第七章 抽象与模块

7.1 列表（元组）传参

```
def add(x,y):
    return x+y

def maxL(*x): #*x表示可变参数
    return max(x)

a = [1,2]
b = [1,2,3,4,5,6]

m = add(*a) #*a表示将列表a中元素按位置传递给add作为参数
n = maxL(*b)#*b表示将列表b中元素按位置传递给maxL作为参数

print(m,n)
```

7.2 字典传参

```
def ave(Chinese=60,Math=60):
    return (Chinese+Math)/2


def avem(Chinese=60,Math=60, **kwds):#**kwds可变命名参数
    return ((Chinese+Math)+sum(kwds.values()))/(len(kwds)+2)

para = {"Chinese":90, "Math":70}

param = {"Chinese":90, "Math":70, "English":85, "History":90}

print(ave(**para)) #**para将para的键和值作为参数传递给函数ave
print(avem(**param))
```

7.2 不同类型参数定义顺序

位置参数，可变参数，命名参数，可变命名参数

```
def foo(x,*args,y = 1,  **ky):
    print(x)
    print(y)
    print(args)
    print(ky)

foo(1,2,3,4,5,6,7,8,9,10,y = 10000,  z = 10)
```

7.3 map函数

map函数遍历序列每个值进行操作，返回一个map对象

语法:map(func, seq)

```
#单序列map
listVar = ['a', 'b', 'c']
mapObj = map(lambda x:x.upper(), listVar)
print(mapObj)
print(list(mapObj))

#多序列map
s = [1,2,3]
print(list(map(lambda x,y,z:x*y*z ,s,s,s)))
```

7.4 filter函数

filter函数，过滤出序列中的元素。

- 语法:filter(function, iterable)
- function -- 判断函数。
- iterable -- 可迭代对象
- 返回filter对象

```
#过滤出大于5的元素
tupleVar = (1,2,3,4,5,6,7,8)
print("原始序列是:", tupleVar)
filterObj = filter(lambda x: x>5, tupleVar)
print("filter直接返回filter对象:", filterObj)
result = list(filterObj)
print("过滤出大于5的元素组成一个列表", result)
```

7.5 reduce函数

reduce() 函数会对参数序列中元素进行累积，返回一个值。

- 函数将一个数据对象（列表，元组等）中的所有数据进行下列操作：用传给 reduce 中的函数 function（有两个参数）先对集合中的第 1、2 个元素进行操作，得到的结果再与第三个数据用 function 函数运算，最后得到一个结果。

语法:

- from functools import reduce
- reduce(function, iterable[, initializer])
- function是操作函数，有两个参数
- Iterable是迭代对象
- Initializer是初始值

7.5 reduce函数

```
from functools import reduce

def comput_number_of_digits(s):
    count = 0
    for item in s:
        if item.isdigit():
            count = count + 1
    return count

listVar = ["a123", "bcd1.12", "craft3.14"]
total_count = reduce(lambda x, y: x+comput_number_of_digits(y), listVar, 0)
print(total_count)
```

7.6 模块的创建

模块是Python中的一个重要概念。随着编写的程序越来越长，可以将这些代码分成几个文件，这样更易于代码的维护。当把一些相关的代码存放在一个文件中时，就创建了一个模块。模块中的定义可以被导入到其他模块中从而被其他模块所使用，这就使得我们可以在多个程序中使用已经编写好的函数而无需将函数复制到每个程序中。

总的来说，模块就是包含Python定义和声明的文件。文件名就是模块名加上.py的扩展名。

模块有一些内置属性，用于存储模块的某些信息，如__name__，__doc__，等等。__name__属性用来取得模块的名称。

7.6 模块的创建

```
def add(*x):
    "加法"
    return sum(x)

def ave(*x):
    return sum(x)/len(x)

def sub(*x):
    return x[0]- sum(x[1:])

def mul(*x):
    from functools import reduce
    return reduce(lambda x, y: x*y, x, 1)

if __name__=="__main__":
    x = [1,2,3,4]
    print( add(*x) )
    print( ave(*x) )
    print( sub(*x) )
    print( mul(*x) )
```

7.7 模块的导入

Python以模块为单位来组织代码。Python标准库自身就内置了许多标准模块，还有非常丰富的第三方模块以供用户使用。当然，用户也可以自己编写模块。

要在模块外部使用模块内定义的函数，首先要导入该模块。使用import语句可以导入一个模块，格式为“import 模块名 [as 别名]”。

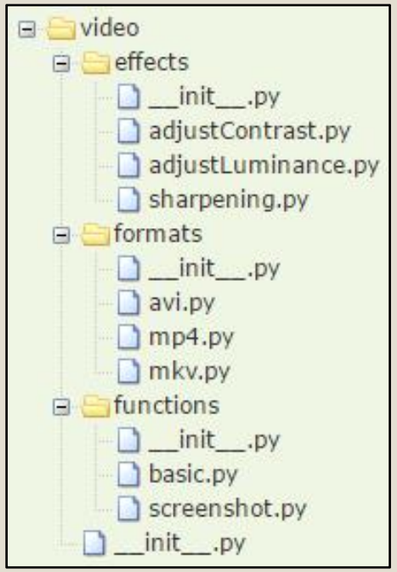
如果频繁地使用一个函数而不想总是带着模块名进行调用，则可以将其赋给一个本地变量。

此外，还可以在import后添加as子句来作为模块的别名。

Python还支持另外一种语法，即“from 模块名 import 对象名 [as 别名]”。使用这种格式仅导入明确指定的对象，可以减少访问速度，同时不需要使用模块名进行调用。如果想要使用这一语法导入模块下的全部对象，则可以使用星号来替代对象名。

7.8 包与包内引用

- 包是一种通过“点分模块名称”管理Python模块命名空间的方式。
- 可以将这些模块按照某种方式组织在一个目录下，构成一个包结构。
- 用户可以从包中导入单独的模块或模块中的对象。



7.8 包与包内引用

子模块之间同样需要互相引用

不在当前子包内的模块如何引用

- 绝对导入： import video.functions.basic
- 相对导入： from ..functions import basic

7.9 第三方包的安装

pip：方便的包管理工具

安装pip：

- 下载pip
- 安装pip： python setup.py install

使用pip：

- pip install <PackageName> 安装包
- pip show <PackageName> 查看已安装的包信息
- pip list 列出已安装的所有包
- pip list --outdated 列出需要更新的包
- pip install --upgrade <PackageName> 升级包
- pip uninstall <Package> 卸载包