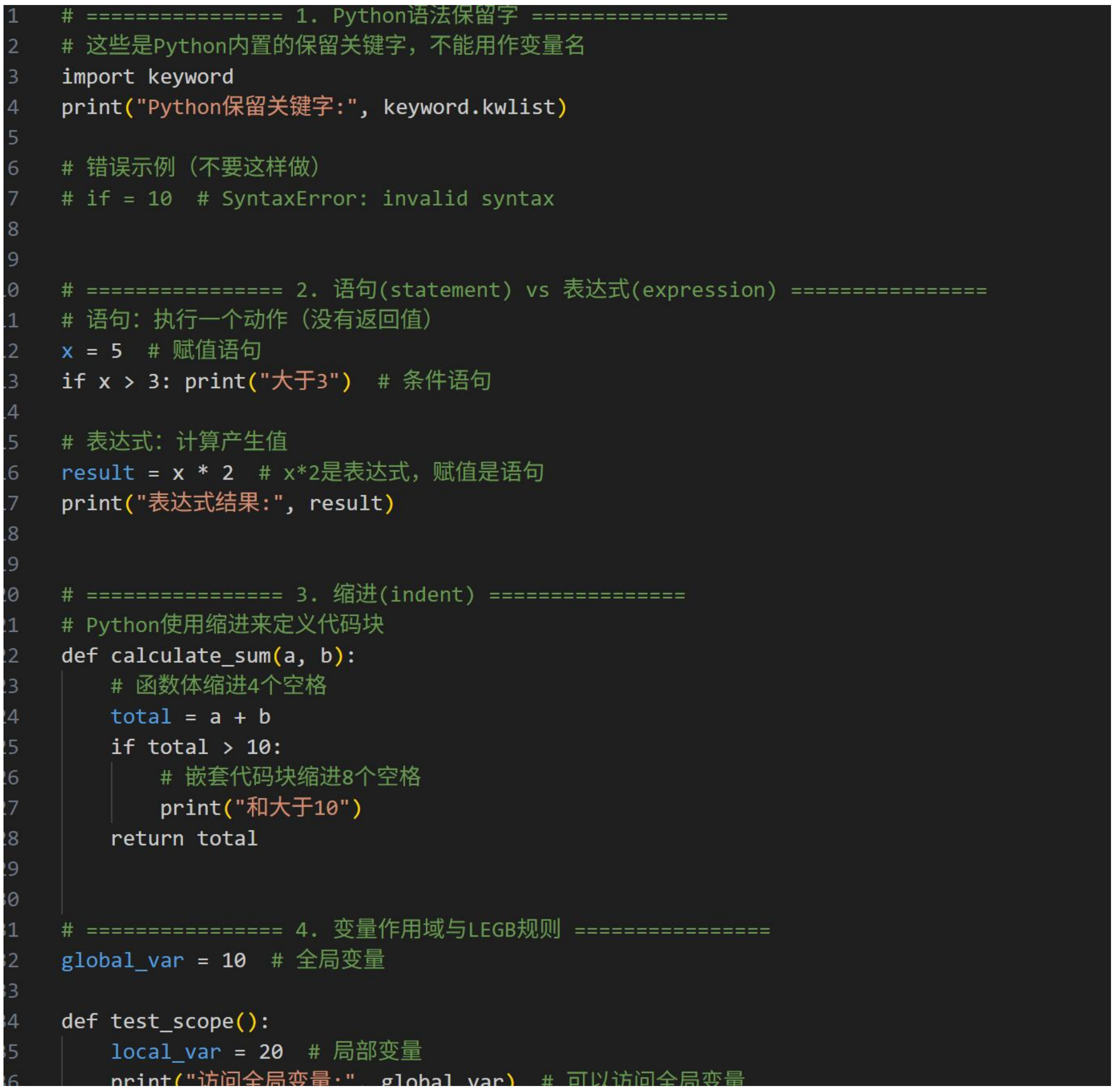
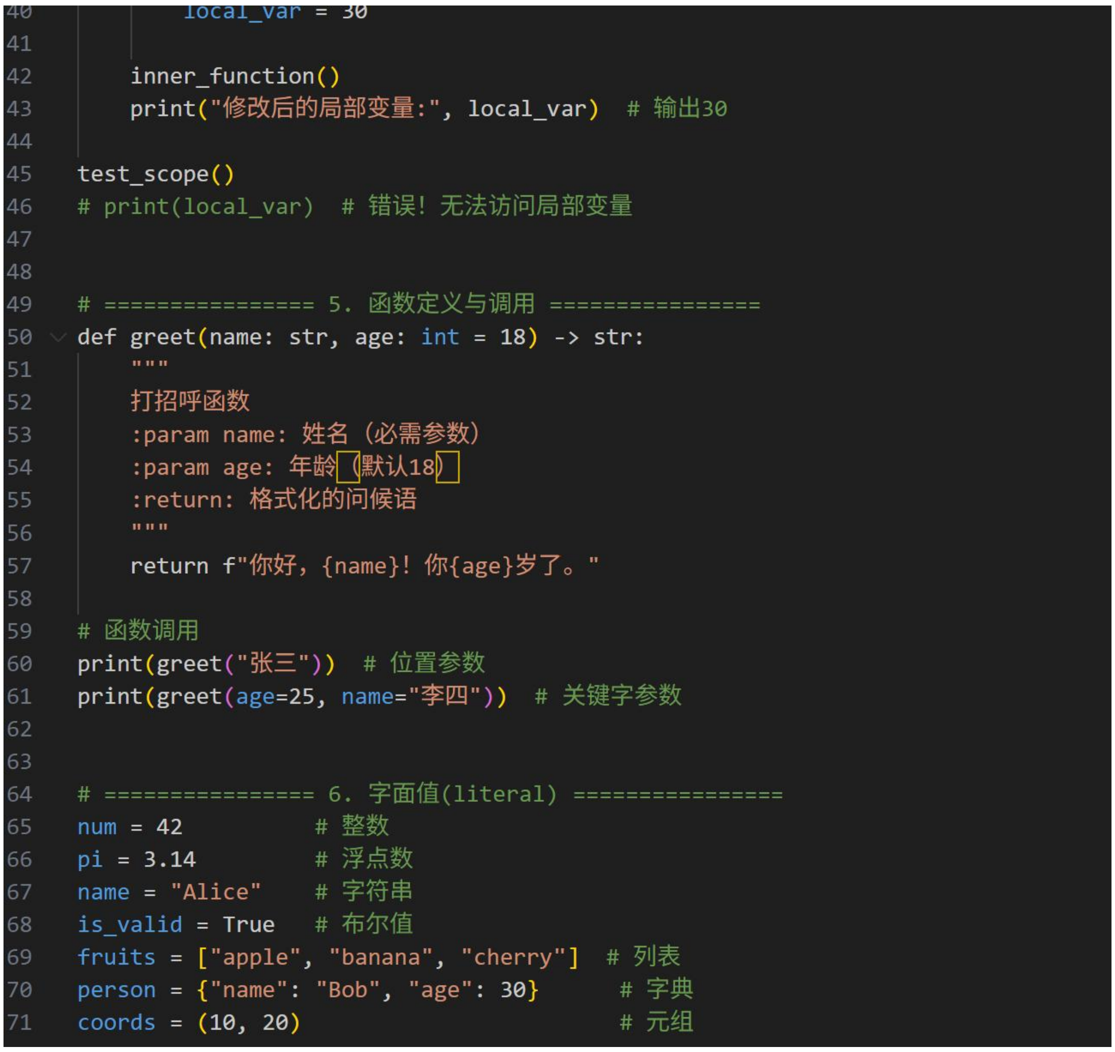
Week04 学习报告

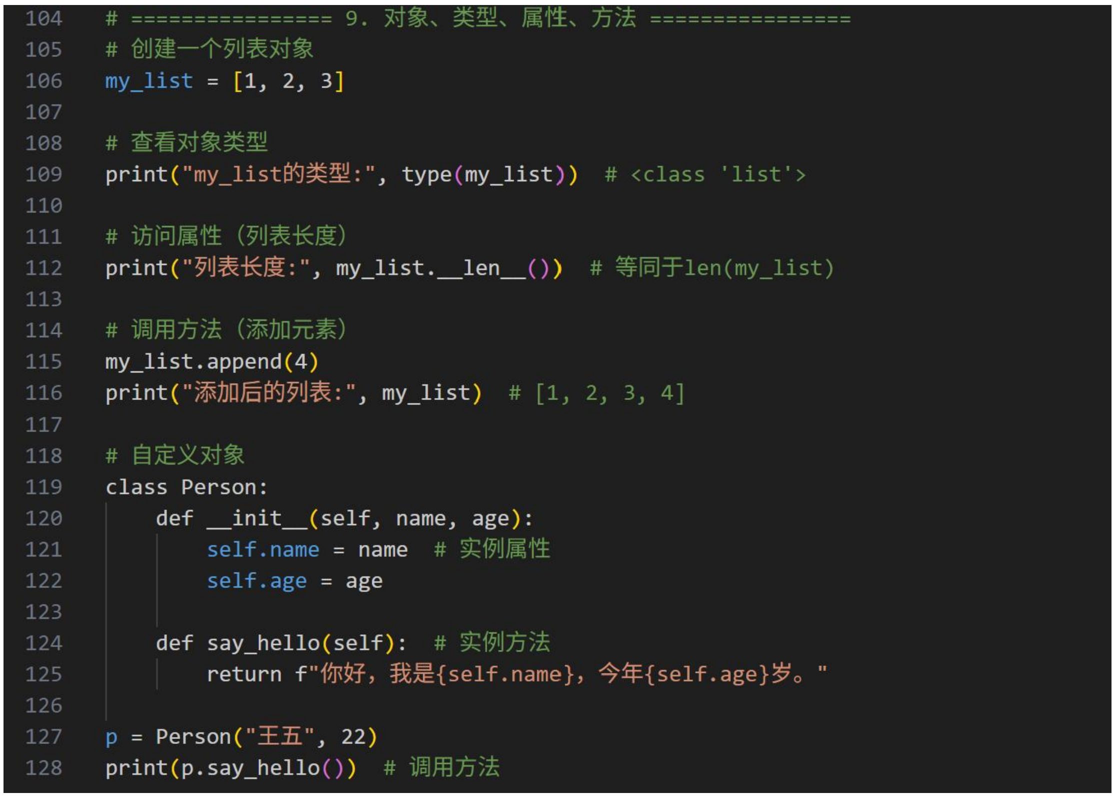
Python 语法保留字（reserved key words）▪ 语句（statement）和表达式（expression）▪ 缩进（indent）▪ 局部变量（local variable）、全局变量（global variable）、LEGB 规则 ▪ 函数（function）的定义（define）和调用（call）▪ 字面值（literal）（字符串（str）、整数（int）、列表（list）、字典（dict）、元组（tuple））▪ 运算符（operator）▪ 形参（parameter）、实参（argument）、返回值（return value）▪ 对象（object）、类型（type）、属性（attribute）、方法（method）



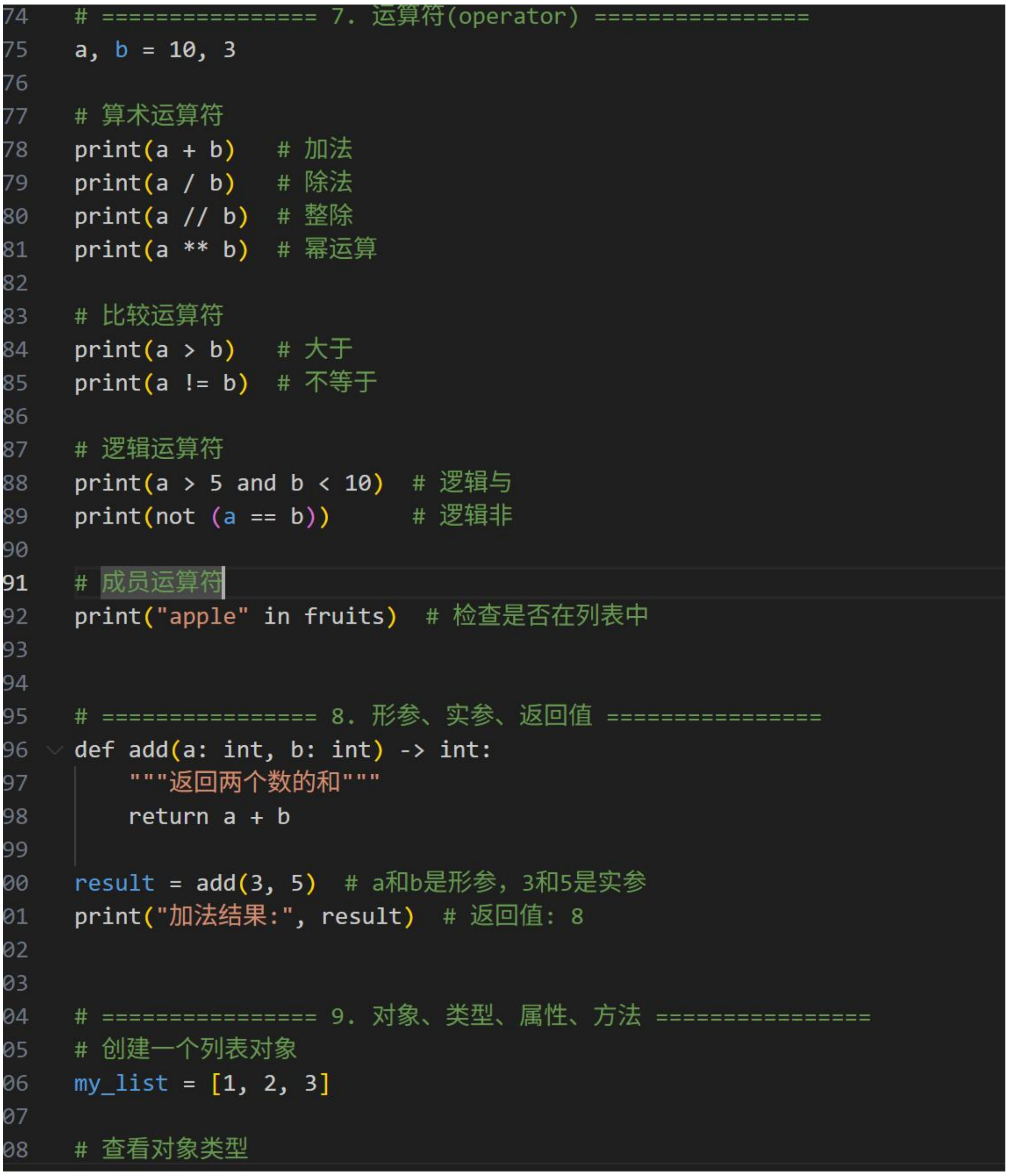
```python
# ================ 1. Python语法保留字 ================
# 这些是Python内置的保留关键字，不能用作变量名
import keyword
print("Python保留关键字:", keyword.kwlist)

# 错误示例（不要这样做）
# if = 10  # SyntaxError: invalid syntax


# ================ 2. 语句(statement) vs 表达式(expression) ================
# 语句：执行一个动作（没有返回值）
x = 5  # 赋值语句
if x > 3: print("大于3")  # 条件语句

# 表达式：计算产生值
result = x * 2  # x*2是表达式，赋值是语句
print("表达式结果:", result)


# ================ 3. 缩进(indent) ================
# Python使用缩进来定义代码块
def calculate_sum(a, b):
    # 函数体缩进4个空格
    total = a + b
    if total > 10:
        # 嵌套代码块缩进8个空格
        print("和大于10")
    return total


# ================ 4. 变量作用域与LEGB规则 ================
global_var = 10  # 全局变量

def test_scope():
    local_var = 20  # 局部变量
    print("访问全局变量:", global_var)  # 可以访问全局变量
```



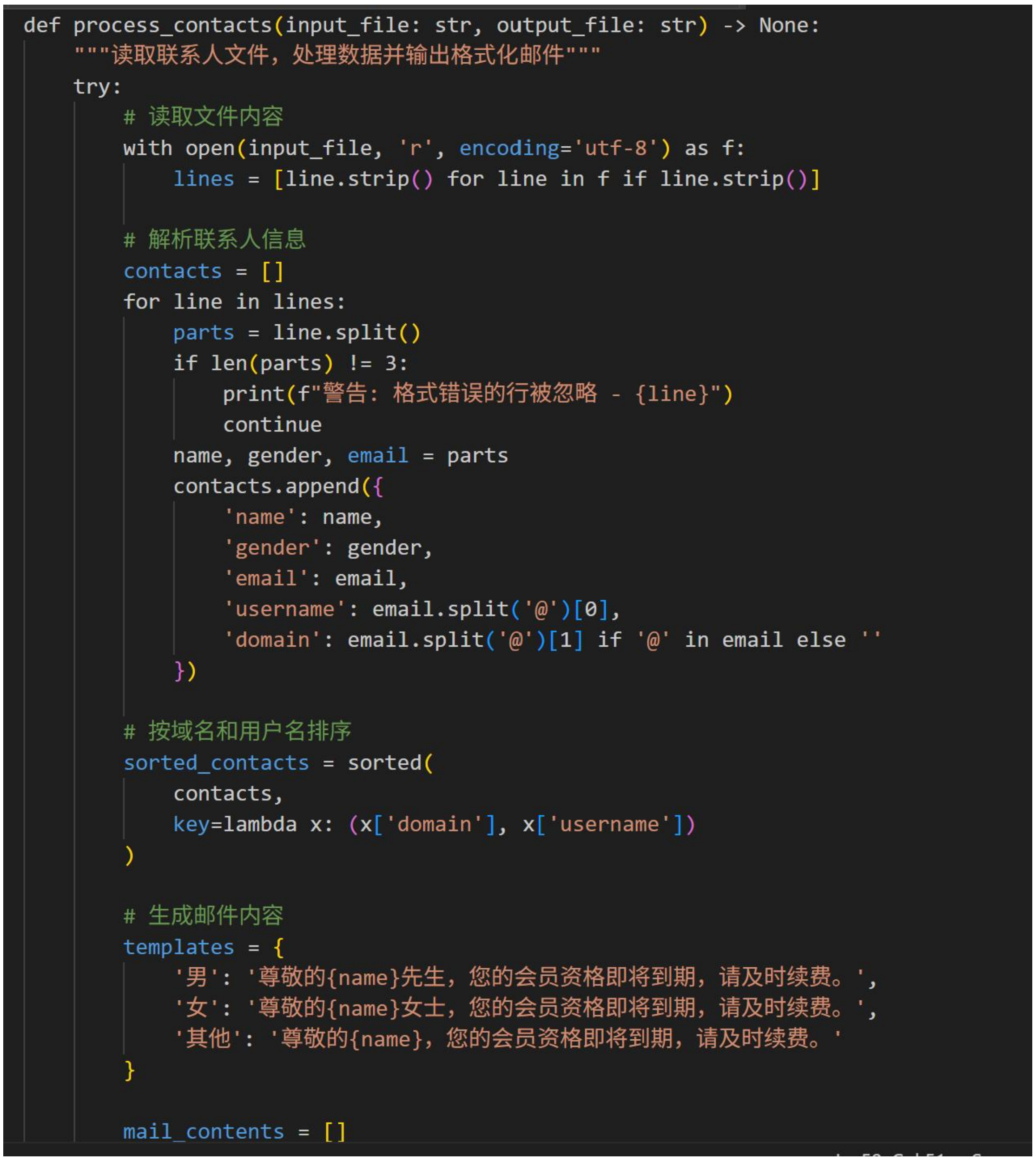
```python
        local_var = 30

    inner_function()
    print("修改后的局部变量:", local_var)  # 输出30

test_scope()
# print(local_var)  # 错误！无法访问局部变量


# ================ 5. 函数定义与调用 ================
def greet(name: str, age: int = 18) -> str:
    """
    打招呼函数
    :param name: 姓名（必需参数）
    :param age: 年龄（默认18）
    :return: 格式化的问候语
    """
    return f"你好，{name}！你{age}岁了。"

# 函数调用
print(greet("张三"))  # 位置参数
print(greet(age=25, name="李四"))  # 关键字参数


# ================ 6. 字面值(literal) ================
num = 42          # 整数
pi = 3.14         # 浮点数
name = "Alice"    # 字符串
is_valid = True   # 布尔值
fruits = ["apple", "banana", "cherry"]  # 列表
person = {"name": "Bob", "age": 30}      # 字典
coords = (10, 20)                        # 元组
```



```python
# ================ 9. 对象、类型、属性、方法 ================
# 创建一个列表对象
my_list = [1, 2, 3]

# 查看对象类型
print("my_list的类型:", type(my_list))  # <class 'list'>

# 访问属性（列表长度）
print("列表长度:", my_list.__len__())  # 等同于len(my_list)

# 调用方法（添加元素）
my_list.append(4)
print("添加后的列表:", my_list)  # [1, 2, 3, 4]

# 自定义对象
class Person:
    def __init__(self, name, age):
        self.name = name  # 实例属性
        self.age = age

    def say_hello(self):  # 实例方法
        return f"你好，我是{self.name}，今年{self.age}岁。"

p = Person("王五", 22)
print(p.say_hello())  # 调用方法
```



```python
# ================ 7. 运算符(operator) ================
a, b = 10, 3

# 算术运算符
print(a + b)   # 加法
print(a / b)   # 除法
print(a // b)  # 整除
print(a ** b)  # 幂运算

# 比较运算符
print(a > b)   # 大于
print(a != b)  # 不等于

# 逻辑运算符
print(a > 5 and b < 10)  # 逻辑与
print(not (a == b))      # 逻辑非

# 成员运算符
print("apple" in fruits)  # 检查是否在列表中


# ================ 8. 形参、实参、返回值 ================
def add(a: int, b: int) -> int:
    """返回两个数的和"""
    return a + b

result = add(3, 5)  # a和b是形参，3和5是实参
print("加法结果:", result)  # 返回值: 8


# ================ 9. 对象、类型、属性、方法 ================
# 创建一个列表对象
my_list = [1, 2, 3]

# 查看对象类型
```



```python
def process_contacts(input_file: str, output_file: str) -> None:
    """读取联系人文件，处理数据并输出格式化邮件"""
    try:
        # 读取文件内容
        with open(input_file, 'r', encoding='utf-8') as f:
            lines = [line.strip() for line in f if line.strip()]

        # 解析联系人信息
        contacts = []
        for line in lines:
            parts = line.split()
            if len(parts) != 3:
                print(f"警告: 格式错误的行被忽略 - {line}")
                continue
            name, gender, email = parts
            contacts.append({
                'name': name,
                'gender': gender,
                'email': email,
                'username': email.split('@')[0],
                'domain': email.split('@')[1] if '@' in email else ''
            })

        # 按域名和用户名排序
        sorted_contacts = sorted(
            contacts,
            key=lambda x: (x['domain'], x['username'])
        )

        # 生成邮件内容
        templates = {
            '男': '尊敬的{name}先生，您的会员资格即将到期，请及时续费。',
            '女': '尊敬的{name}女士，您的会员资格即将到期，请及时续费。',
            '其他': '尊敬的{name}，您的会员资格即将到期，请及时续费。'
        }

        mail_contents = []
```

用于读取联系人信息并生成格式化的邮件通知

代码说明：

文件读取：使用 **UTF-8** 编码打开文件，自动过滤空行

数据解析：

处理每行数据，提取姓名、性别和邮箱
自动识别邮箱中的用户名和域名部分
对格式错误的数据提供警告但不中断处理

排序逻辑：

先按域名排序（**126.com** 在前，**163.com** 在后）

再按用户名排序（字母序）

邮件生成：

根据性别选择不同的称呼模板

支持扩展其他性别选项（如 "其他"）

自动处理邮箱格式（包含尖括号）
错误处理：
文件不存在时给出明确提示

资料阅读

1. Variables

You can think about variables as words that store a value. Simple as that.

In Python, it is really easy to define a variable and set a value to it. Imagine you want to store number 1 in a variable called “one.” Let’s do it:

```
one = 1
```

How simple was that? You just assigned the value 1 to the variable “one.”

```
two = 2
some_number = 10000
```

Besides integers, we can also use booleans (True / False), strings, float, and so many other data types.

```
# booleans
true_boolean = True
false_boolean = False

# string
my_name = "Leandro Tk"

# float
book_price = 15.80
```

2. Control Flow: conditional statements

“**If**” uses an expression to evaluate whether a statement is True or False. If it is True, it executes what is inside the “if” statement. For example:

```
if True:
  print("Hello Python If")

if 2 > 1:
  print("2 is greater than 1")
```

1 is not greater than **2**, so the code inside the “**else**” statement will be executed.

You can also use an “**elif**” statement:

```
if 1 > 2:
  print("1 is greater than 2")
elif 2 > 1:
  print("1 is not greater than 2")
else:
  print("1 is equal to 2")
```

The **while** loop needs a “**loop condition.**” If it stays True, it continues iterating. In this example, when `num` is `11` the **loop condition** equals `False`.

Another basic bit of code to better understand it:

```
loop_condition = True

while loop_condition:
    print("Loop Condition keeps: %s" %(loop_condition))
    loop_condition = False
```