# Python 程序设计实验报告

## 实验一：python 程序基础

### 实验名称：字符串类型的应用

### 实验人员：[朱品赞 / 2410211124 / 24计科3班]

### 实验日期：[2025年 09 月 26 日]

## 一、实验目的

1. 掌握字符串的常见操作。
2. 掌握字符串格式方法 format。
3. 利用字符串类型解决实际问题。

## 二、实验内容及要求

1. **基础实验**
   - **任务 1**：改写文本进度条案例，输入一个值，比如 10，文本进度条就按 10% 的变化到 100%，输入 5，进度条就按 5% 的变化直到 100%。
   - **任务 2**：敏感词替换。输入敏感词，输入一句话，将其中的敏感词用 * 替换后输出这句话。

## 三、实验步骤与代码实现

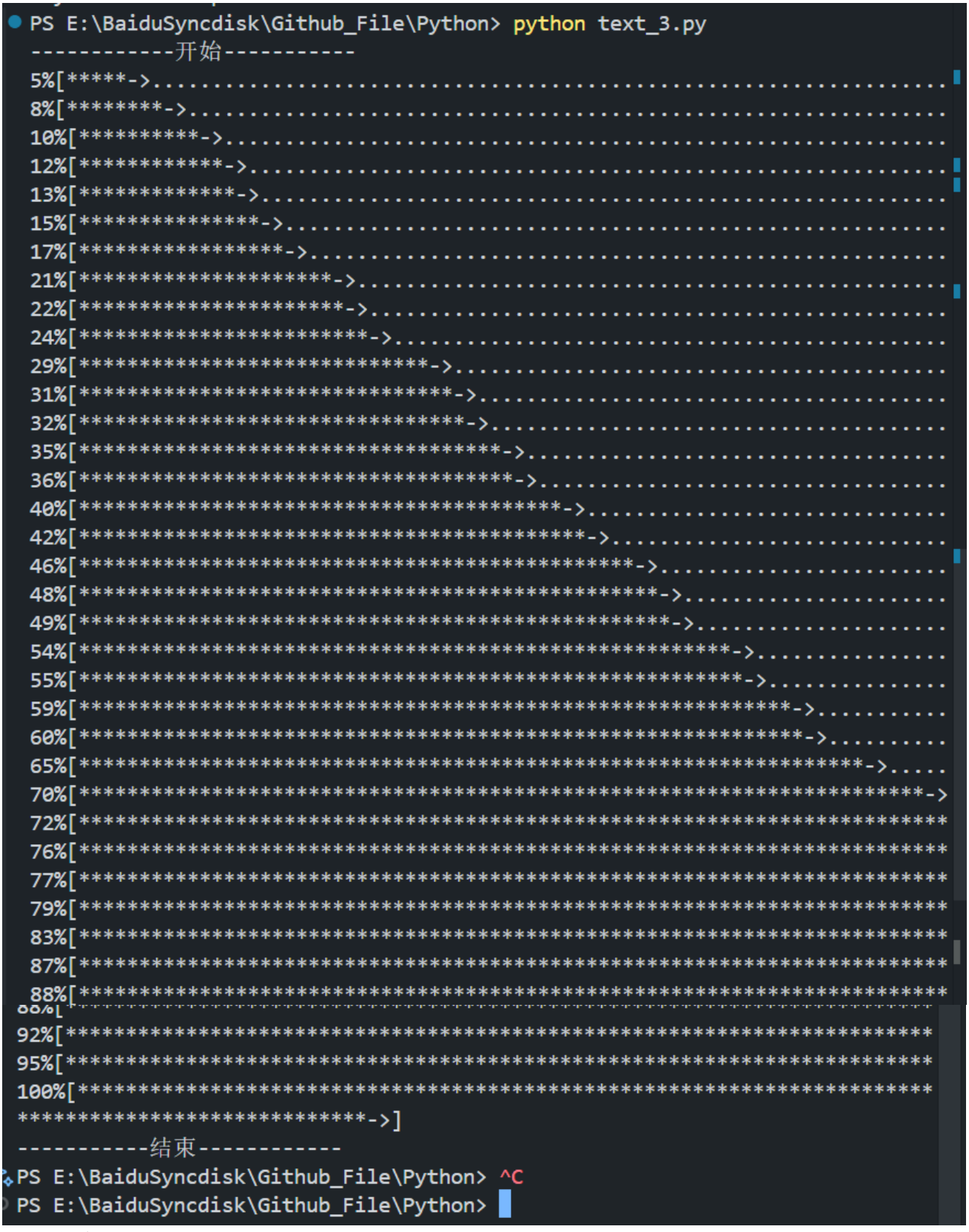
1. **任务 1**

```
1. **启动程序**：运行程序后，首先显示居中的 “开始” 提示，两侧用横线填充。
2. **初始化进度**：设置初始进度为 0%，准备开始进度更新。
3. **循环更新进度**：程序自动生成 1-5 之间的随机增幅，逐步累加进度值（超过 100% 时自动设为 100%）。
4. **动态显示进度条**：用星号 “*” 表示已完成部分，圆点 “.” 表示未完成部分，实时刷新显示当前进度百分比。
5. **控制更新速度**：每次进度更新后，随机暂停 0.2-0.8 秒，形成动态加载效果。
6. **完成进度**：当进度达到 100% 时，显示居中的 “结束” 提示，结束程序。
```

```python
import time
import random

print("开始".center(25, '-'))
```

```
progress = 0
while progress < 100:
    # 随机增长1-5个百分点
    increment = random.randint(1, 5)
    progress += increment
    if progress > 100:
        progress = 100

    b='*'*progress
    c='.'*(100-progress)
    print("\r{0}%[{1}->{2}]".format(progress,b,c),end='')

    # 随机延时0.2-0.8秒
    sleep_time = random.uniform(0.2, 0.8)
    time.sleep(sleep_time)
print()
print("{:-^25}".format("结束"))
```

**代码运行结果：

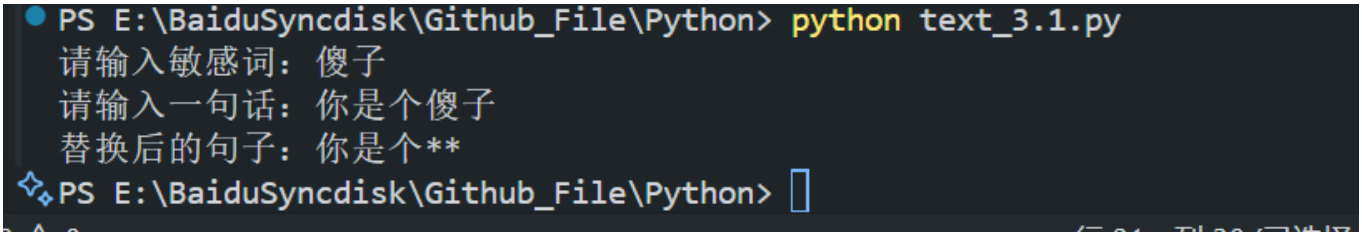



2. **任务 2**
3. **输入敏感词**：在程序提示下，手动输入需要屏蔽的敏感词（如 “违规”）。
4. **输入待处理句子**：根据提示，输入需要检测并替换敏感词的句子（如 “这是违规内容”）。
5. **生成匹配长度的星号**：程序自动计算敏感词的字符个数，生成相同长度的星号（如敏感词 “违规” 对应 “**”）。

6. **替换敏感词**：程序在输入的句子中，找到所有敏感词并替换成对应星号。
7. **查看结果**：程序直接输出替换后的句子，完成操作。

```python
# 敏感词替换程序
# 输入敏感词，输入一句话，将其中的敏感词用 * 替换后输出这句话

# 输入敏感词
sensitive_word = input("请输入敏感词：")

# 输入一句话
sentence = input("请输入一句话：")

# 替换敏感词为星号
# 计算敏感词的长度，用相同数量的星号替换
star_replacement = "*" * len(sensitive_word)

# 执行替换
result_sentence = sentence.replace(sensitive_word, star_replacement)

# 输出结果
print(f"替换后的句子：{result_sentence}")
```

**代码运行结果：




## 四、遇到的问题及解决

1. **任务 1 相关问题**：在编写进度条代码时，起初没有考虑到进度增长后可能超过 100% 的情况，导致进度数值异常。
   - **解决**：添加了 `if progress > 100: progress = 100` 的判断语句，确保进度始终在 0 到 100 之间。
2. **任务 2 相关问题**：输入敏感词和句子后，替换操作没有生效，检查发现是 `replace` 方法使用时，参数传递有误，误将敏感词变量名写错。
   - **解决**：仔细核对变量名，确保 `sentence.replace(sensitive_word, star_replacement)` 中 `sensitive_word` 是正确的敏感词输入变量。

## 五、实验总结

1. 熟练掌握了字符串的常见操作，如 `center` 用于字符串居中显示， `replace` 用于字符串替换等，以及字符串格式化方法 `format` 的使用，能灵活运用这些操作来构建所需的字符串输出。
2. 通过调试解决问题，提升了自己排查代码错误的能力，学会了在遇到问题时，从变量赋值、函数参数传递等基础环节逐步检查，定位并解决问题。
3. 后续需加强对 Python 中不同数据类型操作细节的掌握，以及在复杂逻辑场景下，代码的规范性和健壮性编写，比如在处理用户输入时，增加更多的异常处理机制，确保程序在各种输入情况下都能稳定运行。