

用函数实现模块化程序设计

函数的创建和调用

1. 创建函数

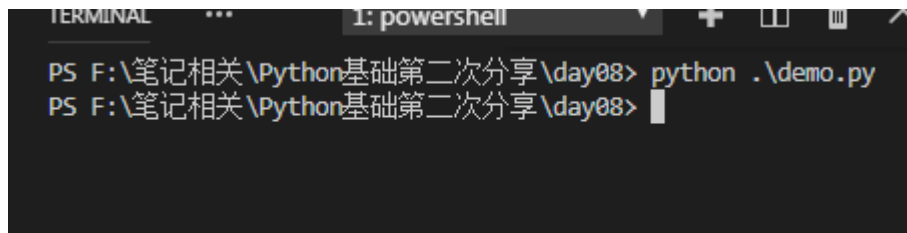
语法：

```
def functionname([形参]):
```

```
    执行代码块
```

这里需要注意的是即使形参没有，括号也是必须的。

```
def hello(name):  
    print("{}你好!".format(name))
```



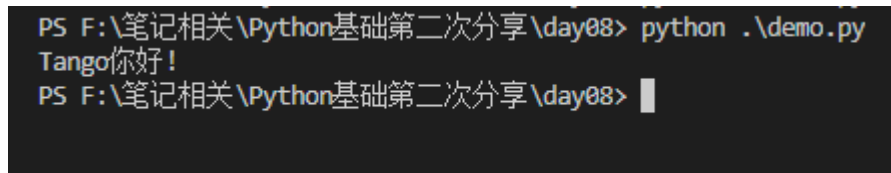
```
TERMINAL ... 1: powershell  
PS F:\笔记相关\Python基础第二次分享\day08> python .\demo.py  
PS F:\笔记相关\Python基础第二次分享\day08> 
```

我们运行这个文件，发现不会输出任何内容，因为还没有调用这个方法。

2. 调用函数

调用函数就是执行这个函数。

```
def hello(name):  
    print("{}你好!".format(name))  
  
myname = 'Tango'  
hello(myname) #调用函数
```



```
PS F:\笔记相关\Python基础第二次分享\day08> python .\demo.py  
Tango你好!  
PS F:\笔记相关\Python基础第二次分享\day08> 
```

这是就会看到执行结果了。

3. pass空语句

pass语句表示空语句，它不做任何事，一般是用来站位的。就像我们在自习室用书站位置一样。

在Python3中3个"."也是表示站位。

```
def sayBye(name):  
    pass #空语句，什么都不做
```

参数传递

1. 形参与实参

- 形参：在定义函数时括号里面的参数为形式参数，也称形参
- 实参：在调用函数时，传入的参数为实际参数，也称实参

2. 位置参数

位置参数也称为必备参数，必须按照正确的顺序传到函数中，且数量也必须相等。

```
def userInfo(name, age, height):  
    '''  
    定义一个方法，接收名字，年龄以及身高  
    并打印出来  
    '''  
    print("姓名:{}\n年龄:{}\n身高:{}".format(name, age, height))  
  
# 调用函数  
userInfo('路飞', 18, 175)
```

以为函数有3个参数，所以必须传3个实参进去。

3. 关键字参数

关键字参数是指使用形参名字来确定输入的参数，这种方式不需要位置相同

4. 为参数设置默认值

在定义函数时，还可以为形参指定默认值，如果不传该参数则使用默认值

```
def userInfo2(name, age, height=170):  
    '''  
    定义一个方法，接收名字，年龄以及身高  
    并打印出来  
    height=170 就是为参数指定了默认值  
    '''  
    print("姓名:{}\n年龄:{}\n身高:{}".format(name, age, height))  
  
userInfo2(age = 18, name='路飞', height= 175)  
print('-----使用默认值-----')  
userInfo2(age = 18, name='路飞')
```

5. 可变参数

- *参数

这种形式表示接受任意多个实参并将其放到一个元组中。

```
def hello(*name):  
    '''  
    注意这里的形参前面有一个*  
    表示接受任意数量的name  
    '''  
    print('形参name的类型', type(name))
```

```

    for item in name:
        print("{}，你好".format(item))
print('-----一个实参-----')
hello('路飞')
print('-----2个实参-----')
hello('路飞', '乔巴')
print('-----3个实参-----')
hello('路飞', '娜美', '乔巴')

```

```

PS F:\笔记相关\Python基础第二次分享\day08> python .\demo2.py
-----一个实参-----
形参name的类型 <class 'tuple'>
路飞，你好
-----2个实参-----
形参name的类型 <class 'tuple'>
路飞，你好
乔巴，你好
-----3个实参-----
形参name的类型 <class 'tuple'>
路飞，你好
娜美，你好
乔巴，你好
PS F:\笔记相关\Python基础第二次分享\day08> ^A

```

o **参数

这种形式接收任意多的显示赋值的实际参数，并将其放到一个字典中

```

def userinfo(**info):
    for key, val in info.items():
        print("{}的年龄是{}".format(key, val))

userinfo(路飞='18', 乔巴='6')
print("-----")
dict1 = {'路飞':18, '乔巴':6}
userinfo(**dict1) # 实参前面也需要加**

```

```

PS F:\笔记相关\Python基础第二次分享\day08> python .\demo3.py
路飞的年龄是18
乔巴的年龄是6
-----
路飞的年龄是18
乔巴的年龄是6
PS F:\笔记相关\Python基础第二次分享\day08> 

```

返回值

当我们想获取函数的执行结果时，就需要让函数有返回值，在Python中可以使用return来指定函数返回的内容。

```
def sum(a, b):
    '''计算a的b次方，并返回'''
    return a**b

print('{}的{}次方是{}'.format(2, 2, sum(2, 2)))
print('{}的{}次方是{}'.format(2, 3, sum(2, 3)))
```

```
PS F:\笔记相关\Python基础第二次分享\day08> python .\demo4.py
2的2次方是4
2的3次方是8
PS F:\笔记相关\Python基础第二次分享\day08> ^A
```

变量的作用域

1. 局部变量

局部变量指的是在函数内部定义的变量，在函数外无法使用。

```
def msg():
    message = "你好"
    print("局部变量的内容:", message)

msg()
print("messaging的内容", message)
```

```
PS F:\笔记相关\Python基础第二次分享\day08> python .\demo5.py
局部变量的内容: 你好
Traceback (most recent call last):
  File ".\demo5.py", line 6, in <module>
    print("messaging的内容", message)
NameError: name 'message' is not defined
PS F:\笔记相关\Python基础第二次分享\day08> ^A
```

我们发现函数内部的内容打印出来了，但是在函数外部打印是，发现报错了。

2. 全局遍历

在函数外部定义的变量，无论函数，还是函数外都可以使用

```
message = '你好'

def msg():
    print("函数内部: ", message)

msg()
print("函数外部: ", message)
```

```
PS F:\笔记相关\Python基础第二次分享\day08> python .\demo6.py
函数内部: 你好
函数外部: 你好
PS F:\笔记相关\Python基础第二次分享\day08> █
```

在函数内部可以用global将一个局部变量转换成全局变量。

```
def showMsg():
    global errMsg
    errMsg = "我变成了全局变量"
showMsg()
print(errMsg)
```

注意的是，要使用这个变量前，必须先执行showMsg()函数，否则还是会报错的。

匿名函数

匿名函数也称lambda表达式，关于这部分大家可以自己在网上查一下用法。它的首要用途是指定短小的回调函数。

```
import math # 导入数学模块

def circlearea(r): #计算圆形的面积
    result = math.pi * r * r
    return result

r = 10
print("半径为10的圆的面积是:", circlearea(10))

# 用匿名函数
r = 5
result = lambda r:math.pi * r * r
print("半径为5的圆的面积是", result(r))
```

```
PS F:\笔记相关\Python基础第二次分享\day08> python .\demo7.py
半径为10的圆的面积是: 314.1592653589793
半径为5的圆的面积是 78.53981633974483
PS F:\笔记相关\Python基础第二次分享\day08> |
```