nsd1909-py02-day03

hashlib模块

- 用于计算数据的哈希值。
- 哈希即hash的音译，它是一个单向加密的算法
  - 给定相同的数据，一定可以得到相同的乱码
  - 不能通过乱码反向推出原始数据
  - 用于存储加密的密码，也可以用于计算文件的完整性

```
# 计算md5值
>>> import hashlib
>>> m = hashlib.md5(b'123456')
>>> m.hexdigest()   # 123456的哈希值
'e10adc3949ba59abbe56e057f20f883e'

# 数据量很大的时候，可以采用分批计算
>>> m1 = hashlib.md5()
>>> m1.update(b'12')
>>> m1.update(b'34')
>>> m1.update(b'56')
>>> m1.hexdigest()
'e10adc3949ba59abbe56e057f20f883e'
```

tarfile模块

- 实现tar包功能、压缩解压缩

```
>>> import tarfile
# 压缩
>>> tar = tarfile.open('/var/tmp/mytest.tar.gz', 'w:gz')
>>> tar.add('/etc/security')
>>> tar.add('/etc/hosts')
>>> tar.close()

# 解压缩
>>> tar = tarfile.open('/var/tmp/mytest.tar.gz')
>>> tar.extractall(path='/var/tmp')  # 不指定解压位置，将会解到当前目录
>>> tar.close()
```

OOP

- OOP：面向对象的编程

```
>>> class Bear:      # 定义类，class是关键字
...   pass
...
>>> b1 = Bear()      # 创建实例
>>> b1.name = '熊大'
>>> b1.size = 'L'
>>> b1.name
'熊大'
>>> b1.size
```

组合

- 两个类明显不同，一个类是另一个类的组件

子类

- 类可以派生出子类，也可以说是子类继承于父类

多重继承

- 子类可以有多个父类
- 子类的实例拥有所有类的方法
- 实例查找方法时，顺序是自下向上，自左向右

特殊方法

- 在class中，为了实现其内部功能，它们拥有很多以双下划线开头和结尾的特殊方法

```
# dir可以查看对象属性
>>> type(10)   # 10是int的实例
<class 'int'>
>>> dir(10)    # 查看10的属性
>>> dir([])    # 查看列表实例的属性
```

- 这些特殊方法被称作magic魔法方法。一般不用用户管理。

正则表达式

- 例：给mac地址加冒号。
- 思路：
  - 找到mac地址
  - mac地址每两个数字分一组
  - 组之间加冒号

```
192.168.1.1     000C29123456
192.168.1.2     525400A3B8C1

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

re模块

```
>>> import re
# match只能从字符串开头匹配
>>> 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'>
>>> re.match('f..', 'seafood')     # 匹配不到，返回 None
>>> print(re.match('f..', 'seafood'))
None
# search应用更多，可以在字符串任意位置匹配
>>> m = re.search('f..', 'seafood')
>>> m.group()   # 通过group方法取出匹配内容
'foo'

# search只匹配到第一个满足条件的字符串， findall匹配到全部，返回列表
>>> re.findall('f..', 'seafood is food')
['foo', 'foo']
# finditer返回所有匹配对象的生成器
>>> re.finditer('f..', 'seafood is food')
<callable_iterator object at 0x7f5764824828>
>>> list(re.finditer('f..', 'seafood is food'))
[<_sre.SRE_Match object; span=(3, 6), match='foo'>, <_sre.SRE_Match object; span=(11,
14), match='foo'>]
>>> for m in re.finditer('f..', 'seafood is food'):
...   m.group()
...
'foo'
'foo'

# 字符串的切割方法，只能指定一个分隔符
>>> s = 'hello-world-ni-hao.tar.gz'
>>> s.split('-')
['hello', 'world', 'ni', 'hao.tar.gz']
>>> s.split('.')
['hello-world-ni-hao', 'tar', 'gz']
# 正则的切割符，可以指定多个
>>> re.split('-|\.', s)
['hello', 'world', 'ni', 'hao', 'tar', 'gz']

# 字符串的替换方法
>>> s.replace('-', '/')
'hello/world/ni/hao.tar.gz'

# 正则表达式的替换
>>> re.sub('-|\.', '/', s)
'hello/world/ni/hao/tar/gz'

# 当有大量匹配时，为了有更好的效率，可以将正则的模式先进行编译
>>> patt = re.compile('f..')
>>> m = patt.search('seafood')
>>> m.group()
'foo'
>>> patt.findall('seafood is food')
['foo', 'foo']
```