nsd1907_py02_day03

模块

搜索路径

- 在导入模块时，python到sys.path定义的目录下搜寻

```
(nsd1907) [root@room8pc16 day03]# mkdir /tmp/mymods
(nsd1907) [root@room8pc16 day03]# vim /tmp/mymods/star.py
hi = "Hello World!"

def pstar(n=30):
    print('*' * n)

if __name__ == '__main__':
    pstar()

(nsd1907) [root@room8pc16 day03]# python
>>> import star   # 报错
>>> import sys
>>> sys.path
>>> sys.path.append('/tmp/mymods')
>>> import star   # 成功
```

- 还可以定义PYTHONPATH环境变量，来指定模块搜索路径

```
(nsd1907) [root@room8pc16 day03]# export PYTHONPATH=/tmp/mymods
(nsd1907) [root@room8pc16 day03]# python
>>> import sys
>>> sys.path
['', '/tmp/mymods', '/usr/local/lib/python36.zip', '/usr/local/lib/python3.6',
'/usr/local/lib/python3.6/lib-dynload', '/root/nsd1907/lib/python3.6/site-packages']
```

- 目录也可以作为一个特殊的模块，称作包

> 注意：在python2中，目录下必须有一个名为__init__.py的文件，才能成为包，该文件即使是空的，也必须存在。

```
(nsd1907) [root@room8pc16 day03]# mkdir mypkgs
(nsd1907) [root@room8pc16 day03]# vim mypkgs/mytest.py
hi = 'Hello China'
(nsd1907) [root@room8pc16 day03]# ls
mypkgs
(nsd1907) [root@room8pc16 day03]# python
>>> import mytest   # Error
>>> import mypkgs.mytest   # 正确
>>> mypkgs.mytest.hi
'Hello China'
```

tarfile模块

- 用于实现压缩、解压缩

```
>>> import tarfile
# 压缩
>>> tar = tarfile.open('/tmp/myfile.tar.gz', 'w:gz')
>>> tar.add('/etc/hosts')
>>> tar.add('/etc/security')
>>> tar.close()
# 解压缩，打开时，python自动判断是什么压缩格式
>>> tar = tarfile.open('/tmp/myfile.tar.gz')
>>> tar.extractall(path='/tmp/mymods')
>>> tar.close()
```

hashlib模块

```
[root@room8pc16 day02]# echo -n 123456 > /tmp/1.txt
[root@room8pc16 day02]# md5sum /tmp/1.txt
e10adc3949ba59abbe56e057f20f883e  /tmp/1.txt
>>> import hashlib
>>> m = hashlib.md5(b'123456')   # 参数必须是bytes类型
>>> m.hexdigest()
'e10adc3949ba59abbe56e057f20f883e'

>>> m1 = hashlib.md5()
>>> m1.update(b'12')
>>> m1.update(b'34')
>>> m1.update(b'56')
>>> m1.hexdigest()
'e10adc3949ba59abbe56e057f20f883e'
```

获取文件绝对路径

```
>>> import os
>>> for path, folders, files in os.walk('/tmp/demo/security'):
...   for file in files:
...     os.path.join(path, file)
```

OOP

- 面向对象的编程
- 在python中，一切皆对象。对象由属性和行为构成。

组合

- 两个类明显不同
- 一个类是另一个类的组件

子类

- 两个类有很多一样的地方
- 某一个类与另一个类又有不同
- 子类可以继承父类的方法
- 子类可以有多个父类
    - 父子类有同名方法，查找顺序是自下向上，自左向右，先找到的执行

魔法方法

- 以__开头和结尾的内部方法，称作magic

正则表达式

为mac地址加冒号：

- 找到MAC地址
- 每两个数字分一组
- 组之间加冒号

```
192.168.1.1     00000C123456
192.168.1.2     525400A3B328

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

re模块

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

# search在字符串中匹配正则，匹配到返回匹配对象
>>> 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方法返回匹配到的内容
'foo'

# 返回所有的匹配
>>> re.findall('f..', 'seafood is food')
['foo', 'foo']
# finditer返回匹配对象构成的生成器
>>> 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'

# 以-或.作为分隔符切割字符串
>>> re.split('-|\.', 'hello-world-china.com.cn')
['hello', 'world', 'china', 'com', 'cn']

# 把X替换成tedu
>>> re.sub('X', 'tedu', 'X web site is X.cn')
'tedu web site is tedu.cn'

# 为了提升匹配效率，最好将正则表达式先编译
>>> patt = re.compile('f..')
>>> patt.match('food')
<_sre.SRE_Match object; span=(0, 3), match='foo'>
>>> patt.search('seafood')
<_sre.SRE_Match object; span=(3, 6), match='foo'>
>>> m = patt.search('seafood')
>>> m.group()
'foo'
```