上海大学

SHANGHAI UNIVERSITY

Python 计算实验报告

组　　号　第十组

实 验 序 号　**1**

学　　号　**22121630**

姓　　名　汪江豪

日　　期　**2025 年 3 月 20 日**

Python 计算实验一

一、实验目的与要求

1.熟悉 Python 的开发调试环境；

2.熟悉 Python 外部库的调用；

3.掌握 Python 语言基本语法；

4.熟悉 Python 的数据结构.

二、实验环境

1.操作系统 windows11

2. vscode 编辑器

三、实验内容

1、Python 代码理解 polygon.py：
（1）运行和阅读代码；
（2）理解代码功能；
（3）修改代码，练习调用文件中其他几个图形函数。

2、输入输出：编写脚本文件，设计友好的用户输入输出提示，用户输入一个时间（24 小时制，包含时、分、秒），输出 1 秒后的时间。

3、反序对：如果一个单词是另一个单词的反向序列，则称这两个单词为“反向对”。编写代码输出 word.txt 中词汇表包含的反向对。

4、文本分析算法设计：
（1）设计 Python 程序读入一个英文单词组成的文本文件，统计该文本文件中各个单词出现的次数。设计测试用例验证代码的正确性。
（2）设计 Python 程序读入一个英文单词组成的文本文件，统计其中包含的某给定关键词列表中各个单词出现的频率。设计测试用例验证代码的正确性。

四、实验内容的设计与实现

1. 该程序使用自定义包 swampy 实现了绘图功能。

• 主函数中，创建了一个 TurtleWorld 实例，用于绘制图形环境。创建了一个 Turtle 实例 bob，表示一个乌龟。

• square 函数实现了正方形的绘制，接收 turtle 对象和正方形边长 length。

• polyline 实现了绘制指定线段数的折线，接收 turtle，线段数 n，线段长度 length,旋转角度 angle。
• polygon 实现了正 n 边形的绘制。接收边数 n，边长 length。
• arc 实现了绘制圆弧，接收半径 r，弧度 angle
• 实现了圆的绘制，接收半径 r

2. 实现对时间的计算，关键点在于时间取余调整。
2.1 • 可以设计一个函数，接收 hour,minute,second 作为参数，计算下一秒的时间。
• 首先 second+1，判断如果为 60，second=0,minute+1.
• 接着判断 minute>59,为真则 minute=0，hour+1,
• 最后判断 hour>23,为真则 hour=0.
2.2 对于输入处理，可以输入纯字符串，然后进行如下处理：

```
    hour, minute, second = map(int, input_time.split())
```

通过切分，转换为 3 个 int 数据即可。
2.3 可添加 try except 异常处理，对于非法输入，抛出异常，提示按照 HH MM SS 格式输入。

3. 本题计算反序对和回文单词，分别统计并计算运行时间。
3.1 对于数据的读取，使用 python 内置的 open,read 读取 txt 文件，将所有单词放到一个列表 words 中，使用 splitlines 对不同单词进行分割。
3.2 对于回文单词的判断，遍历列表每个单词 word，使用切片操作进行字符串反转，如下：

```
        reversed_word = word[::-1]
```

判断如果和原字符串相等，即为回文单词，添加到答案中，最后返回答案列表。
3.3 对于反序对的判断。可以先将 words 放入集合中。因为集合使用哈希方式存储，查找效率为 O(1)。

```
    word_set = set(words) # 将单词列表转换为集合，提高查找效率
```

同样使用切片操作反转每个单词，结果放入 reversed_word，判断如果 reversed_word 在集合 word_set 中，并且本身不为回文单词，则添加答案。之后需要从集合中删除原单词和反转后的单词，避免重复查找。
3.4 对于判断函数运行时间部分，可以写一个函数装饰器，

```
# 定义装饰器
def timecal(func):
    def wrapper(*args, **kwargs):
        start_time = time.time()
        result = func(*args, **kwargs)
        end_time = time.time()
        print(f'{func.__name__}运行时间: {(end_time - start_time) * 1000}ms')
        return result
    return wrapper
```

用来计算每个功能函数的运行时间。只需要在每个功能函数开头添加@[装饰器名称]即可。这样在调用原功能函数名时，自动使用装饰器方法来计算函数运行时间。

4. 对于两个小问的要求，可以放入一个函数中实现。添加一个参数并初始化 key_words=None。

可兼容用户给定关键词列表或直接查询所有单词出现次数。

4.1 对于文件内容的读取，使用正则表达式分割单词。\b 匹配每个单词的开头和结尾，\w+匹配一个及以上若干数字和字母和下划线，结果存入列表 words。

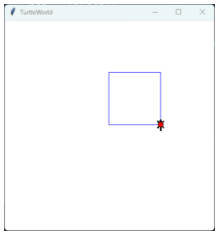
```
        words = re.findall(r'\b\w+\b', file.read().lower())
```

4.2 直接 Counter(words)统计所有单词出现次数。对于有关键词列表的，使用列表推导式即可，如下：

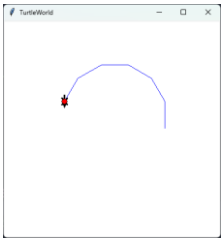
```
        keyword_counts = Counter(word for word in words if word in keywords)
```

五、测试用例

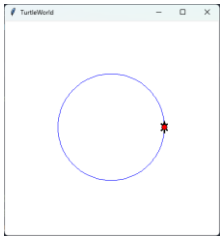
1. square(bob, 100)



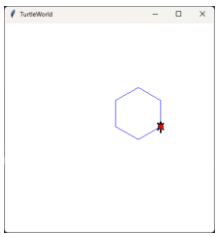
Polyline(bob, 6, 50, 30)



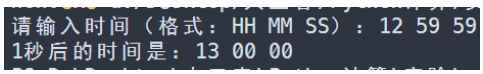
Circle(bob, 100)



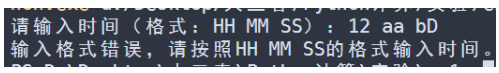
Polygon(bob, n=6, length=50)



2. 合法输入：



非法输入：



3. 反序对函数运行时间：

```
find_reverse_pairs运行时间： 30.26866912841797ms
[('abut', 'tuba'),
 ('ad', 'da'),
 ('ados', 'soda'),
```

回文单词判断函数运行时间：

```
is_reversed_word运行时间： 8.179664611816406ms
['aa',
 'aba',
 'aga',
```

4. 测试文本：

```
Hello world
Hello Python
This is a test file for word counting
Python is great for data analysis
Count the words in this file
```

结果(1):

```
Counter({'hello': 2,
         'python': 2,
         'this': 2,
         'is': 2,
         'file': 2,
         'for': 2,
         'world': 1,
         'a': 1,
         'test': 1,
         'word': 1,
         'counting': 1,
         'great': 1,
         'data': 1,
         'analysis': 1,
         'count': 1,
         'the': 1,
         'words': 1,
         'in': 1})
```

结果(2):

关键词列表：　　keywords = ['hello', 'python']

结果：

```
{'hello': 0.5, 'python': 0.5}
```

六、收获与体会

本次实验完成后，我加深了对 python 基础数据结构的理解。包括列表、字典、集合，以及在 python 中万物皆为对象。且 python 在变量定义是不用显示声明类型，根据值自动推导变量类型。且 python 中有强大的字符串处理函数和庞大的第三方库支持。比如 map 转换序列类型，split()分割字符串，splitline()按行分割，re.findall()可使用正则表达式分割文本内容，Counter 容器统计元素数量，set 集合以哈希方式存储元素，具有极高的查找效率等。

在 python 中，对于字符串的切片操作也极为强大，比如 word[::-1],指定从结尾开始反转字符串等。这些函数极大地提升了处理效率。

Python 语言的高抽象性，极大地便利了初学者，但其对底层实现原理进行了封装，不太利于程序员理解其底层原理。总之，今后对于 python 的学习，不仅要了解其功能，还要多多

关注函数底层的实现原理。