nsd1906_py02_day02

函数

函数参数

- 写为key=val形式的参数称作关键字参数
- 直接写为arg形式的参数称作位置参数

```
>>> def func1(name, age):
...   print('%s is %s years old.' % (name, age))
...
>>> func1('bob', 20)    # OK
>>> func1(20, 'bob')    # 语法正确，语义不对
>>> func1(age=20, name='bob')   # OK
>>> func1(age=20, 'bob')   # 语法错误, 关键字参数必须在后
>>> func1(20, name='bob')  # Error, name得到了多个值
>>> func1('bob', age=20)   # OK
```

参数组

- 定义参数时，参数名前面加上*表示使用元组接收参数
- 定义参数时，参数名前面加上**表示使用字典接收参数

```
>>> def func1(*args):
...   print(args)
...
>>> func1()
()
>>> func1('hao')
('hao',)
>>> func1('hao', 123)
('hao', 123)
>>> func1('hao', 123, 'bob', 'alice')
('hao', 123, 'bob', 'alice')

>>> def func2(**kwargs):
...   print(kwargs)
...
>>> func2()
{}
>>> func2(name='bob')
{'name': 'bob'}
>>> func2(name='bob', age=20)
{'name': 'bob', 'age': 20}
```

- 传参时，*表示把序列对象拆开

- 传参时，**表示把字典对象拆开

```
>>> def func3(x, y):
...   return x + y
...
>>> nums = [10, 20]
>>> func3(*nums)   # func3(10, 20)
30

>>> def func4(name, age):
...   print('%s is %s years old.' % (name, age))
...
>>> adict = {'name': 'alice', 'age': 18}
>>> func4(**adict)   # func4(name='alice', age=18)
alice is 18 years old.
```

匿名函数

```
>>> def add(x, y):
...   return x + y
# 可以改写为
>>> myadd = lambda x, y: x + y
>>> add(10, 5)
15
>>> myadd(10, 5)
15
```

filter函数

- 它接受两个参数。filter(func, seq)
- 第一个参数是函数，如func
- 第二个参数是序列对象
- func它必须接受一个参数，返回值必须是True或False
- filter函数工作时，将序列对象中的每个值作为func的参数进行过滤，结果为真的保留，为假的舍弃

map函数

- 它接受两个参数。map(func, seq)
- 第一个参数是函数，如func
- 第二个参数是序列对象
- func它必须接受一个参数，它将接收到的数据进行处理，然后返回

变量

- 在函数外面定义的变量是全局变量。全局变量从定义开始，到程序结束，任意地方可见可用。

```
>>> x = 10
>>> def func1():
...   print(x)
...
>>> func1()
10
```

- 在函数内定义的变量是局部变量。局部变量只能在函数内部使用。

```
>>> def func2():
...   a = 100
...   print(a)
...
>>> func2()
100
>>> a
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'a' is not defined
```

- 如果局部和全局有同名变量。局部变量将会遮盖住全局变量。

```
>>> def func3():
...   x = 'hello world!'
...   print(x)
...
>>> func3()
hello world!
>>> x   # 全局变量x没有受到影响
10
```

- 如果希望通过函数改变全局变量，需要使用关键字global

```
>>> def func4():
...   global x
...   x = 10000
...   print(x)
...
>>> func4()
10000
>>> x
10000
```

偏函数

- 改造现有函数，生成新函数
- 改造时，可以将现有函数的一些参数固定

```
>>> def func5(a, b, c, d, e):   # 函数必须有5个参数
...   return a + b + c + d + e
```

```
...
>>> func5(10, 20, 30, 40, 2)    # 每次传参，前4个数固定
102
>>> func5(10, 20, 30, 40, 8)
108
>>> from functools import partial
>>> myadd = partial(func5, 10, 20, 30, 40)
>>> myadd(2)
102
>>> myadd(8)
108

>>> int('11001010', base=2)
202
>>> int2 = partial(int, base=2)
>>> int2('1010')
10
>>> int2('1111')
15
```

递归函数

- 函数内部包含对自己的调用

```
# 数字阶乘
5!=5x4x3x2x1
5!=5x4!
5!=5x4x3!
5!=5x4x3x2!
5!=5x4x3x2x1!
1!=1
# 一个数字的阶乘，就是这个数乘以它下一个数的阶乘。1的阶乘是1
>>> def func1(x):
...   if x == 1:
...     return 1
...   return x * func1(x - 1)
...
>>> func1(5)
120
```

生成器

- 可以通过生成器表达式得到生成器

```
>>> nums = (randint(1, 100) for i in range(10))
>>> for i in nums:
...   print(i)
```

- 使有函数的形式
  - 常规函数通过return返回一个最终结果
  - 生成器通过yield语句，返回多个中间结果

```
>>> def mygen():
...   yield 10
...   a = 100 + 5
...   yield a
...   yield 200
...
>>> mg = mygen()
>>> list(mg)
[10, 105, 200]
>>> mg = mygen()
>>> for i in mg:
...   print(i)
```

模块

导入模块时，python到指定路径下搜寻

- sys.path定义的路径
- PYTHONPATH环境变量定义的路径

```
>>> import sys
>>> sys.path
['', '/usr/local/lib/python36.zip', '/usr/local/lib/python3.6',
'/usr/local/lib/python3.6/lib-dynload', '/root/nsd1906/lib/python3.6/site-packages']

(nsd1906) [root@room8pc16 day02]# cp qsort.py /var/tmp/
(nsd1906) [root@room8pc16 day02]# cd /tmp/
(nsd1906) [root@room8pc16 tmp]# python
>>> import qsort   # 报错
(nsd1906) [root@room8pc16 tmp]# export PYTHONPATH=/var/tmp
(nsd1906) [root@room8pc16 tmp]# python
>>> import qsort   # 成功
```

内置模块

- tarfile模块用于压缩、解压缩
- hashlib模块用于计算哈希值

```
# 创建压缩文件
>>> import tarfile
>>> tar = tarfile.open('/tmp/mytest.tar.gz', 'w:gz')
>>> tar.add('/etc/hosts')
>>> tar.add('/etc/security')
>>> tar.close()
(nsd1906) [root@room8pc16 day02]# tar tvzf /tmp/mytest.tar.gz

# 解压，打开时不用指定压缩格式
>>> tar = tarfile.open('/tmp/mytest.tar.gz')
>>> tar.extractall(path='/var/tmp')  # 解压到指定位置，默认为当前目录
>>> tar.close()
```

```
# 计算数据的哈希值
>>> import hashlib
>>> m = hashlib.md5(b'123456')
>>> m.hexdigest()
'e10adc3949ba59abbe56e057f20f883e'

>>> with open('/etc/passwd', 'rb') as fobj:
...   data = fobj.read()
...
>>> m = hashlib.md5(data)
>>> m.hexdigest()
'687950834a69d8e7ab0ba9e9111eeb1f'

>>> m = hashlib.md5()
>>> m.update(b'12')
>>> m.update(b'34')
>>> m.update(b'56')
>>> m.hexdigest()
'e10adc3949ba59abbe56e057f20f883e'
```