Lập trình Python

Bài 7: Kiểu từ điển, module và package, hàm lambda, map và filter

Tóm tắt nội dung bài trước

- Python có kiểu dữ liệu tập hợp (set) lấy cảm hứng từ tập hợp trong toán học. Set có hai đặc điểm chính:
  - Các dữ liệu con bên trong nó đôi một khác nhau
  - Chỉ chứa các dữ liệu loại bất biến (immutable)
- Tập hợp không có tính thứ tự, vì vậy không có phép toán chỉ mục và cắt lát, tuy vậy vẫn có thể duyệt các phần tử con trong tập hợp bằng for
- Python cung cấp nhiều phương thức, phép so sánh và phép toán hữu ích cho tập hợp
- Tập tĩnh (frozenset) là tập hợp bất biến, không thể thay đổi sau khi khởi tạo xong
- Set là kiểu mutable còn frozenset là kiểu immutable

Nội dung

1. Dictionary (từ điển)
2. Module và Package
   - Module math
3. Lambda, map và filter
4. Bài tập

Phần 1

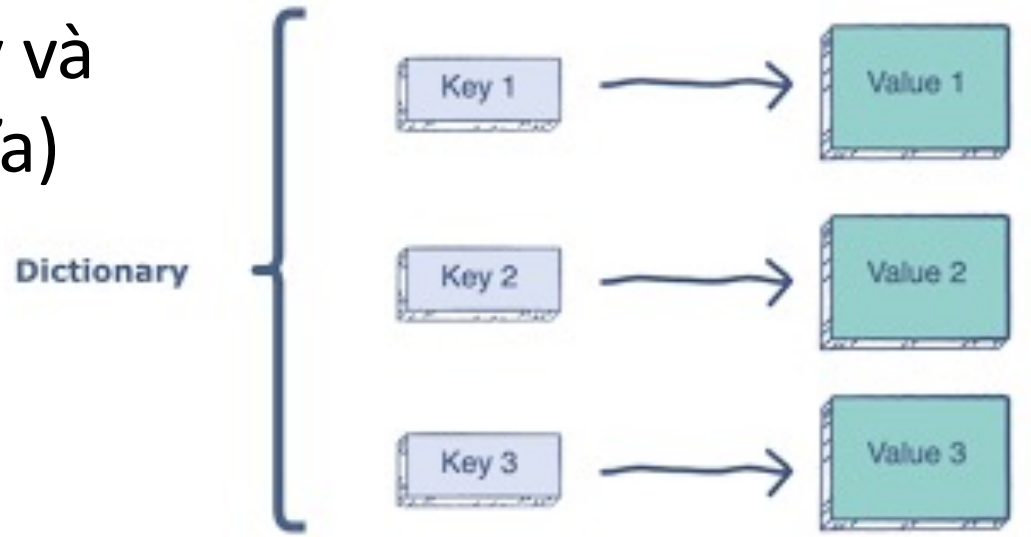
Dictionary (từ điển)

Dictionary (từ điển)

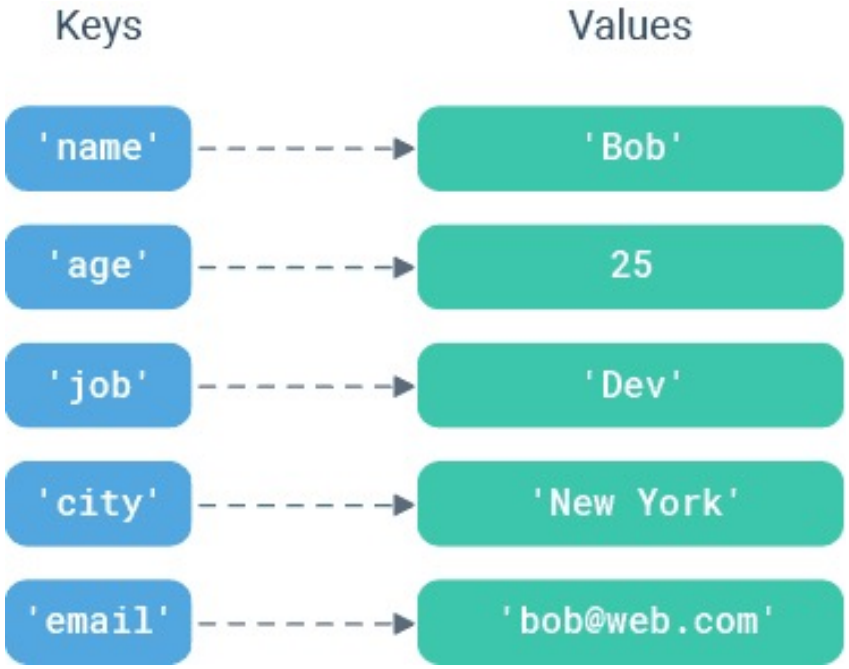
- Từ điển trong cuộc sống: các mục từ tra cứu ngữ nghĩa
  - Các mục từ được sắp thứ tự
  - Các mục từ thường khác nhau, một số từ điển cho phép các mục từ lớn chia thành nhiều mục từ con
- Dictionary trong Python lấy cảm hứng từ từ điển trong cuộc sống:
  - Một mục là một cặp (pair) khóa (key) và giá trị (value)
    - Tương đương với khái niệm mục từ và ngữ nghĩa trong từ điển thông thường
  - Các khóa (key) không được trùng nhau, như vậy có thể xem từ điển như một loại set
  - Các khóa không sắp thứ tự như từ điển thông thường
  - Chỉ dữ liệu bất biến (immutable) mới được dùng làm khóa

Dictionary (từ điển)

- Từ điển là ánh xạ giữa key và value (giữa từ và giải nghĩa)



- Ví dụ:



Dictionary (từ điển)

- Một cặp trong từ điển viết ở dạng `<key> : <value>`
- Từ điển có thể khai báo trực tiếp dùng cú pháp của set

```
d1 = { }                          # từ điển rỗng
d2 = { 1: 'one', 2: 'two' }       # từ điển các cặp số-chuỗi
d3 = { 'one': 1, 'two': 2 }       # từ điển các cặp chuỗi-số
d4 = { 'tên': 'nam', 'sđt': 0}    # từ điển hỗ hợp
```

- Như vậy Python coi từ điển là dạng mở rộng của tập hợp
- Trường hợp lấy dữ liệu từ nguồn khác, cách thích hợp nhất là sử dụng hàm khởi tạo dict()

```
d5 = dict(d4)          # lấy dữ liệu từ d4
print(d5)              # {'tên': 'nam', 'sđt': 0}
d6 = dict()            # tạo từ điển rỗng
print(d6)              # {}
```

Dictionary (từ điển)

- Python cũng cho phép tạo từ điển bằng bộ suy diễn từ điển, với cú pháp tương tự như bộ suy diễn danh sách
- Ví dụ: tạo từ điển gồm các bộ khóa là số tự nhiên nhỏ hơn N và giá trị tương ứng là lập phương của nó

```
n = int(input('N = '))
d = { i: i * i * i for i in range(n) }
print(d)
```

- Ví dụ: tạo từ điển có khóa là các ký tự xuất hiện trong từ S và giá trị là số lần xuất hiện của ký tự đó trong S

```
S = input('S = ')
d = { w : S.count(w) for w in S }
print(d)
```

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

Dictionary (từ điển)

- Một số phép toán / phương thức thường dùng
  - get(k): lấy về value tương ứng với key k
    - Khác phép [k] 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)
```

Truy xuất dữ liệu theo khóa

- Từ điển cho phép lấy giá trị tương ứng với khóa k bằng phương thức get(k), nhưng có thể dùng chỉ mục (là các key):

```
d = { 1: 'one', 2: 'two', 3: 'three' }
print('d.get(3) = ', d.get(3))          # d.get(3) =  three
print('d[3] = ', d[3])                  # d[3] =  three
print('d.get(9) = ', d.get(9))          # d.get(9) =  None
print('d[9] = ', d[9])                  # KeyError
d[3] = 'ba'       # cập nhật cặp 3: 'three' thành 3: 'ba'
d[4] = 'bốn'      # thêm mới cặp 4: 'bốn'
print(d)          # {1: 'one', 2: 'two', 3: 'ba', 4: 'bốn'}
```

- Chú ý: hai cách này không tương đương. Khi dữ liệu không có trong từ điển, get(k) trả về None còn chỉ mục phát sinh ngoại lệ KeyError

Xóa dữ liệu trong từ điển

- Python có nhiều cách để xóa dữ liệu trong từ điển hoặc lấy dữ liệu ra khỏi từ điển
- Các phương thức pop, popitem, clear được thiết kế cho những tình huống lập trình khác nhau

```
d = { 1: 'one', 3: 'three', 2: 'two', 0: 'zero' }
del d[1]                    # xóa bỏ mục 1: 'one'
print(d)                    # {3: 'three', 2: 'two', 0: 'zero'}
print(d.pop(2))             # 'two'
print(d.popitem())          # (0, 'zero')
d.clear()                   # xóa rỗng từ điển
print(d)                    # {}
```

Kiểm tra dữ liệu trong từ điển

- Phép kiểm tra dữ liệu trong từ điển **chỉ làm việc với khóa**

```
dic = { 1: 'one', 3: 'three', 2: 'two', 0: 'zero' }
# kiểm tra có 2 trong từ điển không? True
print(2 in dic)
# kiểm tra không có 'one' trong từ điển phải không? True
print('one' not in dic)
# kiểm tra cặp (3, 'three') có trong từ điển không? False
print((3, 'three') in dic)
```

Duyệt dữ liệu trong từ điển

- Python cung cấp nhiều phương thức lấy về các nhóm dữ liệu trong từ điển, ta có thể dễ dàng duyệt chúng với vòng lặp for

```
d = { 1: 'one', 3: 'three', 2: 'two', 0: 'zero' }
# duyệt từ điển theo khóa: 1, 3, 2, 0
for i in d: print(i)
# duyệt từ điển theo khóa: 1, 3, 2, 0
for i in d.keys(): print(i)
# duyệt theo cặp khóa-giá trị: (1, 'one'), (3, 'three'),..
for i in d.items(): print(i)
# duyệt theo giá trị: 'one', 'three', 'two', 'zero'
for i in d.values(): print(i)
```

Phần 2

Module và Package

Module (khối)

- 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

Module math

- 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

Module math

- 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 (*)
  - lcm(a, b): trả về bội số chung nhỏ 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 $\sqrt[2]{x}$

Module math

- Math cung cấp một số hàm lượng giác:
  - degrees(x): chuyển x từ radians sang độ
    - `math.degrees(math.pi/2)#90.0`
  - radians(x): chuyển x từ độ sang radians
  - cos(x): trả về cos x (độ đo radians)
    - `math.cos(math.pi/2) #6.123233995736766e-17`
  - sin(x): trả về sin x (độ đo radians)
    - `math.sin(math.pi/2) #1.0`
  - tan(x): trả về tang x (độ đo radians)
  - acos(x): trả về arc cos x (độ đo radians)
    - `math.acos(0.0) #1.5707963267948966 (math.pi/2)`
  - asin(x): trả về arc sin x (độ đo radians)
    - `math.asin(1.0) #1.5707963267948966 (math.pi/2)`
  - atan(x): trả về arc tang x (độ đo radians)

Phần 3

Lambda, map và filter

Biểu thức lambda

- Một dạng viết ngắn của hàm, tất nhiên có nhiều hạn chế
- Cú pháp:

```
lambda <tham số>: <biểu thức thay thế>
```

- Ví dụ:

```
lambda x, y: x + y
```

- Tương đương với:

```
def tong(x, y):
    return x + y
```

- Chú ý:
  - Phần thân của lambda chỉ là một biểu thức, không thể viết nhiều dòng hoặc quá nhiều lệnh
  - Đôi khi gọi là hàm vô danh (anonymous function, không hẳn)

Biểu thức lambda

- Biểu thức lambda có thể gán cho một biến và sử dụng như một hàm
- Ví dụ:

```
cộng = lambda x, y: x + y    # biến cộng
print(cộng(10, 20))          # in ra 30
```

- Một chút "loạn" trong thuật ngữ:
  - Cách viết các hàm (function, def,…) được gọi là "lập trình thủ tục" (procedural programming)
  - Lối viết biểu thức lambda (và một số dạng khác) được gọi là "lập trình hàm" (functional programming)
- Lập trình hàm là trào lưu chủ đạo hiện nay, do tính ngắn gọn và mạnh mẽ

Phép ánh xạ (map)

- Hàm "map" cho phép ánh xạ từ kiểu tuần tự sang một danh sách thông qua một hàm ánh xạ nào đó
- Cú pháp:

```
map(hàm-ánh-xạ, kiểu-tuần-tự)
```

- Ví dụ:

```
def binhphuong(x):
    return x * x

L = [1, 2, 3, 4]
P = list(map(binhphuong, L))
print(P) # [1, 4, 9, 16]
```

- Cách thực hiện: thực hiện hàm `binhphuong` với từng phần tử của `L`

Phép lọc (filter)

- Hàm "filter" cho phép lọc các phần tử trong một kiểu tuần tự (trả về danh sách các phần tử thỏa được)
- Cú pháp:

```
filter(hàm-lọc, kiểu-tuần-tự)
```

- Ví dụ:

```
def sole(x):
    return x % 2 == 1

L = [1, 2, 3, 4]
P = list(filter(sole, L))
print(P)
```

- Cách thực hiện: trả về danh sách các phần tử thỏa mãn điều kiện do hàm `sole` đánh giá

Lambda, map và filter

- Biểu thức lambda giúp tăng sức mạnh của map và filter
  - Ngoài ra còn các hàm khác như sort, reduce, accumulate,…
- Ví dụ với map:

```
def binhphuong(x):
    return x * x

L = [1, 2, 3, 4]
P = list(map(binhphuong, L))
print(P)
```

- Dùng biểu thức lambda:

```
L = [1, 2, 3, 4]
P = list(map(lambda x: x * x, L))
print(P)
```

Lambda, map và filter

- Ví dụ với filter:

```
def sole(x):
    return x % 2 == 1

L = [1, 2, 3, 4]
P = list(filter(sole, L))
print(P)
```

- Dùng biểu thức lambda:

```
L = [1, 2, 3, 4]
P = list(map(lambda x: x % 2 == 1, L))
print(P)
```

Lambda, map và filter

- Biểu thức lambda đôi khi còn "ảo" hơn nữa

```
from math import pi, sin, cos, tan

L = (sin, cos, tan)
print(list(map(lambda x: x(pi), L)))
```

- Sử dụng với sắp xếp:

```
L = [1, 4, 3, 5, 6, 7, 2, 3, 5, 4, 2, 6]
L.sort(key = lambda x: (x % 2, x))
print(L)
```

- Đi xa hơn nữa:

```
L = [1, 4, 3, 5, 6, 7, 2, 3, 5, 4, 2, 6]
L.sort(key = lambda x: (x % 2, x if x % 2 else -x))
print(L)
```

Phần 4

Bài tập

Bài tập

1. 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.
2. Nhập một từ điển D, hãy in ra các value khác nhau trong từ điển.
3. Nhập một từ điển D có các value là các số nguyên, hãy in ra màn hình 3 giá trị value lớn nhất.
4. 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 }

Bài tập

5. Nhập từ điển prices lưu trữ giá của các loại trái cây và từ điển stock lưu trữ số lượng tồn của từng loại. Sau đó hãy in ra thứ tự các loại trái cây còn trong cửa hàng giảm dần theo tổng giá trị của từng loại.
   Ví dụ:

   Dữ liệu nhập vào cho ra từ điển như sau:
   - prices = { "banana": 4, "apple": 2, "orange": 1.5, "pear": 3 }
   - stock = { "banana": 6, "orange": 32, "pear": 15 }

   Kết quả in ra thứ tự:

orange	48
pear	45
banana	24
apple	0

Bài tập

6. Tạo ra một từ điển lưu lượng mưa trung bình trong các tháng từ năm 2002 đến 2021. Quy cách như sau:
   - Từ điển có 12 mục, khóa của mỗi mục là một tháng
   - Giá trị ứng với khóa là danh sách 20 số đại diện cho 20 năm
   - Lượng mưa là số thực ngẫu nhiên từ 100 đến 4000
7. Nhập 2 từ điển A có N cặp (key, value) và B có M cặp (key, value). Từ A và B hãy tạo từ điển C theo quy tắc sau:
   - Một mục trong C thì key phải xuất hiện trong A hoặc B
   - Nếu key chỉ xuất hiện trong A hoặc trong B thì value là value tương ứng trong A (hoặc B)
   - Nếu key xuất hiện trong cả A và B thì value là max của value tương ứng trong A và B