学习目标

1. 掌握函数的定义与调用;
2. 掌握函数实参和形参的使用方法;
3. 了解函数的返回值;
4. 掌握缺省参数的使用方法;

目录

第 1 章 函数------简介

一、函数的作用

函数的作用，在开发程序时，使用函数可以提高编写的效率以及代码的重用。

函数的使用包含两个步骤：

1. 定义函数 —— 封装独立的功能 ；

2. 调用函数 —— 执行函数的代码 。

二、函数基本使用

1. 函数的定义

定义函数的格式如下：

```
def 函数名():
    函数封装的代码
    ……
```

1. def 是英文 define 的缩写 ；

2. 函数名称应该能够表达函数封装代码的功能，方便后续的调用 ；

3. 函数名称的命名应该符合标识符的命名规则 ；

2. 函数调用

通过 **函数名()** 即可完成函数的调用

三、函数演练

1. 需求

- 编写一个 hello 的函数，封装三行代码 ；
- 在函数下方调用hello 函数 。

```
# 定义函数hello
def hello():
    print("hello world")
    print("hello python")
    print("hello itcast")
# hello 函数定义完成
# 调用hello 函数


hello()
```

注意：因为函数体相对比较独立，函数定义的上方，应该和其他代码（包括注释）保留两个空行 。

定义好函数之后，函数内的代码并不会执行，只表示这个函数封装了一段代码而已 。

调用函数后，函数的代码才会执行。如果不主动调用函数，函数是不会主动执行的。

2. 思考

- 能否将函数调用放在函数定义的上方？

不能！

因为在调用函数之前，必须要提前定义函数。

1. 课堂练习---

定义一个函数，名字叫 my_func1

调用函数结果为显示 20 个连续的星号

```
********************
```

第 2 章 函数------参数

一、需求

1. 开发一个 my_sum 的函数 ；

2. 函数能够实现两个数字的求和功能;

```
def my_sum():
    num1 = 12
    num2 = 15
    result = num1 + num2
    print(result)


my_sum()
```

二、存在什么问题

函数只能处理固定数值的相加 ，如果能够把需要计算的数字，在调用函数时，传递到函数内部就好了！

三、函数参数的使用

- 在函数名的后面的小括号内部填写参数 ；
- 多个参数之间使用 ，分隔 。
- 带参数的求和函数

```
def my_sum1(num1, num2):
    result = num1 + num2
    print(result)


my_sum1(5, 9)
```

四、参数的作用

函数：把具有独立功能的代码块组织为一个小模块，在需要的时候调用；

函数的参数：增加函数的通用性，针对相同的数据处理逻辑，能够适应更多的数据；

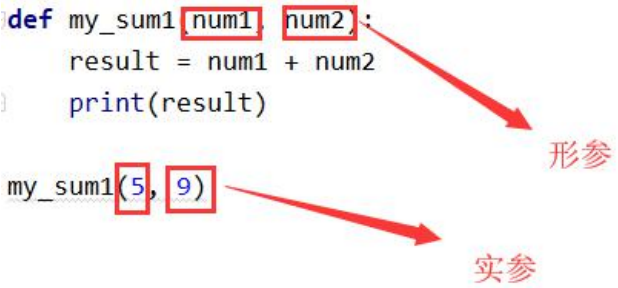
1. 在函数内部，把参数当做变量使用，进行需要的数据处理

2. 函数调用时，按照函数定义的参数顺序，把希望在函数内部处理的数据，通过参数传递

五、形参和实参

形参：定义函数时，小括号中的参数，是用来接收参数用的，在函数内部作为变量使用；

实参：调用函数时，小括号中的参数，是用来把数据传递到 函数内部用的 。



1. 课堂练习---

定义一个函数，名字叫 my_func2，有一个参数 num1；

调用 my_func2 时，num1 为 1，输出个*号，num1 为 5，输出 5 个*号；

举例：调用函数 my_func2(3)应该输出如下结果:

***

第 3 章 函数------返回值

在程序开发中，有时候会希望一个函数执行结束后，告诉调用者一个结果，以便调用者针对具体的结果做后续的处理。

返回值是函数完成工作后，最后给调用者的一个结果。

一、return 关键字

在函数中使用 return 关键字可以返回结果。

调用函数一方，可以使用变量来接收函数的返回结果 。

注意： return 表示返回，后续的代码都不会被执行

- 带返回值的 my_sum2 函数

```
def my_sum2(num1, num2):
    result = num1 + num2
    return result

    # return 后面的代码，将不会被执行
    print("test")


a = my_sum2(5, 9)
print(a)
```

- 返回参数中的最大值

```
def my_max(num1, num2):
    if num1 > num2:
        return num1
    else:
        return num2


a = my_max(5, 9)
print(a)
```

二、函数参数和返回值演练

1. 定义一个函数，有两个参数，start 和 stop，start 代表开始范围，stop 代表终止范围，求这个范围中所有整数相加的和

```
def my_sum1(start, stop):
    sum = 0
    a = start
    while a <= stop:
        sum += a
        a += 1
    return sum


print(my_sum1(3, 9))
```

2. 定义一个函数能够根据半径计算圆的面积

```
def circular(r):
    pai = 3.14
    s = pai * r **2
    return s


print(circular(1))
```

3. 课堂练习---

定义一个函数，名字叫 my_squar，功能为计算矩形的面积，有两个参数 height 与 width，分别代表矩形的高和宽；

函数返回值为矩形的面积；

如调用 my_squar(3, 4)，函数返回值为 12。

4. 课堂练习---

定义一个函数，名字叫 my_func，有两个参数 num1 与 num2，当 num1 能被 num2 整除时，返回值为 True,否则返回值为 False。

如：调用 my_func(8, 4)，函数返回值为 True。

如：调用 my_func(9, 4)，函数返回值为 False。

第 4 章 函数------嵌套调用

一个函数里面又调用 了另外一个函数，这就是函数嵌套调用。

如果函数 test2 中，调用了另外一个函数 test1

那么执行到调用 test1 函数时，会先把函数 test1 中的任务都执行完

才会回到 test2 中调用函数 test1 的位置，继续执行后续代码

```
def test1():
    print("我是 test1")

def test2():
    # 先执行函数 test1 的代码
    test1()
    # test1 函数执行完毕后，再执行下面代码
    print("我是 test2")

test2()
```

第 5 章 函数------变量的作用域

一、局部变量和全局变量

局部变量是在函数内部定义的变量，只能在函数内部使用 ；

全局变量是在函数外部定义的变量（没有定义在某一个函数内），所有函数内部都可以使用这个变量 。

提示：在其他的开发语言中，大多不推荐使用全局变量 —— 可变范围太大，导致程序不好维护！

二、局部变量

局部变量是在函数内部 定义的变量，只能在函数内部使用；

函数执行结束后，函数内部的局部变量，会被系统回收；

不同的函数，可以定义相同的名字的局部变量，彼此之间不会产生影响 ；

三、局部变量的作用

在函数内部使用，临时保存函数内部需要使用的数据

```
def my_func1():
    a = 10

def my_func2():
    a = 20
    my_func1() # 调用 my_func2 函数，不会影响 a 的值
    print("a = %d" % a)


my_func2()
```

四、局部变量的生命周期

所谓生命周期就是变量从被创建到被系统回收的过程 ；

局部变量在函数执行时才会被创建 ；

函数执行结束后局部变量被系统回收 ；

局部变量在生命周期内，可以用来存储 函数内部临时使用到的数据 。

五、全局变量

全局变量是在函数外部定义的变量，所有函数内部都可以使用这个变量 。

为了保证所有的函数都能够正确使用到全局变量，应该将全局变量定义放在其他函数上方。

```
# 定义一个全局变量 num
num = 100
def my_func1():
    print(num)


def my_func2():
    print(num)


my_func1()
my_func2()
```

六、全局变量与局部变量重名

```
# 定义一个全局变量 num
num = 100
def my_func1():
    # 函数内部定义一个变量和全局变量重名
    num = 1

# 全局变量 num 的值并没有改变
my_func1()
print(num)
```

注意：只是在函数内部定义了一个局部变量而已，只是变量名相同 —— 在函数内部不能直接修改全局变量的值。

七、global 关键字

如果在函数中需要修改全局变量，需要使用 global 进行声明

```
# 定义一个全局变量 num
num = 100
def my_func1():
    # 函数内部使用 global 关键字声明全局变量 num
    global num
    num = 1

# 全局变量 num 的值被 my_func1 函数改变
my_func1()
print(num)
```

1. 课堂练习---

定义一个全局变量 name=”张三”，定义一个函数 my_test1，在函数 my_test1 内部修改全局变量 name 的值为”李四”

第 6 章 函数------参数进阶

一、形参和实参的值传递

如果函数的参数为数字，字符串，在函数内部，针对形参使用赋值语句，不会影响调用函数时传递的实参的值。

```
def my_test(num):
    num += 1
    print("num = %d" % num)


a = 100
my_test(a)
print("a = %d" % a)
```

如果函数参数为列表，集合，字典等类型。函数内部修改了参数的内容，会影响到外部的数据 。

```
def my_test(list1):
    list1[0] = "刘备"

a = ["周瑜", 100, 3]
my_test(a)
print(a)
```

1. 课堂练习---

定义一个函数，参数为列表类型，调用函数过后，删除列表所有值

```
list1 = [2,5,3]
myfunc(list1)
print(list1)
```

```
# 结果为
[]
```

二、缺省参数

定义函数时，可以给 某个参数指定一个默认值，具有默认值的参数就叫做缺省参数 。

调用函数时，如果没有传入缺省参数的值，则在函数内部使用定义函数时指定的参数默认值 。

函数的缺省参数，将常见的值设置为参数的缺省值，从而简化函数的调用 。

1. 单个缺省参数

```
# 第二个参数的缺省值为100
def my_func1(a, b = 100):
    print("a = %d, b = %d" % (a, b))

my_func1(1, 2)
# 没有写第二个参数，第二个参数采用缺省值100
my_func1(1)
```

2. 多个缺省参数

当函数有多个缺省参数，调用函数的时候，对于不使用默认值的缺省参数可以通过：**参数 = 值** 的方式，明确具体参数的值。

```
def my_func2(a = 100, b = 200, c = 300):
    print("a = %d, b = %d, c = %d" % (a, b, c))
```

```
my_func2(1, 2, 3)
# 没有写前两个参数，前二个参数采用默认值,第三个参数值为3
my_func2(c = 3)
```

- 缺省参数的注意事项：

1) 缺省参数的定义位置

必须保证带有默认值的缺省参数在参数列表末尾 。

所以，以下定义是错误的！

def print_info(name, gender=True, title):

第 7 章 函数------匿名函数

用lambda 关键词能创建小型匿名函数。这种函数得名于省略了用def 声明函数的标准步骤。

lambda 函数的语法只包含一个语句，如下：

lambda [arg1 [,arg2,.....argn]]:expression

如下实例：

sum = lambda arg1, arg2: arg1 + arg2

- lambda 案例 1

```
# 简化版的 sum 求和函数
my_sum = lambda a, b: a + b
num = my_sum(3, 6)
```

```
print(num)
```

- lambda 案例 2

```
# 简化版的 Lambda 函数，求最大值
num = (lambda a, b: a if a > b else b)(3, 6)
print(num)
```

- Lambda 函数只能返回一个表达式的值 ；
- 匿名函数不能直接调用print，因为 lambda 需要一个表达式 。