函数

非结构化编程：
    所有的代码逻辑都写在一起---逻辑混乱--代码太多，太杂
结构化编程：
    将有特殊功能的代码抽出,封装成函数；重复性使用
函数：有一组代码组成，完成某个特定的功能

- 函数的定义（创建函数）

关键字：
    def 函数名（参数）： #参数可以为多个，逗号隔开，没有参数时--无参
        函数体
函数的调用：
    函数名(参数)

- 函数的运行机制

当调用函数的时候，解释器（PVM）会根据函数名找到函数定义的位置，进而到函数内部依次执行内部的代码，执行完毕之后返回到函数的调用处

- 函数的特点

1. 避免了代码的冗余--多余代码
2. 提高了代码的可维护性
3. 提高了代码的可重用性
4. 提高了代码的灵活性

- 函数定义的位置

1. 全局变量类似
        定义在文件的开头（文件的最左边（没有缩进））
2. 局部变量类似
        可以定义在任何函数内部

- 函数的结构

1. 函数的定义---函数的声明
        def 函数名():
2. 函数体（函数的实现）
        函数具体内容
        核心代码---业务逻辑
3. 函数的返回值
        return 返回的内容
        谁调用就返回给谁
4. 函数的调用
        函数名()

函数的参数

1. 形式参数：形参--没有具体的值
        函数定义时的参数
2. 实际参数：实参
        调用函数时传递的参数

如果函数没有定义参数---无参
函数的参数相当于一个局部变量

- 位置参数（形参）

根据参数的位置传递参数----一一对应---不能多也不能少参数

```
def fun1(b,a):
    print('a:%d b:%d'%(a,b))
fun1(10,20)
```

位置---一一对应

- 关键字参数

定义了关键字的参数---实参

```
def fun1(b,a):
    print('a:%d b:%d'%(a,b))

fun1(a=10,b=20)#a=10,b=20
```

跟位置无关

- 位置参数和关键字参数混用

1. 位置参数要在关键字参数之前，如果返回来解释器(PVM)会认为关键字参数之后的参数全都是关键字参数

```
def fun1(a,b):
    print('a:%d b:%d'%(a,b))

fun1(20,b=10)
```

2. 实际参数只能赋值一次

```
def fun1(a,b):
    print('a:%d b:%d'%(a,b))

fun1(20,a=10)  #报错
```

- 默认参数

设置了默认值的参数--形参
不需要传递参数，使用默认值
如果传递了参数，会把默认值覆盖掉

```
def fun1(a=10,b=20):
    print('a:%d b:%d'%(a,b))
fun1()
```

默认参数必须放在位置参数的后面

- 可变长参数

只定义一个参数但是可以传递多个参数n

1. 在参数前加上*

```
def fun1(*a):
    print(a)  #将参数封装成了元组
fun1(100,200,300)
```

2. *在参数前加***

将参数封装成字典

```
def fun1(**a):
    print(a)
fun1(c=100,d=200,e=300) #关键字参数传参
```

- 可变长参数和位置参数混用

如果可变长参数在位置参数前，需要使用关键字参数赋值（实参）

```
def fun1(*a,b):
    print(a,b)

fun1(100,200,300,400,b=500)
```

函数的文档

函数的文档注释：三引号字符串中的内容可以作为函数的文档注释（函数的第一行）

1. 函数名.__doc__ ----返回函数文档的注释
2. help(函数名)