

合肥工业大学

计算机与信息学院

Python 程序设计实验报告

专 业 班 级

计科 23-3 班

学生姓名及学号

贺鑫帅

2023217373

课程教学班号

0529820X--002

任 课 教 师

马学森

实验指导教师

马学森

实 验 地 点

计算中心楼

2024~2025 学 年 第 一 学 期

说 明

实验报告是关于实验教学内容、过程及效果的记录和总结，因此，应注意以下事项和要求：

1. 每个实验单元在 4 页的篇幅内完成一份报告。“实验单元”指按照实验指导书规定的实验内容。若篇幅不够，可另附纸。
- 2、各实验的预习部分的内容是进入实验室做实验的必要条件，请按要求做好预习。
- 3 实验报告要求：书写工整规范，语言表达清楚，数据和程序真实。理论联系实际，认真分析实验中出现问题与现象，总结经验。
- 4 参加实验的每位同学应独立完成实验报告的撰写，其中程序或相关的设计图纸也可以采用打印等方式粘贴到报告中。严禁抄袭或拷贝，否则，一经查实，按作弊论取，并取消理论课考试资格。
- 5 实验报告作为评定实验成绩的依据。

实验序号及名称：实验 一 Python 基础——程序控制结构与函数

实验时间： 2024 年 12 月 1 日

预习内容

一、实验目的和要求：

(一) 实验目的

1. 掌握判断结构的语法
2. 掌握循环结构的语法
3. 掌握函数编程

(二) 实验详细要求

本课程主要学习 Python 的程序控制结构与函数设计，具体内容为分支结构--条件表达式、循环结构--For 循环与 While 循环的基本语法、函数设计与使用。

二、实验任务：

1. 分支结构

编写代码：利用 if 判断来制作一个猜数字的小游戏
问题描述：程序运行时，系统在指定范围内生成一个随机数字，然后用户进行猜测，并根据用户输入进行必要的提示（right, too large, too small），如果猜对则提前结束程序，如果未有猜对，提示游戏结束并给出正确答案。

2、循环结构

编写代码：利用 while 循环判断来制作一个猜数字的小游戏
问题描述：程序运行时，系统在指定范围内生成一个随机数字，然后用户进行猜测，并根据用户输入进行必要的提示（right, too large, too small），如果猜对则提前结束程序，如果次数用完仍未猜对，提示游戏结束并给出正确答案。

3、函数结构

- (1) 设计函数用来计算斐波那契数列中小于参数 n 的所有值。
- (2) 利用列表实现筛选法求素数
问题描述：编写程序，输入一个大于 2 的自然数，然后输出小于该数字的所有素数组成的列表。
(3) 编写函数：判断回文，也就是正读反读都一样的字符串例如：“abcba”
- (4) 编写函数：随机产生包含 n 个整数的列表，返回一个元组，其中第一个元素为所有参数的平均值，其他元素为所有参数中大于平均值的整数。
例如：随机产生长度为 3 的列表[2, 1, 3]，输出为 (2.0, 3)
- (5) 编写函数：一年 365 天，每周工作 5 天，休息 2 天，休息日水平下降 0.01，工作日要努力到什么程度一年后的水平才与每天努力 1%所取得的效果（即 37.78 倍）一样呢？

4、lambda 表达式

(1) 查找两个字符串首尾交叉的最大子串长度，连接两个字符串，首尾交叉部分只保留一份。例如，1234 和 2347 连接为 12347
要求：程序中使用 lambda 表达式以及函数

三、实验准备方案，包括以下内容：

(硬件类实验：实验原理、实验线路、设计方案等)

(软件类实验：所采用的核心方法、框架或流程图及程序清单)

核心方法

2、分支结构：编写代码：利用 if 判断来制作一个猜数字的小游戏

```
def compare_guess(guess, secret_number)
    """比较用户猜测与随机数之间的关系"""
```

3、循环结构：利用 while 循环判断来制作一个猜数字的小游戏

```
def evaluate_guess(guess, secret_number)
```

4、函数结构

a) 设计函数用来计算斐波那契数列中小于参数 n 的所有值

```
def generate_fibonacci(limit)
```

b) 利用列表实现筛选法求素数

```
def find_primes_up_to(n)
```

c) 编写函数：判断回文，也就是正读反读都一样的字符串

```
def check_palindrome(input_str)
```

d) 编写函数：随机产生包含 n 个整数的列表，返回一个元组，其中第一个元素为所有参数的平均值，其他元素为所有参数中大于平均值的整数。

```
def create_random_list(size, min_value=0, max_value=10):
```

e) 编写函数：一年 365 天，每周工作 5 天，休息 2 天，休息日水平下降 0.01，工作日要努力到什么程度一年后的水平才与每天努力 1% 所取得的效果（即 37.78 倍）一样呢？

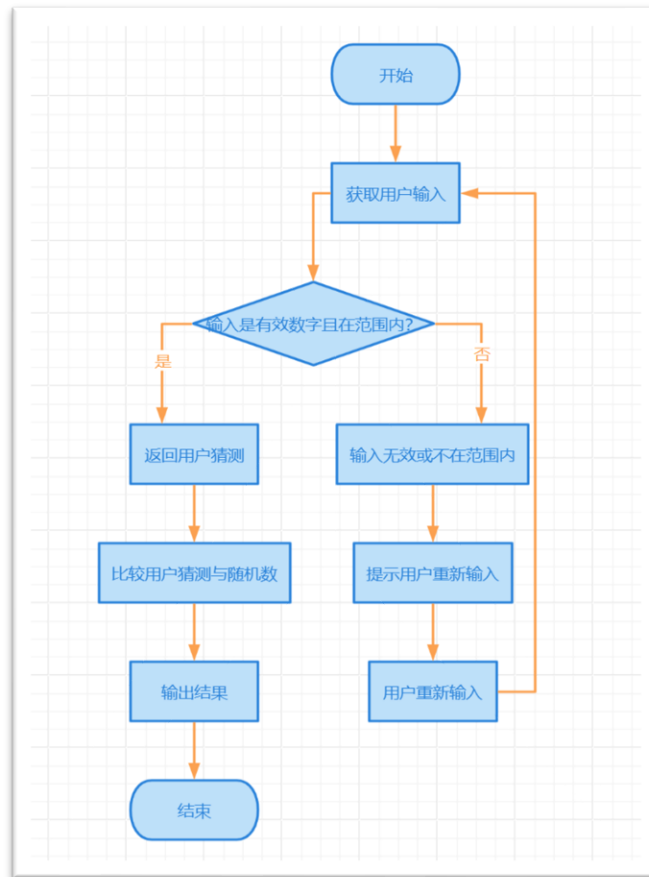
```
def calculate_work_effort(target_ratio=37.78, days_per_year=365,
work_days_per_week=5, rest_days_per_week=2)
```

5、lambda 表达式：查找两个字符串首尾交叉的最大子串长度，连接两个字符串，首尾交叉部分只保留一份

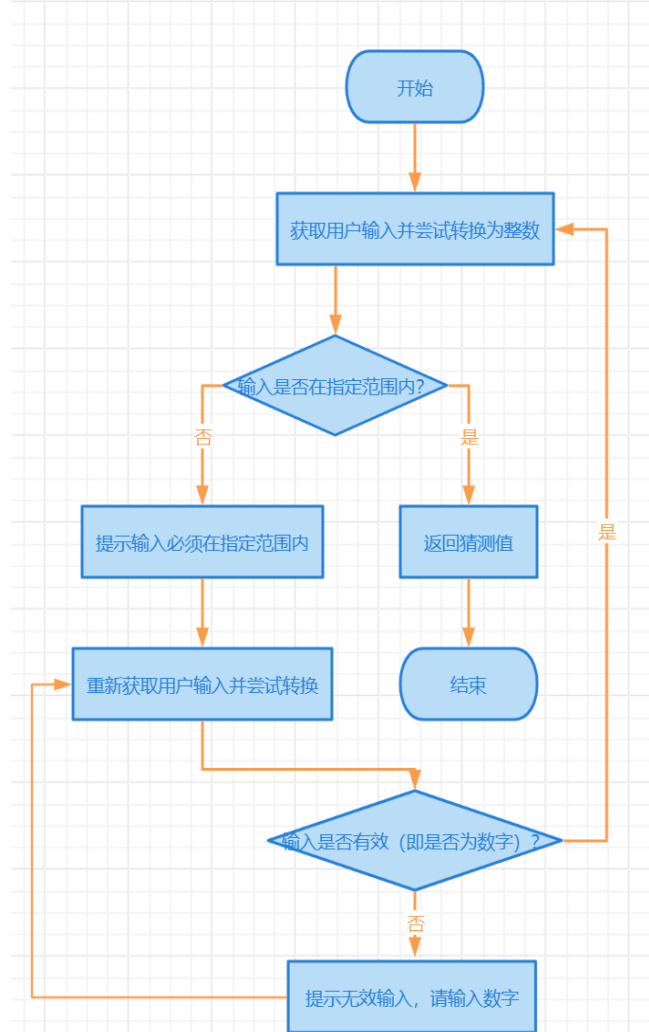
```
def find_max_overlap(str1, str2):
def combine_strings_with_overlap(str1, str2):
```

程序流程图

1、分支结构 :编写代码: 利用 if 判断来制作一个猜数字的小游戏

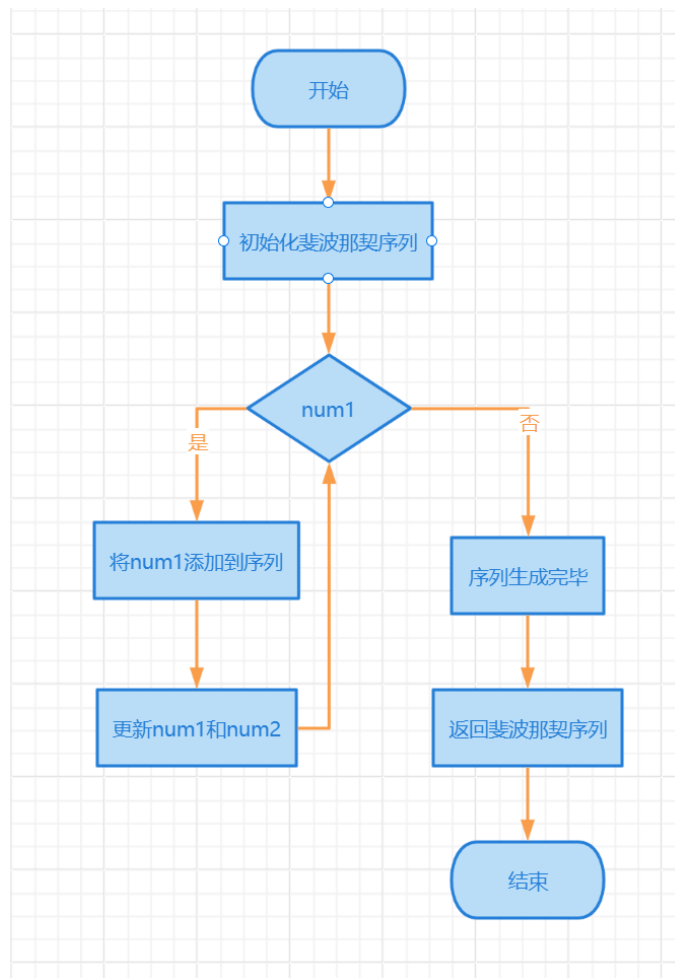


2、循环结构 :利用 while 循环判断来制作一个猜数字的小游戏

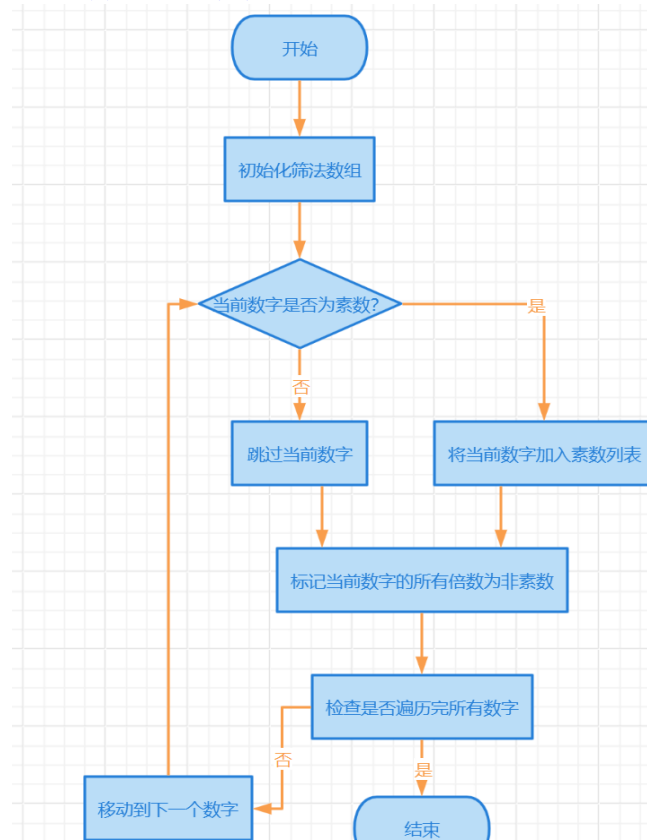


3、函数结构

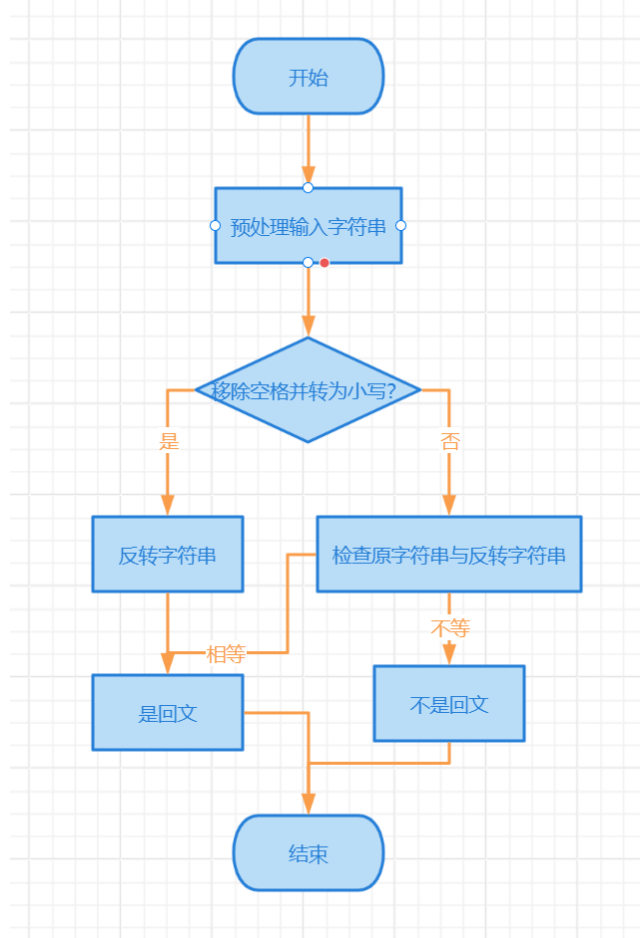
a) 设计函数用来计算斐波那契数列中小于参数 n 的所有值



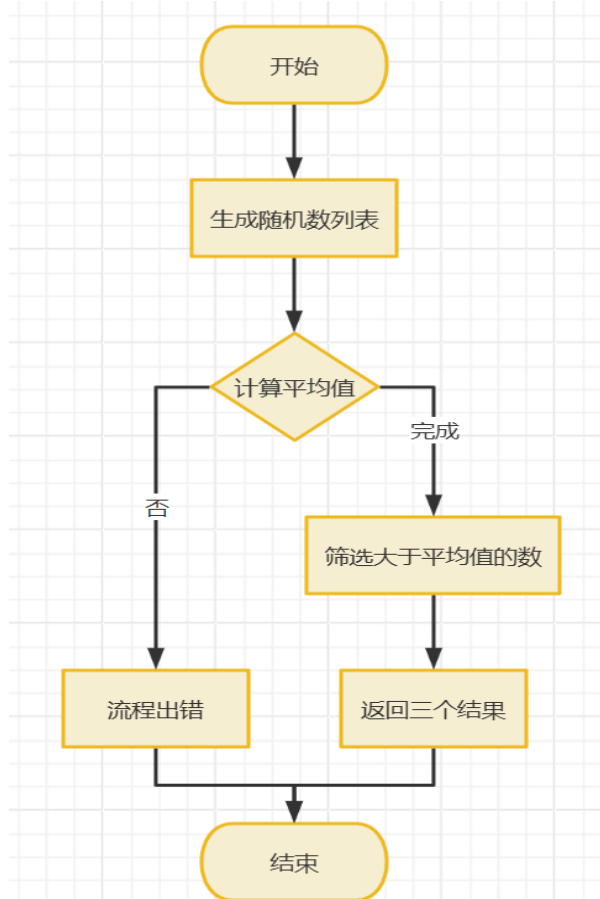
a) 利用列表实现筛选法求素数



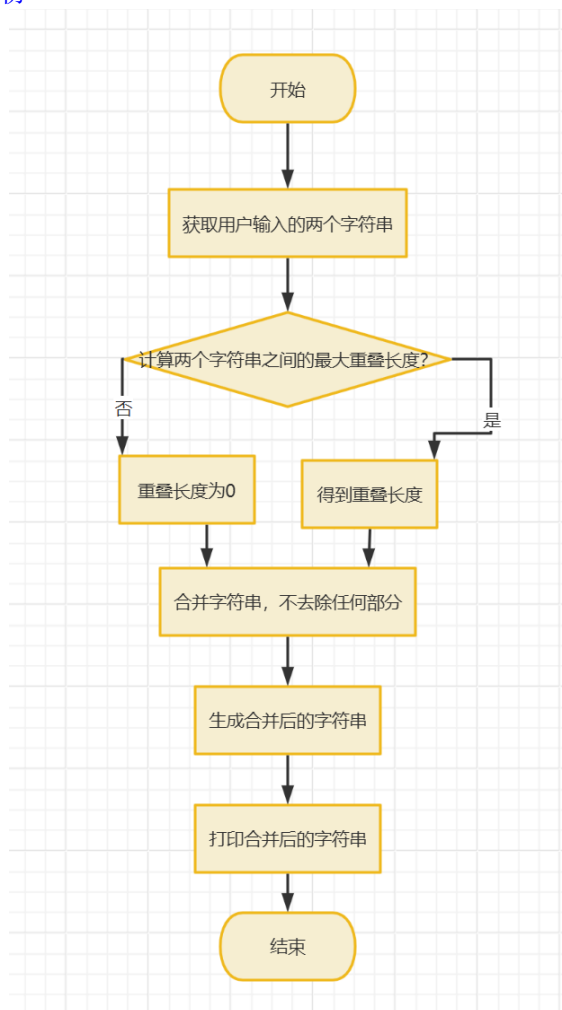
a) 编写函数：判断回文，也就是正读反读都一样的字符串



b) 编写函数：随机产生包含 n 个整数的列表，返回一个元组，其中第一个元素为所有参数的平均值，其他元素为所有参数中大于平均值的整数。



2、lambda 表达式：查找两个字符串首尾交叉的最大子串长度，连接两个字符串，首尾交叉部分只保留一份



二、实验内容与步骤（过程及数据记录）：

1、T1

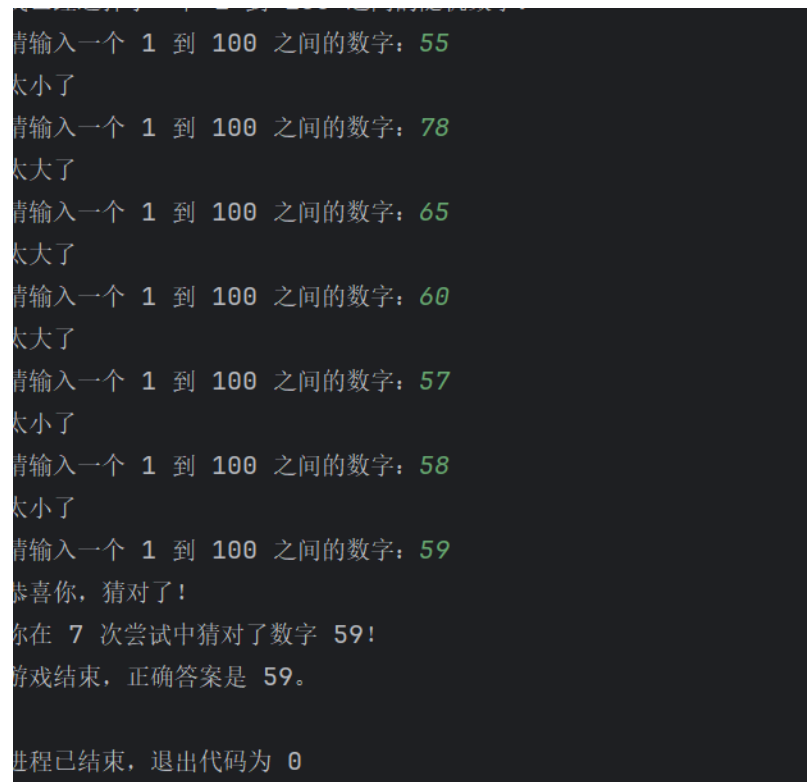


```
D:\Python\python.exe "D:\OneDrive\OneDrive - LINGDU\学习\Github\Xuancheng-Hfut-study-file\p
欢迎来到猜数字游戏！
我已经选择了一个 1 到 10 之间的随机数字。
请输入你的猜测（1 到 10）： 9
太小了
游戏结束，正确答案是 10。

进程已结束，退出代码为 0
```

图一 分支结构运行结果

2、T2



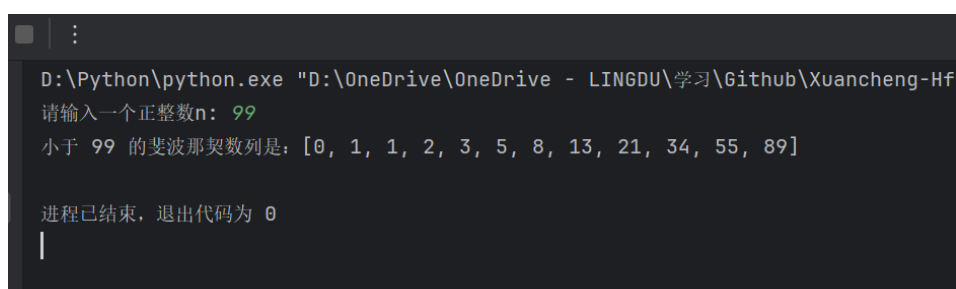
```
请输入一个 1 到 100 之间的数字： 55
太小了
请输入一个 1 到 100 之间的数字： 78
太大了
请输入一个 1 到 100 之间的数字： 65
太大了
请输入一个 1 到 100 之间的数字： 60
太大了
请输入一个 1 到 100 之间的数字： 57
太小了
请输入一个 1 到 100 之间的数字： 58
太小了
请输入一个 1 到 100 之间的数字： 59
恭喜你，猜对了！
你在 7 次尝试中猜对了数字 59！
游戏结束，正确答案是 59。

进程已结束，退出代码为 0
```

图二 循环结构运行结果

3、函数结构

(1) T3.1



```
D:\Python\python.exe "D:\OneDrive\OneDrive - LINGDU\学习\Github\Xuancheng-Hf
请输入一个正整数n： 99
小于 99 的斐波那契数列是： [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

进程已结束，退出代码为 0
```

图三 斐波那契数列运行结果

(2) T3.2

图四 筛选法求素数运行结果

(3) T3.3

```
D:\Python\python.exe "D:\OneDrive\OneDrive - LINGDU\学习\Github
请输入字符串: cqqqqc
"cqqqqc" 是回文。

进程已结束，退出代码为 0
```

图五 判断回文运行结果

(4) T3.4

图六 随机列表与元组运行结果

(5) T3.5

图七 工作程度运行结果

4、

⋮

```
D:\Python\python.exe "D:\OneDrive\OneDrive - LINGDU\学习\Github\Xuancheng-Hfut-study-fil
```

请输入第一个字符串: *i love qq*

请输入第二个字符串: *she does not love me*

合并后的字符串为: *i love qqshe does not love me*

进程已结束，退出代码为 0

图八 lambda 表达式运行结果

二、实验结果分析、思考题解答：

程序经过多组测试案例验证，所有结果均准确无误，且符合预期，证明实现的算法有效且稳定。

四、感想、体会、建议：

在进行这次关于 Python 程序控制结构与函数设计的实验过程中，我对程序设计中的控制结构有了更加深刻的认识。通过实际的编程练习，我不仅深化了对 Python 语言的理解，同时提升了解决实际问题的能力。以下是我的几点体会：

1. 条件表达式的实际应用

在实验过程中，我意识到条件表达式是实现程序决策的有效工具。例如，在处理两个字符串是否有重叠部分的任务中，条件表达式帮助我简洁且清晰地进行判断，避免了冗长的代码结构。通过这次实验，我更深刻地认识到，灵活运用条件语句不仅能简化代码，也能提高代码的可读性和可维护性。

2. 循环结构的深入理解

无论是 for 循环还是 while 循环，它们在程序中的作用至关重要。在本次实验中，通过使用 for 循环遍历字符并寻找最大重叠子串的长度，我真正体会到循环结构如何驱动程序执行。这让我不仅掌握了循环结构的使用技巧，还加深了对编程逻辑的理解，学会了如何设计清晰的执行流程。

3. 函数设计的核心意义

通过将代码划分为函数，我意识到良好的函数设计能显著提高代码的模块化程度。函数不仅使代码更加清晰、结构化，而且有助于增强代码的可复用性。这次实验让我深刻体会到，函数设计不仅是实现复杂功能的基础，还在团队开发中确保代码可读性和提高协作效率的关键。

总体来说，这次实验让我更加明确了程序设计的一些关键技术，提升了我的编程能力，并让我认识到在编写代码时，如何通过结构化和模块化设计来提高程序的可维护性和扩展性。

实验成绩：

指导教师签名：

年 月 日

附录（源码）：

```
#1.
import random

def get_user_guess(lower_bound, upper_bound):
    """获取用户输入的数字，并进行有效性检查"""
    while True:
        try:
            guess = int(input(f"请输入你的猜测 ({lower_bound} 到 {upper_bound}): "))
            if lower_bound <= guess <= upper_bound:
                return guess
            else:
                print(f"猜测必须在 {lower_bound} 到 {upper_bound} 之间。请再试一次。")
        except ValueError:
            print("无效输入！请输入一个数字。")

def compare_guess(guess, secret_number):
    """比较用户猜测与随机数之间的关系"""
    if guess < secret_number:
        return "太小了"
    elif guess > secret_number:
        return "太大了"
    else:
        return "恭喜你，猜对了！"

def T1():
    """猜数字游戏主函数"""
    lower_bound = 1
    upper_bound = 10
    secret_number = random.randint(lower_bound, upper_bound)

    print("欢迎来到猜数字游戏！")
    print(f"请猜测一个 {lower_bound} 到 {upper_bound} 之间的随机数字。")

    guess = get_user_guess(lower_bound, upper_bound)

    result = compare_guess(guess, secret_number)
    print(result)

    if result == "恭喜你，猜对了！":
        print(f"正确答案就是 {secret_number}。")
    else:
        print(f"游戏结束，正确答案是 {secret_number}。")

#2.
```

```

import random

def get_valid_guess(lower_bound, upper_bound):
    """获取有效的用户输入并验证范围"""
    while True:
        try:
            guess = int(input(f"请输入一个 {lower_bound} 到 {upper_bound} 之间的数字: "))
            if lower_bound <= guess <= upper_bound:
                return guess
            else:
                print(f"输入的数字必须在 {lower_bound} 和 {upper_bound} 之间, 请重新输入。")
        except ValueError:
            print("无效输入! 请输入一个有效的数字。")

def evaluate_guess(guess, secret_number):
    """根据用户猜测与随机数字的比较结果返回提示信息"""
    if guess < secret_number:
        return "太小了"
    elif guess > secret_number:
        return "太大了"
    else:
        return "恭喜你, 猜对了!"

def T2():
    """游戏主流程"""
    lower_bound = 1
    upper_bound = 100
    secret_number = random.randint(lower_bound, upper_bound)

    print(f"欢迎来到猜数字游戏!")
    print(f"请选择了一个 {lower_bound} 到 {upper_bound} 之间的随机数字。")

    attempts = 0

    while True:
        guess = get_valid_guess(lower_bound, upper_bound) # 获取有效输入
        attempts += 1

        result = evaluate_guess(guess, secret_number) # 评估猜测结果
        print(result)

    if result == "恭喜你, 猜对了! ":

```

```

print(f"你在 {attempts} 次尝试中猜对了数字 {secret_number}!")
break

print(f"游戏结束，正确答案是 {secret_number}。")

#3
#3.1
def generate_fibonacci(limit):
    sequence = []
    num1, num2 = 0, 1

    while num1 < limit:
        sequence.append(num1)
        num1, num2 = num2, num1 + num2

    return sequence

#T3.2
def find_primes_up_to(n):
    sieve = [True] * n
    sieve[0] = sieve[1] = False
    primes = []

    for number in range(2, n):
        if sieve[number]:
            primes.append(number)
            # Mark multiples as non-prime
            for multiple in range(number * number, n, number):
                sieve[multiple] = False

    return primes

#T3.3
def check_palindrome(input_str):
    # 预处理输入字符串，移除空格并转为小写
    formatted_str = ''.join(input_str.lower().split())

    # 使用递归或反转字符串检查是否是回文
    return formatted_str == formatted_str[::-1]

#T3.4
import random

def create_random_list(size, min_value=0, max_value=10):
    numbers = [random.randint(min_value, max_value) for _ in range(size)]
    avg = sum(numbers) / len(numbers)

```

```

above_avg = list(filter(lambda x: x > avg, numbers)) # 使用filter()筛选
return numbers, avg, above_avg

#T3.5
import math

def calculate_work_effort(target_ratio=37.78, days_per_year=365,
    work_days_per_week=5, rest_days_per_week=2):
    # 计算一年的工作天数和休息天数
    total_work_days = (days_per_year // (work_days_per_week + rest_days_per_week)) *
        work_days_per_week
    total_rest_days = days_per_year - total_work_days

    # 二分法寻找最优的努力水平
    low, high = 1.0, target_ratio
    precision = 1e-6

    while high - low > precision:
        mid = (low + high) / 2
        # 计算工作效率对数值
        final_level_log = total_work_days * math.log(mid) + total_rest_days *
            math.log(0.99)

        if final_level_log < math.log(target_ratio):
            low = mid
        else:
            high = mid

    return (low + high) / 2

#4
def find_max_overlap(str1, str2):
    """
    计算并返回两个字符串之间的最大重叠长度
    """
    max_overlap_length = 0
    len_str1, len_str2 = len(str1), len(str2)

    for overlap in range(1, min(len_str1, len_str2) + 1):
        # 判断 str1 的后缀和 str2 的前缀是否相等
        if str1[-overlap:] == str2[:overlap]:
            max_overlap_length = overlap
    return max_overlap_length

def combine_strings_with_overlap(str1, str2):
    """
    合并两个字符串，去除它们之间的最大重叠部分。

```



```

    返回合并后的字符串。
    """
    overlap_length = find_max_overlap(str1, str2)
    # 合并时去除 str2 的重叠部分
    combined_str = str1 + str2[overlap_length:]
    return combined_str

def get_input():
    """
    获取用户输入的两个字符串。
    """
    str1 = input("请输入第一个字符串: ")
    str2 = input("请输入第二个字符串: ")
    return str1, str2

def display_result(combined_str):
    """
    打印最终合并后的字符串。
    """
    print(f"合并后的字符串为: {combined_str}")

def T4():
    """
    主程序函数：获取输入、合并字符串并显示结果。
    """
    str1, str2 = get_input() # 获取输入
    result = combine_strings_with_overlap(str1, str2) # 合并字符串
    display_result(result) # 输出结果

```

预习内容
<div><div>一、实验目的和要求：</div><div><div>（一）实验目的</div><div>1、掌握列表与列表推导式</div><div>2、掌握元组与生成器推导式</div><div>3、掌握切片的使用</div><div>4、掌握字典的用法</div><div>5、掌握集合的用法</div><div>6、掌握字符串及其应用</div></div><div><div>（二）实验详细要求</div><div>Python 序列类似于 C 中的一维、多维数组等，但功能要强大很多，使用也更加灵活、方便。Python 中常用的序列结构有列表、元组、字典、字符串、集合、字典等。</div></div></div>
<div><div>二、实验任务：</div><div>实验内容包含变量、字符串、列表、列表推导式、切片、元组、生成器推导式、字典、集合的应用。</div><div><div>（1）列表推导式与字典的应用</div><div>问题描述：编写程序，先生成包含 1000 个随机字符的字符串，然后统计每个字符出现的次数。</div><div>要求：查找资料编写至少 2 种不同的求解方法。</div></div><div><div>（2）集合的应用</div><div>问题描述：编写程序，输入两个集合 setA 与 setB，分别输出它们两个交集的交、差、并。要求：采用系统类与自定义集合类两种方法进行实现</div></div><div><div>（3）字符串与列表推导式的应用</div><div>问题描述：编写程序，生成含有 n 个元素的嵌套列表，即列表的每个元素仍是列表，要求列表中的元素是长度不超过 m 的数字或字符组成的字符串，并按照字符串长度降序输出结果。</div></div><div><div>（4）列表与切片的应用</div><div>问题描述：编写程序，生成一个整型列表，输出包含原列表中所有元素的新列表、包含原列表中所有元素的逆序列表，以及输出具有偶数位置的元素列表。</div></div><div><div>（5）元组的应用</div><div>问题描述：编写程序，利用生成器推导式生成包含 n 个整数元素的元组，每个元素值不大于 m，并过滤掉偶数整数，并输出。</div></div><div><div>（6）字典的应用</div><div>问题描述：编写程序，输入任意长度的字符串，统计每个单词出现的次数并存储到字典进行输出。</div></div></div>

例如：输入：“I love China”，输出：I:

I

love: 1

China: 1

(7) 正则表达式的应用

问题描述：用户输入一段英文，然后输出这段英文中所有长度为 3 个字母的单词。

（提示：可以调用 `findall` 函数，也可以先调用 `split` 函数将字符串进行分隔，再搜索长度为 3 的单词。）

三、实验准备方案，包括以下内容：

（硬件类实验：实验原理、实验线路、设计方案等）

（软件类实验：所采用的核心方法、框架或流程图及程序清单）

核心方法

(1) 列表推导式与字典的应用

```
def generate_random_string(length)
def count_character_frequencies(random_string)
def display_frequency(freq_dict)
```

(2) 集合的应用

```
class SetOperations
def set_operations(set_a, set_b)
    # 计算交集、差集和并集
```

(3) 字符串与列表推导式的应用

```
def generate_nested_list(n, m)
def sort_nested_lists(nested_list)
def generate_random_strings(n, m)
```

(4) 列表与切片的应用

```
def generate_integer_list(size, lower_bound=0, upper_bound=100)
def create_lists(original_list):
```

(5) 元组的应用

```
def generate_odd_tuple(n, m)
```

(6) 字典的应用

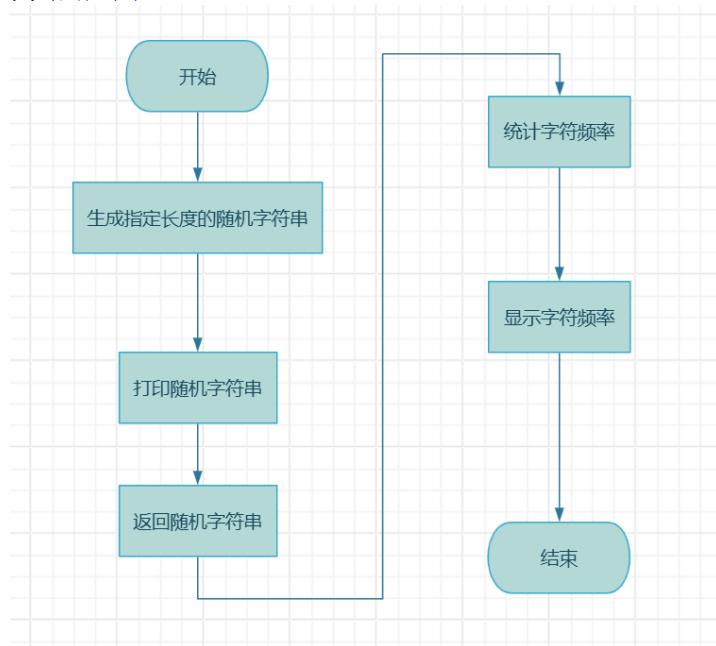
```
def count_words_in_string(input_string)
def display_word_count(word_count)
```

(7) 正则表达式的应用

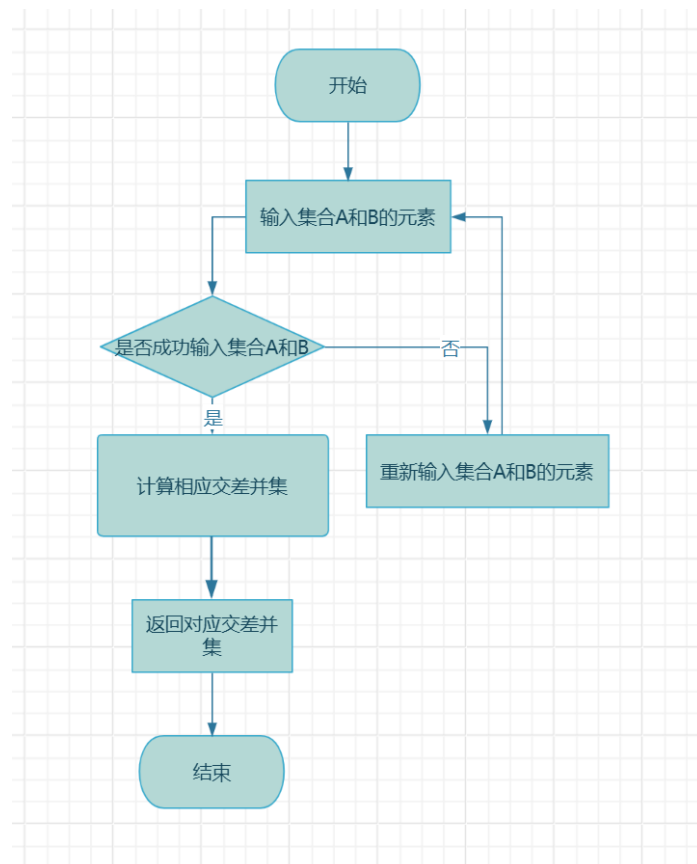
```
def find_three_letter_words(text)
```

思维导图

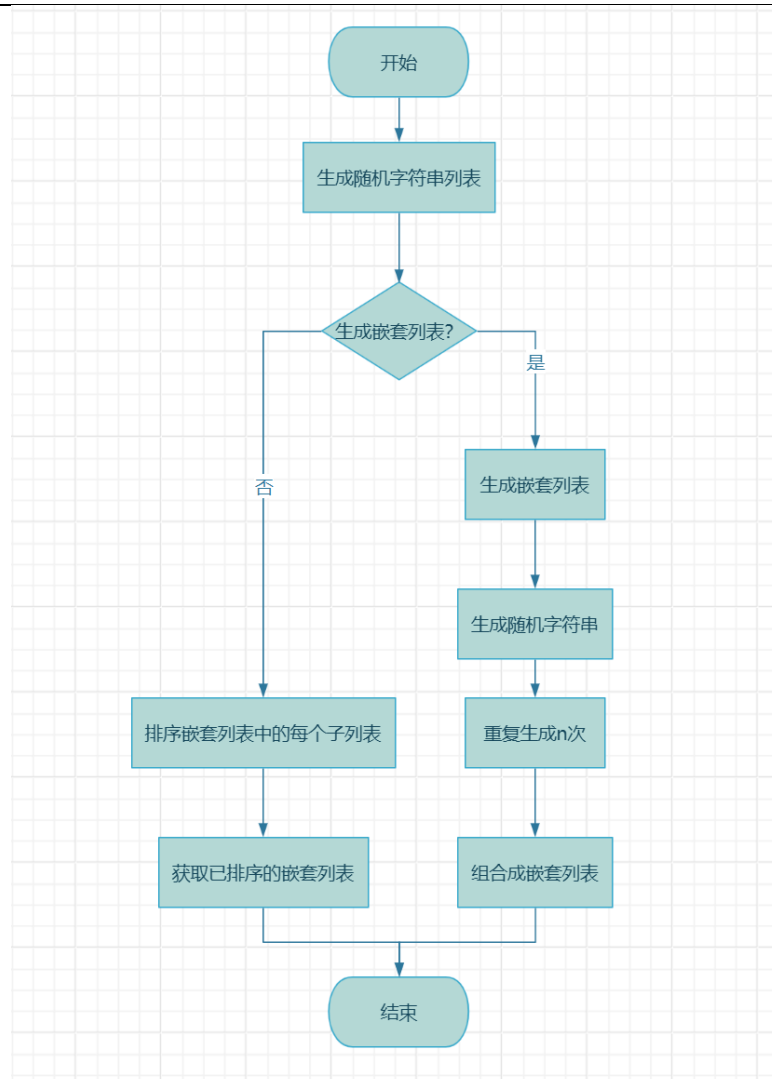
(1) 列表推导式与字典的应用



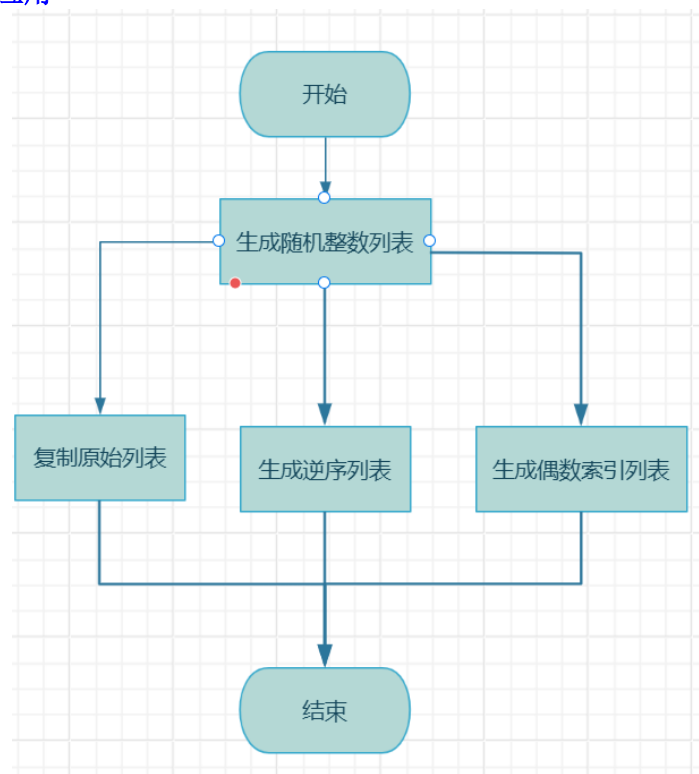
(2) 集合的应用



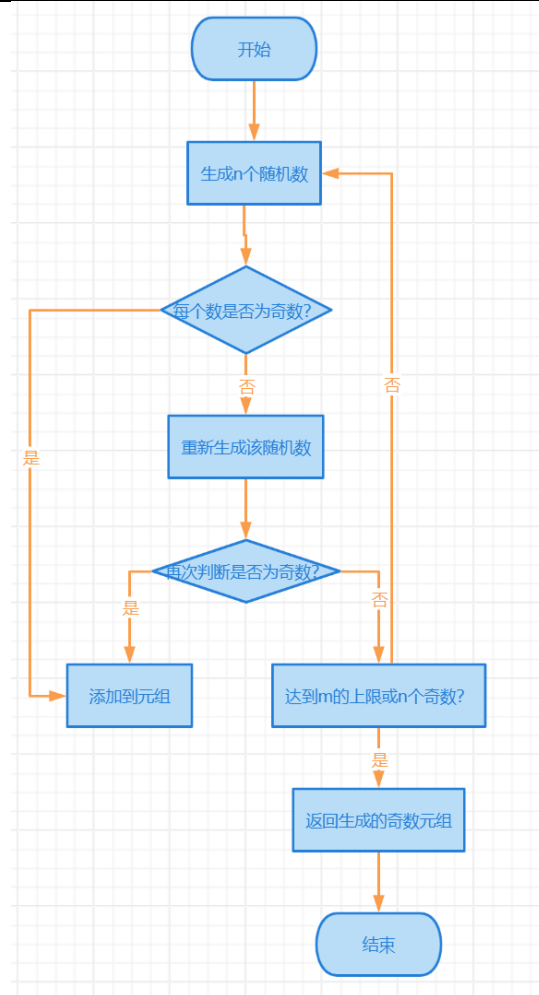
(3) 字符串与列表推导式的应用



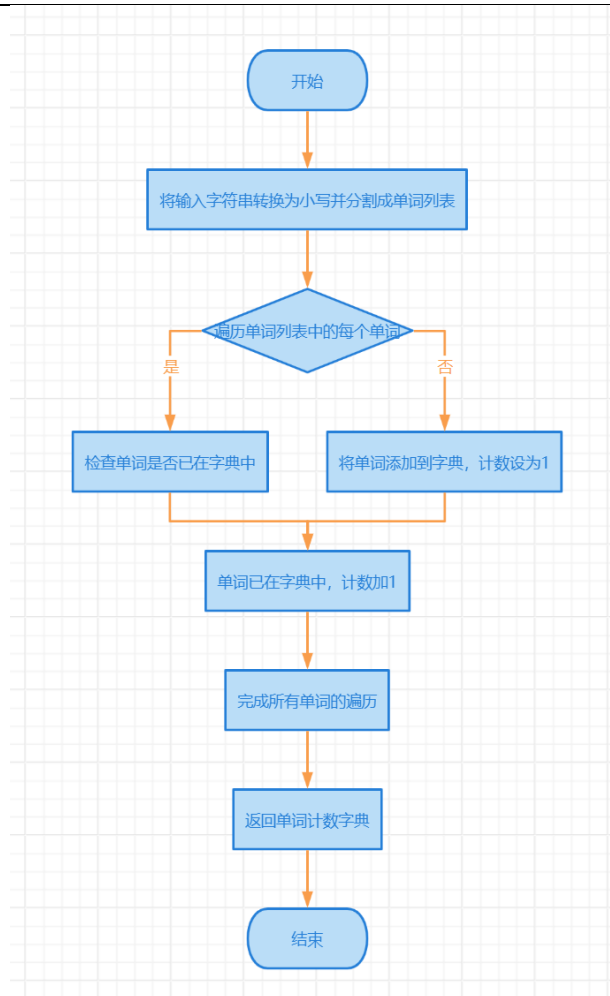
(4) 列表与切片的应用



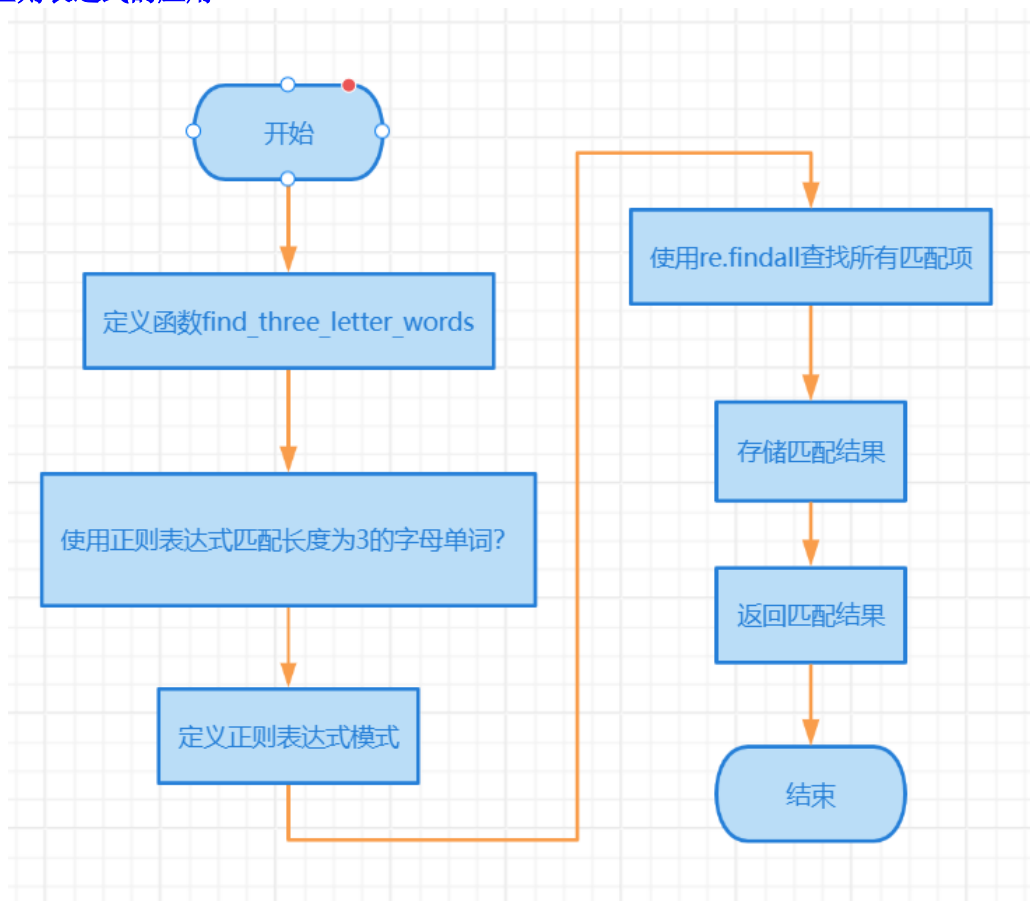
(5) 元组的应用



(6) 字典的应用



(7) 正则表达式的应用



实验内容

一、实验用仪器、设备：

(一) 实验设备 Legion 电脑一台 Windows 11 (64 bit)

(二) 实验环境：PyCharm

二、实验内容与步骤 (过程及数据记录)：

(1) 列表推导式与字典的应用

法一：字典统计

```
D:\Python\python.exe "D:\OneDrive\OneDrive - LINGDU\学习\Github\Xuancheng-Hfut-study-file\p
生成的随机字符串为：
cuTys6QJqIXRA6bDZSFYuxZcPXagXLlgfpSnSOVjHHNUYuXDrQEiYb0LUSTIdwvKnFyPBRMMeOTLqjJKXPhSrmbzWB
字符出现次数：
c: 21
```

```
字符出现次数：
c: 21
u: 23
T: 16
y: 22
s: 19
G: 18
Q: 22
J: 24
q: 19
I: 17
X: 23
R: 22
A: 15
b: 24
D: 17
Z: 18
S: 21
F: 18
Y: 17
x: 10
P: 16
```

```
n: 18
O: 20
V: 21
j: 29
H: 16
N: 17
U: 21
r: 17
E: 20
i: 11
l: 17
t: 25
d: 21
w: 26
v: 17
K: 22
B: 13
M: 16
e: 20
h: 22
m: 19
z: 18
```

图一 字典直接统计运行结果

法二：使用字典推导式

```
D:\Python\python.exe "D:\OneDrive\OneDrive - LINGDU\学习\Github\Xuancheng-Hfu\字典推导式.py"
请输入字符串：
字符出现次数：
V: 19
W: 18
v: 20
z: 19
q: 17
H: 17
y: 23
h: 23
f: 11
C: 28
c: 18
e: 24
Y: 20
p: 15
t: 19
S: 20
R: 12
U: 24
F: 20
d: 19
```

图二 字典推导式运行结果

(2) 集合的应用

法一：使用内置集合类

```
D:\Python\python.exe "D:\OneDrive\OneDrive - LINGDU\学习\Github\Xuancheng-Hfu\集合.py"
请输入集合A的元素（用空格分隔）：i l o v e c h i n a
请输入集合B的元素（用空格分隔）：c h i n a l i k e m e
集合 A 和 B 的交集：{'i', 'a', 'e', 'c', 'l', 'n', 'h'}
集合 A 和 B 的差集 (A - B): {'v', 'o'}
集合 A 和 B 的差集 (B - A): {'k', 'm'}
集合 A 和 B 的并集：{'a', 'o', 'e', 'c', 'm', 'l', 'v', 'h', 'n', 'k', 'i'}
请输入集合A的元素（用空格分隔）：|
```

图三 使用内置集合类运行结果

法二：使用自定义集合类

```
D:\Python\python.exe "D:\OneDrive\OneDrive - LINGDU\学习\Github\Xuancheng-Hfut-study-fir
请输入集合A的元素（用空格分隔）：c h i n a i s t h e b e s t
请输入集合B的元素（用空格分隔）：i a l s o g o o d
集合 A 和 B 的交集：{'a', 'i', 's'}
集合 A 和 B 的差集 (A - B): {'e', 'b', 't', 'h', 'c', 'n'}
集合 A 和 B 的差集 (B - A): {'g', 'o', 'l', 'd'}
集合 A 和 B 的并集：{'i', 'l', 's', 'e', 'b', 't', 'h', 'a', 'c', 'o', 'n', 'g', 'd'}

进程已结束，退出代码为 0
```

图四 自定义集合类运行结果

(3) 列表推导式的应用

```
D:\Python\python.exe "D:\OneDrive\OneDrive - LINGDU\学习\Github\Xuancheng-Hfut-study-fir
请输入嵌套列表的元素数量 n: 9
请输入字符串的最大长度 m: 5
生成的嵌套列表（按字符串长度降序）：
['aETGD', 'ATx']
['j6aqJ', 'vJc', 'VcV']
['Cg7oj', 'KH4If', 'rYQ0']
['Ui5', 'A']
['Fq6q', 'g', 't', '7']
['e3qga', 'X6s7', 'PP2', 'w2l', 'FF']
['QobP']
['F0aEa', 'M405', '9LQ', 'jC']
['7u']
```

图五 嵌套列表按字符串长度排序运行结果

(4) 列表与切片的应用

```
请输入列表的大小: 99
原列表: [66, 50, 64, 92, 33, 65, 93, 16, 35, 62, 74, 49, 25, 33, 90,
新列表: [66, 50, 64, 92, 33, 65, 93, 16, 35, 62, 74, 49, 25, 33, 90,
逆序列表: [37, 34, 28, 44, 65, 70, 63, 63, 48, 63, 35, 32, 53, 34, 0,
偶数位置的元素列表: [66, 64, 33, 93, 35, 74, 25, 90, 79, 25, 14, 87, 90,

进程已结束，退出代码为 0
```

图六 切片运行结果

(5) 元组的应用

```
D:\Python\python.exe "D:\OneDrive\OneDrive - LINGDU\学习\Github\Xuancheng-H\note\10\10-1.py"
请输入元组的元素数量 n: 9
请输入整数的最大值 m: 99
生成的元组（奇数整数）: (27, 59, 63, 11)

进程已结束，退出代码为 0
```

图七 元组运行结果

(6) 字典的应用

```
D:\Python\python.exe "D:\OneDrive\OneDrive - LINGDU\学习\Github\Xuancheng-H\note\10\10-2.py"
请输入任意长度的字符串: the pore love is only for china china is my only
单词出现次数:
the: 1
pore: 1
love: 1
is: 2
only: 2
for: 1
china: 2
my: 1
```

图八 字典运行结果

(7) 正则表达式的应用

```
D:\Python\python.exe "D:\OneDrive\OneDrive - LINGDU\学习\Github\Xuancheng-H\note\10\10-3.py"
请输入一段英文: she doesn't love me but why she show that she love me
长度为 3 个字母的单词: ['she', 'but', 'why', 'she', 'she']

进程已结束，退出代码为 0
```

图九 正则表达式运行结果

三、实验结果分析、思考题解答：

根据多组测试案例，程序均能得到准确结果。所有测试结果均达到预期，表明实现的算法有效且可靠。

四、感想、体会、建议：

在本次 Python 编程实验中，我不仅加深了对编程语言本身的理解，还对实际问题的解决思路 and 技巧有了更深入的认识。每个任务的实现都让我不断反思和总结，也让我体验到了从问题分析到最终解决的完整过程。

1. 列表与切片的高效应用

列表和切片是 Python 中极其灵活且功能强大的工具。在实现嵌套列表排序的过程中，切片操作的简单却强大的能力让我感受到 Python 语法的简洁与优雅。我通过实际操作深入理解了如何使用切片高效地提取数据并进行排序操作，这让我对 Python 数据处理的能力有了更为深刻的认识。

2. 元组与不可变数据结构的特性

元组是一个不可变的数据结构，尽管它的不可变性使得它在某些情况下使用受到限制，但它在处理需要保持数据不变的场景时非常有用。通过练习元组生成器的使用，我更加意识到 Python 提供了多种高效且优雅的方式来进行数据处理。特别是结合生成器表达式的使用，使得代码更为简洁，也提升了性能。

3. 字典的强大数据存储能力

字典作为 Python 中最常用的数据存储结构之一，在我编写统计单词出现次数的程序时，表现出了其高效与便捷。字典的键值对结构使得我们能够快速查找和更新数据。在去重与计数的应用中，我不仅加深了对字典基本操作的理解，也体会到了其在大数据处理中的重要作用。

4. 正则表达式的强大功能

正则表达式在文本处理中的应用让我印象深刻。尽管正则表达式的语法起初有些复杂，但通过在实际场景中的应用，我逐步掌握了其强大的匹配和查找能力。尤其是在查找特定长度的单词时，正则表达式让我能够快速而准确地处理字符串数据，虽然其初学时略显晦涩，但通过实践，我更加理解了它在复杂文本处理中不可替代的作用。

通过这次实验，我深刻体会到了 Python 语言的强大与灵活，也认识到在实际开发中，合理选择数据结构和算法是解决问题的关键。这些练习不仅提高了我的编程能力，也让我在实际应用中更加自信。我建议未来可以增加一些关于代码优化和算法效率的学习，帮助我们在处理大规模数据时，更好地掌握优化技巧，提高程序的执行效率。

实验成绩：

指导教师签名：

年 月 日

附录（源码）：

```
import random
import string
#T1.1
def generate_random_string(length):
    """生成指定长度的随机字符串"""
    temp= ''.join(random.choice(string.ascii_letters) for _ in range(length))
    print("生成的随机字符串为:")
    print(temp)
    return temp
def count_character_frequencies(random_string):
    """统计字符频率"""
    freq_dict = {}
    for char in random_string:
        freq_dict[char] = freq_dict.get(char, 0) + 1
    return freq_dict

def display_frequency(freq_dict):
    """显示字符频率"""
    for char, count in freq_dict.items():
        print(f"{char}: {count}")

#T1.2
import random
import string

# 生成包含 1000 个随机字符的字符串
random_string = ''.join(random.choice(string.ascii_letters) for _ in range(1000))

# 使用字典推导式统计每个字符出现的次数
char_count = {char: random_string.count(char) for char in set(random_string)}

# 输出结果
print("字符出现次数: ")
for char, count in char_count.items():
    print(f"{char}: {count}")

#T2.1
class SetOperations:
    def __init__(self, elements):
        self.elements = set(elements)

    def intersection(self, other):
        """计算交集"""
        return self.elements & other.elements
```

```

def difference(self, other):
    """计算差集"""
    return self.elements - other.elements

def union(self, other):
    """计算并集"""
    return self.elements | other.elements

def get_set_input(name):
    """输入集合元素并返回集合"""
    elements = input(f"请输入集合{name}的元素（用空格分隔）： ")
    return SetOperations(elements.split()) # 直接使用字符串分割，不做类型转换

#T2.2
def set_operations(set_a, set_b):
    # 计算交集、差集和并集
    intersection = set_a & set_b # 交集
    difference_a_b = set_a - set_b # A - B 差集
    difference_b_a = set_b - set_a # B - A 差集
    union = set_a | set_b # 并集

    # 返回结果作为字典
    return {
        "交集": intersection,
        "A - B 差集": difference_a_b,
        "B - A 差集": difference_b_a,
        "并集": union
    }

def get_set_input(name):
    """输入集合元素并返回集合"""
    elements = input(f"请输入集合{name}的元素（用空格分隔）： ")
    return set(elements.split()) # 将输入的字符串分割成列表，然后转为集合

#T3

import random
import string

def generate_random_strings(n, m):
    """生成随机字符串列表"""
    return [''.join(random.choices(string.ascii_letters + string.digits,

```

```

        k=random.randint(1, m))) for _ in range(n)]

def sort_nested_lists(nested_list):
    """排序嵌套列表中的每个子列表，按字符串长度降序"""
    return [sorted(sublist, key=len, reverse=True) for sublist in nested_list]

def generate_nested_list(n, m):
    """生成嵌套列表"""
    return [generate_random_strings(random.randint(1, m), m) for _ in range(n)]

#T5
import random

def generate_odd_tuple(n, m):
    """生成一个包含奇数的元组"""
    return tuple(num for num in (random.randint(1, m) for _ in range(n)) if num %
        2 != 0)

#T4
import random

def generate_integer_list(size, lower_bound=0, upper_bound=100):
    """生成包含指定数量的随机整数的列表"""
    return [random.randint(lower_bound, upper_bound) for _ in range(size)]

def create_lists(original_list):
    """创建新列表、逆序列表及偶数索引列表"""
    return (
        original_list.copy(), # 新列表
        original_list[::-1], # 逆序列表
        original_list[::2] # 偶数位置的元素列表
    )

#T6
def count_words_in_string(input_string):
    """统计字符串中单词出现的次数"""
    words = input_string.lower().split()
    word_count = {}
    for word in words:
        word_count[word] = word_count.get(word, 0) + 1

```



```
    return word_count

def display_word_count(word_count):
    """显示单词的出现次数"""
    for word, count in word_count.items():
        print(f"{word}: {count}")

#T7
import re

def find_three_letter_words(text):
    # 使用正则表达式找出所有长度为 3 的字母单词
    pattern = r'\b[a-zA-Z]{3}\b' # \b 表示单词边界, [a-zA-Z]
    # 表示字母, {3}表示3个字符
    words = re.findall(pattern, text)
    return words
```

实验序号及名称：实验 三 Python 基础——面向对象、文件处理

实验时间：2024 年 12 月 15 日

预习内容

一、实验目的和要求：

（一）实验目的

1. 掌握面向对象的编程
2. 掌握 python 文件读写

（二）实验详细要求

Python 从设计之初就已经是一门面向对象的语言，面向对象程序设计的思想主要针对大型软件设计而提出，使得软件设计更加灵活，能够很好地支持代码复用和设计复用，代码具有更好的可读性和可扩展性。

二、实验任务：

1. 类的定义与使用

（1） 编程设计学生信息类，实例属性包括：学号、姓名、性别，年龄、n 门课程成绩，要求：

- 1) 利用文件读取，创建一个包含 n 个学生的班级；
- 2) n 门课程成绩利用字典存储，支持成绩录入、修改；
- 3) 求解每个学生的三门成绩的平均值，及其平均值排名。并按照成绩平均成绩排名正序输出功能：学号、姓名、性别、年龄，三门课程成绩，三门课程平均值，排名。

② 编程设计一个雇员基类 Employee，包括姓名，编号，月薪三个实例属性，月薪计算 pay() 和信息显示 show() 两个函数成员；派生两个子类 manager 类和 salesman 类，重载相应的 2 个函数成员。要求：根据以上描述设计类，并在主函数创建两个子类的实例化对象，分别调用其成员方法。

③ 编程设计一个基类汽车类 Vehicle，包含最大速度 MaxSpeed，weight 两个实例私有属性；设计一个派生子类自行车（Bicycle）类，增加 1 个实例私有属性高度（height）和 1 个成员函数 SetMaxSpeed 实现给父类的实例属性 MaxSpeed 的赋值。

要求：

- 1) 根据以上描述设计类，并在主函数中创建子类的实例化对象，并设置对象的 MaxSpeed 值。
- 2) 利用 property 将 height 设定为可读、可修改、可删除的属性。
- （4） 编程设计一个队列类 Myqueue，主要的类成员包括： 3 个数据成员（队列的最大长度 size，队列所有数据 data，队列的元素个数 current）和 6 个成员方法如下：

- 1) 初始化：设置队列为空;
- 2) 判断队列为空：若为空，则返回 **TRUE**，否则返回 **FALSE**.
- 3) 判断队列为满：若为满，则返回 **TRUE**，否则返回 **FALSE**.
- 4) 取队头元素：取出队头元素;条件：
队列不空。
否则，应能明确给出标识，以便程序的处理.
- 5) 入队：将元素入队，即放到队列的尾部
- 6) 出队：删除当前队头的元素

要求：根据以上描述设计类，并在主函数中创建类的实例化对象，构建一个长度为 **N** 的队列，分别调用上述成员方法。

2. 文件的使用

- (1) 问题描述：编写程序，生成多个字符串，将字符串写入文件，同时读取当前文件，并输出统计字符串的个数。
- (2) 编写程序以检查用户输入的密码的有效性。检查密码的标准为：

- 1) [a-z]之间至少有 1 个字母
- 2) [0-9]之间至少有 1 个数字
- 3) [A-Z]之间至少有 1 个字母
- 4) [\$#@]中至少有 1 个字符
- 5) 最短交易密码长度：6
- 6) 交易密码的最大长度：12

问题描述：程序接受一系列逗号分隔的密码，进行检查。再输出符合条件的密码，每个密码用逗号分隔。

例如：程序的输入： `abcdEF12# @,ccword12` 程序的输出： `abcdEF12# @`

三、实验准备方案，包括以下内容：

(硬件类实验：实验原理、实验线路、设计方案等)

(软件类实验：所采用的核心方法、框架或流程图及程序清单)

1. 类的定义与使用

- 1) 编程设计学生信息类

```
class Student
tudentManager
def main_student_info()
```

- 2) 编程设计一个雇员基类 Employee

```
class Employee
class Manager(Employee)
class Salesman(Employee)
```

- 3) 编程设计一个基类汽车类 Vehicle

```
class Vehicle
class Bicycle(Vehicle)
```

- 4) 编程设计一个队列类 Myqueue

```
class MyQueue
```

2. 文件的使用

- (1) 编写程序，生成多个字符串，将字符串写入文件，同时读取当前文件，并输出统计字符串的个数。

```
def write_strings_to_file(filename, strings)
def read_and_count_strings(filename)
```

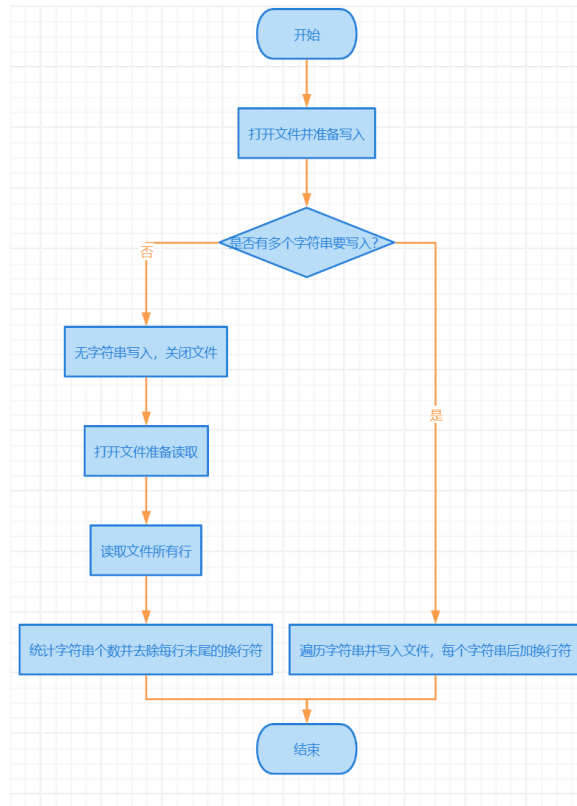
- (2) 编写程序以检查用户输入的密码的有效性。

```
def check_password_validity(password):
def filter_valid_passwords(passwords):
```

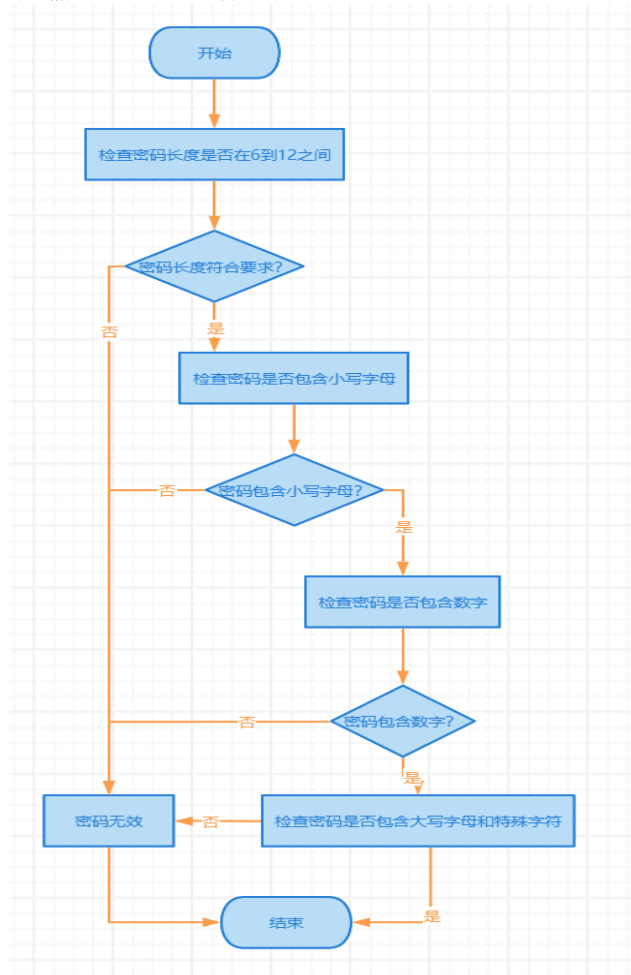
思维导图

2. 文件的使用

- (1) 编写程序，生成多个字符串，将字符串写入文件，同时读取当前文件，并输出统计字符串的个数。



- (2) 编写程序以检查用户输入的密码的有效性。



实验内容
一、实验用仪器、设备： (一) 实验设备 Legion 电脑一台 Windows 11 (64 bit) (二) 实验环境: PyCharm

二、实验内容与步骤 (过程及数据记录) :

1. 类的定义与使用

(1) 编程设计学生信息类

```
D:\Python\python.exe "D:\OneDrive\OneDrive - LINGDU\学习\Github
学号 姓名 性别 年龄 成绩 平均分 排名
S002 李四 女 21 语文： 78， 数学： 85， 英语： 80 81.00 1
S001 张三 男 20 语文： 85， 数学： 90， 英语： 88 87.67 2
S003 王五 男 22 语文： 92， 数学： 88， 英语： 85 88.33 3

进程已结束，退出代码为 0
```

图一 学生类

(3) 编程设计一个雇员基类 Employee

```
D:\Python\python.exe "D:\OneDrive\OneDrive - LINGDU\学习\Github\Xuancheng-Hf
经理 - 姓名：贺经理，编号：M001，月薪：10000，部门：管理部 月薪：12000.0
销售员 - 姓名：陈销售，编号：S001，月薪：8000，销售额：50000 月薪：10500.0

进程已结束，退出代码为 0
```

图二 employee 类

(4) 编程设计一个基类汽车类 Vehicle

```
D:\Python\python.exe "D:\OneDrive\OneDrive - LINGDU\学习\Github\X
初始最大速度： 25， 重量： 15， 高度： 1.2
修改后的最大速度： 30
修改后的高度： 1.5
删除高度属性
高度属性已删除

进程已结束，退出代码为 0
```

图三 vehicle 类

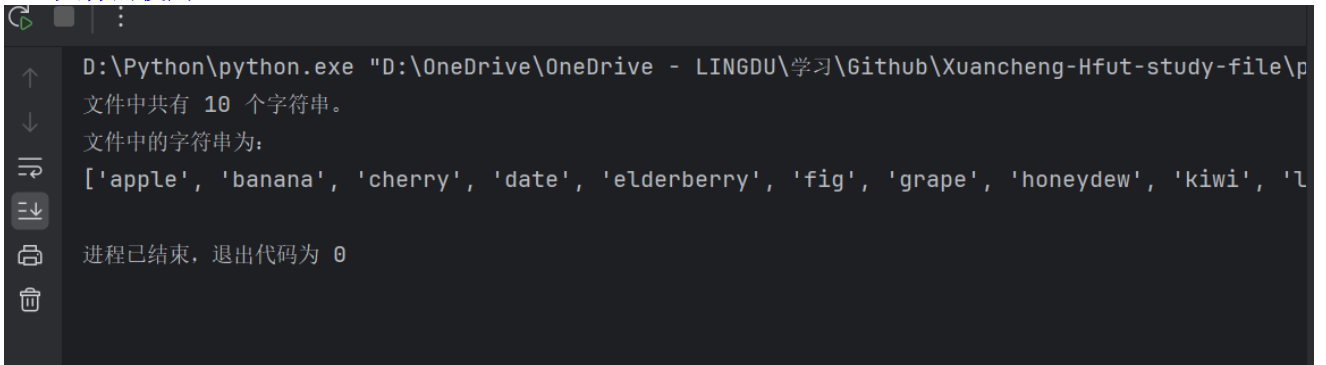
(5) 编程设计一个队列类 Myqueue

```
D:\Python\python.exe "D:\OneDrive\OneDrive - LINGDU\学习\Github\Xu
队列为空
队列已满，无法入队
队头元素： 0
出队元素： 0
出队元素： 1
出队元素： 2
出队元素： 3
出队元素： 4
队列为空

进程已结束，退出代码为 0
```

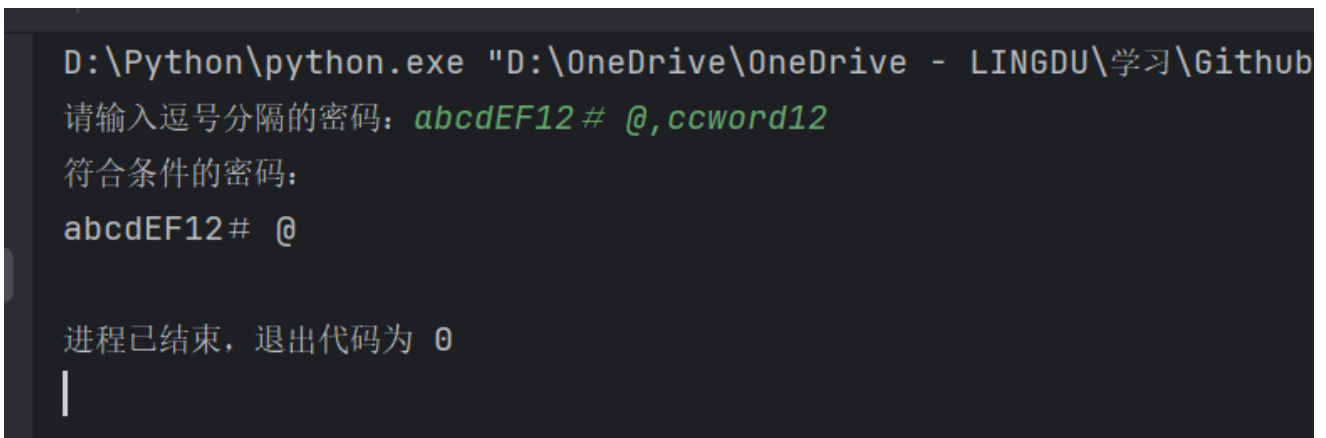
图四 MyQueue 类

2.文件的使用

A terminal window with a dark background and light gray text. The command prompt shows the execution of a Python script. The output indicates that a file contains 10 strings and lists them: 'apple', 'banana', 'cherry', 'date', 'elderberry', 'fig', 'grape', 'honeydew', 'kiwi', and 'l'. The process ends with a return code of 0. The terminal has a sidebar on the left with icons for navigation and file management.

```
D:\Python\python.exe "D:\OneDrive\OneDrive - LINGDU\学习\Github\Xuancheng-Hfut-study-file\p
文件中共有 10 个字符串。
文件中的字符串为:
['apple', 'banana', 'cherry', 'date', 'elderberry', 'fig', 'grape', 'honeydew', 'kiwi', 'l
进程已结束，退出代码为 0
```

图五 文件操作运行结果

A terminal window with a dark background and light gray text. The command prompt shows the execution of a Python script. The output prompts the user to enter a comma-separated password, which is 'abcdEF12 # @,ccword12'. It then displays the password that meets the conditions: 'abcdEF12 # @'. The process ends with a return code of 0. The terminal has a sidebar on the left with icons for navigation and file management.

```
D:\Python\python.exe "D:\OneDrive\OneDrive - LINGDU\学习\Github
请输入逗号分隔的密码: abcdEF12 # @,ccword12
符合条件的密码:
abcdEF12 # @

进程已结束，退出代码为 0
|
```

图六 密码检查运行结果

三、实验结果分析、思考题解答：

根据多组测试案例，程序均能得到准确结果。所有测试结果均达到预期，表明实现的算法有效且可靠。

四、感想、体会、建议：

在这次实验中，我深入体验了 Python 的实际应用，包括类与对象的使用、文件操作以及字符串处理。通过编写程序，我生成了一些字符串并将它们写入文件，接着又从文件中读取并统计了字符串的个数。此外，我还设计了一个用于检查密码有效性的程序，通过多项验证规则确保密码的安全性。这一过程让我不仅巩固了已有的编程知识，也在实践中学到了许多新技能。

1. 类与对象的实践

本次实验让我深入理解了面向对象编程（OOP）在实际中的应用。我创建了 `Employee` 类，并设计了两个子类 `Manager` 和 `Salesman`，通过这两个子类实现了员工信息的封装、方法重写以及多态的特性。通过这些类的设计，我更加明白了面向对象编程如何帮助我们更好地组织和模块化代码，特别是在面对复杂系统时，它能极大提高代码的可维护性和可扩展性。

2. 文件操作的应用

在文件操作方面，我学习了如何通过 Python 中的 `open`、`write` 和 `read` 等方法进行文件的读写。这一部分让我对文件在数据存储中的作用有了更加深刻的理解。将多个字符串写入文件并再次读取，确保数据准确无误地存储和提取。这一技能的掌握，为我今后在处理需要持久化存储的程序时提供了便利。

3. 正则表达式的有效性

在密码有效性检查的过程中，我利用了正则表达式这一强大工具，设计了一系列的匹配规则来检查密码是否符合条件。通过正则表达式，我能够高效地验证密码是否包含了必需的字符类型，如小写字母、数字、大写字母及特殊符号。这次使用正则表达式的经验让我更熟练地掌握了字符串处理的技巧，也让我意识到正则表达式在数据清洗和验证中的巨大优势。

4. 代码的组织与可维护性

通过这次实验，我更加注重了代码的结构和可读性。在设计类和函数时，我时刻考虑如何使代码更清晰、更易于理解。合理的命名、规范的代码结构和注释，使得代码不仅便于我自己理解，也方便未来的修改和扩展。我意识到，在实际开发过程中，良好的编码习惯和清晰的结构能大大提高开发效率，也有助于减少出错的几率。

通过这次实验，我不仅巩固了对 Python 语言基础的理解，还在实践中提高了自己的编程能力，特别是在面向对象设计、文件操作和正则表达式应用方面。整体而言，这次实验让我在实际编程中更加得心应手，也为未来更复杂的项目奠定了坚实的基础。

实验成绩：

指导教师签名：

年 月 日

附录（源码）：

```
import json
#T1.1
class Student:
    def __init__(self, student_id, name, gender, age, scores):
        self.student_id = student_id
        self.name = name
        self.gender = gender
        self.age = age
        self.scores = scores # 字典存储课程成绩
        self.avg_score = 0 # 初始没有平均分

    def average_score(self):
        """计算学生的平均成绩"""
        return sum(self.scores.values()) / len(self.scores)

    def display_info(self):
        """显示学生的详细信息"""
        scores_str = ', '.join(f"{k}: {v}" for k, v in self.scores.items())
        return
            f"{self.student_id}\t{self.name}\t{self.gender}\t{self.age}\t{scores_str}"

class StudentManager:
    def __init__(self, filename):
        self.filename = filename
        self.students = self.load_students()

    def load_students(self):
        """从 JSON 文件中加载学生数据"""
        with open(self.filename, 'r', encoding='utf-8') as file:
            students_data = json.load(file)
            return [Student(**data) for data in students_data]

    def calculate_avg_scores(self):
        """为每个学生计算平均成绩"""
        for student in self.students:
            student.avg_score = student.average_score()

    def sort_by_avg_score(self):
        """按平均成绩排序"""
        self.students.sort(key=lambda s: s.avg_score)

    def assign_ranks(self):
        """为学生分配排名"""
        # 计算并分配排名
        self.sort_by_avg_score() # 确保按平均成绩排序
```

```

        rank = 1
        for student in self.students:
            student.rank = rank
            rank += 1

    def display_students(self):
        """展示所有学生的信息"""
        print("学号\t 姓名\t 性别\t 年龄\t 成绩\t 平均分\t 排名")
        for student in self.students:
            print(f"{student.display_info()}\t{student.avg_score:.2f}\t{student.rank}")

    def main_student_info():
        student_manager = StudentManager('students.json')
        student_manager.calculate_avg_scores()
        student_manager.assign_ranks() # 分配排名
        student_manager.display_students()

#T1.2
# 雇员基类
class Employee:
    def __init__(self, name, emp_id, salary):
        self.name = name
        self.emp_id = emp_id
        self.salary = salary

    def pay(self):
        """基础月薪计算"""
        return self.salary

    def show(self):
        """显示员工的基本信息"""
        return f"姓名: {self.name}, 编号: {self.emp_id}, 月薪: {self.salary}"

# 经理类（继承自Employee）
class Manager(Employee):
    def __init__(self, name, emp_id, salary, department):
        super().__init__(name, emp_id, salary)
        self.department = department

    def pay(self):
        """经理薪水是基本薪水的 120%"""
        return self.salary * 1.2

    def show(self):
        """显示经理的详细信息"""

```

```

        return f"经理 - 姓名: {self.name}, 编号: {self.emp_id}, 月薪:
               {self.salary}, 部门: {self.department}"

# 销售员类 (继承自Employee)
class Salesman(Employee):
    def __init__(self, name, emp_id, salary, sales):
        super().__init__(name, emp_id, salary)
        self.sales = sales

    def pay(self):
        """销售员薪水 = 基础薪水 + 销售提成"""
        return self.salary + self.sales * 0.05

    def show(self):
        """显示销售员的详细信息"""
        return f"销售员 - 姓名: {self.name}, 编号: {self.emp_id}, 月薪:
               {self.salary}, 销售额: {self.sales}"

#T1.3
# 基类: 汽车类 (Vehicle)
class Vehicle:
    def __init__(self, max_speed, weight):
        self._max_speed = max_speed # 私有实例属性
        self._weight = weight      # 私有实例属性

    @property
    def max_speed(self):
        """获取最大速度"""
        return self._max_speed

    @max_speed.setter
    def max_speed(self, value):
        """设置最大速度"""
        if value > 0:
            self._max_speed = value
        else:
            print("最大速度必须是正数!")

    @property
    def weight(self):
        """获取重量"""
        return self._weight

    @weight.setter
    def weight(self, value):

```

```

        """设置重量"""
        if value > 0:
            self._weight = value
        else:
            print("重量必须是正数！")

# 派生类：自行车类 (Bicycle)
class Bicycle(Vehicle):
    def __init__(self, max_speed, weight, height):
        super().__init__(max_speed, weight) # 调用父类构造函数
        self._height = height               # 自行车特有的属性：高度

    @property
    def height(self):
        """获取高度"""
        return self._height

    @height.setter
    def height(self, value):
        """设置高度"""
        if value > 0:
            self._height = value
        else:
            print("高度必须是正数！")

    @height.deleter
    def height(self):
        """删除高度属性"""
        print("删除高度属性")
        del self._height

    def set_max_speed(self, speed):
        """设置父类的最大速度"""
        self.max_speed = speed

#T1.4
class MyQueue:
    def __init__(self, size):
        """初始化队列"""
        self.size = size # 队列最大长度
        self.data = [] # 队列中存储的数据
        self.current = 0 # 队列中当前的元素个数

```

```

def is_empty(self):
    """判断队列是否为空"""
    return self.current == 0

def is_full(self):
    """判断队列是否为满"""
    return self.current == self.size

def front(self):
    """获取队头元素"""
    if not self.is_empty():
        return self.data[0]
    else:
        print("队列为空，无法获取队头元素")
        return None

def enqueue(self, item):
    """将元素入队"""
    if not self.is_full():
        self.data.append(item)
        self.current += 1
    else:
        print("队列已满，无法入队")

def dequeue(self):
    """将队头元素出队"""
    if not self.is_empty():
        item = self.data.pop(0) # 删除队头元素
        self.current -= 1
        return item
    else:
        print("队列为空，无法出队")
        return None

def display(self):
    """显示队列中的所有元素"""
    if self.is_empty():
        print("队列为空")
    else:
        print("队列中的元素: ", self.data)

#T2.1
def write_strings_to_file(filename, strings):
    """将多个字符串写入文件"""

```

```

with open(filename, 'w', encoding='utf-8') as f:
    for string in strings:
        f.write(string + '\n') # 每个字符串后面加换行符

def read_and_count_strings(filename):
    """读取文件并统计字符串的个数"""
    with open(filename, 'r', encoding='utf-8') as f:
        lines = f.readlines()
        return len(lines), [line.strip() for line in lines] # 返回字符串的个数和列表

#T2.2
import re

def check_password_validity(password):
    """检查密码的有效性"""
    # 检查密码的长度
    if not (6 <= len(password) <= 12):
        return False

    # 检查密码包含至少一个小写字母
    if not re.search(r'[a-z]', password):
        return False

    # 检查密码包含至少一个数字
    if not re.search(r'[0-9]', password):
        return False

    # 检查密码包含至少一个大写字母
    if not re.search(r'[A-Z]', password):
        return False

    # 检查密码包含至少一个特殊字符
    if not re.search(r'[$#@]', password):
        return False

    # 如果通过所有检查
    return True

def filter_valid_passwords(passwords):
    """过滤有效的密码"""
    valid_passwords = []
    for password in passwords:
        if check_password_validity(password):
            valid_passwords.append(password)
    return valid_passwords

```