Python程序设计实验报告（实验四）

- **实验名称**：程序控制结构
- **实验人员**：[朱品赞 / 2410211124 / 计科3班]
- **实验日期**：[2025 年 10 月 12 日]

一、实验目的

1. 掌握程序的三大结构
2. 掌握异常处理的应用
3. 利用分支循环解决实际问题

二、实验内容及要求

（一）基础实验

1. 任务 1：成绩评定程序

- **具体要求**：接收用户输入的学生姓名（字符串）和语文、数学、英语三科成绩（整数，0-100分），若成绩超出范围则提示 “成绩无效(0-100)，请重新输入”，直至输入正确。计算三科总分（整数）和平均分（保留 1 位小数，浮点数），使用分支结构评定等级：平均分≥90 为 “A”，80≤平均分<90 为 “B”，70≤平均分<80 为 “C”，60≤平均分<70 为 “D”，平均分<60 为 “E”。
- **输出格式**：“姓名：XXX，总分：XXX，平均分：X.X，等级：X”

2. 任务 2：字符串密码强度检测

- **具体要求**：输入一串密码；根据下列要求判断密码的强度。
  - 弱强度：长度<6，或仅由数字组成，或仅由字母组成，打印 “密码强度：弱！”
  - 中强度：长度≥6，且同时包含数字和字母，打印 “密码强度：中”
  - 高强度：长度≥8，且包含数字、字母和特殊符号（仅限!@#$%^&*），打印 “密码强度：高，设置成功！”
- 对于密码强度为弱和中的，要求再次设置密码，直到强度为高为止。

三、实验步骤与代码实现

（一）任务 1

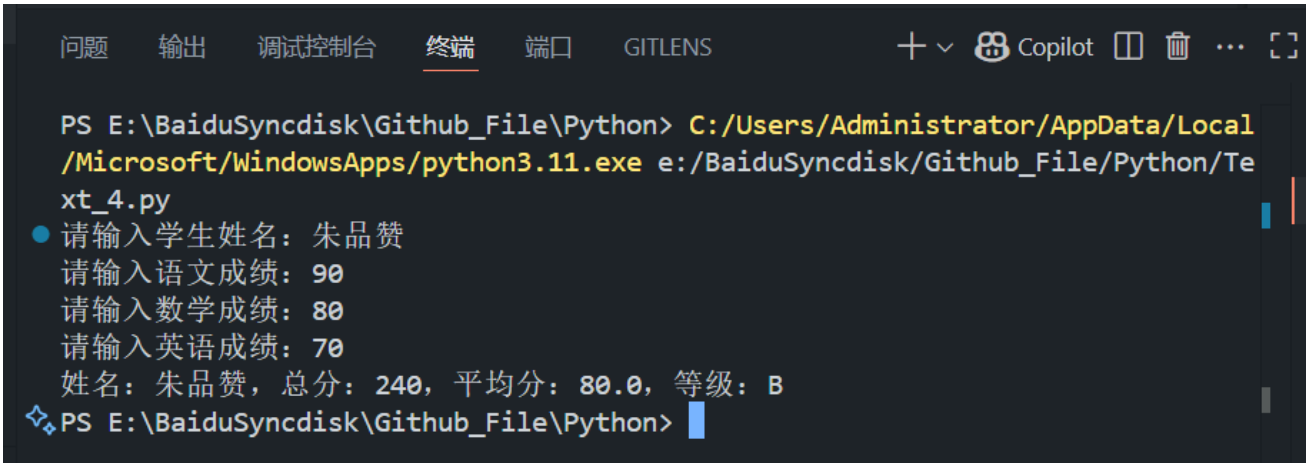
1. 新建 Python 文件；
2. 利用 print 函数输出每行的内容；

3. 运行程序验证结果。

```
def get_valid_score(subject):
    """获取有效成绩（0-100分）"""
    while True:
        try:
            score = int(input(f"请输入{subject}成绩："))
            if 0 <= score <= 100:
                return score
            else:
                print("成绩无效(0-100)，请重新输入")
        except ValueError:
            print("请输入有效的整数成绩")


def calculate_grade(average_score):
    """根据平均分评定等级"""
    if average_score >= 90:
        return "A"
    elif average_score >= 80:
        return "B"
    elif average_score >= 70:
        return "C"
    elif average_score >= 60:
        return "D"
```

```
    else:
        return "E"


def main():
    # 获取学生姓名
    name = input("请输入学生姓名：")
    # 获取三科成绩（确保在0-100范围内）
    chinese_score = get_valid_score("语文")
    math_score = get_valid_score("数学")
    english_score = get_valid_score("英语")
    # 计算总分和平均分
    total_score = chinese_score + math_score + english_score
    average_score = total_score / 3.0
    # 评定等级
    grade = calculate_grade(average_score)
    # 输出结果
    print(f"姓名：{name}，总分：{total_score}，平均分：{average_score:.1f}，等级：
{grade}")


# 运行程序
if __name__ == "__main__":
    main()
```

- **运行结果：**



（二）任务 2

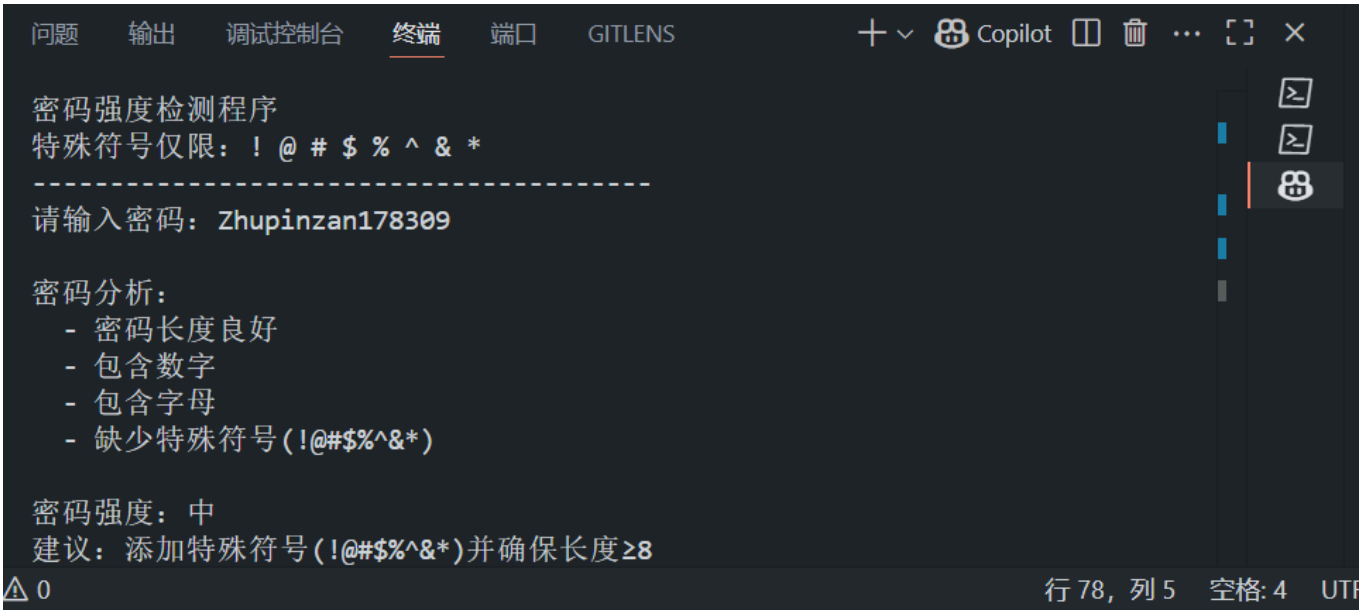
```
def check_password_strength_detailed(password):
    """详细检测密码强度并返回反馈"""
    length = len(password)
    has_digit = any(char.isdigit() for char in password)
    has_alpha = any(char.isalpha() for char in password)
    has_special = any(char in '!@#$%^&*' for char in password)
    # 提供详细反馈
    feedback = []
    if length < 6:
        feedback.append("密码长度至少需要6个字符")
    elif length < 8:
        feedback.append("密码长度达到6个字符，但建议使用8个或更多字符")
    else:
        feedback.append("密码长度良好")
    if not has_digit:
```

```
        feedback.append("缺少数字")
    else:
        feedback.append("包含数字")
    if not has_alpha:
        feedback.append("缺少字母")
    else:
        feedback.append("包含字母")
    if not has_special:
        feedback.append("缺少特殊符号(!@#$%^&*)")
    else:
        feedback.append("包含特殊符号")
    # 判断强度
    if length < 6 or (has_digit and not has_alpha and not has_special) or
(has_alpha and not has_digit and not has_special):
        strength = "弱"
    elif length >= 6 and has_digit and has_alpha and not has_special:
        strength = "中"
    elif length >= 8 and has_digit and has_alpha and has_special:
        strength = "高"
    else:
        strength = "弱"
    return strength, feedback


def main_detailed():
```

```
    print("密码强度检测程序")
    print("特殊符号仅限：! @ # $ % ^ & *")
    print("-" * 40)
    while True:
        password = input("请输入密码：")
        strength, feedback = check_password_strength_detailed(password)
        print("\n密码分析：")
        for item in feedback:
            print(f"  - {item}")
        if strength == "高":
            print("\n密码强度：高，设置成功！")
            break
        elif strength == "中":
            print("\n密码强度：中")
            print("建议：添加特殊符号(!@#$%^&*)并确保长度≥8")
        else:
            print("\n密码强度：弱！")
            print("建议：确保长度≥6，同时包含数字和字母")
        print("\n" + "="*40)


# 可以选择使用哪个版本
if __name__ == "__main__":
    # 使用基础版本
```

```
# main()

# 或者使用详细版本

main_detailed()
```

运行结果：



四、遇到的问题及解决

1. **问题**：在成绩评定程序中，最初没有处理非数字输入的情况，当用户输入字母或其他字符时程序会崩溃
   解决：添加了try-except异常处理机制，捕获ValueError异常，提示用户输入有效的整数成绩
2. **问题**：密码强度检测时，对"仅由数字组成"和"仅由字母组成"的判断逻辑不够准确，容易误判
   解决：使用更严谨的条件判断，通过 `has_digit and not has_alpha and not has_special` 来确保确实是纯数字，类似方法判断纯字母
3.

五、实验总结

1. **熟练掌握了**Python的基本输入输出处理、条件分支结构、循环控制结构，以及字符串的基本操作和类型判断方法
2. **通过调试解决问题**，加深了对程序健壮性的理解，学会了如何通过异常处理来增强程序的容错能力，避免因用户输入不当导致的程序崩溃
3. **后续需加强**对复杂条件逻辑的梳理能力，特别是在多条件组合判断时，要确保逻辑的完备性和准确性；同时需要学习更多字符串处理的高级技巧，提高代码的效率和可读性
4. **体会到**良好的用户交互设计的重要性，通过清晰的提示信息和反馈，能够显著提升用户体验，减少用户的困惑和操作错误

5. **认识到**模块化编程的价值，将功能分解为独立的函数，不仅使代码结构更清晰，也便于后续的测试和维护