nsd1902_py01_day04

python官方手册页：

https://docs.python.org/zh-cn/3/ -> 标准库参考

shutil模块

```
>>> import shutil
# copyfileobj只是了解底层原理，实际代码不需要使用
>>> f1 = open('/bin/ls', 'rb')
>>> f2 = open('/tmp/list4', 'wb')
>>> shutil.copyfileobj(f1, f2)
>>> f1.close()
>>> f2.close()

# shutil.copyfile只拷贝内容
>>> shutil.copyfile('/bin/ls', '/tmp/list5')
# shutil.copy既拷贝内容，又拷贝权限
>>> shutil.copy('/bin/ls', '/tmp/list6')
# shutil.copy2相当于系统命令cp -p
>>> shutil.copy2('/bin/ls', '/tmp/list7')
# shutil.move => 相当于系统命令mv
>>> shutil.move('/tmp/list7', '/var/tmp/list')
# copytree相当于cp -r
>>> shutil.copytree('/etc/security', '/tmp/security')
>>> shutil.move('/tmp/security', '/var/tmp/auquan')
# rmtree 相当于rm -rf
>>> shutil.rmtree('/var/tmp/auquan')

# 删除单个文件的函数在os模块
>>> import os
>>> os.remove('/tmp/list5')
# 改变属主属组
>>> shutil.chown('/tmp/list', user='bob', group='bob')
```

subprocess模块：用于调用系统命令

```
>>> subprocess.run(['ls', '/home'])
bob  lisi  Student  wangwu  zhangsan
CompletedProcess(args=['ls', '/home'], returncode=0)

>>> subprocess.run('ls /home', shell=True)
bob  lisi  Student  wangwu  zhangsan
```

```
CompletedProcess(args='ls /home', returncode=0)

>>> subprocess.run(['ls', '~'])
ls: 无法访问~: 没有那个文件或目录
CompletedProcess(args=['ls', '~'], returncode=2)

>>> subprocess.run('ls ~', shell=True)

# subprocess.run的返回值
>>> rc = subprocess.run('ping -c2 192.168.4.254 &> /dev/null', shell=True)
>>> rc.returncode   # 就是系统命令的$?
0

# 捕获输出
>>> rc = subprocess.run('id root; id john', shell=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
>>> rc.returncode
1
>>> rc.stdout
b'uid=0(root) gid=0(root) \xe7\xbb\x84=0(root)\n'
>>> rc.stderr
b'id: john: no such user\n'
# 将bytes类型转成str类型
>>> rc.stdout.decode()
'uid=0(root) gid=0(root) 组=0(root)\n'
```

python语法风格

```
>>> x = y = 10
>>> a, b = 10, 20
>>> x, y = (100, 200)
>>> m, n = [1, 2]
>>> a, b = b, a   # a和b的值互换

# python的关键字
>>> import keyword
>>> keyword.kwlist
>>> 'pass' in keyword.kwlist
True
>>> keyword.iskeyword('pass')
True
```

内建函数不是关键字，可以被覆盖，但是最好不覆盖它。

内建函数：https://docs.python.org/zh-cn/3/library/functions.html

python模块布局

```
#!/usr/local/bin/python3
"""
文档字符串，用于帮助
```

```
"""
import os    # 导入模块
import string

debug = True   # 全局变量
all_chs = string.ascii_letters + string.digits

class MyClass:    # 类的声明
    pass

def my_func():    # 函数声明
    pass

if __name__ == '__main__':    # 程序主体代码
    mc = MyClass()
    my_func()
```

编程思路

1. 发呆。思考程序运行方式：交互？非交互？

```
# filename: /etc/hosts
文件已存在，请重试
# filename: /tmp/abc.txt
请输入内容，输入end表示结束
(end to quit)> Hello World!
(end to quit)> the end.
(end to quit)> bye bye.
(end to quit)> end
```

2. 分析程序有哪些功能，将功能写成函数，编写出大致的框架

```
def get_fname():
    '用于获取文件名'
    pass

def get_content():
    '用于获取文件内容，将文件内容以列表形式返回'
    pass

def wfile(fname, content):
    '将内容与到文件'
    pass
```

3. 编写程序主体，依次调用各个函数

```
if __name__ == '__main__':
    fname = get_fname()
    content = get_content()
    wfile(fname, content)
```

4. 完成每个函数功能。

序列对象

内建函数

```
# list用于转换成列表
>>> list('hello')
['h', 'e', 'l', 'l', 'o']
>>> list(range(1, 10))
[1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list((10, 20, 30))
[10, 20, 30]

# tuple用于转换成元组
>>> tuple('hello')
('h', 'e', 'l', 'l', 'o')
>>> tuple(range(1, 10))
(1, 2, 3, 4, 5, 6, 7, 8, 9)
>>> tuple(['bob', 'tom', 'jerry'])
('bob', 'tom', 'jerry')

# str用于转成字符串
>>> str(100)
'100'
>>> str([100, 200])
'[100, 200]'
```

常用于序列对象的方法：

```
>>> from random import randint
>>> num_list = [randint(1, 100) for i in range(5)]
>>> num_list
[53, 95, 37, 50, 54]

# reversed用于翻转
>>> list(reversed(num_list))
[54, 50, 37, 95, 53]
>>> for i in reversed(num_list):
...     print(i)

# sort排序
>>> sorted(num_list)
[37, 50, 53, 54, 95]
>>> sorted(num_list, reverse=True)   # 降序
[95, 54, 53, 50, 37]
```

```
# enumerate返回下标和元素
>>> list(enumerate(num_list))
[(0, 53), (1, 95), (2, 37), (3, 50), (4, 54)]
>>> for data in enumerate(num_list):
...     print(data)
>>> for ind, num in enumerate(num_list):
...     print(ind, num)
```

字符编码：

ASCII（American Standard Code for Information Interchange，美国信息交换标准代码。

ISO-8859-1(Latin1)：欧洲常用的字符编码

中国采用的字符编码：GB2312 / GBK / GB18030

ISO国际标准化组织发布了标准的万国码：unicode。UTF8就是一种实现方式

字符串格式化

```
>>> '%s is %s years old' % ('tom', 20)
'tom is 20 years old'
>>> '%s is %d years old' % ('tom', 20)
'tom is 20 years old'
>>> '%10s%8s' % ('name', 'age')   # 第一列宽度为10，第二列8
'      name     age'
>>> '%10s%8s' % ('tom', 20)
'       tom      20'
>>> '%-10s%-8s' % ('name', 'age')
'name      age     '
>>> '%-10s%-8s' % ('tom', 20)
'tom       20      '

>>> '%d' % (5 / 3)
'1'
>>> '%f' % (5 / 3)   # 浮点数
'1.666667'
>>> '%.2f' % (5 / 3)   # 保留两位小数
'1.67'
>>> '%6.2f' % (5 / 3)   # 总宽度为6，小数位2位
'  1.67'
>>> '%#o' % 10   # 8进制
'0o12'
>>> '%#x' % 10    # 16进制
'0xa'
>>> '%e' % 128000   # 科学计数法
'1.280000e+05'
```

通过字符串的format方法实现格式化（了解）

```
>>> '{} is {} years old'.format('tom', 20)
'tom is 20 years old'
>>> '{} is {} years old'.format(20, 'tom')
'20 is tom years old'
>>> '{1} is {0} years old'.format(20, 'tom')
'tom is 20 years old'
>>> '{0[1]} is {0[0]} years old'.format([20, 'tom'])
'tom is 20 years old'
>>> '{:<10}{:<8}'.format('tom', 20)  # 左对齐，宽度为10、8
'tom       20      '
>>> '{:>10}{:>8}'.format('tom', 20)   # 右对齐
'       tom      20'
```

原始字符串

```
>>> win_path = 'c:\temp'
>>> print(win_path)   # \t将被认为是tab
c:  emp
>>> win_path = 'c:\\temp'   # \\真正表示一个\
>>> print(win_path)
c:\temp
>>> wpath = r'c:\temp\new'  # 原始字符串，字符串中的字符都表示字面本身含义
>>> print(wpath)
c:\temp\new
>>> wpath
'c:\\temp\\new'
```

字符串方法

```
# 去除字符串两端空白字符
>>> ' \thello world!\n'.strip()
'hello world!'
# 去除字符串左边空白字符
>>> ' \thello world!\n'.lstrip()
'hello world!\n'
# 去除字符串右边空白字符
>>> ' \thello world!\n'.rstrip()
' \thello world!'

>>> hi = 'hello world'
>>> hi.upper()   # 将字符串中的小写字母转成大写
'HELLO WORLD'
>>> 'HELLO WORLD'.lower()   # 将字符串中的大写字母转成小写
'hello world'
>>> hi.center(30)   # 居中
'         hello world          '
>>> hi.center(30, '*')
'*********hello world**********'
>>> hi.ljust(30)
'hello world                   '
>>> hi.ljust(30, '#')
```

```
'hello world###################'
>>> hi.rjust(30, '@')
'@@@@@@@@@@@@@@@@@@@@hello world'
>>> hi.startswith('h')   # 字符串以h开头吗？
True
>>> hi.startswith('he')
True
>>> hi.endswith('o')   # 字符串以o结尾吗？
False
>>> hi.replace('l', 'm')   # 把所有的l替换成m
'hemmo wormd'
>>> hi.replace('ll', 'm')
'hemo world'
>>> hi.split()   # 默认以空格进行切割
['hello', 'world']
>>> 'hello.tar.gz'.split('.')    # 以点作为分隔符切割
['hello', 'tar', 'gz']
>>> str_list = ['hello', 'tar', 'gz']
>>> '.'.join(str_list)   # 以点为分隔符进行字符串拼接
'hello.tar.gz'
>>> '-'.join(str_list)
'hello-tar-gz'
>>> ''.join(str_list)
'hellotargz'
>>> hi.islower()   # 判断字符串内的字母都是小写的吗？
True
>>> 'Hao123'.isupper()   # 字符串内的字母都是大写的吗？
False
>>> 'hao123'.isdigit()   # 所有的字符都是数字字符吗？
False
>>> '123'.isdigit()
True
```