第 4 周练习题目

一. 记录日志装饰器练习题

好的日志对一个软件的重要性是显而易见的。如果函数的入口都要写一行代码来记录日志，这种方式实在是太低效了。 那么请你创建一个装饰器， 功能实现函数运行时自动产生日志记录。 日志格式如下:

程序运行时间 主机短名 程序名称: 函数[%s]运行结果为[%s]

产生的日志文件并不直接显示在屏幕上， 而是保存在 file.log 文件中， 便于后期软件运行结果的分析.

二. 斐波那契数列的装饰器练习：实现高速缓存递归

在数学上，**费波那契数列**是以递归的方法来定义：

$$F_0 = 0.$$

$$F_1 = 1.$$

$$F_n = F_{n-1} + F_{n-2}, n \geq 2.$$

用文字来说，就是费波那契数列由 0 和 1 开始，之后的费波那契系数就是由之前的两数相加而得出。首几个费波那契系数是：

0,1,1,2,3,5,8,13,21,34,55,89,144,233.....（OEIS 中的数列 A000045）

装饰器 1： 添加高速缓存的装饰器 num_cache

如果第一次计算 F(5) = F(4) + F(3) = 5

第二次计算 F(6) = F(5) + F(4)

显然 F(5)已经计算过了， F(4)也已经计算了，我们可否添加一个装饰器， 专门用来存储费波那契数列已经计算过的缓存， 后期计算时， 判断缓存中是否已经计算过了， 如果计算过，直接用缓存中的计算结果．如果没有计算过， 则开始计算并将计算的结果保存在缓存中．

装饰器 2： 程序运行计时器的装饰器 timeit

该装饰器用来测试有无高速缓存求斐波那契数列， 它们两者运行的时间效率高低．

```python
@num_cache
@timeit
def fibonacci(n):
    if(n<=2):
        return 1
    return fibonacci(n-1)+fibonacci(n-2)


@num_cache
def fibonacci(n):
    if(n<=2):
        return 1
    return fibonacci(n-1)+fibonacci(n-2)
```

三. Leetcode 字符串练习题目：比较版本号

本题难度较高， 可网络查询思路， 但务必要能理解其思路并自行编写代码

题目需求:

比较两个版本号 *version1* 和 *version2*。
如果 *version1* > *version2* 返回 `1`，如果 *version1* < *version2* 返回 `-1`，除此之外返回 `0`。

你可以假设版本字符串非空，并且只包含数字和 `.` 字符。

`.` 字符不代表小数点，而是用于分隔数字序列。

例如，`2.5` 不是"两个半"，也不是"差一半到三"，而是第二版中的第五个小版本。

你可以假设版本号的每一级的默认修订版号为 `0`。例如，版本号 `3.4` 的第一级（大版本）和第二级（小版本）修订号分别为 `3` 和 `4`。其第三级和第四级修订号均为 `0`。

测试案例:

示例 1:

```
输入: version1 = "0.1", version2 = "1.1"
输出: -1
```

示例 2:

```
输入: version1 = "1.0.1", version2 = "1"
输出: 1
```

示例 3:

```
输入: version1 = "7.5.2.4", version2 = "7.5.3"
输出: -1
```

示例 4：

```
输入：version1 = "1.01", version2 = "1.001"
输出：0
解释：忽略前导零，“01” 和 “001” 表示相同的数字 “1”。
```

示例 5：

```
输入：version1 = "1.0", version2 = "1.0.0"
输出：0
解释：version1 没有第三级修订号，这意味着它的第三级修订号默认为 “0”。
```

提示：

1). 版本字符串由以点（.）分隔的数字字符串组成。这个数字字符串可能有前导零。

2). 版本字符串不以点开始或结束，并且其中不会

四. 模块与包练习题： 微信好友数据分析与展示

1. 已知 itchat 可以获取好友的信息， 此处统计好友的省份分布；

2. 获取分布好友最多的 5 个省份；

3. 将省份分布的数量基于 pyecharts 模块以条形图的方式展示;

附加需求: 将每个省份的好友备注名(RemarkName)存入依次存入对应省份的文件中；

e.g.

文件： 陕西省.txt

文件： 山东省.txt

4. 将上述 编写的代码封装为模块， 并实现模块的制作与发布;