

# **Module và Package**

## **Nội dung**

1. Dictionary (từ điển)
2. Set (tập hợp)
3. Module và Package

## Dictionary (từ điển)

- Một danh sách các từ khóa (key) và giá trị của nó (value):
  - Mỗi cặp **key-value** được xem như là **một phần tử**.
  - Xác định từ điển trong cặp **{}**
  - Các key không được trùng nhau (chỉ có các giá trị duy nhất)
  - Key: một kiểu dữ liệu không thay đổi (immutable) như chuỗi, số hoặc tuple.
  - Key và value được phân biệt riêng rẽ bởi một dấu hai chấm **(:)**
  - Các phần tử phân biệt nhau bởi một dấu phẩy **(,)**

3

## Dictionary (từ điển)

- Cú pháp khai báo từ điển

**< tên từ điển > = { Key: values }**

Ví dụ: `dic1 = { 1: 'e', 2: 'bc', 3: 'xyztk' }`

`dic = {}` # Khai báo một từ điển rỗng

`dic = { [ 1, 2, 3 ]: "abc" }` # **Lỗi** do Key là dữ liệu thay đổi kiểu

- Thêm phần tử vào từ điển

**< tên từ điển > [ Key ] = values**

- Ví dụ: `dic1[4] = 'xyz'`

Ghi chú: Không có sự khác biệt khi dùng dấu nháy đơn hay dấu nháy kép

4

## Truy cập các giá trị trong từ điển

- Truy cập các giá trị trong từ điển
  - **<tên từ điển>[Key]**
  - VD1: `print(dic1[1])`      # kết quả 'e'
  - VD2: `dic1[2]="abc"`      # Kết quả từ điển  
`dic1={1:'e',2:'abc',3:'three'}`
  - Nếu cố gắng truy cập vào phần tử không có trong từ điển thì sẽ báo lỗi
  - VD3: `print(dic1[10])`      #sẽ báo lỗi

## Dictionary (từ điển)

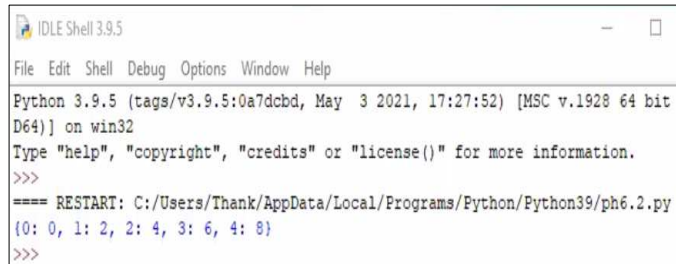
- **deld[k]**: xóa key k (và value tương ứng)
- **len(d)**: trả về độ dài của từ điển (số cặp key-value)
- **cmd (d1,d2)**: So sánh các phần tử của cả hai từ điển d
- **kind**: trả về True nếu có key k trong từ điển d
- **k not in d**: trả về True nếu không có key k trong từ điển
- **pop(k)**: trả về value tương ứng với key k và xóa cặp này đi
- **popitem()**: trả về (và xóa) một cặp (key, value) tùy ý

### Ví dụ

Tạo một từ điển có tên là dic chứa 5 phần tử mà giá trị của từng phần tử chia hết cho 2. Biết rằng, giá trị của các phần tử nằm trong khoảng từ 0 đến 10.

#### Hướng dẫn:

```
Dict = {}  
d=0  
for i in range(0,10):  
    if i%2==0:  
        Dict[d] = i  
        d = d + 1  
print(Dict)
```



```
IDLE Shell 3.9.5  
File Edit Shell Debug Options Window Help  
Python 3.9.5 (tags/v3.9.5:0a7dcdb, May 3 2021, 17:27:52) [MSC v.1928 64 bit  
D64] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
==== RESTART: C:/Users/Thank/AppData/Local/Programs/Python/Python39/ph6.2.py  
{0: 0, 1: 2, 2: 4, 3: 6, 4: 8}  
>>>
```

### Ví dụ

Tìm trong dic (ở ví dụ slide trên) các phần tử mà giá trị của chúng chia hết cho 3.

#### Hướng dẫn:

```
dem = 0  
dic1 = {}  
  
for i in range(0, 10):  
    if i % 2 == 0:  
        dic1[dem] = i  
        dem += 1  
print(dic1)  
# chia hết cho 3  
for i in range(0, dem):  
    if(dic1[i] % 3 == 0):  
        print("==> So chia het cho 3 la:", dic1[i])
```

## Dictionary (từ điển)

- **get(k)**: lấy về value tương ứng với key k
- **update(w)**: ghép các nội dung từ từ điển w vào từ điển hiện tại (nếu key trùng thì lấy value từ w)
- **items()**: trả về list các cặp (key,value)
- **keys()**: trả về các key của từ điển
- **values()**: trả về các value của từ điển
- **zip(l1, l2)**: ghép 2 danh sách thành 1 từ điển

## Ví dụ

Viết chương trình tạo 1 từ điển với key là các số tự nhiên từ 1 đến 20 còn các values là bình phương của các key tương ứng.

Hướng dẫn:

```
dic1 = {}
```

```
for i in range(1, 21):  
    dic1[i] = i * i
```

```
print(dic1)
```

### Ví dụ

- Nhập một string S, hãy tạo từ điển D trong đó key là các chữ xuất hiện trong S còn value tương ứng là số lần xuất hiện các chữ đó trong S.

- **Hướng dẫn:**

```
a = (input("Nhập chuỗi: "));  
dic = {};  
t=0;  
for x in a:  
    dic[x]=a.count(x);  
    t = t+1;  
  
print(dic);
```

11

### Set (tập hợp)

- Set=tập hợp các đối tượng (không trùng nhau)
- Khai báo trực tiếp bằng cách liệt kê các phần tử con đặt trong cặp ngoặc nhọn ({ }), ngăn cách bởi phẩy (,)

```
>>> a = {'xyz', 'abc', 'xyz', 'c'}  
>>> print(a)  
{'xyz', 'abc', 'c'} # xóa trùng nhau
```

- Tạo set bằng constructor

```
s1 = set([1, 2, 3, 4]) # {1, 2, 3, 4}  
s2 = set((1, 1, 1)) # {1}  
s3 = s1 - s2 # {2, 3, 4}  
s4 = set(range(1,50)) # {1, 2, 3,..., 48, 49}
```

12

## Khởi tạo

- Tạo set bằng set comprehension

```
# a = {'r', 'd'}  
a = {x for x in 'abracadabra' if x not in 'abc'}
```

- Set không thể chứa những đối tượng mutable (có thể bị thay đổi), mặc dù chính set lại có thể thay đổi

```
a.add("abc") # {(1, 2), "abc", (2, 3)}
```

- Frozen set giống set, nhưng không thể bị thay đổi

```
b = frozenset(((1, 2), (2, 3))) # {(1, 2), (2, 3)}  
b.add("abc") # lỗi
```

## Set (tập hợp)

```
a = set('abracadabra') # {'d', 'r', 'c', 'b', 'a'}  
b = set('alacazam') # {'z', 'c', 'm', 'l', 'a'}
```

**Phép Hiệu:** thuộc a nhưng không thuộc b

```
print(a - b) # {'r', 'd', 'b'}
```

**Phép Hợp:** thuộc a hoặc b

```
# {'a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'}  
print(a | b)
```

**Phép Giao:** thuộc cả a và b

```
print(a & b) # {'a', 'c'}
```

**Phép Xor:** thuộc hoặc a, hoặc b nhưng không phải cả 2

```
# {'r', 'd', 'b', 'm', 'z', 'l'}  
print(a ^ b)
```

### Set (tập hợp)

- **add(e)**: thêm e vào tập hợp
- **clear()**: xóa mọi phần tử trong tập hợp
- **copy()**: tạo một bản sao của tập hợp
- **difference(x)**: tương đương với phép trừ đi x
- **difference\_update(x)**: loại bỏ những phần tử trong x khỏi tập
- **discard(e)**: bỏ e khỏi tập
- **remove(e)**: bỏ e khỏi tập, báo lỗi nếu không tìm thấy e
- **union(x)**: tương đương với phép hợp với x
- **intersection(x)**: tương đương với phép giao với x

### Set (tập hợp)

- **isdisjoint(x)**: trả về True nếu tập không có phần chung nào với x
- **issubset(x)**: trả về True nếu tập là con của x, tương đương với phép so sánh  $\leq x$
- **issuperset(x)**: trả về True nếu x là tập con của tập, tương đương với phép so sánh  $\geq x$
- **pop()**: lấy một phần tử ra khỏi tập (không biết trước)
- **symmetric\_difference(x)**: tương đương với phép  $\Delta x$



## Module

- Module trong Python là các file mà có các code tương tự nhau, hay có liên quan với nhau:
  - **Khả năng tái sử dụng**: Module có thể được sử dụng ở trong phần Python code khác, do đó làm tăng tính tái sử dụng code.
  - **Khả năng phân loại**: Các kiểu thuộc tính tương tự nhau có thể được đặt trong một Module.

## Module

- Một file mã nguồn trong python được xem là một module
  - Có phần mở rộng **.py**
  - Mọi hàm, biến, kiểu trong file là các thành phần của module
- Sử dụng module:
  - Khai báo import module đó:  
**import <tên-module>**
  - Có thể khai báo **import** cùng lúc nhiều module cách nhau bởi dấu phẩy
  - Nếu muốn sử dụng các hàm, biến trong module => viết tường minh tên module đó
  - Hoặc có thể import riêng một hàm hoặc nhiều hàm:  
**from <tên-module> import fuc1, fuc2,..., fucN**

## Module

### Các Module có sẵn:

- math,
- random,
- threading,
- collections,
- os,
- mailbox,
- string, time ..

## Ví dụ

Tạo module nhập 2 số a, b và tính tổng.

### Hướng dẫn:

Bước 1: Tạo file "module.py" với code:

```
def tong(a, b):  
    return a+b
```

Bước 2: Tạo file "tinh.py" với code:

```
import module  
print(module.tong(3, 2))
```

**Lưu ý: file module.py và file tinh.py lưu cùng thư mục.**

## Package

- Package là một tập hợp các Module, sub-package, ... tương tự nhau. Đó là một cấu trúc có thứ bậc của thư mục và file.
- Để tạo một package, chỉ cần tạo ra một thư mục, với tên thư mục chính là tên của package và trong thư mục này nhất định phải có một file có tên **init.py**.
- File **init.py** giống như các constructor => được gọi ra đầu tiên khi chúng ta import package đó.

## Package

- Package = Thư mục các module (lưu trữ vật lý)

```
import numpy
A = array([1, 2, 3])    # lỗi
A = numpy.array([1, 2, 3]) # ok
import numpy as np
B = np.array([1, 2, 3]) # ok
from numpy import array
C=array([1, 2, 3])      # ok
```

- Gom, nhóm các hàm, biến, lớp xử lý cùng một chủ đề, giúp phân cấp và sử dụng dễ dàng hơn

Tạo package nhập 2 số a, b và tính tổng.

**Ví dụ**

**Hướng dẫn:**

Bước 1: Tạo thư mục có tên "**my\_package**"

Bước 2: Tạo file "**init.py**" trong thư mục **my\_package** với code:

```
def nhap():  
    a = int(input("Hay nhap a: "))  
    b = int(input("Hay nhap b: "))  
    print("==> Ket qua la:", a + b)
```

Bước 3: Tạo file **Demo.py** lưu cùng thư mục với **my-package** với code:

```
import my_package.init  
my_package.init.nhap()
```

**Hoặc:** from my\_package.init import \*  
nhap()

Lưu ý: tên package là chữ thường. \_\_\_\_\_