※ Python3 包模块函数方法

4. 包模块函数方法

4.2 函数方法作用域

§ 理解变量

```
a='abc' #变量a指向字符串对象

a=['a','b','c'] #变量a也可以指向列表对象
```

以上代码中，变量 a 是没有类型，它仅仅是一个对象的引用（一个指针）

- 回顾一下,python中的number,string,tuple,这三个类型都是不可以变对象
- list,dict,set都是可变对象

```
a=1.0 #将float类型(不可变对象)数值1.0赋值给变量a
b=a #变量b与变量a相同
b=3 #将int类型数值3赋值给变量b
print(a) #此时思考 a变量是什么?? 答案貌似很简单当然还是1.0
```

```
a=[1,2,3,4,5] #将list类型(可变类型).赋值给变量a
b=a #变量a赋值给变量b
b[0]=3 #将int类型数值3赋值给变量a
print(a) #此时再思考 a变量是什么??
```

上一行代码改变变量b的值,变量a指向的值也改变了,这就是变量仅表示指向对象的意思

*通常更准确的说法是,改变的不是b的值,而是改变变量b指向对象的值

§ 理解函数

你可以定义一个由自己想要功能的函数，以下是简单的规则：

- 函数代码块以 def 关键词开头，后接函数标识符名称和圆括号 ()。
- 任何传入参数和自变量必须放在圆括号中间，圆括号之间可以用于定义参数。
- 函数的第一行语句可以选择性地使用文档字符串—用于存放函数说明。
- 函数内容以冒号起始，并且缩进。
- return [表达式] 结束函数，选择性地返回一个值给调用方。不带表达式的return相当于返回 None。

参数

以下是调用函数时可使用的正式参数类型：

不带参数,必需参数,关键字参数,默认参数,不定长参数

- 没有带参数_函数实例

```
def myprint(): #构造一个没有参数的函数,该函数没有返回值,
    print('hello')
    return
a = myprint #将方法赋值给变量a
a() #变量名称后带()表示调用方法
```

- 带必需参数_函数实例

```
def mymax(x,y): #实现输出最大值
    return  x if x>=y else y

mymax(1,100)
```

```
def mysort(x,y): #实现排序的方法
    if x<=y:
        a,b=x,y
    else:
        a,b=y,x
    return(a,b)

mysort(4,2)
```

- 带关键字参数_函数实例
    - 关键字参数和函数调用关系紧密，函数调用使用关键字参数来确定传入的参数值。
    - 使用关键字参数允许函数调用时参数的顺序与声明时不一致，因为 Python 解释器能够用参数名匹配参数值。

```
def myprint(name,age): #构造带有关键字参数的函数
    print('我的名字%s,今年%d岁'%(name,age))
    return
a=myprint #将方法赋值给变量a

a('python',2) #正常调用方法时,传入必需参数

a(age=2,name='python') #可以无序传入带关键字的参数
```

- 带不定长参数_函数实例
  用在函数需要处理不定长度的参数时

```
def mymaxs( *var ): #声明一个可以比较不限定数量并且能返回该最大值的函数
    a= list(var) #先转换为列表类型
    a.sort() #对列表进行排序
    return a[-1] #返回列表最后一个值

mymaxs(1.0,2,8,12335,31,3)
```

```
def mymaxs(**vardict ): #加了两个星号 ** 的参数会以字典的形式导入。
    print(vardict)

mymaxs(x=3,dfdf='ui')
```

```
def f(x,*,y,z): #声明函数时,带+号后面的参数必须带关键字
    return x*y+z

f(1,y=2,z=5)
```

- 匿名函数 (lambda)
  - ambda 只是一个表达式，函数体比 def 简单很多。
  - lambda的主体是一个表达式，而不是一个代码块。仅仅能在lambda表达式中封装有限的逻辑进去。

```
lambda x:x+1 #关键字lambda后面跟参数,冒号后面跟表达式,

mysum = lambda x,y,z:x+y+z #将lambda函数返回值赋值给变量sum
mysum(10,2,5)
```

```
mymax = lambda *v:list(v)# 创建一个返回最大数的lambda表达式
a=mymax(10,2,5,235,6,5,4,8)
a.sort()
print(a[-1])
```

匿名函数的应用

```python
a=map(lambda x:x*x,[1,1,2,3]) #lambda与map函数的应用 map(函数,列表)
print(list(a))
```

```python
a=[x*x for x in [1,1,2,3]] # 等效于上一行语句
print(list(a))
```python
a=filter(lambda x: x % 3 == 0, [1,2,3]) #lambda与filter函数的应用 filter(函数,列表)
print(list(a))

a= [x for x in [1,2,3] if x%3==0] # 等效于上一行语句
print(a)
```

§ 理解作用域

- 定义在函数内部的变量拥有一个局部作用域，定义在函数外的拥有全局作用域。
- 局部变量只能在其被声明的函数内部访问，而全局变量可以在整个程序范围内访问。
- 调用函数时，所有在函数内声明的变量名称都将被加入到作用域中。

```python
x = 0 # 全局变量x
def add ( a, b ):
    x = a + b # 函数内也有x,但与全局变量x不一样,
    print ("函数内是局部变量x =", x)
    return x


add( 10, 20 )
print ("函数外是全局变量x =", x)
```

如果想在函数内调用全局变量 则需要加关键字 global

```python
x = 0 # 全局变量x
def add ( a, b ):
    global x #声明调用全局变量
    x = a + b # 函数内也有x,但与全局变量x不一样,
    print ("函数内是局部变量x =", x)
    return x


add( 10, 20 )
print ("函数外是全局变量x =", x)
```

如果要修改嵌套作用域（enclosing 作用域，外层非全局作用域）中的变量则需要 nonlocal 关键字了，如下实例：
```

```
x=0 # 全局变量x
def outer():
    x = 10 # 函数outer()的局部变量x
    def inner():
        x = 100 # 函数inner()的局部变量x
        print('这调用的是函数inner()的局部变量x=',x)
    inner()# 函数体outer调用自建函数inner()
    print('这调用的是函数inner()的局部变量x=',x)
outer()
print('这调用的是全局变量x=',x)
```

```
x=0 # 全局变量x
def outer():
    x = 10 # 函数outer()的局部变量x
    def inner():
        nonlocal x   # nonlocal关键字声明调用是函数outer()的局变量x
        x = 100 # 函数inner()的局部变量x
        print('这调用的是函数inner()的局部变量x=',x)
    inner()# 函数体outer调用自建函数inner()
    print('这调用的是函数inner()的局部变量x=',x)
outer()
print('这调用的是全局变量x=',x)
```