



# NHẬP MÔN TƯ DUY TÍNH TOÁN

Bài 7: Một số chủ đề thú vị với python

# Tóm tắt nội dung bài trước



- Ngoại lệ: tình huống bất thường không như dự kiến xảy ra khi thực hiện chương trình
- Xử lý ngoại lệ: try-except-else-finally
- Sinh ra một ngoại lệ: raise <lỗi>
- Python có các hàm hữu ích làm việc với tập tin, quá trình xử lý thường gồm 3 bước (mở file – xử lý – đóng file)
- Hai loại file cơ bản: văn bản (text) và nhị phân (binary)
- Tập tin văn bản thường được xử lý theo từng dòng
  - Python cung cấp cơ chế tự nhiên cho phép xử lý nội dung tập tin theo loại (sử dụng vòng for)
- Tập tin nhị phân có thể xem như một khối các byte
- Làm việc với tập tin có thể phát sinh ngoại lệ IOError



1. Module và Package
2. Set (tập hợp) và Frozenset (tập hợp tĩnh)
3. Dictionary (từ điển)
4. Bài tập



Phần 1

# Module và Package

- Trong python, file mã nguồn 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:
  - Có thể sử dụng các thành phần trong các module khác bằng cách import (nhập/nạp) module đó, đây là phương pháp cơ bản để tái sử dụng lại mã nguồn
  - Cú pháp: `import <tên-module>`
  - Có thể import cùng lúc nhiều module cách nhau bởi dấu phẩy
  - Nếu muốn sử dụng các hàm, biến trong module thì cần viết tường minh tên module đó
  - Có thể import riêng một hoặc nhiều hàm từ một module, cú pháp: `from <tên-module> import fuc1, fuc2,... fucN`

# Package (gói)



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

```
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
```

- Module và Package giúp quản lý tốt hơn mã nguồn
- Nhóm các hàm, biến, lớp xử lý cùng một chủ đề, giúp phân cấp và sử dụng dễ dàng hơn
- Giải quyết tranh chấp định danh của thư viện khác nhau
- Python có rất nhiều các package hỗ trợ mọi nhu cầu xử lý

- Một module rất thông dụng của python: `import math`
- Math có nhiều hằng số định nghĩa sẵn:
  - `pi`: 3.141592...
  - `e`: 2.718281...
  - `tau`: 6.283185... ( $2 * pi$ )
  - `inf`: dương vô cùng (âm vô cùng là `-math.inf`)
  - `nan`: not a number (tương đương với `float('nan')`)
- Math chứa nhiều hàm toán học:
  - `ceil(x)`: trả về số nguyên nhỏ nhất nhưng không nhỏ hơn  $x$
  - `copysign(x, y)`: copy dấu của  $y$  gán sang  $x$ 
    - Ví dụ: `copysign(1.0, -0.0)` trả về `-1`
  - `fabs(x)`: trả về trị tuyệt đối của  $x$

- Math chứa nhiều hàm toán học (tiếp...):
  - **factorial**(x): trả về  $x!$
  - **floor**(x): trả về số nguyên lớn nhất nhưng không vượt quá x
  - **gcd**(a, b): trả về ước số chung lớn nhất của a và b
  - **isinf**(x): trả về True nếu x là dương/âm vô cùng
  - **isnan**(x): trả về True nếu x là NaN (not a number)
  - **trunc**(x): trả về phần nguyên của x
  - **exp**(x): trả về  $e^x$
  - **log**(x[, y]): trả về  $\log_y x$ , mặc định  $y = e$
  - **log10**(x): trả về  $\log_{10} x$
  - **pow**(x, y): trả về  $x^y$
  - **sqrt**(x): trả về  $\sqrt{x}$



- Math cung cấp một số hàm lượng giác:
  - `degrees(x)`: chuyển  $x$  từ radians sang độ
  - `radians(x)`: chuyển  $x$  từ độ sang radians
  - `acos(x)`: trả về arc cos  $x$  (độ đo radians)
  - `asin(x)`: trả về arc sin  $x$  (độ đo radians)
  - `atan(x)`: trả về arc tang  $x$  (độ đo radians)
  - `cos(x)`: trả về cos  $x$  (độ đo radians)
  - `sin(x)`: trả về sin  $x$  (độ đo radians)
  - `tan(x)`: trả về tang  $x$  (độ đo radians)



Phần 2

# Set (tập hợp) và Frozenset (tập hợp tĩnh)

- 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

```
>>> basket = {'apple', 'orange', 'apple', 'pear'}  
>>> print(basket)  
{'orange', 'pear', 'apple'}      # 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(1,100))     # {1, 2, 3,..., 98, 99}
```

- 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 = set(([1,2], [2,3])) # lỗi
```

```
a = set((1,2), (2,3)) # {(1, 2), (2, 3)}
```

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

- Frozenset giống set, nhưng không thể bị thay đổi

```
b = frozenset((1,2), (2,3)) # {(1, 2), (2, 3)}
```

```
b.add("abc") # lỗi
```

# Các phép toán trên set

```
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)
```

# Các phương thức của set



- Một số phương thức thường hay sử dụng
  - `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  $x$

# Các phương thức của set



- Một số phương thức thường hay sử dụng
  - `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  $\wedge x$



Phần 3

# Dictionary (từ điển)



# Dictionary (từ điển)



- Từ điển là một danh sách các từ (key) và định nghĩa của nó (value)
  - Yêu cầu các key không được trùng nhau, như vậy có thể xem từ điển như một loại set
- Từ điển có thể khai báo theo cú pháp của set

```
>>> dic = {1:'one', 2:'two', 3:'three'}
```

```
>>> print(dic[1])
```

```
'one'
```

```
>>> dic[4]='four'
```

```
>>> print(dic)
```

```
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
```

# Dictionary (từ điển)



- Chú ý: chỉ những loại dữ liệu immutable (không thể thay đổi) mới có thể dùng làm key của từ điển

```
dic = { (1,2,3):"abc", 3.1415:"abc" }
```

```
dic = { [1,2,3]:"abc" } # lỗi
```

- Một số phép toán / phương thức thường dùng
  - `len(d)`: trả về độ dài của từ điển (số cặp key-value)
  - `del d[k]`: xóa key k (và value tương ứng)
  - `k in d`: trả về True nếu có key k trong từ điển
  - `k not in d`: trả về True nếu không có key k trong từ điển
  - `pop(k)`: trả về value tương ứng với k và xóa cặp này đi
  - `popitem()`: trả về (và xóa) một cặp (key, value) tùy ý

- Một số phép toán / phương thức thường dùng
  - `get(k)`: lấy về value tương ứng với key k
    - Khác phép `[]` ở chỗ `get` trả về `None` nếu `k` không phải là key
  - `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
  - `pop(k)`: trả về value tương ứng với `k` và xóa cặp này đi
  - `popitem()`: trả về (và xóa) một cặp (key, value) tùy ý

# Dictionary (từ điển)



- Dùng zip để ghép 2 list thành từ điển

```
>>> l1 = ["a", "b", "c"]
>>> l2 = [1, 2, 3]
>>> c = zip(l1, l2)
>>> for i in c:
...     print(i)
...
('a', 1)
('b', 2)
('c', 3)
```



Phần 4

# Bài tập

1. Tạo một tập hợp gồm các phần tử từ 0 đến 99, in chúng ra màn hình.
2. Tạo một tập hợp gồm các số nguyên lẻ trong khoảng từ 1 đến 199, in chúng ra màn hình.
3. Tạo một tập hợp gồm các số nhập vào từ bàn phím (nhập trên 1 dòng, cách nhau bởi ký tự trống), tìm và in ra số phần tử của tập, giá trị lớn nhất và nhỏ nhất trong tập.
4. Cho D là từ điển định nghĩa cách đọc các chữ số ở tiếng Anh, hãy in ra các value của D theo thứ tự tăng dần.
5. Nhập một từ điển D, hãy in ra các value khác nhau trong từ điển.

6. Nhập một từ điển D có các value là các số nguyên, hãy in ra màn hình 3 giá trị value lớn nhất.
7. 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
  - Ví dụ: S = “dai hoc thuy loi”  
D = { ‘d’:1, ‘a’:1, ‘i’:2, ‘ ’:3, ‘h’:2,  
‘o’:2, ‘c’:1, ‘t’:1, ‘u’:1, ‘y’:1, ‘l’:1 }