函数

软件测试Python课程V5.0

学习目标

Learning Objectives

1. 掌握函数的定义及调用方法
2. 掌握模块和包的用法
3. 理解变量的引用
4. 理解可变和不可变类型
5. 掌握局部变量和全局变量
6. 掌握缺省参数、多值参数和匿名函数

目录

Contents

函数

01

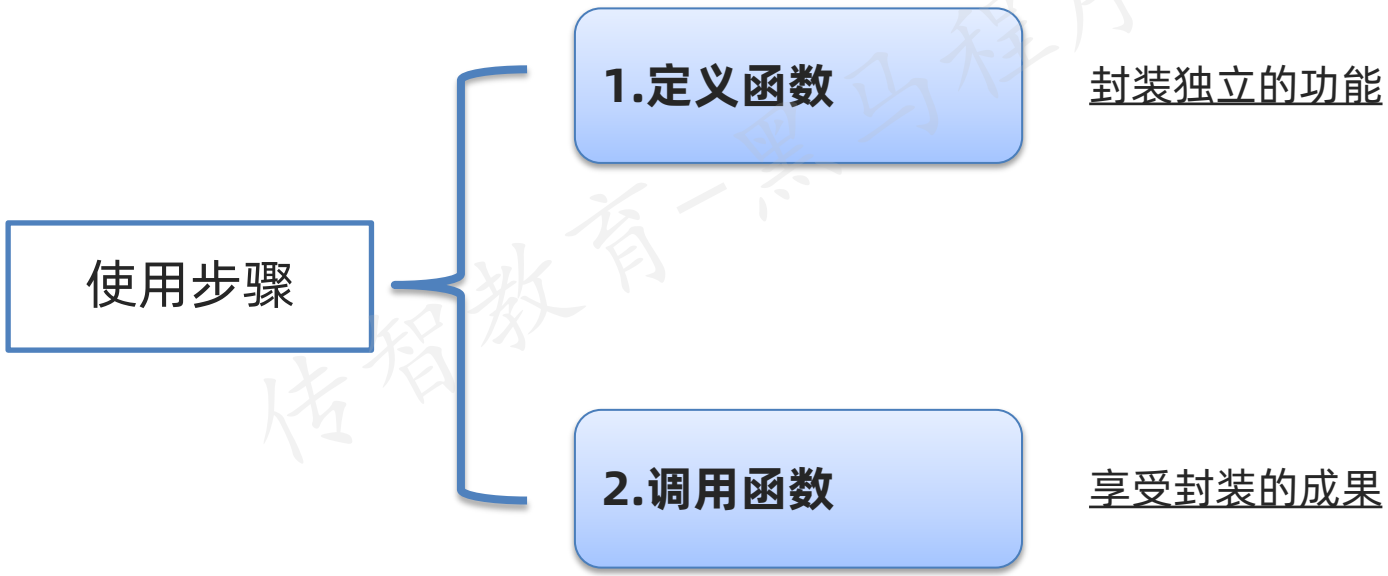
- 函数基本使用
- 函数参数
- 函数返回值
- 函数的嵌套调用

学习目标

1. 掌握如何定义函数
2. 掌握如何调用函数

函数介绍

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



作用	对具备相同逻辑的代码进行封装，提高代码的编写效率，实现对代码的重用

定义函数

语法格式

```
def 函数名():
    函数封装的代码
    ......
```

说明:

1. def 是英文 define 的缩写
2. 函数名最好能够表达函数内部封装的代码的功能, 方便后续的获取调用[见名知意]
3. 函数名命名遵循标识符命名规则: 字母, 数字, 下划线, 不能以数字开头, 不用使用系统关键字

调用函数

语法格式

函数名()

说明:

1. 只定义函数, 不调用函数, 函数永远不会被执行
2. 不能将 函数调用 放在 函数定义 的上方，否则将会出现错误

案例

第一个函数演练

需求：

1. 编写一个打招呼 say_hello 的函数，封装三行打招呼的代码
2. 在函数下方调用打招呼的代码

```
# 定义函数
def say_hello():
    print("hello 1")
    print("hello 2")
    print("hello 3")

# 调用函数
say_hello()
```

函数的文档注释

语法格式

```
def func():
    """函数文档注释"""
    pass
```

说明：

- 在开发中，如果希望给函数添加注释，应该在 定义函数的下方，使用 连续的三对引号
- 在 连续的三对引号 之间编写对函数的说明文字
- 在 函数调用 位置，使用快捷键 CTRL + Q 可以查看函数的说明信息

函数的参数

演练需求：

1. 开发一个 sum_2_num 的函数
2. 函数能够实现 两个数字的求和 功能

```
def sum_2_num():
    num1 = 10
    num2 = 20
    result = num1 + num2
    print(f"{num1} + {num2} = {result}")


sum_2_num()
```


思考一下存在什么问题？

答：函数只能处理**固定数值**的相加

如何解决？

答：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！

函数参数的使用

语法格式

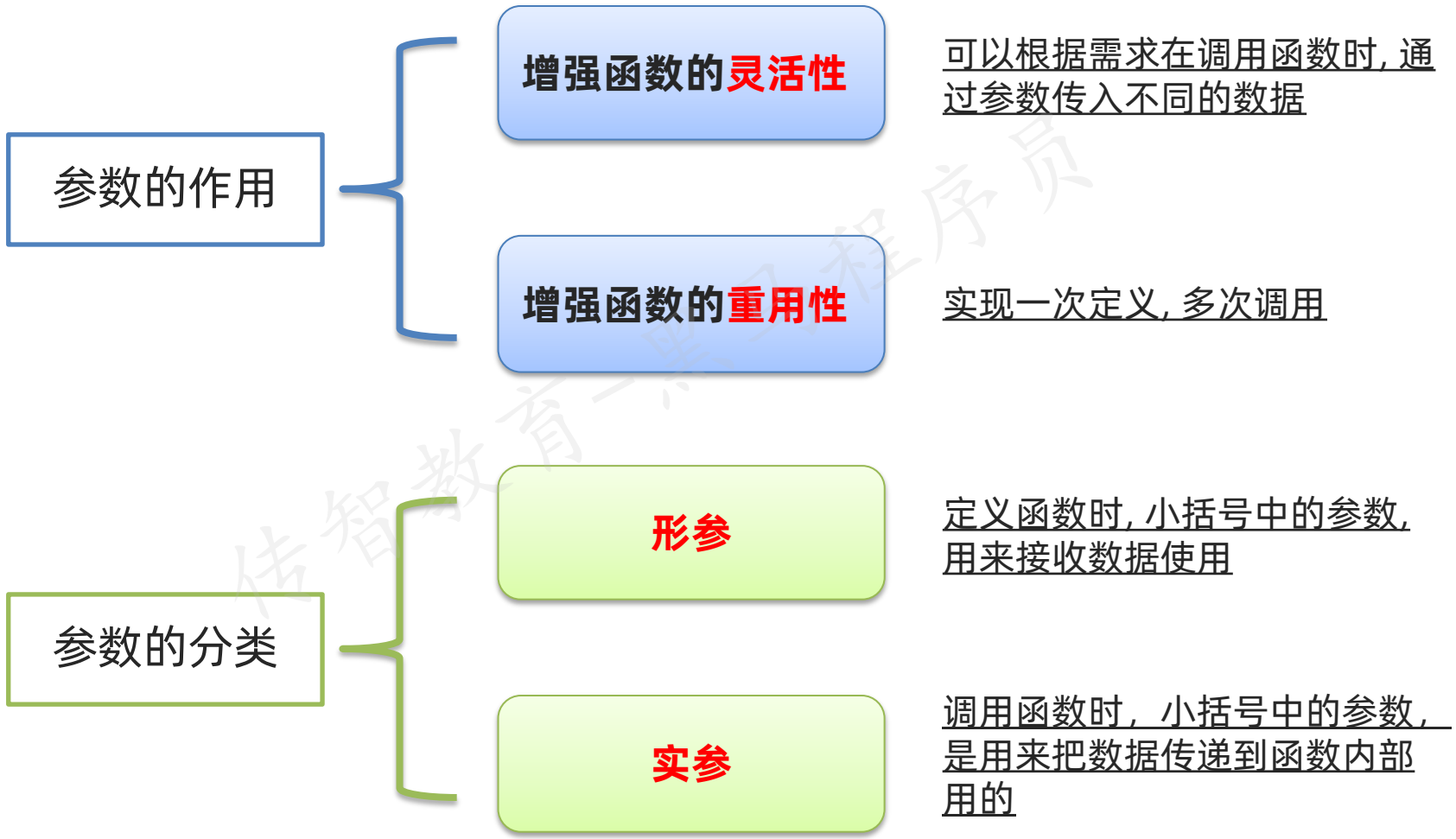
```
def 函数名(参数1, 参数2...):
    函数封装的代码，可以使用传递进来的参数
    pass
```

说明：
1. 在函数名的后面的小括号内部填写**参数**
2. 多个参数之间使用 , 分隔

```
def sum_2_num(num1, num2):
    result = num1 + num2
    print(f"{num1} + {num2} = {result}")

sum_2_num(10, 20)
```

函数参数的作用和分类



函数的返回值

语法格式

```
def 函数名(参数...):
    ...
    return xxx
```

说明	如果想在函数调用处获取函数定义内部的某个数据, 需要使用 return 关键字返回结果

```
def demo():
    name = "admin"
    print(name)  # admin


result = demo()
print(result)  # None
```

错误样例: 没有返回值的函数, 在函数调用处获取的结果都是 **None**

函数的返回值

注意事项

- return 关键字可以将函数定义内部的任意数据，返回给函数调用处
- return 关键字也可以单独使用，表示提前结束函数调用，但不返回任何数据

```
def demo():
    name = "admin"
    return name
    print(name)  # 注意：return 表示返回，后续的代码都不会被执行

result = demo()
print(result)  # admin
```

案例

获取最大值

需求：设计一个函数用于获取两个数中的较大数，数据来自于函数的参数

实现思路分析：

1. 定义一个函数，例如：get_max()
2. 为函数定义两个参数，用于接收两个数字
3. 使用判断语句分两种情况对两个数字的大小关系进行处理
4. 准备测试数据，调用函数

案例

函数返回值

```
def get_max(x, y):
    """获取两个数字中的较大值"""
    if x > y:
        return x
    else:
        return y

max_num = get_max(10, 20)
print(f"max_num={max_num}")
```

函数的嵌套调用

函数嵌套调用：一个函数里面又调用了另外一个函数

需求:

1. 定义名为test01的函数，打印当前函数的名称
2. 定义名为test02的函数，打印当前函数的名称，并调用test01函数
3. 在main代码块中调用test02函数

```
def test01():
    print("test01")

def test02():
    print("test02")
    test01()

if __name__ == '__main__':
    test02()
```

案例

函数嵌套调用

需求:

1. 定义名为 input_username 的函数, 获取用户输入的用户名
2. 定义名为 input_password 的函数, 获取用户输入的密码
3. 定义名为 login 的函数, 判断获取的用户名和密码信息
4. 要求当获取的用户名为:admin 并且密码为: 123456 时, 输出“登录成功!”，否则提示“用户名或密码错误!”

案例

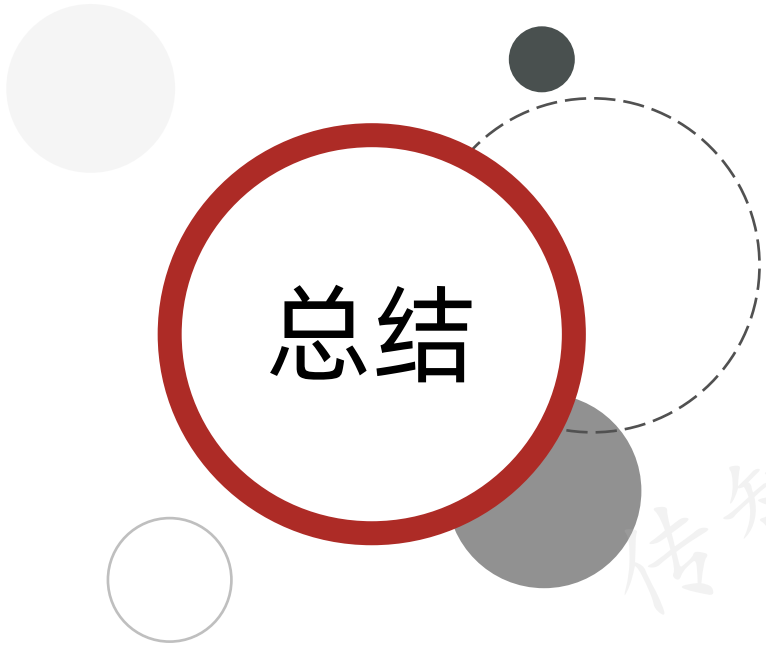
函数嵌套调用

```
def input_username():
    return input("请输入用户名：")

def input_password():
    return input("请输入密码：")

def login():
    username = input_username()
    password = input_password()
    if username == "admin" and password == "123456":
        return "登录成功！"
    else:
        return "用户名或密码错误!"

result = login()
print("result=", result)
```



1. 函数的作用?

2. 函数的使用步骤?

目录

Contents

02

模块和包

- 模块
- 包

学习目标

1. 掌握模块的用法
2. 掌握包的用法

模块的概念(Module)

- 每一个以 `.py` 结尾的 Python 代码文件都是一个模块
- `模块名` 同样也是一个 `标识符`，需要符合标识符的命名规则
- 在模块中定义的 `全局变量`、`函数`、`类` 都是提供给外界直接使用的 `工具`
- 模块就好比是工具包，要想使用这个工具包中的工具，就需要先 `导入模块`

模块的导入方式-import

```
import 模块名1, 模块名2

# 建议：在导入模块时，每个导入独占一行
import 模块名1
import 模块名2

# 如果模块的名字太长，可以使用 as 指定模块的名称，以方便在代码中的使用
import 模块名1 as 模块别名
```

注意：模块别名应该符合大驼峰命名法

使用方式	导入之后，通过 模块名. 使用模块提供的工具（全局变量、函数、类）

模块的导入方式-from...import

- 如果希望 从某一个模块 中，导入 部分 工具，就可以使用 from ... import 的方式
- import 模块名 是 一次性 把模块中 所有工具全部导入，并且通过 模块名/别名 访问

```
# 从 模块 导入 某一个工具
from 模块名1 import 工具名
```

使用方式	不需要通过 模块名.调用，可以直接使用模块提供的工具（全局变量、函数、类）

注意事项

1. 允许一行导入多个工具名: from 模块名 import 工具名1, 工具名2
2. 如果两个模块, 存在同名的工具, 那么后导入模块的工具, 会覆盖掉先导入的工具

模块的导入顺序

Python 解释器在导入模块时的顺序是：

1. 查找当前目录下指定模块名的文件，如果有就直接导入
2. 当前目录下如果没有，则再查找系统目录下的模块

注意：给模块命名时，务必不要和系统模块重名，否则会直接影响系统模块的导入！

__name__ 属性

```
# 根据 __name__ 判断是否执行下方代码
if __name__ == "__main__":
    pass
```

作用: 处于该条件下的代码, 只有是在当前模块内执行时, 才会被运行

原理: Python 内置属性 __name__ 的主要作用是记录一个字符串信息

- 当被其他模块导入后运行时, __name__ 属性值为模块名
- 当在模块内运行时, __name__ 属性值为 __main__

注意	一般情况下, 都需要将模块内的调试代码置于该条件下, 以防止被其他模块导入后, 调试代码也参与执行

包：(Package)是一个包含多个模块的特殊目录

- 目录下有一个特殊的文件 `__init__.py`
- 包名 的命名方式和 变量名 一致
- 作用：python文件较多时，**方便分目录管理维护**

提示：在 PyCharm 中支持直接创建包，工程根目录上鼠标右键 -> New -> Python Package

导包的常用方式

```
# 方式一
import 包名.模块名
包名.模块名.工具名

# 方式二
from 包名 import 模块名
模块名.工具名

# 方式三
from 包名.模块名 import 工具名
工具名
```

案例演练

包操作案例

需求：

1. 新建一个 hm_message 的 包
2. 在目录下，新建两个文件 send_message 和 receive_message
3. 在 send_message 文件中定义一个 send 函数
4. 在 receive_message 文件中定义一个 receive 函数
5. 在外部定义一个 test_message 文件，编写测试代码调用发消息和收消息的函数

案例演练

包操作案例

```
# /hm_message/send_message.py

def send():
    print("发消息")
```

```
# /hm_message/receive_message.py

def receive():
    print("收消息")
```

```
# /test_message.py

from hm_message import receive_message
from hm_message import send_message

send_message.send()
receive_message.receive()
```

注意：导入方式有多种实现

1. 什么是模块?

2. 什么是包?

3. 如何导入模块?

案例

模拟开发登录功能并测试

需求：

1. 模拟开发人员实现登录功能，定义一个函数能够接收用户输入的用户名、密码、验证码，根据不同的测试数据返回不同的验证结果
2. 已知正确的登录信息（用户名：admin、密码：123456、验证码：8888）
3. 要求1：创建login_service.py模块开发登录功能代码，并存放到service包中
4. 要求2：创建test_login.py模块开发测试代码，并存放到script包中
5. 要求3：至少测试3种测试场景

目录

Contents

变量进阶

03

- 引用的概念
- 可变和不可变类型
- 局部变量和全局变量

学习目标

1. 理解引用的概念
2. 掌握可变和不可变类型
3. 掌握局部变量和全局变量的用法

引用的概念

- **变量** 和 **数据** 是分开存储的
- **数据** 保存在内存中的一个位置
- **变量** 中保存着数据在内存中的地址
- 变量中记录数据的地址，就叫做 **引用**

提示：使用 id() 函数可以查看变量中保存数据所在的 **内存地址**

注意

当给一个变量重新赋值的时候，**本质上是修改了数据的引用**

- 变量 不再 对之前的数据引用
- 变量 改为 对新赋值的数据引用

变量的引用示例

```
# 定义一个变量a，并赋值为1
a = 1

# 将变量a赋值为2
a = 2

# 将变量a的值赋值给b
b = a
```

函数的参数和返回值的传递

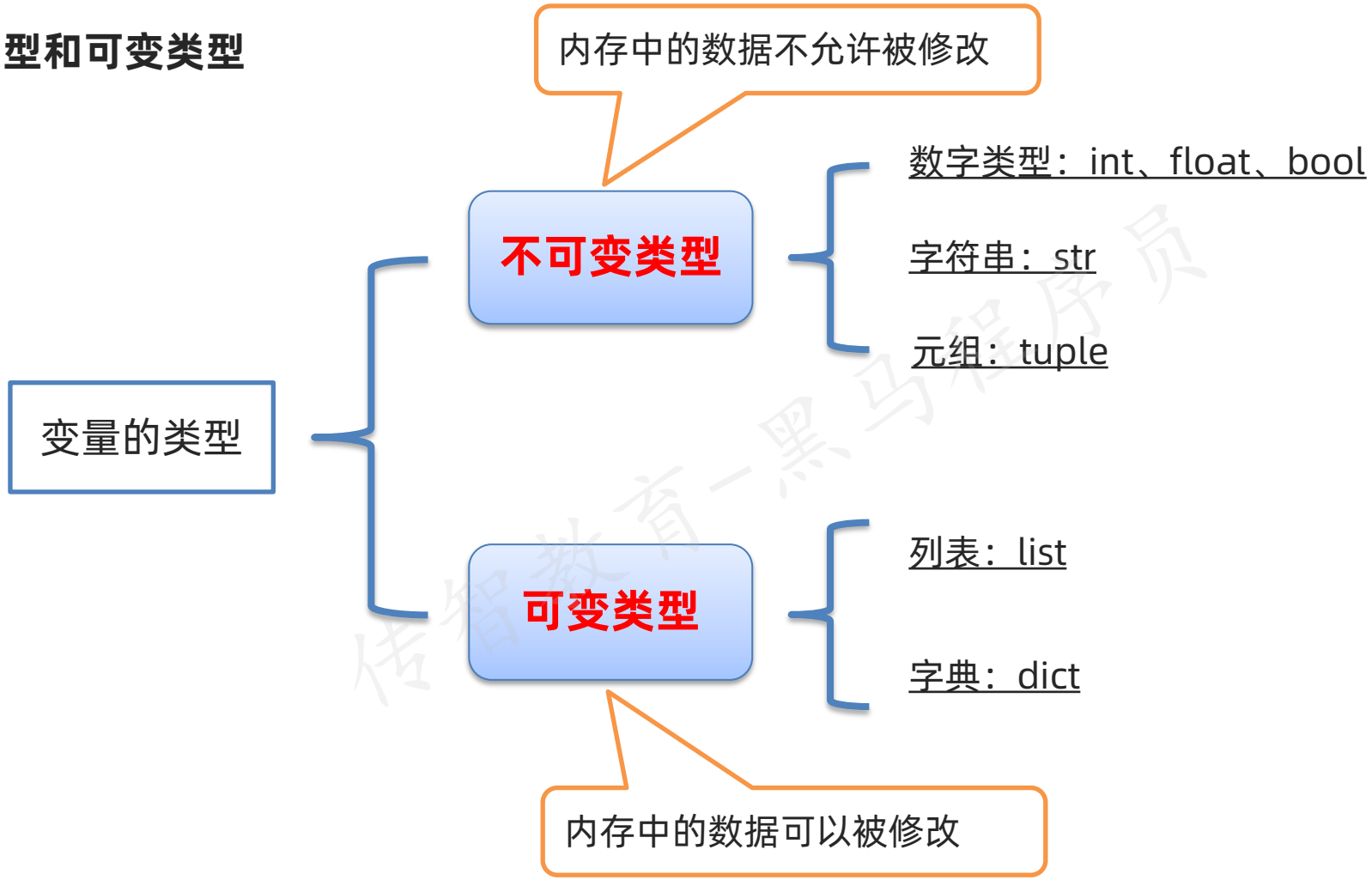
说明	在Python中，函数的 **参数** 和**返回值** 传递都是引用信息

```
def test(num):
    print(f"{num} 在函数内的内存地址是 {id(num)}")
    result = 100
    print(f"返回值 {result} 在内存中的地址是 {id(result)}")
    return result


a = 10
print(f"调用函数前 内存地址是 {id(a)}")
r = test(a)
print(f"调用函数后 实参内存地址是 {id(a)}")
print(f"调用函数后 返回值内存地址是 {id(r)}")
```

变量的进阶

不可变类型和可变类型



不可变类型和可变类型

不可变类型

```
a = 10
print(id(a))

a = "hello"
print(id(a))

a = "hello" + "666"
print(id(a))
```

可变类型

```
demo_list = [1, 2, 3]
print(f"定义列表后的内存地址 {id(demo_list)}")
demo_list.append(999)
print(f"修改数据后的内存地址 {id(demo_list)}")

demo_dict = {"name": "小明"}
print(f"定义字典后的内存地址 {id(demo_dict)}")
demo_dict["age"] = 18
demo_dict["name"] = "老王"
print(f"修改数据后的内存地址 {id(demo_dict)}")
```

局部变量和全局变量

注意 在开发中，大多不推荐使用全局变量，可变范围太大，会导致程序不好维护！

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，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

生命周期

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

案例

局部变量

需求:

1. 定义函数 login_front 实现前台系统登录
2. 定义函数 login_back 实现后台登录
3. 要求使用同样的变量名, 分别记录前后台登录信息中的账号和密码

案例

局部变量

```
def login_front():
    """前台系统登录"""
    username = "tom"
    password = "123"
    print(f"前台系统登录 username={username}, password={password}")

def login_back():
    """后台系统登录"""
    username = "admin"
    password = "666"
    print(f"后台系统登录 username={username}, password={password}")

if __name__ == '__main__':
    login_front()
    login_back()
```

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

```
num = 10  # 定义一个全局变量

def demo1():
    print("demo1.num=", num)

def demo2():
    print("demo2.num=", num)

if __name__ == '__main__':
    demo1()
    demo2()
```

函数中变量获取顺序：

1. 首先查找函数内部是否存在指定名称的局部变量，如果有直接使用
2. 内部如果没有，则查找函数外部是否存在指定名称的全局变量，如果有直接使用
3. 如果还没有，则程序报错

近水楼台先得月！

变量的进阶

在函数内部修改全局变量的值

```
num = 10

def demo1():
    # 只是定义了一个局部变量，不会修改到全局变量
    num = 100
    print("demo1.num=", num)

def demo2():
    print("demo2.num=", num)

demo1()
demo2()
```

提示：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在其他函数的上方

```
num = 10

def demo1():
    num = 100
    print("demo1.num=", num)

def demo2():
    global num  # 告诉解释器 num 是一个全局变量
    num = 200
    print("demo2.num=", num)

demo1()
demo2()
print("num=", num)
```

注意：在函数中需要修改全局变量，需要使用 **global** 进行声明

1. 理解引用的概念
2. 可变和不可变类型？
3. 局部变量和全局变量的用法？

目录

Contents

函数进阶

04

- 缺省参数
- 多值参数
- 匿名函数

学习目标

1. 掌握缺省参数的用法
2. 知道多值参数的用法
3. 掌握如何定义匿名函数

缺省参数

缺省参数：定义函数时，给某个参数指定一个默认值，具有默认值的参数

说明：
- 调用函数时，如果没有传入缺省参数的值，则使用默认值
- 将常见的值设置为参数的缺省值，从而**简化函数的调用**

```
num_list = [6, 3, 9]

# 默认就是升序排序，因为这种应用需求更多
num_list.sort()

# 只有当需要降序排序时，才需要传递 `reverse` 参数
num_list.sort(reverse=True)
```

缺省参数

语法格式

```
def print_info(name, gender="男生"):
    print(f"{name} 是 {gender}")
```

注意

1. 必须保证 缺省参数 在参数列表末尾
2. 在 调用函数时，如果有 多个缺省参数，需要指定参数名，这样解释器才能够知道参数的对应关系

缺省参数

```
def print_info(name, title="", gender="男生"):
    """
    :param title: 职位
    :param name: 班上同学的姓名
    :param gender: 性别
    """
    print(f"{title}{name} 是 {gender}")


# 提示：在指定缺省参数的默认值时，应该使用最常见的值作为默认值！
print_info("小明")
print_info("老王", title="班长")
print_info("小美", gender="女生")
```

多值参数

说明	如果一个函数能够处理的参数个数是不确定的, 可以考虑使用多值参数

语法格式

```
def demo(num, *args):
    print(num)
    print(args)

demo(1, 2, 3, 4, 5)
```

提示：*args 用于接收元组类型数据, args为习惯命名,可以自定义

案例

多值参数

需求:

1. 定义一个函数 sum_numbers，可以接收的 任意多个整数

2. 功能要求：将传递的 所有数字累加 并且返回累加结果

```
def sum_numbers(*args):
    num = 0
    # 遍历 args 元组求和
    for n in args:
        num += n
    return num

print(sum_numbers(1, 2, 3))
```

元组的拆包

思考：在调用带有多值参数的函数时，如何将一个**元组变量**直接传递给args参数？

回答：可以使用 **拆包**，简化参数的传递

元组拆包的方式：在元组变量前，增加一个 *

案例

元组拆包

需求:

1. 定义一个函数 sum_numbers，可以接收的 任意多个整数

2. 定义一个元组(1, 2, 3)，调用上面定义好的函数对元组中所有数据进行求和

案例

元组拆包

```
def sum_numbers(*args):
    num = 0
    for n in args:
        num += n
    return num


# 需要将一个元组变量传递给函数对应的参数
nums = (1, 2, 3)
result = sum_numbers(*nums)
print(f"result={result}")
```

匿名函数

lambda 参数1, 参数n : 表达式

- 用 lambda 关键字创建的简化型函数, 不同于使用 def 声明的函数, 匿名函数没有函数名称
- 匿名函数能接收任何数量的参数, 但只能返回一个表达式的值

```
sum = lambda arg1, arg2: arg1 + arg2

# 调用sum函数
print("result=", sum(10, 20))  # result= 30
print("result=", sum(20, 20))  # result= 40
```

案例

匿名函数

需求:

1. 已知用户信息数据如下:

```
user_list = [
    {'name': '张三', 'age': 22, 'title': '测试工程师'},
    {'name': '李四', 'age': 24, 'title': '开发工程师'},
    {'name': '王五', 'age': 21, 'title': '测试工程师'}
]
```

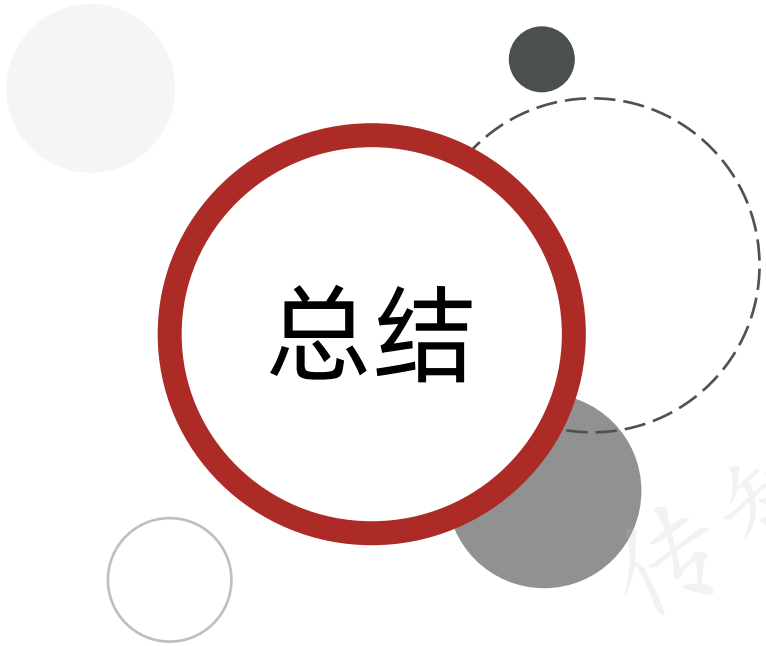
2. 要求按照年龄对用户信息数据进行排序

案例

匿名函数

```
user_list = [
    {'name': '张三', 'age': 22, 'title': '测试工程师'},
    {'name': '李四', 'age': 24, 'title': '开发工程师'},
    {'name': '王五', 'age': 21, 'title': '测试工程师'}
]


# 按照age进行排序
user_list.sort(key=lambda x: x["age"])
print(user_list)
```



1. 何为缺省参数?

2. 何为多值参数?

3. 匿名函数的定义格式?

案例

学生信息管理系统

需求：按照以下要求完成代码的编写

第一步：录入学生信息

1)提示用户在控制台输入3个学生的信息，学生信息包含姓名、年龄

2)要求：封装录入单个学生信息的函数，并返回学生的信息

第二步：展示学生列表信息

1)封装打印学生信息的函数，格式要求如右图：

```
-----学生列表信息-----
1        张三        19
2        李四        18
3        王五        22
----------------------
```

第三步：统计学生总数

1)封装获取学生总数的函数，并对该函数进行调用和数据打印

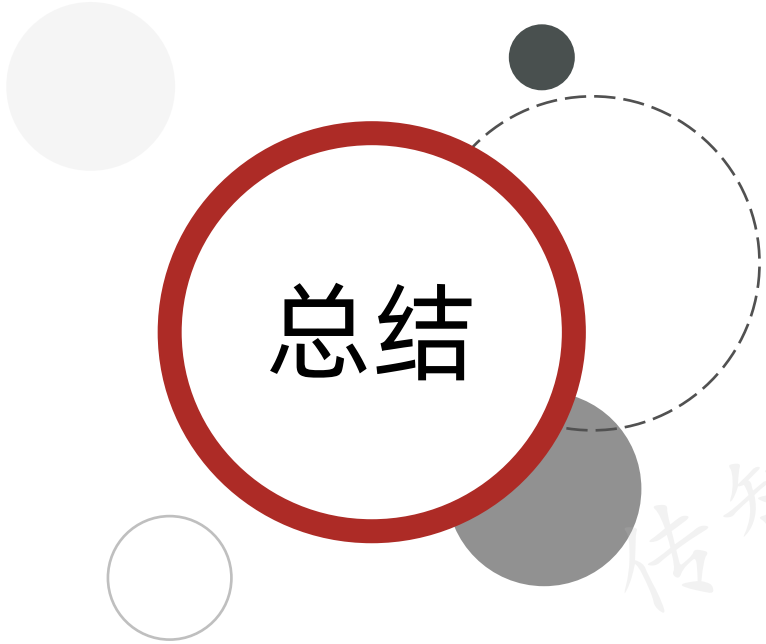
第四步：查询学生信息

1)封装根据学生姓名查询学生信息的函数

2)提示用户“请输入要查询的学生姓名：”

3)如果存在，直接在控制台打印学生信息，格式为：“姓名：张三，年龄：25”

4)如果不存在，直接在控制台打印“对不起，名字叫【张三】的学生不存在”



1. 掌握函数的定义及调用方法
2. 理解变量的引用
3. 理解可变和不可变类型
4. 掌握局部变量和全局变量
5. 掌握缺省参数、多值参数和匿名函数

传智教育旗下高端IT教育品牌

黑马程序员-软件测试