

## 实验三 PYTHON 数据结构与函数定义

### 一、目的和要求

1. 熟悉 Python 的函数定义;
2. 熟悉 Python 的数据结构;
3. 掌握 Python 语言基本语法;

### 二、实验环境

1. Win 7 操作系统;
2. Python 2.7.X, IDLE、PyCharm 等开发环境;

### 三、实验内容

#### (一) 验证实验 (每个同学完成)

1. 运行调试第二章各小节例示代码及课后练习的程序设计题, 检查运行结果是否正确, 记录实验结果。

2. 运行调试第五章各小节例示代码及课后练习的程序设计题, 检查运行结果是否正确, 记录实验结果。

3. 阅读和运行 `reversepair.py`:

(1) 比较该代码和你的小组在实验二中所完成的反序词代码。

(2) 阅读和自学 Python 文档中模块 `bisect` 的使用方法。

4. 函数的作用:

(1) 编写一个函数, 绘制如下的表格。

```
+-----+-----+
|       |       |
|       |       |
|       |       |
+-----+-----+
|       |       |
|       |       |
|       |       |
+-----+-----+
```

比较 Python2.x 和 Python3.x 中 `print` 的差异。

(提示, 同一行打印多个值, 可以使用逗号分隔不同的值: `print '+', '—'`; 如果

值序列的结尾有一个逗号，则 Python 输出不会换行，如：

```
print '+',  
print '—'
```

两个语句的输出是 ‘+—’)

(2) 阅读和运行 grid.py (支持 Python2.x, 请修改代码使得 Python3.x 下同样可以正常运行), 分析 grid 所实现功能中各个函数之间的调用关系, 绘制这种调用关系的流程图 (可用 Visio 等软件绘制, 流程图放到实验报告中)。比较 grid.py 的实现方法和 3 (1) 中你所实现的绘制表格函数的差异, 并且把学习体会写在实验报告中。

## (二) 设计实验 (小组完成, 代码提交, 算法设计和测试写入实验报告)

5. 编写 Ackermann 函数的递归实现 Ack (m,n)

$$A(m,n) = \begin{cases} n + 1 & \text{if } m = 0 \\ A(m-1, 1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1, A(m, n-1)) & \text{if } m > 0 \text{ and } n > 0. \end{cases}$$

测试 Ack (3, 4) 的值, 阅读 [https://en.wikipedia.org/wiki/Ackermann\\_function](https://en.wikipedia.org/wiki/Ackermann_function), 分析 m 和 n 取值对函数值计算的影响, 深入理解递归。

6. 计算 N 名同学生日相同的概率有多大。提示: 用 input 接收 N, 用 random 模块中的 randint 生成随机数作为这 N 名同学的生日, 模拟 1000 次, 计算出现了相同生日情况的次数 M, 用 P=M/1000 作为对相同生日概率的估计, 分析 N 和 P 之间的关系。

## (三) 综合实验 (小组完成, 代码提交, 算法设计和测试写入实验报告)

7. 参照验证实验 3 中反序词实现的例示代码, 设计 Python 程序找出 words.txt 中最长的“可缩减单词”(所谓“可缩减单词”是指: 每次删除单词的一个字母, 剩下的字母依序排列仍然是一个单词, 直至单字母单词‘a’或者‘i’)。提示:

(1) 可缩减单词例示:

sprite —> spite —> spit —> pit —> it —> i

(2) 如果递归求解, 可以引入单词空字符串”作为基准。

(3) 一个单词的子单词不是可缩减的单词, 则该单词也不是可缩减单词。

因此, 记录已经查找到的可缩减单词可以提速整个问题的求解。