LOGO

LẬP TRÌNH CHO KHOA HỌC DỮ LIỆU

Bài 6. Xử lý dữ liệu trong Python

Nội dung

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

Dictionary (từ điển)

- Từ điển là 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 { }
  - Yêu cầu các key không được trùng nhau (chỉ có các giá trị duy nhất)
  - Key phải là 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 phảy (,)

Dictionary (từ điển)

- Cú pháp khai báo từ điển
  - <tên từ điển>={Key:values}
  - VD1: dic1={1:'one',2:'two',3:'three'}
  - VD2: dic2={} # Khai báo một từ điển rỗng
  - VD3: dic3={[1,2,3]:"abc"} # lỗi do Key là kiểu dữ liệu thay đổi
  - Thêm phần tử vào từ điển
  - <tên từ điển>[Key]=values
  - VD4: dic1[4]= 'four'

<te
<ten_dictionary>[key]

- Truy cập các giá trị trong từ điển
- <tên từ điển>[Key]
- VD1: print(dic1[1]) # kết quả 'one'
- VD2: dic1[2]="abc" # Kết quả từ điển

dic1={1:'one',2:'abc',3:'three'}

- Nếu cố gắng truy cập vào phần tử không có trong từ điển thì sẽ báo lỗi
- VD3: print(dic1[4]) #sẽ báo lỗi

Dictionary (từ điển)

- 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)
  - cmd (d1,d2): So sánh các phần tử của cả hai từ điển d
  - k in d: 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

Dictionary (từ điển)

- 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
  - zip(l1,l2): ghép 2 danh sách thành 1 từ điển

Dictionary (từ điển)

- VD1: Viết chương trình tạo 1 từ điển với key là các số từ nhiên từ 1 đến 20 còn các values là bình phương của các key tương ứng.
- VD2: Viết chương trình nhập 1 từ điển từ bàn phím
- VD3: 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

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

```
>>> 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}
```

Set (tập hợp)

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)}
```

- Frozenset 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)

Phép toán

```
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)

Phương thức

- 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

Set (tập hợp)

Phương thức

- 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 <=x
  - issuperset(x): trả về True nếu x là tập con của tập, tương đương với phép so sánh >=x
  - pop(): lấy một phần tử ra khỏi tập (không biết trước)
  - symmetric_difference(x): tương đương với phép ^x

Module và Package

Module

- Module được sử dụng để tổ chức Python code một cách logic để giúp bạn dễ dàng hiểu và sử dụng code đó hơn.
- Nếu nội dung của một quyển sách không được lập chỉ mục hoặc phân loại thành các chương riêng, thì quyển sách này có thể trở nên nhàm chán và gây khó khăn cho độc giả khi đọc và hiểu nó. Tương tự, Module trong Python là các file mà có các code tương tự nhau, hay có liên quan với nhau. Chúng có lợi thế sau:
  - 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 và Package

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 thì cần viết tường minh tên module đó
  - Hoặc có thể import riêng một hàm hoặc nhiều hàm, cú pháp: `from <tên-module> import fuc1, fuc2,..., fucN`

Module

- **Các Module dựng sẵn:**
  - math, random, threading, collections, os, mailbox, string, time ..
  - Mỗi Module này đã định nghĩa sẵn rất nhiều hàm để có thể sử dụng thực hiện các tính năng khác nhau

```
import math
a=4.6
print math.ceil(a)
print math.floor(a)
b=9
print math.sqrt(b)
print math.exp(3.0)
print math.log(2.0)
print math.pow(2.0,3.0)
print math.sin(0)
print math.cos(0)
print math.tan(45)
```

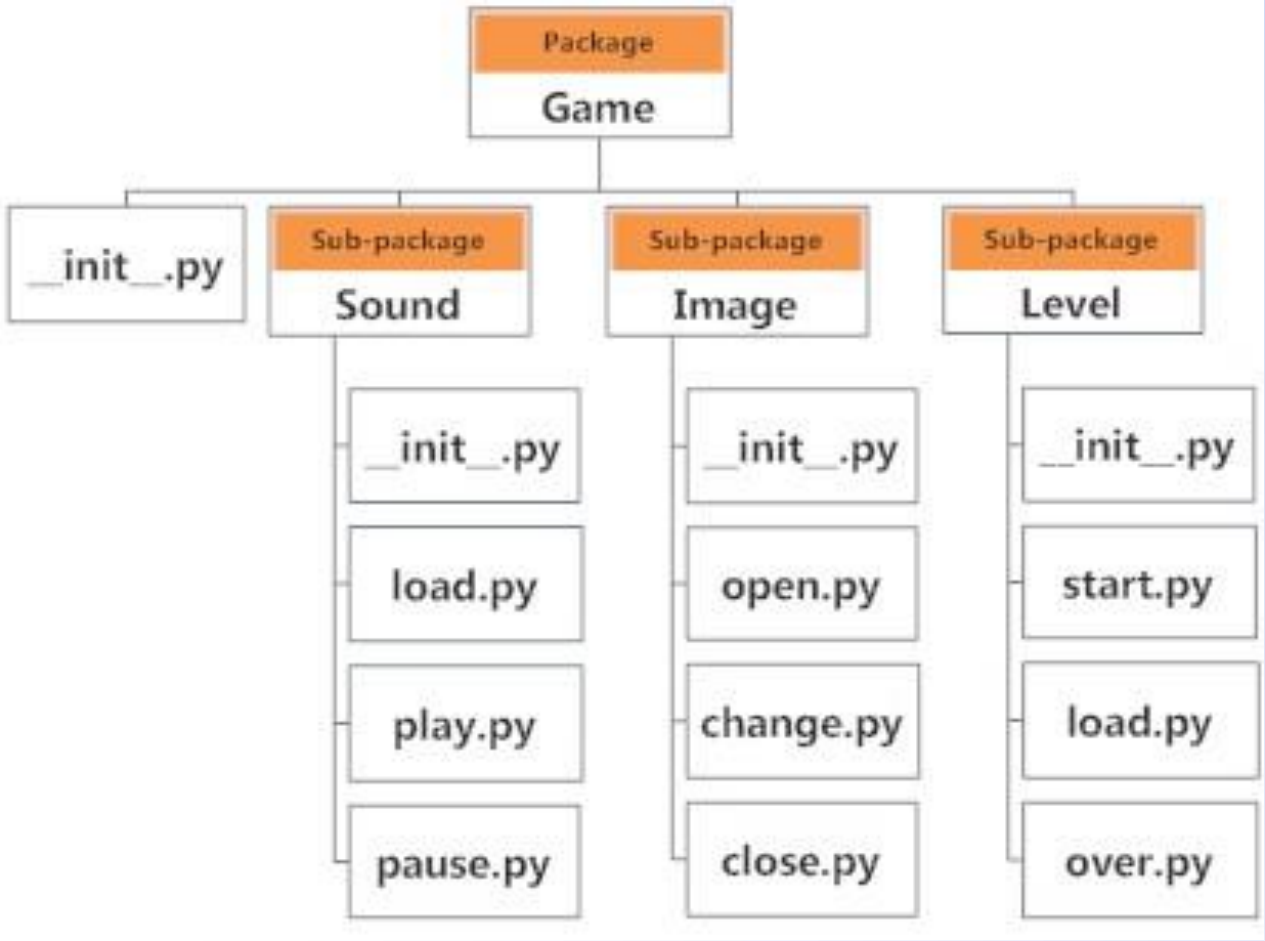
Module và Package

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.
- Build package trong Python: Để tạo ra một package trong python thì mọi người 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 này nó giống như các constructor, và nó sẽ được gọi ra đầu tiên khi chúng ta import package đó.

Module và Package

Package



Module và Package

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
```

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


LOGO
CẢM ƠN!