nsd1903_py02_day03

备份程序

- 备份的源文件是/tmp/demo/security目录
- 要求可以实现完全备份和增量备份

OOP：面向对象的编程

- 在python中，一切皆对象
- 对象有属性：数据属性（变量），函数属性（方法）
- OOP编程时，需要把某些事物找出它们的共性，抽象成一个类class
- 在具体应用时，再根据类创建实例
- 类名建议使用驼峰的形式，如MyClass

```
class GameRole:
    def __init__(self, nm, wp):
        self.name = nm
        self.weapon = wp
if __name__ == '__main__':
    lb = GameRole('吕布', '方天画戟')  # 创建名为lb的实例
    print(lb.name, lb.weapon)
```

- __init__是类中的特殊方法，称作构造器方法，它在创建实例的时候自动调用
- 方法的第一个参数self，表示实例，不是关键字，可以是任意的合法名称
- 方法中的变量名，就是函数的局部变量，只在该方法中可用
- 绑定到实例上的变量，是实例的属性，可以在类中任意位置应用

组合

- 两个类明显不同
- 一个类是另一个类的组件

继承

- 两个类有很多相同
- 一个类是另一个类的子类
- 子类继承父类的属性
- 父子拥有同名方法，子类优先级高
- 子类可以有多个父类，继承所有父类的方法
- 执行方法时，查找的顺序是自下向上，自左向右

特殊方法

类中有一些以双下划线开头和结尾的特殊方法，也称作magic魔法方法。

re模块

https://jex.im/regulex

```
# 将mac地址加上冒号
192.168.1.1     000C29123456
192.168.1.2     525400A31B2C
192.168.1.3     0002231A08D3

# 思路：找到mac地址、每两个数分一组、组之间加冒号
:%s/\(..\)\(..\)\(..\)\(..\)\(..\)\(..\)$/\1:\2:\3:\4:\5:\6/
```

re模块常用方法

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

# search在字符串中匹配
>>> re.search('f..', 'food')
<_sre.SRE_Match object; span=(0, 3), match='foo'>
>>> re.search('f..', 'seafood')
<_sre.SRE_Match object; span=(3, 6), match='foo'>
>>> m = re.search('f..', 'seafood')
>>> m.group()  # 匹配对象的group方法返回匹配到的字符串
'foo'

# findall可以匹配到所有的内容
>>> re.findall('f..', 'seafood is food')
['foo', 'foo']
# finditer返回匹配对象的迭代器
>>> 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'

# split用于切割
# 以.或-作为分隔符
>>> re.split('\.|-', 'how-are-you.tar.gz')
['how', 'are', 'you', 'tar', 'gz']

# 替换
# 将X替换为python
>>> re.sub('X', 'python', 'X is good. I like X.')
'python is good. I like python.'
```

```
# 当有大量匹配时，提前把模式进行编译，可以得到更好的效率
>>> patt = re.compile('f..')
>>> patt.search('seafood')
<_sre.SRE_Match object; span=(3, 6), match='foo'>
>>> m = patt.search('seafood')
>>> m.group()
'foo'
>>> patt.findall('seafood is food')
['foo', 'foo']
```