Python HomeWork Assignment 3

作业说明

请完成以下任务，内容基于条件语句与循环的基础知识。
每道题需包含:
1. 清晰的代码注释，说明解题思路。
2. 运行结果截图。
3. 答案文件需保存为 '.py' 格式并压缩提交。
作业题目设计注重实用性和逻辑性，请独立完成。

基础练习

1. **任务 1：进击的原神**

   在《原神》的游戏中，冒险家需要管理自己的物资以完成每日任务。

   - 任务要求每位冒险家每天至少携带 `F` 个体力恢复药剂。

   - 当前冒险家拥有的药剂数量为 `H`，每天从材料中可以合成 `C` 个新药剂。

   - 冒险家计划冒险 `D` 天。

   编写一个程序，计算:

   - 是否可以通过每天合成药剂满足需求;

   - 如果不满足，输出还需额外购买的药剂数量。

   示例输入:

   ```
   请输入每天所需药剂数量: 3
   请输入当前药剂数量: 5
   请输入每日合成药剂数量: 2
   请输入计划冒险天数: 4
   ```

   示例输出:

   ```
   合成后药剂不足! 还需购买 2 个药剂。
   ```

2. **任务 2：谁是奶龙**

在一场在线游戏中，团队需要选择一名“奶龙”（专职辅助）来为其他队友治疗。

- 每名玩家可以选择候选的“奶龙”名字并给出支持票数。

- 程序需要统计每名候选者的票数，并找出得票最多的“奶龙”。

编写一个程序，

- 用户输入所有玩家投票结果（格式为名字，以空格分隔，例如：`Alice Bob Alice Eve`）。

- 统计每个名字的得票数。

- 找出得票最多的候选者，并输出结果。

示例输入：

`请输入投票结果：Alice Bob Alice Eve Bob Alice`

示例输出：

`奶龙是 Alice，得票数：3。`

如果出现得票数相同的情况，输出所有得票最高的名字。

3. **任务 3：时代少年团**

时代少年团正在举办一场粉丝投票大赛，每位成员的粉丝需要通过累计投票来为自己的偶像争取第一名。

- 每位成员的票数每天会以某种增长率 `G%` 增加，初始票数为 `V`。

- 程序需要计算连续 `D` 天后，所有成员的最终票数，并找出票数最高的成员。

编写一个程序：

- 用户输入：成员的名字、初始票数 `V` 和每天的增长率 `G%`。

- 程序支持输入多位成员的信息，输入格式为 *姓名：初始票数：增长率*，以逗号分隔。

- 用户输入计划天数 `D`。

- 计算所有成员的最终票数，找出票数最高者。

示例输入：

`请输入成员信息：马嘉祺:100:10，丁程鑫:120:8，宋亚轩:150:5`

`请输入计划天数：5`

示例输出：

`宋亚轩以票数 191.44 位居第一。`

提示：

- 每日票数计算公式为：

当天票数 = 前一天票数 * (1 + 增长率 / 100)。

- 结果保留两位小数。

进阶练习

1. **任务：跳舞的线**

一个著名的音乐游戏《跳舞的线》中，玩家需要根据音乐节奏控制一条线穿越各种障碍。

假设游戏有以下规则：

- 游戏线段有初始长度 L（单位：米，由用户输入）。

- 每个游戏节拍（Turn）会有以下事件发生：

1. 如果音乐节奏快（随机事件）：线段会缩短 10%。

2. 如果玩家成功躲避障碍：线段会延长 A 米（由用户输入）。

3. 如果玩家撞上障碍：线段会缩短 B 米（由用户输入）。

- 游戏共进行 N 个节拍。

编写一个程序模拟该游戏过程，要求：

- 每个节拍，随机生成节奏（快或慢）和玩家状态（躲避成功或撞上障碍）。

- 按照规则计算每个节拍后线段的长度。

- 如果线段长度小于或等于 0，游戏结束，并提示“游戏失败，线段消失。”

- 如果游戏正常结束（所有节拍完成且线段长度仍大于 0），输出最终线段长度。

示例输入：

```
请输入初始线段长度（米）：10
请输入成功躲避障碍的奖励长度（米）：2
请输入撞上障碍的惩罚长度（米）：3
请输入游戏节拍数：5
```

示例输出（随机示例）：

```
第 1 节拍：快节奏，躲避成功，线段长度为 11.0 米
第 2 节拍：慢节奏，撞上障碍，线段长度为 7.0 米
第 3 节拍：快节奏，撞上障碍，线段长度为 3.3 米
第 4 节拍：慢节奏，躲避成功，线段长度为 5.3 米
第 5 节拍：快节奏，躲避成功，线段长度为 6.93 米
```

游戏结束，最终线段长度为 6.93 米。

提示：

- 使用 random.choice() 方法随机生成节奏和玩家状态。
- 使用循环模拟多个节拍，结合条件语句实现逻辑。
- 保留长度结果到小数点后两位。

预习任务：循环与控制语句

在下一次课程中，我们将学习 Python 的循环与控制语句。请提前预习以下内容：

1. while 循环

‘while‘ 循环用于在条件为 True 时重复执行代码块。

语法：

```
while 条件:
    执行的代码块
```

示例：

```
count = 0
while count < 5:
    print(count)
    count += 1
```

2. 防范无限循环

当循环条件一直为 True 时，会导致无限循环。

示例：

```
while True:
    print("这是一个无限循环！")
```

解决方法：确保循环条件在某个时刻会变为 False。

3. for 循环

‘for‘ 循环用于遍历一个序列。

语法：

```
for 变量 in 序列:
    执行的代码块
```

示例:

```
for i in range(1, 6):
    print(i)
```

4. break 与 continue

- ‘break‘：立即退出循环。
- ‘continue‘：跳过当前迭代，继续下一次循环。

示例:

```
for i in range(1, 6):
    if i == 3:
        break
    print(i)
```

预习任务：列表与元组

在下一次课程中，我们将学习 Python 中非常重要的数据结构——列表和元组。
请通过以下内容了解它们的定义、特点、基本操作及应用场景。

1. 列表（List）

列表是 Python 中最常用的数据结构之一，它是一个有序的、可变的元素集合。
特点:

- 列表中的元素可以是任意数据类型（整数、字符串、列表、元组等）。
- 列表是可变的，这意味着我们可以动态地修改列表中的内容。
- 列表中的元素按索引顺序存储，从 0 开始编号。

创建列表:

```
my_list = [1, 2, 3, 4]
empty_list = []
mixed_list = [1, "Python", 3.14]
```

常用操作:

- 访问元素：通过索引访问，例如 $my_list[0]$
- 修改元素：直接使用索引赋值，例如 $my_list[1] = 10$
- 添加元素：

```
my_list.append(5)  # 在列表末尾添加
my_list.insert(2, 20)  # 在索引 2 处插入 20
```

- 删除元素：

```
my_list.pop()  # 删除最后一个元素
my_list.remove(3)  # 删除值为 3 的元素
```

- 遍历列表：使用 for 循环。

```
for item in my_list:
    print(item)
```

2. 元组（Tuple）

元组是另一个重要的数据结构，与列表类似，但它是 ** 不可变的 **。
特点：

- 元组中的元素类型可以混合，与列表类似。
- 元组是不可变的，一旦创建后无法修改其内容。
- 元组的访问方式与列表相同，支持索引和切片操作。

创建元组：

```
my_tuple = (1, 2, 3)
single_element_tuple = (1,)  # 单个元素的元组需要逗号
empty_tuple = ()
```

常用操作：

- 访问元素：与列表相同，使用索引。

- 元组解包：

```
my_tuple = (1, 2, 3)
a, b, c = my_tuple
print(a, b, c)  # 输出 1 2 3
```

- 合并与重复：

```
new_tuple = (1, 2) + (3, 4)  # 合并元组
repeated_tuple = (1, 2) * 3  # 重复元组
```

3. 列表与元组的对比

- 可变性：列表是可变的，元组是不可变的。
- 性能：元组由于不可变，操作速度比列表略快。
- 用途：列表常用于需要动态增删改的场景；元组多用于固定结构的数据。

4. 示例代码

以下是一些简单示例，帮助理解列表和元组的用法：

```
# 列表操作示例
fruits = ["apple", "banana", "cherry"]
fruits.append("orange")
print(fruits)  # 输出 ['apple', 'banana', 'cherry', 'orange']

# 元组操作示例
coordinates = (10.0, 20.0)
x, y = coordinates  # 元组解包
print(f"x: {x}, y: {y}")
```

学习目标

通过本次预习任务，您需要掌握以下内容：

- 列表的定义、基本操作（增删改查）。
- 元组的定义及不可变特性。

- 列表与元组的适用场景及对比。
- 如何遍历列表和元组。

请在正式上课前，熟悉上述内容并运行代码示例，以便更好地理解课堂内容。

Made by Python 乐团