Python函数进阶

目录

Contents

学习目标

Learning Objectives

1. 知道函数如何返回多个返回值

问： 如果一个函数如些两个return（如下所示），程序如何执行？

```
def return_num():
    return 1
    return 2


result = return_num()
print(result)  # 1
```

答： 只执行了第一个return，原因是因为return可以退出当前函数，导致return下方的代码不执行

多个返回值

如果一个函数要有多个返回值，该如何书写代码？

```
def test_return():
    return 1, 2

x, y = test_return()
print(x)    # 结果1
print(y)    # 结果2
```

按照返回值的顺序，写对应顺序的多个变量接收即可

变量之间用逗号隔开

支持不同类型的数据return

目录

Contents

学习目标

Learning Objectives

1. 掌握位置参数
2. 掌握关键字参数
3. 掌握不定长参数
4. 掌握缺省参数

函数参数种类

使用方式上的不同，函数有4中常见参数使用方式：

- 位置参数
- 关键字参数
- 缺省参数
- 不定长参数

位置参数

位置参数：调用函数时根据函数定义的参数位置来传递参数

```
def user_info(name, age, gender):
    print(f'您的名字是{name}, 年龄是{age}, 性别是{gender}')



user_info('TOM', 20, '男')
```

注意：

传递的参数和定义的参数的顺序及个数必须一致

关键字参数

关键字参数：函数调用时通过“键=值”形式传递参数.

作用：可以让函数更加清晰、容易使用，同时也清除了参数的顺序需求.

```
def user_info(name, age, gender)
    print(f"您的名字是：{name}，年龄是：{age}，性别是：{gender}")

# 关键字传参
user_info(name="小明", age=20, gender="男")
# 可以不按照固定顺序
user_info(age=20, gender="男", name="小明")
# 可以和位置参数混用，位置参数必须在前，且匹配参数顺序
user_info("小明", age=20, gender="男")
```

注意：

函数调用时，如果有位置参数时，位置参数必须在关键字参数的前面，但关键字参数之间不存在先后顺序

缺省参数

缺省参数：缺省参数也叫默认参数，用于定义函数，为参数提供默认值，调用函数时可不传该默认参数的值（注意：所有位置参数必须出现在默认参数前，包括函数定义和调用）.

作用：当调用函数时没有传递参数，就会使用默认是用缺省参数对应的值.

```
def user_info(name, age, gender='男'):
    print(f'您的名字是{name}, 年龄是{age}, 性别是{gender}')



user_info('TOM', 20)
user_info('Rose', 18, '女')
```

注意：

函数调用时，如果为缺省参数传值则修改默认参数值，否则使用这个默认值

不定长参数

不定长参数：不定长参数也叫可变参数. 用于不确定调用的时候会传递多少个参数(不传参也可以)的场景.

作用：当调用函数时不确定参数个数时，可以使用不定长参数

不定长参数的类型：

①位置传递

②关键字传递

位置传递

```
def user_info(*args):
    print(args)



# ('TOM',)
user_info('TOM')
# ('TOM', 18)
user_info('TOM', 18)
```

注意：

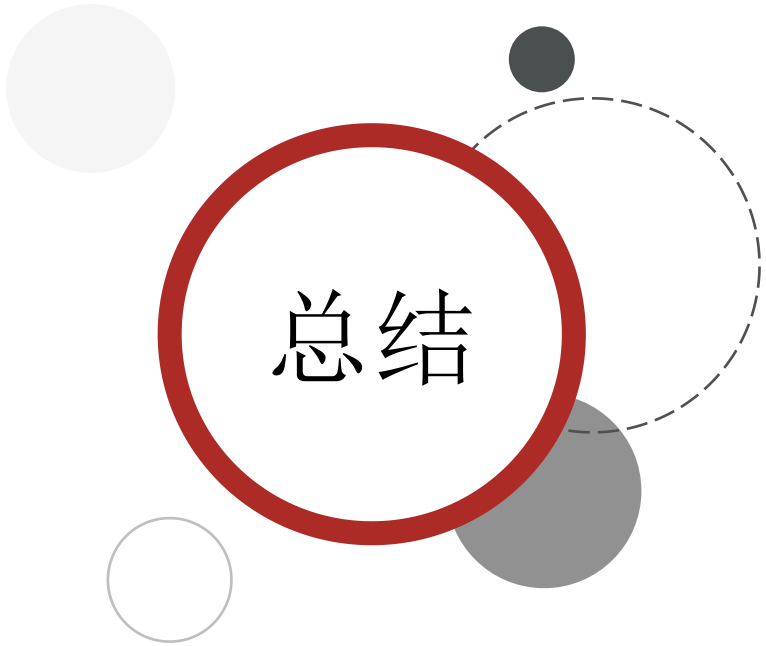
传进的所有参数都会被args变量收集，它会根据传进参数的位置合并为一个元组(tuple)，args是元组类型，这就是位置传递

关键字传递

```
def user_info(**kwargs):
    print(kwargs)



# {'name': 'TOM', 'age': 18, 'id': 110}
user_info(name='TOM', age=18, id=110)
```

注意：

参数是“键=值”形式的形式的情况下，所有的“键=值”都会被kwargs接受，同时会根据“键=值”组成字典.



1. 掌握位置参数

·根据参数位置来传递参数

2. 掌握关键字参数

·通过“键=值”形式传递参数，可以不限参数顺序
·可以和位置参数混用，位置参数需在前

3. 掌握缺省参数

·不传递参数值时会使用默认的参数值
·默认值的参数必须定义在最后

4. 掌握不定长参数

· 位置不定长传递以*号标记一个形式参数，以元组的形式接受参数，形式参数一般命名为args
· 关键字不定长传递以**号标记一个形式参数，以字典的形式接受参数，形式参数一般命名为kwargs

目录

Contents

学习目标

Learning Objectives

1. 掌握函数作为参数传递

函数作为参数传递

在前面的函数学习中，我们一直使用的函数，都是接受数据作为参数传入：

- 数字
- 字符串
- 字典、列表、元组等

其实，我们学习的函数本身，也可以作为参数传入另一个函数内。

函数作为参数传递

如下代码：

```python
def test_func(compute):
    result = compute(1, 2)
    print(result)

def compute(x, y):
    return x + y

test_func(compute)      # 结果：3
```

```python
def test_func(compute):
    result = compute(1, 2)
    print(result)

def compute(x, y):
    return x * y

test_func(compute)      # 结果：2
```

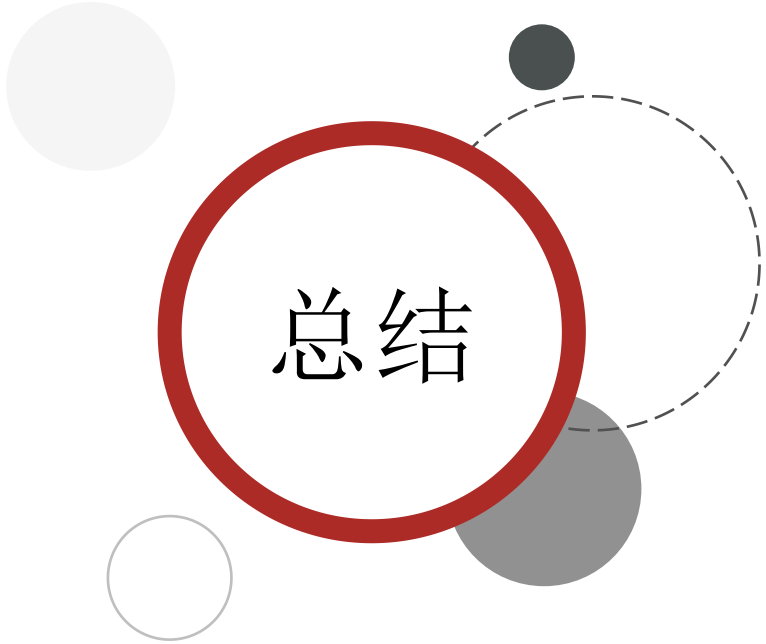
```python
def test_func(compute):
    result = compute(1, 2)
    print(result)

def compute(x, y):
    return x - y

test_func(compute)      # 结果：-1
```

函数compute，作为参数，传入了test_func函数中使用。

- test_func需要一个函数作为参数传入，这个函数需要接收2个数字进行计算，计算逻辑由这个被传入函数决定
- compute函数接收2个数字对其进行计算，compute函数作为参数，传递给了test_func函数使用
- 最终，在test_func函数内部，由传入的compute函数，完成了对数字的计算操作

所以，这是一种，计算逻辑的传递，而非数据的传递。

就像上述代码那样，不仅仅是相加，相见、相除、等任何逻辑都可以自行定义并作为函数传入。



1. 函数本身是可以作为参数，传入另一个函数中进行使用的。

2. 将函数传入的作用在于：传入计算逻辑，而非传入数据。

目录

Contents

学习目标

Learning Objectives

1. 掌握lambda匿名函数的语法

lambda匿名函数

函数的定义中

- def关键字，可以定义带有名称的函数
- lambda关键字，可以定义匿名函数（无名称）

有名称的函数，可以基于名称重复使用。

无名称的匿名函数，只可临时使用一次。

匿名函数定义语法：

```
lambda 传入参数： 函数体(一行代码)
```

- lambda 是关键字，表示定义匿名函数
- 传入参数表示匿名函数的形式参数，如：x，y 表示接收2个形式参数
- 函数体，就是函数的执行逻辑，要注意：只能写一行，无法写多行代码

lambda匿名函数

如下图代码，我们可以：

- 通过def关键字，定义一个函数，并传入，如下图：

```
def test_func(compute):
    result = compute(1, 2)
    print(result)

def compute(x, y):
    return x + y

test_func(compute)      # 结果：3
```

- 也可以通过lambda关键字，传入一个一次性使用的lambda匿名函数

```
def test_func(compute):
    result = compute(1, 2)
    print(result)

test_func(lambda x, y: x + y)       # 结果：3
```

使用def和使用lambda，定义的函数功能完全一致，只是lambda关键字定义的函数是匿名的，无法二次使用

总结

1. 匿名函数使用lambda关键字进行定义

2. 定义语法：

```
lambda 传入参数: 函数体(一行代码)
```

3. 注意事项：

- 匿名函数用于临时构建一个函数，只用一次的场景
- 匿名函数的定义中，函数体只能写一行代码，如果函数体要写多行代码，不可用lambda匿名函数，应使用def定义带名函数

传智教育旗下高端IT教育品牌