nsd1906_py01_day04

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

在标准库参考页面中，“内置函数”就是内建。还可以按ctrl+f搜索模块。

常用系统模块

shutil模块：实现文件的复制、剪切、删除等操作

```
>>> import shutil
# 通过文件对象拷贝文件
>>> f1 = open('/etc/shadow', 'rb')
>>> f2 = open('/tmp/sd', 'wb')
>>> shutil.copyfileobj(f1, f2)
>>> f1.close()
>>> f2.close()

# 直接拷贝文件
>>> shutil.copy('/etc/hosts', '/tmp')
'/tmp/hosts'

# 拷贝目录
>>> shutil.copytree('/etc/security', '/tmp/anquan')

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

# 删除目录
>>> shutil.rmtree('/var/tmp/anquan')

# 改变文件的属主属组
>>> shutil.chown('/tmp/sd', user='bob', group='bob')
>>> help(shutil.chown)
```

subprocess模块：可以调用任何的系统命令

```
>>> subprocess.run('ls -a ~bob', shell=True)
# 执行系统命令，将输出保存到stdout变量中，错误保存到stderr变量中
>>> result = subprocess.run('ls -a ~bob', shell=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
>>> result.stdout
>>> result.stderr
b''
>>> result.returncode   # 即$?
0

>>> result1 = subprocess.run('id natasha', shell=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
>>> result1.stderr
b'id: natasha: no such user\n'
```

python其他语法风格

```
# 链式多重赋值
>>> a = b = 10
>>> a
10
>>> b
10
>>> b = 20
>>> b
20
>>> a
10
>>> alist = blist = [1, 2]
>>> alist
[1, 2]
>>> blist
[1, 2]
>>> blist[0] = 10
>>> blist
[10, 2]
>>> alist
[10, 2]

# 多元赋值
>>> a, b = 'xy'
>>> c, d = (10, 20)
>>> e, f = ['hello', 'world']
>>> g, h = 100, 200
>>> a
'x'
>>> b
'y'
>>> c
10
>>> d
```

```
20
>>> e
'hello'
>>> f
'world'
>>> g
100
>>> h
200

# 交换两个变量的值
>>> a, b = 1, 100
>>> a
1
>>> b
100
>>> t = a   # 其他语言写法
>>> a = b
>>> b = t
>>> a
100
>>> b
1
>>> a, b = b, a   # python写法
>>> a
1
>>> b
100
```

标识符

- 各种各样的名称，如变量、函数、模块、类，统称为标识符
- 合法标识符需要满足的条件：
    - 首字符必须是字母或下划线
    - 其他字符是字母、下划线或数字
    - 区分大小写

关键字

- 为了实现python的语法，python保留了一些名字，叫关键字
- 关键字不能被覆盖

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

内建

- 内建不是关键字
- 但是内建也不建议覆盖

```
>>> type(len)
<class 'builtin_function_or_method'>
>>> type(len)
<class 'builtin_function_or_method'>
>>>
>>> len('abcd')
4
>>> len = 10   # 将len定义为变量，赋值10
>>> len('abcd')   # 报错，因为len已不再是函数，等价于下面的10('abcd')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'int' object is not callable
>>> 10('abcd')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'int' object is not callable
```

模块文件布局

```
#!/usr/bin/python3    # 解释器
"""模块说明文档

用于help帮助时显示
"""

import os             # 模块导入
import time

debug = True
hi = 'Hello World'    # 全局变量的定义

class MyClass:        # 类的定义
    pass

def func1():          # 函数声明
```

```
    pass

def func2():
    pass

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

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/hosts
文件已存在，请重试。
文件名: /etc
文件已存在，请重试。
文件名: /tmp/abc.txt
请输入文件内容，单独输入end表示结束。
(end to quit)> Hello World!
(end to quit)> How are you?
(end to quit)> the end
(end to quit)> end
# ls /tmp/abc.txt
abc.txt
# cat /tmp/abc.txt
Hello World!
How are you?
the end
```

2. 思考程序由哪些功能构成，将这些功能编写成函数。

```
def get_fname():

def get_content():

def wfile(fname, content):

```

3. 编写主程序，按一定的规则调用函数

```
def get_fname():
    '返回一个文件名字符串'

def get_content():
    '返回文件内容的字符串列表'

def wfile(fname, content):
    '将content中的内容写入文件fname中'

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

4. 编写每个函数

序列对象

```
# list将对象转成列表
>>> list('abc')
['a', 'b', 'c']
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
# tuple将对象转成元组
>>> tuple('abc')
('a', 'b', 'c')
>>> tuple(range(10))
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
# str将对象转成字符串
>>> str(10)
'10'

# reversed函数用于反转对象
>>> from random import randint
>>> nums = [randint(1, 100) for i in range(10)]
>>> nums
[68, 48, 98, 78, 43, 78, 94, 5, 53, 11]
>>> reversed(nums)
<list_reverseiterator object at 0x7f480addcc18>
>>> list(reversed(nums))
[11, 53, 5, 94, 78, 43, 78, 98, 48, 68]

# sorted函数用于排序
>>> sorted(nums)
[5, 11, 43, 48, 53, 68, 78, 78, 94, 98]

# enumerate可以同时得到下标和值
>>> for data in enumerate(nums):
...   print(data)

# 分别将下标和值赋值给i和n
>>> for i, n in enumerate(nums):
```

```
...     print(i, n)
```

字符串

- 比较大小，是按照字符的码值比较
- 常用的编码有：ASCII、Lantin-1(ISO8859-1)、GB2312、GB18030、GBK、utf8

```
>>> s = '中'
>>> s
'中'
>>> s.encode()   # 将字符(str)转成bytes类型
b'\xe4\xb8\xad'
>>> a = s.encode()
>>> a
b'\xe4\xb8\xad'
>>> a.decode()   # 将bytes类型转成str类型
'中'
```

字符串格式化操作符

```
# 基础格式
>>> "" % ()       # 如果字符串中只有一项需要替换，()可以省略

>>> '%s is %s years old' % ('tom', 20)
'tom is 20 years old'
>>> '%s is %d years old' % ('tom', 20)  # %d表示需要用整数进行替换
'tom is 20 years old'

>>> '%d is %d years old' % ('tom', 20)  # 错误，tom转不成整数
>>> '%8s%5s' % ('name', 'age')  # 默认右对齐
'    name  age'
>>> '%8s%5s' % ('tom', 20)
'     tom   20'
>>> '%-8s%-5s' % ('name', 'age')  # 数字为负表示左对齐
'name    age  '
>>> '%-8s%-5s' % ('tom', 20)
'tom     20   '

# 以下备查
>>> '%#o' % 10  # 转8进制
'0o12'
>>> '%#x' % 10  # 转16进制
'0xa'
>>> '%f' % (5/3)
'1.666667'
>>> '%.2f' % (5/3)  # 保留2位小数
'1.67'
>>> '%5.2f' % (5/3)  # 输出总宽度为5，小数位2位，不够宽度补空格
' 1.67'
>>> '%e' % 10000  # 科学计数法
```

```
'1.000000e+04'

# 字符串格式化还可以使用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'
```

字符串方法

```
# 没有字符串方法，判断一个字符串是不是全为数字
>>> a = input('data: ')
data: 1234
>>> for i in a:
...   if i not in '0123456789':
...     print(False)
...     break
... else:
...   print(True)
...
True
>>> a = input('data: ')
data: 123a456
>>> for i in a:
...   if i not in '0123456789':
...     print(False)
...     break
... else:
...   print(True)
...
False

# 用字符串方法判断字符串中所有的字符是否为数字
>>> a = '123a456'
>>> b = '123'
>>> a.isdigit()
False
>>> b.isdigit()
True

>>> s1 = '  hello  \n'
>>> s1.strip()   # 去除两端空白字符
'hello'
>>> s1.rstrip()  # 去除右侧空白字符
'  hello'
>>> s1.lstrip()  # 去除左侧空白字符
'hello  \n'
>>> 'hello world hao 123'.split()  # 切割字符串
['hello', 'world', 'hao', '123']
>>> 'hello.world.hao.123'.split('.')
```

```
['hello', 'world', 'hao', '123']
>>> a = ['hello', 'world', 'hao', '123']
>>> '-'.join(a)   # 用-拼接字符串
'hello-world-hao-123'
>>> 'hello world'.replace('l', 'a')  # 将l替换为a
'heaao worad'
>>> 'hello'.center(20)  # 居中
'       hello        '
>>> 'hello'.center(20, '#')
'#######hello########'
>>> 'hello'.ljust(20, '*')  # 左对齐
'hello***************'
>>> 'hello'.rjust(20, '*')  # 右对齐
'***************hello'
>>> 'hello'.upper()   # 转大写
'HELLO'
>>> 'HELLO'.lower()   # 转小写
'hello'
>>> 'hao1233'.islower()   # 字母都是小写的吗？
True
>>> 'hao1233'.isdigit()   # 所有的字符都是数字吗？
False
```

完整的字符串方法参见：https://docs.python.org/zh-cn/3.6/library/stdtypes.html#text-sequence-type-str