nsd1909-py01-day04

shutil模块

用于执行与文件系统相关的命令

```
>>> import shutil
>>> f1 = open('/bin/ls', 'rb')
>>> f2 = open('/tmp/list3', 'wb')
>>> shutil.copyfileobj(f1, f2)
>>> f1.close()
>>> f2.close()

>>> shutil.copy('/etc/hosts', '/tmp/zhuji')  # cp /etc/hosts /tmp/zhuji
'/tmp/zhuji'

>>> shutil.copy2('/etc/hosts', '/tmp/zhuji2')  # cp -p /etc/hosts /tmp/zhuji2
'/tmp/zhuji2'

>>> shutil.copytree('/etc/security', '/tmp/security')  # cp -r
'/tmp/security'

>>> shutil.move('/tmp/security', '/var/tmp/anquan')  # mv
'/var/tmp/anquan'

>>> shutil.rmtree('/var/tmp/anquan')  # rm -rf

>>> shutil.chown('/tmp/zhuji', user='student', group='student')  # chown

# 查看帮助
>>> help(shutil)
>>> help(shutil.chown)
# 官方文档：https://docs.python.org/zh-cn/3/library/index.html
```

subprocess模块

用于执行系统命令

```
# 只要记住以下形式即可
>>> import subprocess
>>> result = subprocess.run('ls ~', shell=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
# 'ls ~'  -> 要执行的命令
# shell=True   -> 在shell环境中运行命令
# stdout=subprocess.PIPE  -> 固定写法，用于将输出保存到 stdout中
# stderr=subprocess.PIPE  -> 固定写法，用于将错误保存到 stderr中
>>> result  # 可以看到result的各种属性
>>> result.args
'ls ~'
>>> result.returncode   # 返回值，即$?
0
>>> result.stdout.decode()  # stdout是bytes类型，转为str类型
>>> result.stderr.decode()  # stderr是bytes类型，转为str类型

>>> result = subprocess.run('id tom', shell=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
>>> result.returncode
1
>>> result.stderr.decode()
'id: tom: no such user\n'
```

python语法风格

```
# 链式多重赋值
>>> x = y = 10
>>> id(x)  # 查看变量在内存中的地址
9360736
>>> id(y)
9360736

>>> l1 = l2 = [1, 2, 3]
>>> id(l1)
140338008994056
>>> id(l2)
140338008994056
>>> l1
[1, 2, 3]
>>> l2
[1, 2, 3]
>>> l2.append(10)
>>> l2
[1, 2, 3, 10]
>>> l1
[1, 2, 3, 10]

# 多元赋值
>>> a, b = 1, 2
>>> c, d = 'mn'
>>> d, f = (100, 200)
>>> g, h = ['tom', 'jerry']
>>> a, b = b, a   # 两个变量的值互换
>>> a
1
>>> b
2
```

关键字

为了实现python的语法，它保留了一些名字，不能被替换

```
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def',
'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import',
'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try',
'while', 'with', 'yield']
```

内建

内建不是关键字，可以被覆盖，但是尽量不要覆盖。

```
>>> len
<built-in function len>
>>> len('abcd')
4
>>> len = 100
>>> len('abcd')  # 报错，因为此时len不再是函数，而是数字100
# https://docs.python.org/zh-cn/3/library/functions.html
```

python模块布局

```
#!/usr/local/bin/python3          # 解释器位置
"文档字符串，用于help"

import string                     # 导入模块
import shutil

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

class MyClass:                     # 定义类
    pass

def my_func():                     # 定义函数
    pass

if __name__ == '__main__':
    mc = MyClass()
    my_func()
```

编程思路

1. 发呆。思考程的工作方式，交互式？非交互？

```
[root@localhost day04]# python3 mkfile.py
文件名： /etc/hosts
文件已存在，请重试。
文件名： /
文件已存在，请重试。
文件名： /tmp/abc.txt
请输入内容，在单独的一行输入 end表示结束。
(end to quit)> Hello World!
(end to quit)> chi le ma?
(end to quit)> the end
(end to quit)> end
[root@localhost day04]# ls /tmp/abc.txt
/tmp/abc.txt
[root@localhost day04]# cat /tmp/abc.txt
Hello World!
chi le ma?
the end
```

2. 分析程序有哪些功能，将这些功能编写成功能函数，形成程序的大体框架。

```
def get_fname():
    '用于获取文件名，返回一个不存在文件名 '

def get_content():
    '用于获取内容，返回一个列表 '

def wfile(fname, content):
    '需要文件名和内容作为参数，将内容写入文件 '
```

3. 编写代码的主体，按一定的规则调用相关函数

```
def get_fname():
    '用于获取文件名，返回一个不存在文件名 '

def get_content():
    '用于获取内容，返回一个列表 '

def wfile(fname, content):
    '需要文件名和内容作为参数，将内容写入文件 '

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

4. 编写每个具体的函数代码

```
import os

def get_fname():
    '用于获取文件名，返回一个不存在文件名 '
    while 1:
        fname = input('文件名： ')
        # os.path.exists(fname)，如果文件已存在返回 True，不存在返回False
        if not os.path.exists(fname):
            break
        print('文件已存在，请重试。 ')

    return fname

def get_content():
    '用于获取内容，返回一个列表 '
    content = []  # 创建一个列表，用于存储用户输入内容

    print('请输入内容，在单独的一行输入 end表示结束。 ')
    while 1:
        line = input('(end to quit)> ')
        if line == 'end':
            break

        # content.append(line + '\n')
        content.append(line)

    return content

def wfile(fname, content):
    '需要文件名和内容作为参数，将内容写入文件 '
    with open(fname, 'w') as fobj:
        fobj.writelines(content)

if __name__ == '__main__':
    fname = get_fname()
    content = get_content()
    content = ['%s\n' % line for line in content]  # 给字串加上\n后，替换content变量
    wfile(fname, content)
```

序列对象相关的函数

- reversed()：翻转序列对象，返回一个新的翻转结果，不会改变原始对象

```
>>> reversed(seq)
<reversed object at 0x7f3f2ecf1be0>
>>> for zifu in reversed(seq):
...     print(zifu)
>>> list(reversed(seq))
['n', 'o', 'h', 't', 'y', 'p']
>>> l = [10, 20, 25, 30]
>>> list(reversed(l))
[30, 25, 20, 10]
>>> l
[10, 20, 25, 30]
```

- sorted()：排序，返回排序后的列表，不会改变对象本身

```
>>> sorted(seq)  # 按字符的大小排序
['h', 'n', 'o', 'p', 't', 'y']
>>> ord('a')   # a的ascii码
97
>>> ord('A')
65
>>> ord('b')
98
>>> l
[10, 20, 25, 30]
>>> sorted(l)
[10, 20, 25, 30]
>>> l
[10, 20, 25, 30]
```

- enumerate()：可以同时得到下标和值

```
>>> list(enumerate(seq))
[(0, 'p'), (1, 'y'), (2, 't'), (3, 'h'), (4, 'o'), (5, 'n')]
>>> for data in enumerate(seq):  # data是元组
...     print(data)
...
(0, 'p')
(1, 'y')
(2, 't')
(3, 'h')
(4, 'o')
(5, 'n')
>>> for i, zifu in enumerate(seq):   # 可以将元组中的两项分别赋值
...   print(i, zifu)
...
0 p
1 y
2 t
3 h
4 o
5 n
```

字符串

格式化操作符

```
# 基本形式：
>>> '' % ()
# 如果''中的占位符只有一项，()可以省略不写
>>> 'Hello %s' % 'tom'
'Hello tom'
# %s是将相应的数据转换成 str替代它
>>> '%s is %s years old' % ('tom', 20)
'tom is 20 years old'
# %d是整数
>>> '%s is %d years old' % ('tom', 20)
'tom is 20 years old'
>>> '%d is %d years old' % ('tom', 20)   # 报错，因为'tom'不能转成整数
>>> '%s is %d years old' % ('tom', 20.5)  # 20.5保留整数
'tom is 20 years old'
>>> '%f' % (5 / 3)      # %f是浮点数
'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，宽度不足6，左侧补空格
'  1.67'

>>> '%10s%5s' % ('name', 'age')   # 第一个字段点10个宽度，第2个占8个宽度，右对齐
'      name  age'
>>> '%10s%5s' % ('tom', 20)
'       tom   20'
>>> '%-10s%-5s' % ('name', 'age')  # 宽度为负，表示左对齐
'name      age  '
>>> '%-10s%-5s' % ('tom', 20)
'tom       20   '
>>> '%#o' % 10   # 转8进制
'0o12'
>>> '%#x' % 10   # 转16进制
'0xa'
>>> '%e' % 12000  # 科学计数法，1.2乘以10的4次方
'1.200000e+04'

# 字符串格式化，还可以使用 format方法，作为了解
>>> '{} 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'
```

原始字符串

也叫真实字符串

```
>>> win_path = 'c:\temp\new.txt'
>>> print(win_path)   # 打印时\t是tab，\n是回车
c:      emp
ew.txt
>>> wpath = r'c:\temp\new.txt'  # 在字符串前加上r，表示字符串中的字符都是其字面意义
>>> print(wpath)
c:\temp\new.txt
>>> wpath              # 直接写变量名，将显示python内部存储时的样式
'c:\\temp\\new.txt'
>>> win_path = 'c:\\temp\\new.txt'
>>> print(win_path)
c:\temp\new.txt
```

字符串方法

```
# strip用于去除字符串两端的空白字符
>>> s1 = ' \thello world\n'
>>> s1.strip()
'hello world'
>>> s1.lstrip()  # 去除字符串左端的空白字符
'hello world\n'
>>> s1.rstrip()  # 去除字符串右端的空白字符
' \thello world'
>>> s2 = 'hello world'
>>> s2.upper()  # 转大写
'HELLO WORLD'
>>> s3 = 'HELLO'
>>> s3.lower()  # 转小写
'hello'
>>> s2.center(50)  # 总宽度50，居中
'                   hello world                    '
>>> s2.center(50, '*')  # 总宽度50，居中，使用*号填充两侧
'*******************hello world********************'
>>> s2.ljust(50, '#')   # 左对齐
'hello world########################################'
>>> s2.rjust(50, '#')   # 右对齐
'########################################hello world'
```

练习：https://www.jianshu.com/c/00c61372c46a

练习例01到43、45、46、112、113、114、118、127