dài
- `max(tuple)`: Trả về giá trị lớn nhất
- `min(tuple)`: Trả về giá trị nhỏ nhất
- `tuple(seq)`: Chuyển một đối tượng `seq(str, list..)` thành tuple
- `sorted(iterable, key=None, reverse=False)`: Trả về một **danh sách** mới chứa các phần tử đã được sắp xếp từ tuple hoặc iterable khác.

```
tup = (1, 9, 6, 7, 5, 3)
ds = sorted(tup)

print(ds)
```

tuple

Phương thức

- `tuple.count(obj)`: Trả về số lần xuất hiện của một obj trong tuple.
- `tuple.index(obj)`: Trả về chỉ số đầu tiên của đối tượng obj trong tuple.

```
tup = (1, 9, 6, 7, 5, 6, 3, 6)  
num = tup.count(6)
```

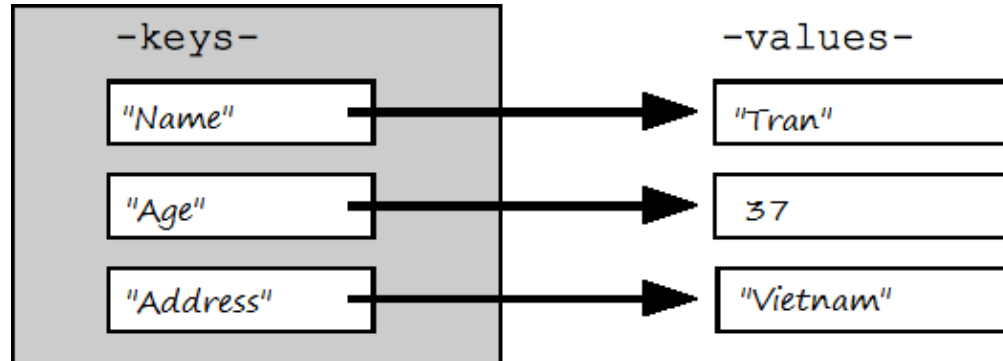
```
tup = (1, 9, 6, 7, 5, 6, 3, 6)  
i = tup.index(6)
```

dictionary (viết tắt dict)

• *dict là gì?*

- Là một kiểu dữ liệu dạng tập hợp, lưu trữ dữ liệu theo cặp key:value.
- Mỗi phần tử được cấu tạo gồm 2 thành phần **khóa (key)** và **giá trị (value)**, ngăn cách với nhau bằng dấu hai chấm “ : ”. Cụ thể:

key : **value**



key-value

Key	Value
Key phải có kiểu dữ liệu không thay đổi như chuỗi, số, hoặc kiểu tuple.	Value của một từ điển có thể thuộc bất kỳ kiểu dữ liệu nào.
Key là duy nhất (không trùng nhau) trong một từ điển.	Value giá trị có thể trùng nhau.

dict

Cú pháp

- Mỗi phần tử ngăn cách nhau bằng dấu phẩy “,” và tất cả phần tử nằm trong cặp dấu ngoặc nhọn “{}”.

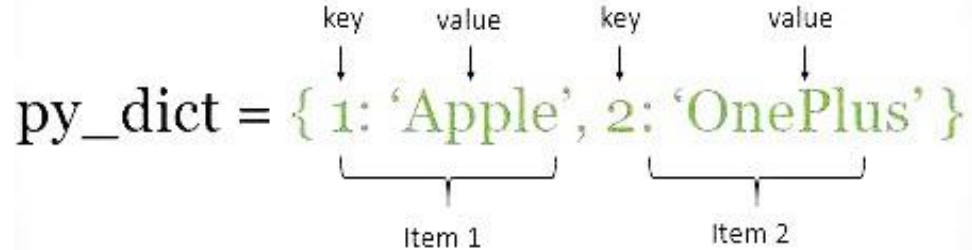
```
dict_name = {  
    key_1 : value_1,  
    key_2 : value_2,  
    key_3 : value_3,  
    ...  
    key_n: value_n  
}
```

- Từ điển rỗng là không có bất kỳ phần tử nào, được quy ước là {}.

dict

Ví dụ

py_dict = { 1: 'Apple', 2: 'OnePlus' }



The diagram illustrates the structure of a Python dictionary. It shows a dictionary named `py_dict` containing two items. Each item is a key-value pair. The first item is `1: 'Apple'`, where `1` is the key and `'Apple'` is the value. The second item is `2: 'OnePlus'`, where `2` is the key and `'OnePlus'` is the value. Arrows point from the labels 'key' and 'value' to the corresponding parts of each pair. Brackets below each pair label them as 'Item 1' and 'Item 2' respectively.

dict

Ví dụ

```
dictUni = {  
    520189: "MIT",  
    "IT": "Information Technology",  
    "SE": "Computer Engineering",  
    "CS": "Computer Science"  
}
```

Ví dụ

```
dict_tt = {  
    'TG' : 'Tiền Giang',  
    'LA' : 'Long An',  
    'ĐN' : 'Đồng Nai',  
    'BT' : 'Bình Thuận',  
    'VL' : 'Vĩnh Long'  
}
```

Các thao tác cơ bản

- Xuất giá trị
- Thêm giá trị
- Xóa
- Cập nhật giá trị
- Giải phóng toàn bộ

dict

Xuất giá trị

- Cú pháp như sau: Dict_name[key]

```
dictUni = {  
    520189: "MIT",  
    "IT": "Information Technology",  
    "SE": "Computer Engineering",  
    "CS": "Computer Science"  
}
```

```
print(dictUni[520189])  
print(dictUni["SE"])
```

dict

Thêm giá trị

- Cú pháp như sau: Dict_name['key mới'] = “giá trị mới”

```
dictUni = {  
    520189: "MIT",  
    "IT": "Information Technology",  
    "SE": "Computer Engineering",  
    "CS": "Computer Science"  
}
```

```
dictUni["DS"] = 'Data Science'  
print(dictUni)
```

Xóa phần tử

- Cú pháp

```
del Dict_name['Key_name']    # loại bỏ phần tử với key là 'Name'  
dict_name.clear()           # loại bỏ toàn bộ item trong dict => {}  
del dict_name                # giải phóng đối tượng dict, print lài thì báo lỗi.
```

Xóa phần tử

```
del dictUni["IT"]  
print(dictUni)
```

```
dictUni.clear()  
print(dictUni)
```

```
del dictUni
```

```
dictUni = {  
    520189: "MIT",  
    "IT": "Information Technology",  
    "SE": "Computer Engineering",  
    "CS": "Computer Science"  
}
```

dict

Cập nhật/sửa giá trị cho dict

Dict_name[“key cũ”] = giá_trị_mới

```
dictUni["CS"] = "Com. Sci."  
print(dictUni)
```

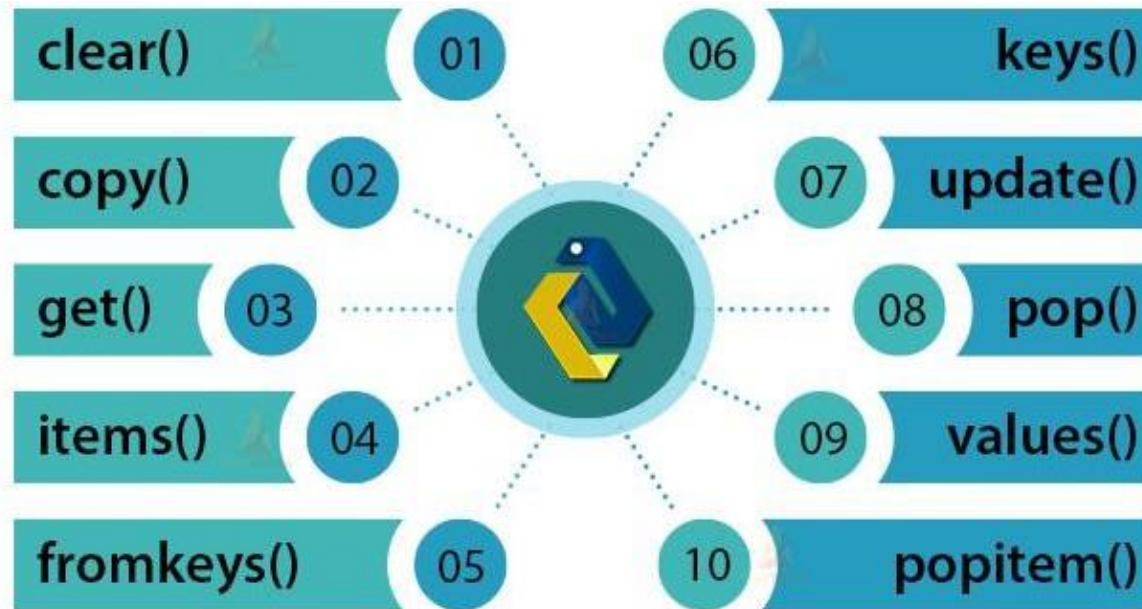

dict

Hàm

- `len(dict)`: Trả về số lượng các phần tử có trong dict.
- `str(dict)`: Chuyển dict thành chuỗi.

dict

Các phương thức xử lý dict



Phương thức

- `dict.clear()`: Loại bỏ toàn bộ phần tử trong dict.
- `dict.copy()`: Trả về 1 bản sao các giá trị của dict
- `dict.get(key, default=None)`: Trả về giá trị tương ứng với key trong dict. Nếu key không có trong dict thì trả về giá trị default (mặc định là None).
- `dict.items()`: Trả về **một danh sách** chứa tất cả các cặp (key, value) trong dict dưới dạng tuple.
- `dict.fromkeys(keys, value=None)`: là một phương thức thuộc lớp **dict**. Gọi phương thức bằng tên lớp dict. Dùng để tạo một dict mới với các key từ danh sách keys và giá trị tương ứng là value. Nếu value không được cung cấp thì các giá trị mặc định là None.

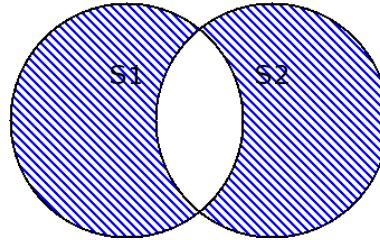
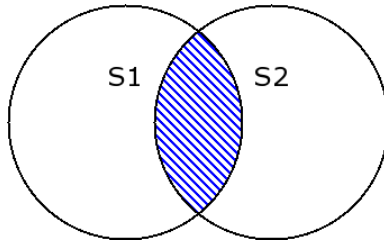
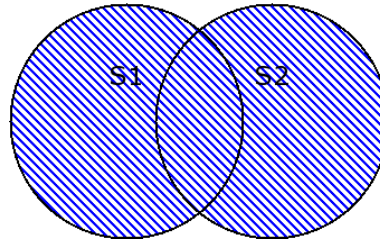
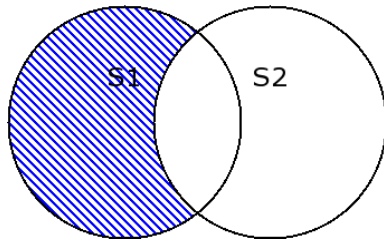
dict

Phương thức

- `dict.keys()`: Trả về danh sách tất cả các key trong dict.
- `dict.update(dict2)`: Thêm tất cả các cặp key-value từ dict2 vào dict. Nếu có các key trùng nhau, giá trị sẽ được cập nhật.
- `dict.pop(key, default=None)`: Loại bỏ một cặp key-value tương ứng với key trong dict và trả về giá trị của key. Nếu key không tồn tại thì trả về giá trị ghi trong default (mặc định là None).
- `dict.values()`: Trả về danh sách tất cả các giá trị trong dict.
- `dict.setdefault(key, default = None)`: Tương tự như `get()`, trả về giá trị tương ứng với key. Nếu key không tồn tại thì thêm key vào dict với giá trị là default.

Giới thiệu

- Kiểu dữ liệu set trong Python là kiểu tập hợp.
- Cấu trúc dữ liệu **không tuần tự**, không có thứ tự và không trùng nhau.



set

Khởi tạo

- Cú pháp:

```
set_name = {ele1, ele2..., eleN}
```

Dùng constructor của Set:

```
set_name = set((ele1, ele2..., eleN))
```

```
set_name = set([ele1, ele2..., eleN])
```

Cách tạo set rỗng:

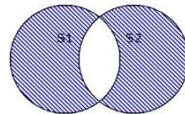
```
set_name = set()
```

Toán tử tập hợp:

- Kiểu set trong Python giống như kiểu tập hợp trong toán học, có các phép giao, hợp, hiệu.

STT	Phép toán	Python
1	Hợp	
2	Giao	&
3	Hiệu	-

- Toán tử **in** cho phép kiểm tra 1 phần tử có nằm trong 1 set hay không.
- Toán tử **^** cho phép trả về 1 tập hợp chứa tất cả các phần tử chỉ tồn tại ở một trong hai Set



set

Ví dụ

```
setA = set(["Cầu", "Sung", "Dừa", "Đủ", "Xoài"])
setB = set(["Đủ", "Sung", "Dừa", "Xoài", "Nho"])
setC = set(["Cầu", "Dừa", "Đủ", "Qua", "Môn"])
setD = set(["Cầu", "Thơm", "Dừa", "Đủ", "Xoài"])

print("Giao của hai tập hợp SetA và SetB là:\n",
      setA & setB)
print("Hợp của hai tập hợp SetC và SetD là:\n",
      setC | setD)
print("Hiệu của hai tập hợp SetA và SetD là:\n",
      setA - setD)
```

Kết quả:

```
C:\Python\python.exe C:/PythonProjects/PyPB/fc_test.py
Giao của hai tập hợp SetA và SetB là:
{'Sung', 'Đủ', 'Dừa', 'Xoài'}
Hợp của hai tập hợp SetC và SetD là:
{'Cầu', 'Dừa', 'Qua', 'Thơm', 'Xoài', 'Môn', 'Đủ'}
Hiệu của hai tập hợp SetA và SetD là:
{'Sung'}
```


Phương thức

- `add(X)`: sẽ chèn một phần tử `X` vào tập hợp.
- `copy()`: copy tập hợp
- `remove(X)`: sẽ xóa một phần tử `X` trong tập hợp.
- `clear()`: sẽ xóa sạch các phần tử trong tập hợp.
- `issubset(Set)` kiểm tra xem một tập hợp có phải là tập hợp con của một tập hợp khác hay không.
- `issuperset(Set)` kiểm tra xem một tập hợp có phải là tập hợp cha của tập hợp khác hay không.

set

Ví dụ

```
setA = set(["Cầu", "Sung", "Dừa", "Đủ", "Xoài"])
setB = set(["Đủ", "Sung", "Dừa", "Xoài", "Nho"])
setC = set(["Cầu", "Dừa", "Đủ", "Qua", "Môn"])
setD = set(["Cầu", "Thơm", "Dừa", "Đủ", "Xoài"])
setE = set(["Xoài", "Dừa", "Đủ"])

setA.add("Nho")
print("Kết quả thêm phần tử Nho vào SetA:\n", setA)

setB.remove("Sung")
print("Kết quả khi remove Sung trong SetB:\n", setB)

print("SetE có phải là con của SetD:\n",
      setE.issubset(setD))
print("SetD là cha của SetE:\n", setD.issuperset(setE))
```

```
C:\Python\python.exe C:/PythonProjects/PyPB/fc_test.py
Kết quả thêm phần tử Nho vào SetA:
{'Nho', 'Dừa', 'Xoài', 'Đủ', 'Sung', 'Cầu'}
Kết quả khi remove Sung trong SetB:
{'Nho', 'Dừa', 'Xoài', 'Đủ'}
SetE có phải là con của SetD:
True
SetD là cha của SetE:
True
```

Khởi tạo hai tập hợp có 10 phần tử là các số nguyên hai chữ số. Sau đó, áp dụng các phương thức sẵn có thực hiện yêu cầu sau:

1. Kiểm tra xem một tập hợp có phải là tập hợp **con** của một tập hợp khác hay không.
2. Kiểm tra xem một tập hợp có phải là tập hợp **cha** của tập hợp khác hay không.

Code it!

boolean

Giới thiệu

- Kiểu boolean (hoặc kiểu bool) là kiểu dữ liệu luận lý, chỉ có hai chân trị đúng (**True**) hoặc sai (**False**).

- Cú pháp khai báo:

var_name = True
Hoặc, **var_name = False**

- Cú pháp lấy chân trị của hàm Boolean:

bool(object*)

- Sử dụng bool khi nào?

boolean

Ví dụ

```
a = 8 > 9
b = 6 > 8 and 10 < 9

print("Chân trị của 8 > 9 là: ", a, "Kiểu: ", type(a))
print("Chân trị của 6>8 and 10<9 là:", b, "Kiểu:", type(b))
```

Kết quả:

```
Chân trị của 8 > 9 là: False Kiểu: <class 'bool'>
Chân trị của 6>8 and 10<9 là: False Kiểu: <class 'bool'>
```

boolean

Chân trị cơ bản thường gặp của hàm bool()

STT	Hàm bool() và đối số	Kết quả
1	<code>print(bool(print("Chuỗi bất kỳ")))</code>	False
2	<code>bool(8)</code>	True
3	<code>bool(1)</code>	True
4	<code>bool(0)</code>	False
5	<code>bool(None)</code>	False
6	<code>bool(giá-trị-rỗng)</code> giá-trị-rỗng gồm danh sách rỗng, tuple rỗng, dictionary rỗng, chuỗi rỗng, tập hợp rỗng...	False

Câu hỏi đặt ra: lý giải kết quả của việc lấy chân trị trên.

None

Giới thiệu

- **None** là một giá trị đại diện cho việc không có giá trị hoặc giá trị không xác định.
- Sử dụng để chỉ ra rằng một biến không được gán giá trị hoặc một hàm không trả về giá trị.
- Sử dụng khi tạo biến nhưng chưa biết **gán** giá trị gì.

```
def myfc():  
    return  
  
print(myfc())
```

Kết quả:

```
C:\Python\python.exe C:/PythonProjects/PyPB/fc_bool.py  
None
```

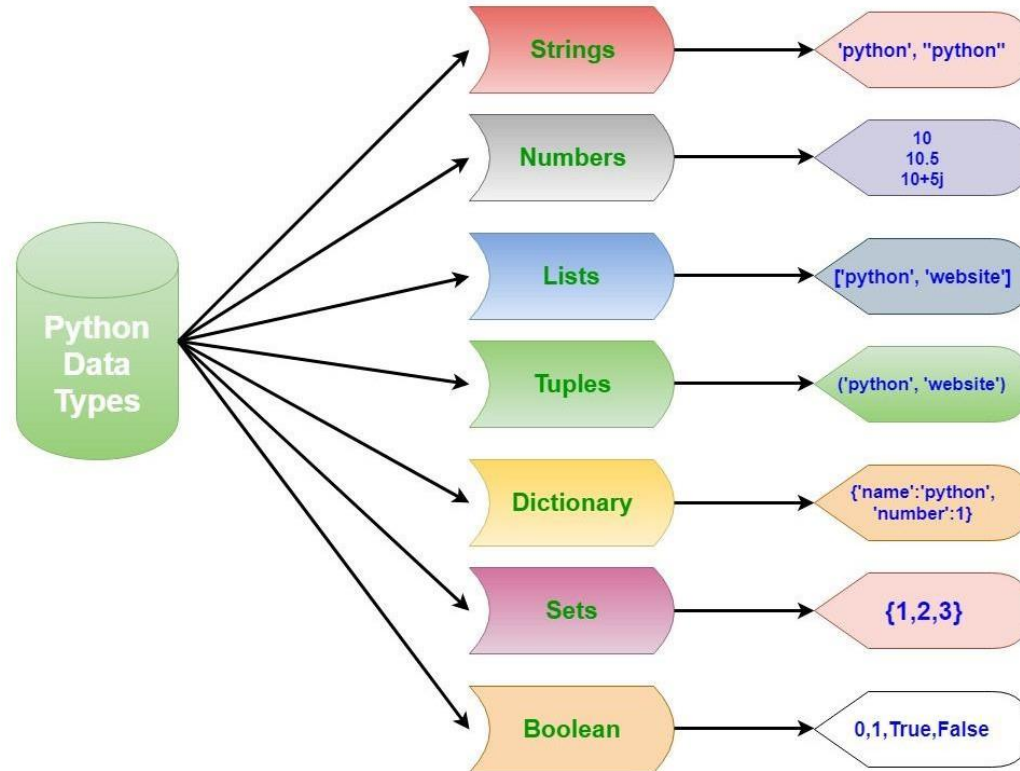

None

Ví dụ

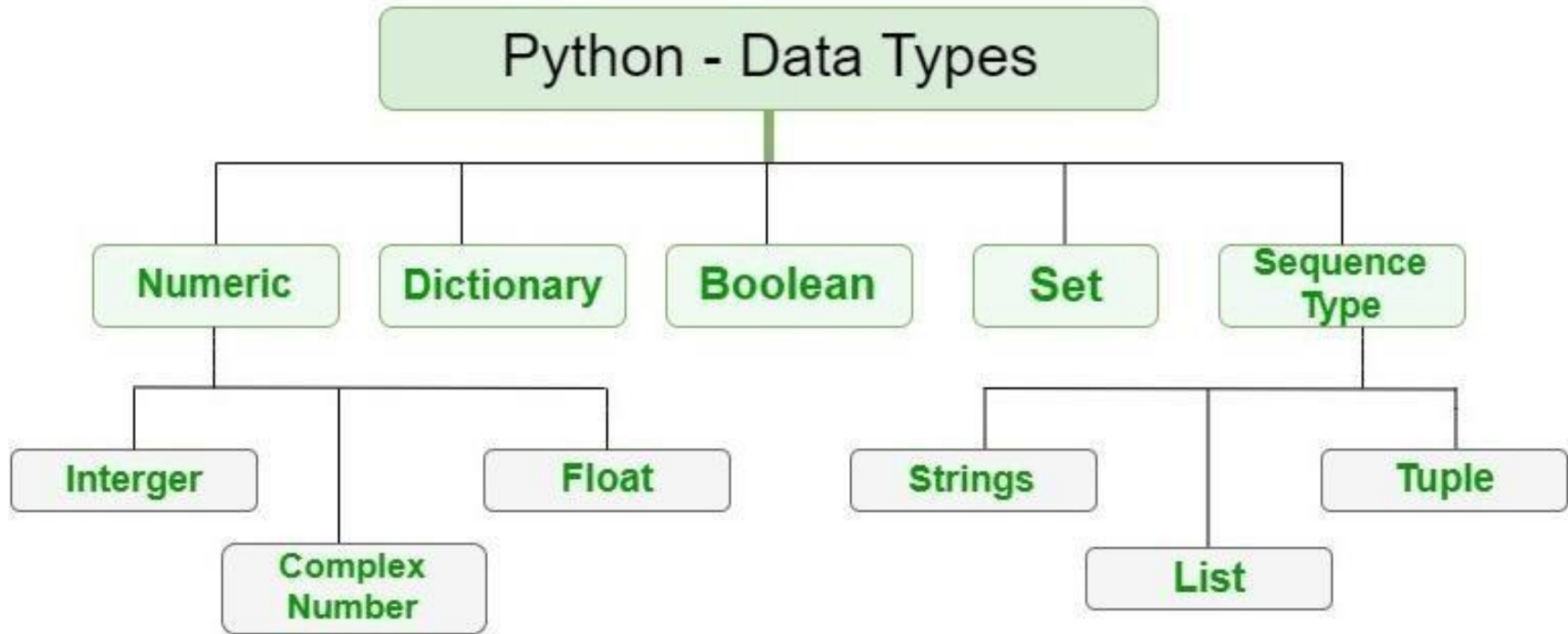
```
name = None  
print(name)  
  
name = 'Văn B'  
print(name)
```

```
value = 5  
  
if value is None:  
    print("Value có giá trị None")  
else:  
    print("Value = ", value)
```

Tóm tắt



Tóm tắt



Chú ý: Giải thích rõ cách phân chia này.

Các kiểu dữ liệu khác

Mô-đun collections

- **Counter**: Đếm số lần xuất hiện của các phần tử trong chuỗi dữ liệu và lưu trữ thành từ điển.
- **defaultdict**: Tương tự như kiểu dict, nhưng tự động tạo ra các giá trị mặc định cho các khóa không tồn tại.
- **OrderedDict**: Lưu trữ các cặp key:value theo thứ tự.
- **deque**: cấu trúc dữ liệu hàng đợi.
- **namedtuple**: Tạo ra một kiểu cấu trúc với các thuộc tính.
- **ChainMap**: Kết hợp nhiều dict lại thành một dict duy nhất.
- **UserDict**: Là một lớp cơ sở cho việc xây dựng tùy chỉnh từ điển bằng cách kế thừa từ UserDict và ghi đè các phương thức cần thiết.

Minh họa Counter

```
from collections import Counter  
  
ds = [1, 5, 8, 3, 7, 0, 6, 1, 5, 7, 1, 8, 7, 3, 8]  
counter = Counter(ds)  
  
print(counter)
```

Mô-đun collections

Minh họa deque

```
from collections import deque

de = deque([3, 6, 9, 2, 8])

de.append(10)    # Thêm cuối
de.appendleft(12) # Thêm đầu

print(de) # Output: deque([12, 3, 6, 9, 2, 8, 10])

print(de[0])    # Xem item đầu
print(de[-1])   # Xem item cuối

de.pop()        # Lấy item cuối ra khỏi queue
de.popleft()    # Lấy item đầu ra khỏi queue

print(de) # Output: deque([3, 6, 9, 2, 8])
```

1. Toán tử số học
2. Toán tử gán
3. Toán tử so sánh
4. Toán tử logic
5. Toán tử kiểm tra tồn tại in và not in
6. Độ ưu tiên

Toán tử số học (nhắc lại)

+	...
-	...
*	...
/	...
%	Chia lấy dư
//	Lấy nguyên
**	Lũy thừa

Toán tử gán

- =** Phép gán
- /=** Chia, rồi gán lại
- +=** Cộng, rồi gán lại
- =** Trừ, rồi gán lại
- *=** Nhân, rồi gán lại
- %=** Chia lấy dư, rồi
- //=** Chia lấy nguyên, rồi gán lại gán lại
- **=** Tính lũy thừa, rồi gán lại

Toán tử so sánh

- < Nhỏ hơn
- > Lớn hơn
- <= Nhỏ hơn hoặc bằng
- >= Lớn hơn hoặc bằng
- = Bằng
- != Khác (Không bằng nhau)

⇒ Kết quả sau khi thực hiện trả về kiểu **bool**

Toán tử logic

and phép **và**

or phép **hoặc**

not phép **phủ định**

⇒ Kết quả sau khi thực hiện trả về kiểu **bool**

- ***Toán tử kiểm tra tồn tại in và not in***

- Sử dụng để kiểm tra xem một giá trị có thuộc (hoặc không thuộc) một chuỗi, danh sách, tập hợp hay không.
- in
- not in

Độ ưu tiên cơ bản

- Dùng dấu (...) thể hiện độ ưu tiên nếu muốn.
- Độ ưu tiên cơ bản được thể hiện theo thứ tự sau:

SỐ HỌC > SO SÁNH > LOGIC > GÁN

Học viên tìm hiểu thêm các độ ưu tiên bên trong của SỐ HỌC, SO SÁNH, LOGIC

CÂU HỎI TỔNG HỢP

1. Phân biệt toán tử định dạng chuỗi và hàm định dạng chuỗi có sẵn trong gói thư viện chuẩn Python? Cho năm ví dụ minh họa tương ứng?
2. Viết chương trình xuất ra số ngẫu nhiên trong một đoạn bất kỳ bất cho trước?
3. Khác biệt cơ bản giữa **list** và **tuple**?
4. Ứng dụng kiểu dữ liệu tuple trong thực tế?

Hôm nay, kết thúc!

- Nghi Hoàng Khoa
- khoanh@uit.edu.vn
- inseclab.uit.edu.vn