nsd1912-py02-day03

OOP

- 面向对象编程
- 实现了数据和行为的统一
- 将现实世界中的对象进行抽象化，创建出一个类class
- 通过class再创建具体的实例，也叫对象

```
>>> class MyClass:
...   pass
...
>>> mc1 = MyClass()
```

组合

- 编写程序时，可以创建多个类
- 当多个类明显不同，但是一个类是另一个类的属性时，可以使用组合

子类和继承

- 两个类有很多相似之处，可以使用子类的方法编写
- 子类自动继承父类（基类）所有的方法

```
class Role:
    def __init__(self, name, weapon):
        "构造器方法，用于绑定数据到实例身上 "
        self.name = name
        self.weapon = weapon

    def show_me(self):
        # 绑定在实例身上的属性可以在类中任何地方使用
        print('我是%s, 惯用%s' % (self.name, self.weapon))

    def speak(self, words):
        # 在方法内的变量和参数，都是局部变量
        print(words)

class Warrior(Role):
    def attack(self, target):
        print('与%s近身肉搏' % target)

class Mage(Role):
    def attack(self, target):
        print('远程打击%s' % target)

if __name__ == '__main__':
    lb = Warrior('吕布', '方天画戟')
    zgl = Mage('诸葛亮', '羽扇')
    lb.show_me()
    zgl.show_me()
    lb.attack('张飞')
    zgl.attack('曹操')
```

多重继承

- 类的父类可以有多个
- 子类将会继承所有父类的方法
- 子类查找方法的顺序是自下向上，自左向右，先查到的优先级最高

魔法方法

- 在OOP中，以双下划线开头结尾的方法都有特殊的作用，称作magic方法，即魔法方法。这些方法大部分是实现内部功能的，不需要深究。
- 通过dir()函数查看某一对象所有的属性

```
>>> dir(10)
>>> dir([])
```

- 需要掌握的三个magic，分别是
  - `__init__`: 构造器方法

- `__str__`: 打印、显示实例时调用
- `__call__`: 使实例可以象函数一样调用

正则

练习：为mac地址加冒号

思路：

- 找到mac地址
- mac地址每两个字符分一组
- 在组之间加冒号

```
192.168.1.1     000C29123456
192.168.1.2     525400A3E81B

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

re模块

```
>>> import re
# 在字符串开头匹配 f..，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'>
>>> re.match('f..', 'seafood')
>>> print(re.match('f..', 'seafood'))
None

# 在字符串中匹配 f..，匹配对象的 group方法返回匹配结果
>>> m = re.search('f..', 'seafood')
>>> m.group()
'foo'

# 返回字符串中匹配到的所有模式，返回列表
>>> re.findall('f..', 'seafood is food')
['foo', 'foo']
# 返回字符串中匹配到的所有模式，返回匹配对象构成的迭代器
>>> for m in re.finditer('f..', 'seafood is food'):
...   m.group()
...
'foo'
'foo'

# 以-和.切割字符串
>>> re.split('-|\.', 'hello-world-nihao.tar.gz')
['hello', 'world', 'nihao', 'tar', 'gz']

# 将x替换成tom
>>> re.sub('x', 'tom', 'hi x. nice to meet you, x')
'hi tom. nice to meet you, tom'

# 将正则表达式的模式提前编译，将会有更好的执行效率
>>> patt = re.compile('f..')
>>> patt.search('seafood')
<_sre.SRE_Match object; span=(3, 6), match='foo'>
>>> patt.findall('seafood is food')
['foo', 'foo']
```