# nsd_1908_py01_day04

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

## shutil模块

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

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

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

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

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

# chown tom.tom /tmp/tch
>>> shutil.chown('/tmp/tch', user='tom', group='tom')

# 查看shutil.chown的帮助
>>> help(shutil.chown)

# 查看shutil模块的帮助
>>> help(shutil)

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

## subprocess模块

- 用于执行系统命令

```
# 在shell环境下执行命令
>>> subprocess.run('ls /tmp', shell=True)

# 将运行结果保存到变量中
>>> result = subprocess.run('ls /tmp', shell=True)
>>> result
CompletedProcess(args='ls /tmp', returncode=0)
```

```
>>> result.args
'ls /tmp'
>>> result.returncode   # 即$?
0

# 将标准输出保存到stdout，错误保存到stderr
>>> result = subprocess.run('id root; id zhangsan', shell=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
>>> result.stdout
b'uid=0(root) gid=0(root) \xe7\xbb\x84=0(root)\n'
>>> result.stderr
b'id: zhangsan: no such user\n'
>>> result.stdout.decode()  # 将bytes类型转成str类型
'uid=0(root) gid=0(root) 组=0(root)\n'
```

## 语法风格

```
# 链式多重赋值
# 注意，因为列表是可变的，所以链式多重赋值，两个列表使用同一内存空间
>>> alist = blist = [1, 2]
>>> alist
[1, 2]
>>> blist
[1, 2]
>>> blist[0] = 10
>>> blist
[10, 2]
>>> alist
[10, 2]
# 链式多重赋值，对数字、字符串不会有像列表一样的影响
>>> a = b = 10
>>> b = 20
>>> a
10
>>> b
20

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

# 交换两个变量的值，其他语言的方式
>>> a, b = 100, 200
>>> t = a
>>> a = b
>>> b = t
>>> a
200
>>> b
```

```
100

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

## 标识符

- 标识符就是各种各样的名字：变量、函数、模块

## 关键字

- python也拥有一些被称作关键字的保留字符
- 关键字不能挪作他用

```
>>> 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']
>>> keyword.iskeyword('pass')
True
```

## 内建

- 虽然built-in不是关键字,但是应该把它当作“系统保留字”
- https://docs.python.org/zh-cn/3/library/functions.html

```
>>> len
<built-in function len>
>>> len('abc')
3
>>> len = 10
>>> len('abc')  # 报错，len是数字10，不再是函数
```

## 模块布局

```
#!/usr/local/bin/python3
"""文档字符串

用于模块文件的帮助信息
"""

# 模块导入
```

```
import os
import time
from string import ascii_letters, digits

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

# 类定义
class MyClass:
    pass

# 函数定义
def pstar(n=30):
    '默认打印30个星号，也可以指定个数'
    print('*' * n)

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

## 编程思路

1. 发呆。思考程序的运行方式。交互？非交互？

```
# python mkfile.py
文件名: /etc
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
请输入内容，在单独的一行输入end结束。
(end to quit): Hello World!
(end to quit): 2nd line.
(end to quit): the end.
(end to quit): end
# ls /tmp/abc.txt
abc.txt
# cat /tmp/abc.txt
Hello World!
2nd line.
the end.
```

2. 思考程序有哪些功能，将功能写为功能函数。

```
def get_fname():
    '用于获取并返回文件名'

def get_content():
    '用于获取并返回文件内容'

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

3. 编写主程序，按逻辑调用相关的函数

```
def get_fname():
    '用于获取并返回文件名'

def get_content():
    '用于获取并返回文件内容'

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

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

4. 完成每一个函数

# 作用于序列对象的函数

```
# reversed()用于翻转序列对象
>>> nums = [2, 10, 8, 6]
>>> list(reversed(nums))
[6, 8, 10, 2]
>>> for i in reversed(nums):
...   print(i)

# sorted()用于排序
>>> sorted(nums)
[2, 6, 8, 10]

# enumerate()可以同时得到下标和值
>>> users = ['tom', 'jerry', 'bob', 'alice']
>>> list(enumerate(users))
[(0, 'tom'), (1, 'jerry'), (2, 'bob'), (3, 'alice')]
>>> for data in enumerate(users):
...   print(data)
...
(0, 'tom')
(1, 'jerry')
(2, 'bob')
```

```
(3, 'alice')
>>> for i, name in enumerate(users):
...     print(i, name)
...
0 tom
1 jerry
2 bob
3 alice
```

# 字符串

- 属于标量、不可变、顺序类型

## 字符串格式化

```
# 基本形式
>>> '' % ()
>>> '%s is %s years old' % ('tom', 20)
'tom is 20 years old'
# 如果字符串中只有一个占位符，后面的()可以省略
>>> 'Hi %s' % 'tom'
'Hi tom'
# %d是整数
>>> '%s is %d years old' % ('tom', 20)
'tom is 20 years old'
>>> '%s is %d years old' % ('tom', 20.5)
'tom is 20 years old'

>>> '%f' % (5 / 3)   # %f为浮点数
'1.666667'
>>> '%.2f' % (5 / 3)  # 小数位取两位
'1.67'
>>> '%6.2f' % (5 / 3)  # 总宽度为6，小数位2位，不够宽度左侧补空格
'  1.67'

>>> '%8s%8s' % ('name', 'age')
'    name     age'
>>> '%8s%8s' % ('bob', 20)
'     bob      20'
>>> '%-8s%-8s' % ('name', 'age')
'name    age     '
>>> '%-8s%-8s' % ('bob', 20)
'bob     20      '

# 以下不常用，了解
>>> '%e' % 1230000   # 科学计数法
'1.230000e+06'
>>> '%#o' % 10   # 转8进制表示
'0o12'
>>> '%#x' % 10   # 转16进制表示
'0xa'
```

```
# 字符串格式化，还可以使用字符串的format方法
>>> '{} is {} years old'.format('bob', 20)
'bob is 20 years old'
>>> '{} is {} years old'.format(20, 'bob')
'20 is bob years old'
>>> '{1} is {0} years old'.format(20, 'bob')
'bob is 20 years old'
# 将['bob', 20]作为一个整体，下标为0的是bob，下标为1的是20
>>> '{0[0]} is {0[1]} years old'.format(['bob', 20])
'bob 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'  # 表示字符串中的所有字符都是其字面本身的意思
>>> print(wpath)
c:\temp
>>> wpath
'c:\\temp'
>>> win_path = 'c:\\temp'
>>> print(win_path)
c:\temp
```

## 字符串方法

- https://docs.python.org/zh-cn/3/library/stdtypes.html#text-sequence-type-str

```
>>> s1 = '\tHello World!    '
>>> s1.strip()   # 去除两端空白字符
'Hello World!'
>>> s1.lstrip()   # 去除左端空白字符
'Hello World!    '
>>> s1.rstrip()   # 去除右端空白字符
'\tHello World!'
>>> s2 = 'hao123'
>>> s2.center(30)
'            hao123            '
>>> s2.center(30, '*')
'************hao123************'
>>> s2.ljust(30)
'hao123                        '
>>> s2.rjust(30)
'                        hao123'
>>> s2.startswith('h')  # 字符串以h开头吗？
True
>>> s2.startswith('ha')  # 字符串以ha开头吗？
True
>>> s2.endswith('a')  # 字符串以a结尾吗？
False
>>> '1234'.isdigit()  # 判断字符串是不是都是数字
```

```
True
>>> s2.isdigit()
False
>>> s2.upper()   # 字母转大写
'HAO123'
>>> 'Hao123'.lower()   # 字母转小写
'hao123'
```