函数

引言

- 什么是函数?
  - 前面在讲解Python数据类型的时候，我们已经接触过函数了。我们说，所谓的函数其实就是Python语言中的一种工具，基于该工具可以完成不同的具体操作。
  - 案例：当你在野外露营的时候，如果想生火，如果你身上恰好带了打火机，则可以直接使用该工具自行完成生火操作，否则，你也可以自己利用现有环境下的资源自行制作取火工具。

函数基础

- 自定义函数的使用要经过两个过程
  - 函数的定义（制定）
  - 函数的调用（使用）
- 函数定义语法

```
def 函数名(参数):
    #内部代码
    return 表达式
```

```
def myFunc(): #函数定义的时候，函数体是不会被执行的
    #函数体
    a = 1
    b = 2
    c = a + b
    print('a和b的计算结果为:',c)
```

- 函数调用语法
  - 函数编写出来就是给人调用的。
  - 要调用一个函数，必须使用函数名后跟圆括号的方式才能调用函数。
  - 调用的同时要根据函数的定义体，提供相应个数和类型的参数，每个参数之间用逗号分隔，否则就会报错。

```
myFunc() #函数调用，函数定义中的函数体才会被执行
```

- 函数定义规范使用

```
def summer(lis):
    """
    这里是函数的说明文档，doc的位置
    :param lis: 参数列表的说明
    :return: 返回值的说明
    """
    total = 0
    for i in lis:
        total += i
    return total
```

返回值

- return语句
  - 当一个函数被调用结束后，该函数势必已经将一组操作执行结束了，如果在操作执行结束后，想要将一个结果返回给调用者，则就可以使用return语句实现。

```
#返回一个表达式
def func():
    return 1 + 2 #返回一个表达式

result = func()
print(result)
```

```
#不写return默认返回None(空)
def func():
    a = 10
    b = 20
    sum = a + b

result = func()
print(result)
```

```
#返回多个结果
def func():
    return 1,'two',3.3
```

```
r1,r2,r3 = func()
print(r1,r2,r3)


######################
def func():
    return 1,'two',3.3

result = func() #使用一个变量接收多个返回值，多个返回值会被封装到一个元组中
print(result)
```

```
#return后面的代码无意义：程序执行到return语句后，表示函数调用结束
def func():
    return 'bobo'
    print('i love bobo') #会执行吗？不会执行的（报错）！
func()
```

```
def outer():
    print('我是外部函数outer')
    def inner():
        print('我是内部函数inner')
#想要调用inner函数?
outer()
```

```
#如何调用inner这个内部函数呢?
def outer():
    print('我是外部函数outer')
    def inner():
        print('我是内部函数inner')
    return inner #返回的是内部函数名，不加括号的!

result = outer() # outer() == inner, result == inner
result() # inner()
```

```
#返回函数调用（了解）
def outer():
    print('我是外部函数outer')
    def inner():
        print('我是内部函数inner')
    return inner()  #return None
outer()
```

函数参数

- 增加函数的通用性

```
#定义一个函数可以计算出两个数据的和
def my_add():
    num1 = 1
    num2 = 2
    return num1 + num2
result = my_add()
print(result)
#局限性：只可以计算指定两个数的和，无法实现具有较高的通用性
```

```
#定义一个函数可以计算出两个数据的和:具有更强的通用性
def my_add(num1,num2): #num1=5,num2=9
    return num1 + num2

result = my_add(5,9)
print(result)
```

- 绝大多数函数在定义的时候需要接收一定数量的参数，然后根据实际调用时提供的参数的不同，输出不同的结果。注意将函数内部的参数名字，定义得和外部变量的名字一样是一种不好的习惯，它容易混淆思维，甚至发生错误。
- 参数的两种称谓
  - 形参(形式参数)
    - 函数定义时，制定的参数叫做形参
  - 实参(实际参数)

        - 函数调用时，传递的参数叫做实参
    - 而我们通常讨论的参数，指的都是形参

参数类型

- 参数的不同种类
    - 定义函数时，参数的名字和位置确定下来，函数的接口就固定了。对于函数的调用者来说，只需要知道如何传递正确的参数，以及函数将返回什么样的值就够了，函数内部的复杂逻辑被封装起来，调用者无需了解。
      Python函数的参数定义灵活度非常大。除了正常定义的位置参数外，还可以使用：
        - 位置参数(重点)
        - 默认参数
        - 动态参数

位置参数

- 也叫必传参数或者顺序参数，是最重要的、也是必须在调用函数时明确提供的参数！位置参数必须按先后顺序，一一对应，个数不多不少的传递！

```
def add(a,b,c):
    return a+b+c

x = y = 5
r1 = add(x,y,x)
r2 = add(4,5,6)
print(r1,r2)
```

- 上面例子中的a，b，c就是位置参数，我们在使用add(4, 5, 6)调用时，就是将4的地址传给a，5的传给b，6的传给c的一一对应传递。
    - 类似add(4, 5, 6, 7)、add(4)这种“画蛇添足”、“缺胳膊少腿”和“嫁错郎”类型的调用都是错误的。
- **注意: Python在做函数参数传递的时候不会对数据类型进行检查，理论上你传什么类型都可以!**

```
def add(a,b,c):
    return a+b+c

add(1,2,'haha')
```

- 但是，上面的add函数，如果你传递了一个字符串和两个数字，结果是弹出异常，因为字符串无法和数字相加。
    - 这就是Python的弱数据类型和动态语言的特点。在简单、方便的时候，需要你自己去实现数据类型检查。

默认参数

- 在函数定义时，如果给某个参数提供一个默认值，这个参数就变成了默认参数，不再是位置参数了。在调用函数的时候，我们可以给默认参数传递一个自定义的值，也可以使用默认值。

```
def power(x,n=2): #x是位置参数，n是默认参数
    return x * n


power(10)
power(10,4)
```

- 上面例子中的n就是个默认参数。默认参数可以简化函数的调用，在为最常用的情况提供简便调用的同时，还可以在特殊情况时传递新的值。
- 默认参数的注意事项：
  - 默认参数必须在位置参数后面！
  - 使用参数名传递参数

```
def
student(name,sex,age,classroom='101',tel='132
3333333',address='...'):
    pass #函数体：pass表示一个空实现

#下述函数调动是否可以?
student('jack','male',17) #正确
student('tom','male',18,'102','666','Beijing'
) #正确
student('marry','female',18,'102',address='SH
') #正确
student('mary','female',address='Bj',18)   #错
误
```

课上练习

```
#定义一个函数，该函数可以将一个列表中的最大值找到
alist = [3,8,5,7,6,9,66,17]

#函数可以接收一个列表，将其内部最大值找到
def max_value_list(items):
    for i in range(len(items)-1):
        if items[i] > items[i+1]:
            items[i],items[i+1] =
items[i+1],items[i]
    return items[-1]

#使用该函数max_value_list将alist列表中的最大值找到
max_value = max_value_list(alist)
print(max_value)
```