# 第六章

# 函数代码的复用

# 本章大纲

1、函数
2、函数的定义
3、函数的使用
4、函数的参数传递
5、函数的返回值
6、变量的作用域
7、代码复用
8、lambda函数

# 1.函数的定义

函数是一段具有特定功能的、可重用的语句组，通过函数名来表示和调用的语句。经过定义，一组语句等价于一个函数，在需要使用这组语句的地方，直接调用函数名称即可。

因此，函数的使用包括两部分：函数的定义和函数的使用。

函数是一种功能抽象。

**语法格式：**

```
def  <函数名> (<参数列表>):
        <函数体>
```

**注意：**

- 括号后面的冒号不能少；
- 即使该函数不需要接收任何参数，也必须保留一对空括号；
- 函数体相对于def关键字必须保持一定的空格缩进。
- 函数形参不需要声明其类型，也不需要指定函数的返回类型；

# 2.函数的调用

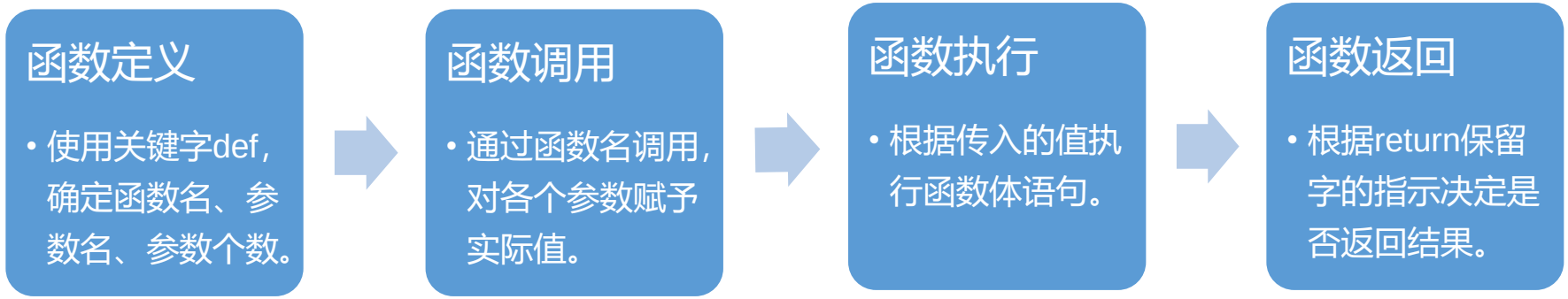
定义后的函数不能直接运行，需要经过“调用”才能运行。调用函数的基本方法如下：

<函数名>(<实际赋值参数列表>)

```
#定义一个sum函数实现计算a+b的和
def sum(a, b):
    s = a+b
    return s
s = sum(2, 3) #调用sum函数
print(s)
```

# 3.函数的使用

通过以上学习，我们可以将函数的使用分为四个部分：



# 4.函数的参数传递

## 1、位置参数

位置参数是比较常用的形式，调用函数时实参和形参的顺序必须一致，并且数量相同。

```
>>> def mul(x,y):
        print(x*y)


>>> mul(2,4)
8
>>> mul(2) #实参和形参的数量必须一致
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    mul(2) #实参和形参的数量必须一致
TypeError: mul() missing 1 required positional argument: 'y'
```

## 2、默认值参数

函数的参数在定义时也可以指定默认值，函数调用时若该位置没有给定实际参数，则使用默认值代替。但需要注意可选参数应当放在非可选参数后面。定义如下：

```
def    〈函数名〉(〈非可选参数列表〉, 〈可选参数〉 = 〈默认值〉):
        〈函数体〉
```

```
>>> def mul(x, y = 10):
        print(x*y)


>>> mul(99)
990
>>> mul(99, 2)
198
```

## 3、关键字参数

Python语言同时支持函数按照参数名称方式传递参数，此种传参方式不必考虑实参和形参的顺序。

调用方式如下：

〈函数名〉(〈参数名称〉 =〈实际值〉):

```
>>> def mul(x,y):
        print(x*y)


>>> mul(y = 2,x = 99)
198
```

## 4、可变长度参数

除了以上传参方式以外，当我们不确定会接受多少个参数的时候可以利用可变长度参数解决。

*param接收任意多个参数放在一个元组中

**param接收任意多个关键字参数放入字典中

定义方式如下：

```
def    <函数名
>(*param):
            <函数体>
def    <函数名
>(**param):
            <函数体>
```

```
>>> def demo1(*p):
        print(p)

>>> demo(1,2,3)
(1, 2, 3)
>>> def demo2(**p):
        print(p)

>>> demo2(a=1,b=2,c=3)
{'c': 3, 'a': 1, 'b': 2}
```

# 5.变量的作用域

根据程序中变量所在的位置和作用范围，变量分为局部变量和全局变量。

局部变量仅在函数内部，且作用域也在函数内部，全局变量的作用域跨越多个函数。

1、局部变量

局部变量指在函数内部使用的变量，仅在函数内部有效，当函数退出时变量将不再存在。

```
def mul(x, y=10):
    z=x*y
    return z
s = mul(99, 2)
print(s) #改行打印s结果为198
print(z) #改行会报错 name 'z' is not defined
```

变量z是函数multiple()内部使用的变量，当函数调用后，变量z将不存在。

## 2、全局变量

全局变量指在函数之外定义的变量，在程序执行全过程有效。全部变量在函数内部使用时，需要提前使用保留字global声明，语法形式如下：

global <全局变量>

```
n = 2 #此处n为全局变量
def mul(x,y=10):
    global n  #在函数内部需要使用是先声明
    n = x*y
mul(99,2)
print(n) #此时全局变量n以发生改变
```

当在函数内部需要修改全局变量，必须使用global先声明，否则默认为局部变量。

当局部变量与全局变量同名时，函数内部会优先使用局部变量。

```
n = 2 #此处n为全局变量
def mul(x, y=10):
    n = 4
    print(n) #此时打印的n为局部变量
mul(99, 2)
print(n)
```

# 6.函数的返回值

- return语句用来结束函数并将程序返回到函数被调用的位置继续执行。
- return语句可以出现在函数中的任何部分。
- return可以同时返回0个或多个函数运算的结果给函数被调用处的变量。
- 当return返回多个值时，返回的值形成元组数据类型。
- 函数也可以没有return代表无返回值。

```
>>> def mul(x, y):
        return x*y

>>> s = mul(99, 2)
>>> print(s)
198
```

```
>>> def mul(x, y):
        return x*y, x+y

>>> s=mul(99, 2)
>>> print(s)
(198, 101)
```

# 7.代码的复用

函数是将一部分代码组织起来供其他代码使用，函数封装的直接好处就是代码复用，任何代码只要输入参数即可调用函数，从而避免相同功能代码在被调用处重复编写。

模块化设计指通过函数的封装功能将程序划分为主程序、子程序和子程序间关系的表达。模块化设计是使用函数设计程度的思考方法，以功能为基本单位，一般由两个基本要求：

- 紧耦合：尽可能合理划分功能模块，功能块内部耦合紧密。
- 松耦合：模块间关系尽可能简单，功能块之间耦合度低。

耦合性指程序中各模块之间想相互关联的程度。

# 8.匿名函数

匿名函数适合处理临时需要一个类似于函数的功能但又不想定义函数的场合，可以省去函数的定义过程和考虑函数的命名，让代码更加简洁，可读性更好。表达式如下：

<函数对象名>=lambda <形式参数列表>:<表达式>

例如：

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

等价于匿名函数：

```
fun = lambda x,y:x+y
```