
# Python 计算实验报告

| | |
|---|---|
| 组　　号 | 第 8 组 |
| 实验序号 | 2 |
| 学　　号 | 19120191 |
| 姓　　名 | 汪雨卿 |
| 日　　期 | 2020 年　4 月　22 日 |

## 一、 实验目的与要求

1. 熟悉Python 的开发调试环境；
2. 熟悉Python 的流程控制；
3. 熟悉Python 的数据结构；
4. 掌握Python 语言基本语法；

## 二、 实验环境

1. 操作系统不限；
2. Python IDLE、PyCharm 等开发环境不限。

## 三、 实验内容

1. Python 流程控制：编写循环控制代码用下面公式逼近圆周率(精确到小数点后 15 位)，并且和 math.pi 的值做比较。

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{9801} \sum_{k=0}^{\infty} \frac{(4k)!(1103 + 26390k)}{k!^4 (396^{4k})}.$$

2. Python 流程控制：阅读 https://en.wikipedia.org/wiki/Koch_snowflake，通过修改 koch.py 绘制其中一种泛化的 Koch 曲线。

3. 生日相同情形的概率分析：

1）生成 M（M>=1000）个班级，每个班级有 N 名同学，用 input 接收 M和 N；

2）用 random 模块中的 randint 生成随机数作为 N 名同学的生日；

3）计算 M 个班级中存在相同生日情况的班级数 Q，用 P=Q/M 作为对相同生日概率的估计；

4）分析 M，N 和 P 之间的关系。

4. 参照验证实验1 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：**

可缩减单词例示：**sprite —〉spite —〉spit—〉pit—〉it—〉i**

如果递归求解，可以引入单词空字符串’’作为基准。

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。

## 四、 实验内容的设计与实现

**题目 1：**

Python 流程控制：编写循环控制代码用下面公式逼近圆周率(精确到小数点后 15 位)，并且和 math.pi 的值做比较。

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{9801} \sum_{k=0}^{\infty} \frac{(4k)!(1103 + 26390k)}{k!^4 (396^{4k})}.$$

**【技术重点】**

- math 库。

  利用 math 库中的阶乘，指数的方式，以及 for 循环的方式实现题目所给出的公式。
- 引入 decimal 库设置小数的长度

  简单了解，decimal 库可以实现浮点数的精确计算，也可以利用其中的函数控制小数的个数。

**【结果】**

所实现的 pi 的计算结果 15 位小数和 math.pi 给出的结果一致精确度较高。同时，对于 k 的大小和计算所需的时间进行判断。当 k<1000 时，用时在 1s 左右；当 k=3000 及以上时，所需的时间成 2 到 3 倍上升。

【代码实现】

```
'''2.0 利用math库，精确度提升'''
def Pai(max):
    getcontext().prec = 16  # 设置数有几位，由于个为有1位，小数15位，故设置一共16位
    k = 0
    m = 2 * math.sqrt(2) / 9801
    sum = 0
    while k < max:
        top = math.factorial(4 * k) * (1103 + 26390 * k)
        bottom = math.factorial(k) ** 4 * (396) ** (4 * k)
        sum = sum + top / bottom
        k += 1
    sum = 1 / (sum * m)
    return sum
```

**题目 2：**

Python 流程控制：阅读 https://en.wikipedia.org/wiki/Koch_snowflake，通过修改 koch.py 绘制其中一种泛化的 Koch 曲线。

**【技术重点】**

- turtle 库。

  利用 turtle 库先绘制所需要的图像。熟悉了 turtle 库常用的函数。

  + turtle.pensize()：设置画笔的宽度；

  + turtle.pencolor()：设置画笔的颜色；

  + turtle. penspeed()：设置画笔的速度（0-10）。

**(1) 画笔运动命令**

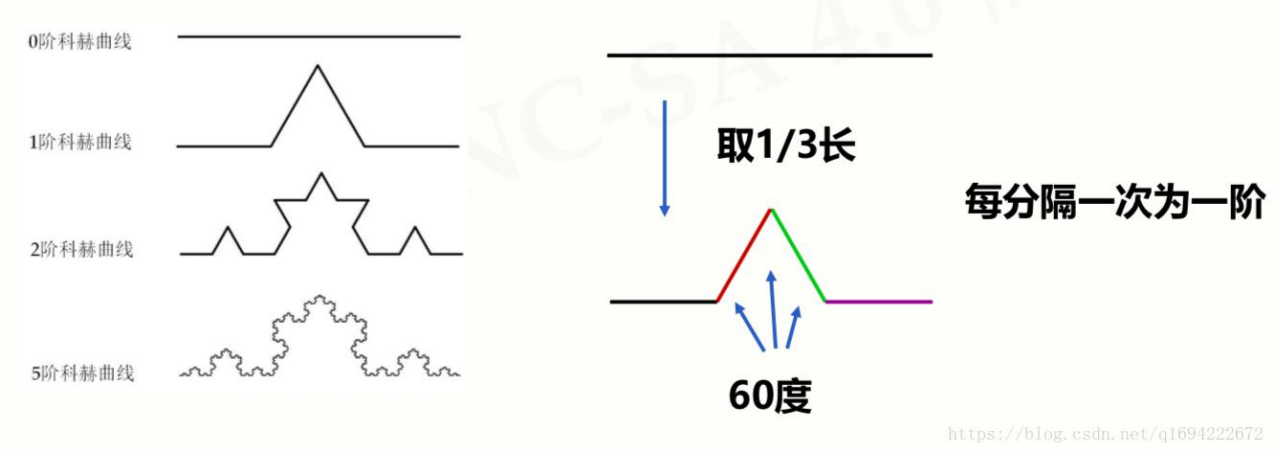
| 命令 | 说明 |
| --- | --- |
| turtle.forward(distance) | 向当前画笔方向移动distance像素长度 |
| turtle.backward(distance) | 向当前画笔相反方向移动distance像素长度 |
| turtle.right(degree) | 顺时针移动degree° |
| turtle.left(degree) | 逆时针移动degree° |
| turtle.pendown() | 移动时绘制图形，缺省时也为绘制 |
| turtle.goto(x,y) | 将画笔移动到坐标为x,y的位置 |
| turtle.penup() | 提起笔移动，不绘制图形，用于另起一个地方绘制 |

- 科赫雪花定义：

设想一个边长为 1 的等边三角形，取每边中间的三分之一，接上去一个形状完全相似的但边长为其三分之一的三角形，结果是一个六角形。现在取六角形的每个边做同样的变换，即在中间三分之一接上更小的三角形，以此重复，直至无穷。外界的变得原来越细微曲折，形状接近理想化的雪花。

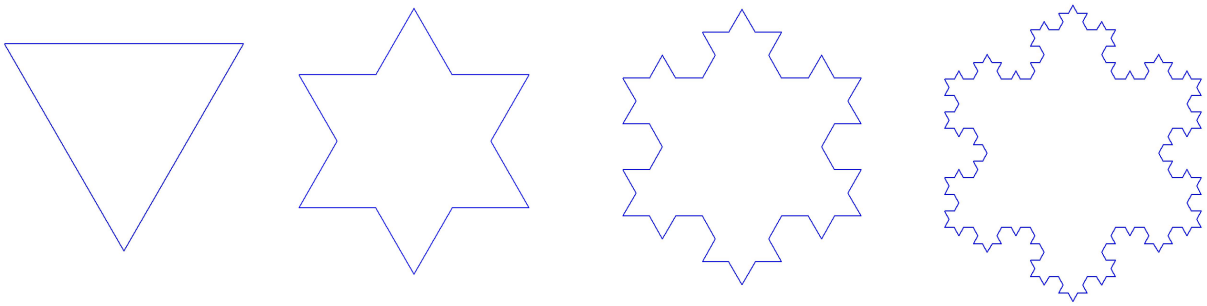
【实现原理】

先定义单边的科赫曲线，在 0 阶的时候科赫曲线时一条直线；当阶数大于 0 的时候，每条边增加一个折，即一条边变为 3 条边。可以利用递归算法实现。




【结果】

可以实现自定义阶数的科赫雪花。以下仅展示 0~4 阶的科赫雪花图样。(N 表示阶数)



N = 0　　N = 1　　N = 2　　N = 3

【代码实现】

```python
# 科赫雪花
import turtle


def kehe(len, n):
    if n == 0:
        turtle.fd(len)  # 0阶时，走直线，长度为len
    else:
        for i in [0, 60, -120, 60]:  # 依次向左转指定角度调用n-1阶的kehe函数
            turtle.left(i)
            kehe(len / 3, n - 1)


def Koch(level):
    turtle.speed(0)  # 设置乌龟的速度
    turtle.penup()  # 拿起画笔
    turtle.goto(-200, 100)  # 设置初始点（-100，100）
    turtle.pensize(2)  # 设置笔的粗细
    turtle.color('blue')  # 设置笔的颜色为蓝色
    turtle.pendown()  # 放下画笔

    for i in range(3):
        kehe(lenth, level)  # 转向画3次
        turtle.right(du)

    turtle.hideturtle()  # 隐藏乌龟（即箭头）
    turtle.done()


if __name__ == '__main__':
    lenth = 500
    du = 120
    level = (input('please enter a number:'))
    level = int(level)
    Koch(level)
```

**题目 3：**生日相同情形的概率分析：

1）生成 M（M>=1000）个班级，每个班级有 N 名同学，用 input 接收 M和 N；

2）用 random 模块中的 randint 生成随机数作为 N 名同学的生日；

3）计算 M 个班级中存在相同生日情况的班级数 Q，用 P=Q/M 作为对相同生日概率的估计；

4）分析 M，N 和 P 之间的关系。

【技术重点】

- Numpy 中的 random 库

  利用 random 库中的 randint 函数构造同学的生日。

  + random(start, stop, step)

- 二维数组的创建方法。

  （避免了一个 list 复制，导致修改某个单元引发其他元素同步修改的情况）

  + 直接创建法

  + 列表生成法

  + numpy 创建法

```
# 直接
test = [0, 0, 0], [0, 0, 0], [0, 0, 0]]
# 列表
test = [[0] * n for _ in range(m)]
# numpy
import numpy as np
test = np.zeros((m, n), dtype=np.int)
```

- **Matplotlib.pyplot 绘制函数图像，通过控制变量的方式，查看数据变化的趋势。**

【实现原理】

1）初始化

首先利用列表生成法创建一个 m * n 的二维矩阵(m 表示班数,n 表示人数)。再利用 randint 对每个班的 n 个学生在（1-365）中随机赋值，模拟一个人的生日。

2）判断一个班是否存在两个生日相同的人。

利用 set 元组中不存在重复元素的特点，比较每个班原来 list 列表和 set 之后的 set 集合的长度。如果被元组化之后的列表长度发生改变，则表明原来的班级中存在生日相同的同学。设置一个计数器记录班级总数。

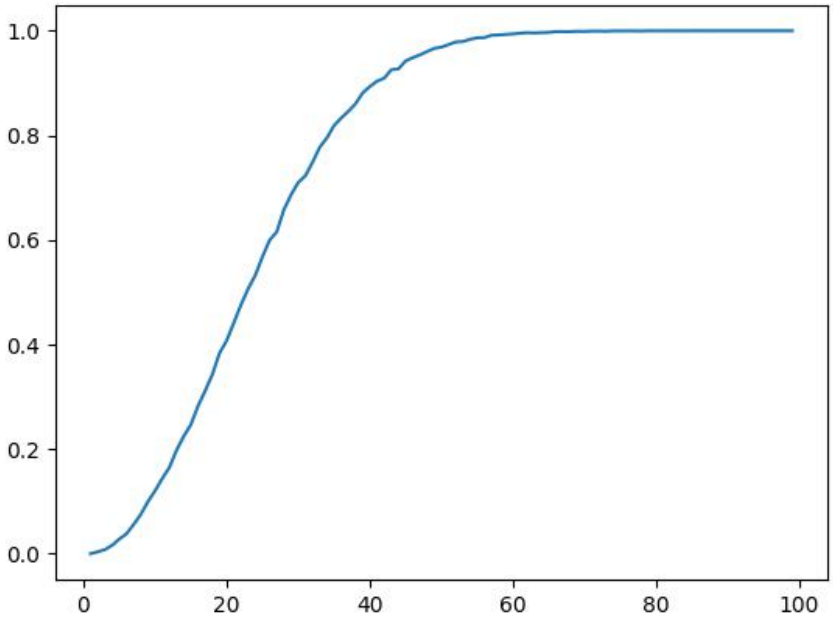
3）输出概率

输出计数器/m 的值，即是所求的概率。

【结果】

显示图像为 班级总数：10000 个，各班人数从 1~100 变化输出的重复生日人数的概率。横坐标为班级人数；纵坐标为有重复生日班级的概率。

从图像分析可得，班级人数越多，可能有重复生日的同学概率越高。当人数超过一定人数之后，重复的概率为 1。




【代码实现】

```
def Birthday(m, n):
    '''创建m个班级，每个班级n个学生，返回m个班中有存在相同生日数的概率'''
    try:
        while 1:
            count = 0
            if m < 1000 or n < 0:  # 判断输入的m和n是否正确
                print("m or n is not correct.")
                break
            my_class = [[0] * n for _ in range(m)]  # 创建m行n列的二维数组，并且全部初始化为0
            for i in range(m):  # 随机生成生日
                for j in range(n):
                    my_class[i][j] = random.randint(1, 366)

            for i in range(m):   # 利用元组化列表后集合的长度变化，判断是否一个班有同样生日的学生。
                if len(my_class[i]) != len(set(my_class[i])):
                    count += 1
            print(count / m)
            break;
    except ValueError:
        print("The number you enter is not the right type.")
```

**题目 4：**

参照验证实验1 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：**

可缩减单词例示：sprite —〉spite —〉spit—〉pit—〉it—〉i

如果递归求解，可以引入单词空字符串’’作为基准。

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。

**【技术重点】**

- 数据结构的选择
  + 优先考虑字典/集合的存储方式，由于集合不能涵盖两个相同的元素。故选择字典。
  + 利用列表传入从文件中读取的每个单词
- 算法的构思：
  + 参考了同学基于树的深度搜索的思想，探索实践完成了本题的代码。
- 对于 n 位的单词，如何存储随机减少一位后产生的 n-1 位的单词
  + 想法 1 利用 string.replace()对长度内的每个位置的字母进行遍历，并替换成“”。
  在调试中，发现问题。由于 python 基于内存的管理方式，replace 会将字符串中出现的所有相同的字母都替换成“”，导致结果异常。
  例：apple 删除第 2 位的’p’会得到’ale’
  + 想法 2 利用切片分割字符串，再拼接。每次取[:i]+[i+1:]的字符，i 在字符串长度内遍历。注意，切片的范围是开闭区间。经过测试，该方法可以避免上述问题的产生，故使用切片方法。

【实现原理】

1）初始化

从 word.txt 中读取文件生成一个 t 列表，在使用 t 列表生成一个对应的字典 t1。t1 中农的每个 value 值初始化为 0。

2）getwords(w)函数

参数：w 是传入的单词。

返回值：t1.get(w,-1) 在字典中查找单词 w，找到返回对应的 value 值，否则返回

-1。

i. 判断传入的单词的长度是否为 1，为 1 则在字典中将它的 value 设置为 1，返回 t1.get(w,-1)；

ii. 对于单词长度大于 2 的情况,先判断是否能在字典中找到它，若找得到，则生成它对应的切片。

iii. 对于切片中的每个元素，重复 i,ii 两步

iv. 判断元素在字典中的值是否大于 0，即是否存在从原始 n 位单词到最后一个字母均为单词的情况。若如此，则把它的 value 值置为：它的字符串长度，把它上一级的值也置为对应的字符串长度。

v. 排序输出结果

利用 max 函数和 lambda()函数输出最大值的结果。

**【结果】**

找到结果：complecting

用时 Time: 1.9192843437194824

【代码实现】

```
def getwords(w):
    """
    :param w: 单词（字符串）
    :return: t1.get(w,-1) 在字典中查找单词w，找到返回对应的value值，否则返回-1.
    """
    if len(w) == 1:  # 只有单词只有一个字母时
        t1[w] = 1
    elif t1.get(w, -1) == 0 and len(w) > 0:  # 单词大于一个字母时
        # minus = [w.replace(w[i], "") for i in range(len(w))]  # 把每一个减少一位的单词存入数组再查找
        minus = [w[:i] + w[i + 1:] for i in range(len(w))]  # 把每一个减少一位的单词存入数组再查找
        # print(w, '的子单词：', minus)

        # 对于每一个子单词，判断子单词在字典中，继续查找
        for k in minus:
            getwords(k)
            if t1.get(k, -1) > 0:
                t1[k] = len(k)
                t1[w] = len(w)
    return t1.get(w, -1)
```

```
with open('words.txt') as f:
    # dict1 = {x.strip() for x in f}
    # t = ['apple', 'split', 'spit', 'pit', 'it','abc','ab','ap','sit']
    # t1 = {x: 0 for x in t}

    start = time.time()
    t = [x.strip() for x in f]
    t1 = {x.strip(): 0 for x in t}
    for i in t:
        getwords(i)
    # print('change dict:', t1)
    for key in max(t1.items(), key=lambda x: x[1]):
        print(key)
    print('Time:', time.time() - start)

'''complecting
 11
Time: 1.9192843437194824'''
```

## 五、 测试用例

题目 4 使用了附件中提供的 word.txt 文档，作为字典来源。

## 六、 收获与体会

本次实验的四道题目，难度是成梯度上升的，在整体完成的过程中有感觉到在学习完基础语法之外，还需要大量的习题练习和不断学习他人写的较好的代码。这样才能对于自己的代码能力有所提升。

这次的习题一，虽然是一道较为简单的条件控制题，但却给我留下了很深的印象。因为，最开始拿到这道题的时候，第一反应并不是调用 Python 自带的 math 库中所含有的阶乘，指数运算等等的数学计算方法，而是使用了最初级的通过 for 循环等方式实现。还引入了 decimal 的计算库。但是最后所实现的精度，并没有很高。在完成实验之后，和小组成员讨论之后，才意识到可以使用 math 库自带的方法，并且也发现自己对于基础库所含有的方法并没有很高的熟悉度。因此，也借此机会熟悉了一下 math 库的使用。

习题二使用的是 Python 的 turtle 库，能够完成一些有图形输出效果的程序是较为趣味的。尤其是熟悉了 turtle 库的简单使用，可以通过简单的指令，更改图像的颜色；通过代码，控制边数等等。这个题目可以说很算是寓教于乐了，在编程之余学习了数学知识，也收获了趣味性和成就感。

习题三应该是基于一个很著名的概率论的题目：生日悖论所改编。整体代码实现的过程中，我学习到了 Python 创建二维数组的方式，粗略的了解了 numpy 库的一部分功能。在完成了题目所要求的

内容之余，我还上网搜索了 Python 绘制坐标的方法。通过现有的数据，通过控制变量的方法绘制了一个概率趋势的坐标图。所得到的结果也和悖论所阐述的结果相一致。本题中，有一个代码的处理方法，让我体会到了 Python 数据结构中转换的灵动。就是通过集合和列表的转换判断是否有相同的成员。最开始我并没有很好的想法，如何处理这部分的内容。甚至第一反应是查找和对比。然后，在参考一些题解的过程中，我了解到了这个方法。它很好的利用了集合中不存在重复元素的特征，通过转换后两者的长度是否存在变化，轻松的处理了这个问题，非常值得学习参考。

最后一个题，也是这次实验难度较高的一道题目。在编写题目之初，我和小组成员在有了自己的想法之后也互相交流果一些思想。但想实现的东西，总存在一些欠缺，导致代码一直存在 bug。之后我们和班上的一个同学讨论过后，我参考了他的一些处理方法，完善了自己的代码问题，最后写出了这个代码。判断整体的处理速度也是相对较快的。

这次的实验 2，相对上次的实验难度更大。也增加了一部分对于算法能力的要求。完成这部分的实验，让我对 Python 的语法和功能使用有了更深入的了解。