nsd1904_py02_day03

《python基础教程 第二版》https://down.51cto.com/data/200004

os.walk()

- os.walk返回值由多个元组构成
- 每个元组由三项构成：(字符串， 列表， 列表)
  - 元组中的第一项，字符串，是路径
  - 元组中的第二项，列表，是路径下的所有目录
  - 元组中的第三项，列表，是路径下的所有文件
- 只要将路径和路径下的文件进行拼接，即可得到文件的路径

```
>>> list(os.walk('/etc/security'))
[
    (
        '/etc/security',
        ['console.apps', 'console.perms.d', 'limits.d', 'namespace.d'],
        ['access.conf', 'chroot.conf', 'console.handlers', 'console.perms', 'group.conf',
'limits.conf', 'namespace.conf', 'namespace.init', 'opasswd', 'pam_env.conf',
'sepermit.conf', 'time.conf', 'pwquality.conf']
    ),
    (
        '/etc/security/console.apps',
        [],
        ['config-util', 'xserver', 'liveinst', 'setup']
    ),
    ('/etc/security/console.perms.d', [], []),
    ('/etc/security/limits.d', [], ['20-nproc.conf']),
    ('/etc/security/namespace.d', [], [])
]
>>> for path, folders, files in os.walk('/etc/security'):
...   for file in files:
...     os.path.join(path, file)
...
```

OOP面向对象的编程

- OOP将现实世界的实体抽象成类class
- 将数据属性和函数属性整合在一起
- class：类，用来描述具有相同的属性和方法的对象的集合
- 实例，对象：通过类定义的数据结构实例。
- 方法：定义在类中的函数。

- 组合：两个类明显不同，其中一个类是另一个类的组件，使用组合。
- 继承：当两个类有很多一样的地方，但是又有一些不同，使用继承
- 多重继承：子类可以有多个父类。
    - 子类将拥有所有父类的方法
    - 如果有重名方法，查找的顺序是自下向上，自左向右
- 魔法方法magic
    - 以双下划线开头和结尾的方法

re模块

正则表达式，把MAC地址加上冒号

思路：

- 找到mac地址：出现在行尾的12个任意字符
- mac地址每两个数分一组，共6组
- 各组之间加冒号

```
192.168.1.1     00000C291234
192.168.1.2     525400A3B231
192.168.1.3     09283A3B328F

:%s/\(..\)\(..\)\(..\)\(..\)\(..\)\(..\)$/\1:\2:\3:\4:\5:\6/
```

re模块的方法

```
>>> import re
>>> re.match('f..', 'food')  # 在food中匹配f..，匹配到返回匹配对象
<_sre.SRE_Match object; span=(0, 3), match='foo'>
>>> print(re.match('f..', 'seafood'))   # 匹配不到返加None
None

>>> m = re.search('f..', 'seafood')  # 在字符串中匹配模式
>>> m.group()   # 返回匹配到的字符串
'foo'
>>> re.findall('f..', 'seafood is food')
['foo', 'foo']
# finditer得到的是由多个匹配对象构成的迭代器
>>> for m in re.finditer('f..', 'seafood is food'):
...   m.group()
...
'foo'
'foo'

>>> s1 = 'hello-world-china.com.cn'
>>> re.split('\.|-', s1)  # 以.或-作为分隔符拆分字符串
['hello', 'world', 'china', 'com', 'cn']
```

```
>>> re.sub('X', 'Niu', 'Hello X. ni hao X')  # 把X替换成Niu
'Hello Niu. ni hao Niu'

# 如果有大量内容需要匹配，把模式先进行编译将会得到更好的效率
>>> patt = re.compile('f..')
>>> m = patt.search('seafood')
>>> m.group()
'foo'
>>> patt.findall('seafood is food')
['foo', 'foo']
```

字典排序

- 字典本身没有顺序
- 将字典转成列表再排序

```
>>> result = {'172.40.58.150': 10, '172.40.58.124': 6, '172.40.58.101': 10, '127.0.0.1':
121, '192.168.4.254': 103, '192.168.2.254': 110, '201.1.1.254': 173, '201.1.2.254': 119,
'172.40.0.54': 391, '172.40.50.116': 244}
>>> ip_list = list(result.items())
>>> ip_list   # 列表由元组构成。需要根据元组的第二项进行排序
[('172.40.58.150', 10), ('172.40.58.124', 6), ('172.40.58.101', 10), ('127.0.0.1', 121),
('192.168.4.254', 103), ('192.168.2.254', 110), ('201.1.1.254', 173), ('201.1.2.254', 119),
('172.40.0.54', 391), ('172.40.50.116', 244)]
# 列表的sort方法，可以指定key。key是一个函数，该函数处理列表中的每一项，将处理结果作为排序依据
>>> def get_item(seq):
...   return seq[-1]
...
>>> ip_list.sort(key=get_item)
>>> ip_list
[('172.40.58.124', 6), ('172.40.58.150', 10), ('172.40.58.101', 10), ('192.168.4.254',
103), ('192.168.2.254', 110), ('201.1.2.254', 119), ('127.0.0.1', 121), ('201.1.1.254',
173), ('172.40.50.116', 244), ('172.40.0.54', 391)]
# 使用匿名函数，并降序排列
>>> ip_list.sort(key=lambda seq: seq[-1], reverse=True)
>>> ip_list
[('172.40.0.54', 391), ('172.40.50.116', 244), ('201.1.1.254', 173), ('127.0.0.1', 121),
('201.1.2.254', 119), ('192.168.2.254', 110), ('192.168.4.254', 103), ('172.40.58.150',
10), ('172.40.58.101', 10), ('172.40.58.124', 6)]
```