nsd1907_py01_day04

shutil模块：用于执行一些Shell操作

```
>>> import shutil
>>> f1 = open('/etc/hosts', 'rb')
>>> f2 = open('/tmp/zj.txt', 'wb')
>>> shutil.copyfileobj(f1, f2)
>>> f1.close()
>>> f2.close()

# 拷贝文件, cp /etc/hosts /tmp/zhuji
>>> shutil.copy('/etc/hosts', '/tmp/zhuji')
# cp -p /etc/hosts /tmp/zhuji
>>> shutil.copy2('/etc/hosts', '/tmp/zhuji2')
# cp -r /etc/security /tmp/anquan
>>> shutil.copytree('/etc/security', '/tmp/anquan')
# mv /tmp/anquan /var/tmp/
>>> shutil.move('/tmp/anquan', '/var/tmp/')
# chown bob.bob /tmp/zhuji
>>> shutil.chown('/tmp/zhuji', user='bob', group='bob')

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

subprocess模块：用于调用系统命令

```
>>> import subprocess
>>> result = subprocess.run('id root', shell=True)
uid=0(root) gid=0(root) 组=0(root)   # 命令结果将会print到屏幕
>>> result.args
'id root'
>>> result.returncode   # returncode就是$?
0
# 如果不希望把命令的执行结果打印在屏幕上，可以使用以下方式：
>>> result = subprocess.run('id root; id john', shell=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
>>> result.stderr
b'id: john: no such user\n'
>>> result.stdout
b'uid=0(root) gid=0(root) \xe7\xbb\x84=0(root)\n'
>>> result.stdout.decode()
'uid=0(root) gid=0(root) 组=0(root)\n'
```

语法风格

```
# 链式多重赋值
>>> a = b = 10
>>> a
10
>>> b
10
>>> alist = blist = [1, 2, 3]
>>> blist.append(4)
>>> blist
[1, 2, 3, 4]
>>> alist
[1, 2, 3, 4]

# 多元赋值
>>> a, b = 10, 20
>>> a
10
>>> b
20
>>> c, d = (10, 20)
>>> c
10
>>> d
20
>>> e, f = [100, 200]
>>> g, h = 'ab'

# 交换两个变量的值
>>> a, b = 100, 200
>>> a
100
>>> b
200
>>> t = a
>>> a = b
>>> b = t
>>> a
200
>>> b
100
>>> a, b = b, a    # python交换变量值的方式
```

关键字

```
>>> import keyword
>>> keyword.kwlist
>>> 'pass' in keyword.kwlist
True
>>> keyword.iskeyword('is')
True
```

内建

内建不是关键字，可以被覆盖

```
>>> len('abcd')
4
>>> len = 100
>>> len('abcd')   # 报错，因为此时的len已经是变量，不是函数了
```

模块文件布局

```
#!/usr/local/bin/python3          # 解释器
"""模块文件名

模块文件的说明文字，即文档字符串，用于help帮助
"""
import sys                        # 模块导入
from random import randint, choice

hi = 'Hello World'                # 全局变量
debug = True


class MyClass:                    # 类的定义
    pass


def func1():                      # 函数定义
    pass


if __name__ == '__main__':        # 主程序代码
    func1()
```

编程思路

1. 发呆。思考程序的动作方式（交互？非交互？）

```
文件名: /
文件已存在，请重试。
文件名: /etc/hosts
文件已存在，请重试。
文件名: /tmp/abc.txt
请输入文件内容，在单独的一行输入end结束。
(end to quit)> hello world!
(end to quit)> how are you?
(end to quit)> the end
(end to quit)> end
# cat /tmp/abc.txt
hello world!
```

```
how are you?
the end
```

2. 分析程序有哪些功能，把这些功能编写成函数

```
def get_fname():
    '用于获取文件名'

def get_content():
    '用于获取内容'

def wfile(fname, content):
    '用于将内容content，写入文件fname'
```

3. 编写程序的主体，按一定的准则调用函数

```
def get_fname():
    '用于获取文件名'

def get_content():
    '用于获取内容'

def wfile(fname, content):
    '用于将内容content，写入文件fname'

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

4. 完成每一个具体的函数

序列对象

包括：字符串、列表、元组

```
# list用于将某些数据转成列表
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list('abc')
['a', 'b', 'c']
# tuple用于将某些数据转成元组
>>> tuple(range(10))
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
>>> tuple('abc')
('a', 'b', 'c')

# reversed用于翻转
>>> alist = [10, 8, 25, 24, 1, 100]
```

```
>>> list(reversed(alist))
[100, 1, 24, 25, 8, 10]
>>> for i in reversed(alist):
...   print(i)
# sorted用于排序，默认升序
>>> sorted(alist)
[1, 8, 10, 24, 25, 100]

# enumerate用于获取下标和值
>>> users = ['tom', 'jerry', 'bob']
>>> list(enumerate(users))
[(0, 'tom'), (1, 'jerry'), (2, 'bob')]
>>> for data in enumerate(users):
...   print(data)
...
(0, 'tom')
(1, 'jerry')
(2, 'bob')
>>> for i, name in enumerate(users):
...   print(i, name)
...
0 tom
1 jerry
2 bob
```

字符串

格式化操作符

```
# 基本形式
>>> '' % ()
# 常用的表示方式
>>> '%s is %s years old' % ('tom', 20)
'tom is 20 years old'
>>> '%s is %d years old' % ('tom', 20)   # %d指的是10进制整数
'tom is 20 years old'
>>> '%d is %d years old' % ('tom', 20)   # 错误，tom不能转成数字
>>> '%s is %d years old' % ('tom', 20.5)
'tom is 20 years old'
>>> '%s is %f years old' % ('tom', 20.5)  # %f是浮点数
'tom is 20.500000 years old'
>>> '%s is %.1f years old' % ('tom', 20.5)  # %.1f指保留1位小数
'tom is 20.5 years old'
>>> '%10s%8s' % ('name', 'age')  # 10表示该字段宽度为10
'      name     age'
>>> '%10s%8s' % ('tom', 20)
'       tom      20'
>>> '%-10s%-8s' % ('name', 'age')  # 负数是左对齐
'name      age     '
>>> '%-10s%-8s' % ('tom', 20)
'tom       20      '
```

```
# 不常用，了解
>>> '%c' % 97   # 将数字根据ascii码转换成对应的字符
'a'
>>> '%c' % 65
'A'
>>> '%#o' % 10   # 转8进制
'0o12'
>>> '%#x' % 10   # 转16进制
'0xa'
>>> '%e' % 10000  # 科学计数法
'1.000000e+04'
>>> '%8d' % 10
'      10'
>>> '%08d' % 10   # 宽度为8，不够的补0
'00000010'
```

字符串格式化还可以使用字符串的format方法

```
>>> '{} 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/真实字符串

```
>>> win_path = 'c:\temp'
>>> print(win_path)
c:      emp
>>> wpath = r'c:\temp'
>>> print(wpath)
c:\temp
>>> wpath
'c:\\temp'
>>> win_path = 'c:\\temp'
>>> print(win_path)
c:\temp
```

字符串方法

```
>>> s = '\thello world!      '
>>> s.strip()   # 去除字符串两端空白字符
'hello world!'
>>> s.lstrip()   # 去除字符串左端空白字符
'hello world!      '
>>> s.rstrip()   # 去除字符串右端空白字符
'\thello world!'
>>> 'hello'.center(30)  # 居中
'            hello             '
```

```
>>> 'hello'.center(30, '*')
'************hello*************'
>>> 'hello'.rjust(30, 'a')
'aaaaaaaaaaaaaaaaaaaaaaaahello'
>>> 'hello'.ljust(30, '#')
'hello########################'
>>> 'hello'.upper()  # 转大写
'HELLO'
>>> 'HELLO'.lower()  # 转小写
'hello'
>>> 'hello'.startswith('h')   # 字符串以h开头吗？
True
>>> 'hello'.startswith('he')  # 字符串以he开头吗？
True
>>> 'hello'.endswith('ab')    # 字符串以ab结尾吗？
False
>>> 'hao123'.islower()   # 字母都是小写的吗？
True
>>> 'hao123'.isupper()   # 字母都是大写的吗？
False
>>> '1234'.isdigit()     # 所有的字符都是数字吗？
True

# 判断是不是所有的字符都是数字字符
>>> s = '1234@11'
>>> for ch in s:
...   if ch not in '0123456789':
...     print(False)
...     break
... else:
...   print(True)
...
False
```