一、函数的定义和使用

1. 函数的基本概念

> **定义**：函数是一段封装好的、可重复使用的代码块，用于实现特定功能。
>
> **作用**：
>
> - 减少代码重复（复用性）。
> - 使程序结构更清晰（模块化）。
> - 便于维护和扩展。

> **生活类比**：像洗衣机的“洗涤”功能——只需设定程序（调用函数），无需关心内部如何转动（函数实现细节）。

2. 函数的定义与调用

(1) 定义函数

> 语法

```
def 函数名(参数列表):
    """函数文档字符串（可选，用于说明功能）"""
    函数体（实现功能的代码）
    return 返回值（可选）
```

> - `def`：定义函数的关键字。
> - 函数名：遵循变量命名规则（字母、数字、下划线，不能以数字开头）。
> - 参数列表：函数接收的输入（可选，无参数则留空）。
> - 函数体：缩进的代码块（4个空格）。
> - `return`：返回函数结果（可选，无 `return` 则默认返回 `None`）。

> 案例 1：定义一个简单函数

```
def say_hello():
    """打印欢迎信息"""
    print("Hello, Python!")

# 调用函数（执行函数体）
say_hello()  # 输出：Hello, Python!
```

(2) 函数的调用

> 通过 `函数名(参数)` 调用，每次调用都会执行函数体中的代码。

```python
# 定义计算加法的函数
def add(a, b):
    """计算两个数的和"""
    result = a + b
    return result  # 返回计算结果

# 调用函数并接收返回值
sum1 = add(3, 5)
sum2 = add(10, 20)
print(sum1)  # 输出：8
print(sum2)  # 输出：30
```

3. 函数的参数

> 参数是函数接收的输入，使函数更灵活（同一函数可处理不同数据）。

(1) 位置参数

> 按参数定义的顺序传递值，必须一一对应。

```python
def introduce(name, age):
    """介绍姓名和年龄"""
    print(f"我叫{name}，今年{age}岁")

# 按位置传递参数
introduce("张三", 20)  # 输出：我叫张三，今年20岁
```

(2) 关键字参数

> 传递参数时指定参数名，可打乱顺序（更清晰）。

```python
# 用关键字参数调用
introduce(age=22, name="李四")  # 输出：我叫李四，今年22岁
```

(3) 默认参数

> 定义函数时给参数设置默认值，调用时可省略该参数。

```python
def greet(name, greeting="你好"):
    """打招呼，默认用"你好" """
    print(f"{greeting}，{name}！")

greet("王五")  # 省略默认参数，输出：你好，王五！
greet("赵六", "早上好")  # 覆盖默认值，输出：早上好，赵六！
```

> 注意：默认参数必须放在位置参数后面（如 `def func(a, b=1)` 正确，`def func(b=1, a)` 错误）。

(4) 不定长参数

> 处理参数数量不确定的情况，有两种形式：

- `*args`：接收多个位置参数，打包为元组。
- `**kwargs`：接收多个关键字参数，打包为字典。

```
# *args 示例：计算任意个数的和
def sum_all(*args):
    print("参数列表：", args)  # args 是元组
    return sum(args)

print(sum_all(1, 2, 3))  # 输出：6
print(sum_all(10, 20, 30, 40))  # 输出：100

# **kwargs 示例：打印用户信息
def print_info(** kwargs):
    print("用户信息：", kwargs)  # kwargs 是字典
    for key, value in kwargs.items():
        print(f"{key}: {value}")

print_info(name="张三", age=20, gender="男")
# 输出：
# 用户信息： {'name': '张三', 'age': 20, 'gender': '男'}
# name: 张三
# age: 20
# gender: 男
```

4. 函数的返回值

用 `return` 语句将结果返回给调用者，可返回任意类型（数字、字符串、列表等），也可返回多个值（打包为元组）。

案例 2：返回多个值

```
def get_user():
    """返回用户信息（姓名、年龄）"""
    name = "张三"
    age = 20
    return name, age  # 多个值自动打包为元组

# 接收多个返回值
user_name, user_age = get_user()
print(user_name, user_age)  # 输出：张三 20
```

案例 3：提前退出函数

`return` 会立即终止函数，其后的代码不再执行。

```
def check_positive(num):
    """检查数字是否为正数"""
    if num <= 0:
        return False  # 非正数直接返回
    return True

print(check_positive(5))  # 输出：True
print(check_positive(-3))  # 输出：False
```

5. 函数的作用域

> 变量的生效范围，分为：
>
> - **局部变量**：在函数内部定义，仅在函数内部有效。
> - **全局变量**：在函数外部定义，整个程序中有效（函数内部可访问，修改需用 `global` 声明）。

```
# 全局变量
global_var = "我是全局变量"

def test_scope():
    # 局部变量
    local_var = "我是局部变量"
    print("函数内访问全局变量：", global_var)
    print("函数内访问局部变量：", local_var)

test_scope()
print("函数外访问全局变量：", global_var)
# print("函数外访问局部变量：", local_var)  # 报错：局部变量在外部不可访问

# 修改全局变量（需用 global 声明）
def modify_global():
    global global_var  # 声明使用全局变量
    global_var = "全局变量被修改了"

modify_global()
print(global_var)  # 输出：全局变量被修改了
```

6. 函数的嵌套调用

> 在一个函数内部调用另一个函数，实现复杂功能的拆分。

> 案例 4：嵌套调用计算圆的面积

```
import math

def circle_radius(diameter):
    """根据直径计算半径"""
    return diameter / 2

def circle_area(diameter):
    """根据直径计算圆的面积（调用 circle_radius 函数）"""
    r = circle_radius(diameter)  # 嵌套调用
    area = math.pi * r **2
    return area

print(circle_area(10))  # 输出：78.53981633974483（直径10的圆面积）
```

7. 综合案例

> 案例 5：学生成绩管理系统（函数版）

> 需求：用函数实现成绩录入、平均分计算、等级评定功能。

```python
def input_scores():
    """录入学生成绩，返回成绩列表"""
    scores = []
    while True:
        score = input("请输入成绩（输入q结束）：")
        if score.lower() == "q":
            break
        scores.append(float(score))
    return scores

def calculate_average(scores):
    """计算平均分"""
    if not scores:  # 处理空列表
        return 0
    return sum(scores) / len(scores)

def get_grade(score):
    """根据分数返回等级"""
    if score >= 90:
        return "A"
    elif score >= 80:
        return "B"
    elif score >= 60:
        return "C"
    else:
        return "D"

# 主程序
scores = input_scores()
if scores:
    avg = calculate_average(scores)
    print(f"平均分：{avg:.1f}")
    print("各分数等级：")
    for s in scores:
        print(f"{s} → {get_grade(s)}")
```

8. 课堂练习

> 1. 编写函数 `is_prime(n)`，判断一个数是否为质数（只能被 1 和自身整除的大于 1 的整数）。
> 2. 定义函数 `reverse_string(s)`，返回字符串 `s` 的反转结果（如输入 "abc" 返回 "cba"）。
> 3. 用函数实现斐波那契数列：定义 `fib(n)`，返回前 `n` 项斐波那契数列（1, 1, 2, 3, 5...）

9. 常见错误与注意事项

1. **参数数量不匹配**：调用函数时传递的参数数量与定义不一致。
2. **默认参数位置错误**：默认参数必须放在位置参数后面（如 `def func(a=1, b)` 错误）。

3. **局部变量与全局变量混淆**：在函数内部修改全局变量未用 `global` 声明，导致创建了同名局部变量。
4. **忘记返回值**：需要返回结果却遗漏 `return` 语句，函数默认返回 `None`。