上海大学

SHANGHAI UNIVERSITY

Python 计算实验报告

组　　号	第 2 组
实验序号	2
学　　号	21120860
姓　　名	刘虚谷
日　　期	2023 年 4 月 27 日

Python 计算实验报告撰写提纲

一、实习目的与要求

1. 熟悉 Python 的流程控制；
2. 熟悉 Python 的数据结构；
3. 掌握 Python 语言基本语法

二、实习环境

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

三、实习内容

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
。
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：
（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。

四、实习内容的设计与实现

1.

```
def digui(k):
    m=1
    for i in range(1,k+1):
        m*=i
    return m
def qiouhe(k):
    return (digui(4*k)*(1103+26390*k))/((digui(k)**4)*(396**(4*k)))
x=int(input('请输入 k:'))
y=0
for i in range(x+1):
    y=y+qiouhe(i)
pi=9801/(2*(2**0.5)*y)
print(pi)
```

第一题的代码很简单，定义了一个迭代函数 digui()用来计算 k 的阶层。

2.

```
def koch(t, n):
    """Draws a koch curve with length n."""
    if n<4:
        fd(t, n)
        return
    m = n/3.0
    koch(t, m)
    lt(t, 90)
    koch(t, m)
    rt(t, 90)
    koch(t, m)
    rt(t, 90)
    koch(t, m)
    lt(t, 90)
    koch(t, m)

def snowflake(t, n):
    """Draws a snowflake (a triangle with a Koch curve for each side)."""
    for i in range(2):
        koch(t, n)
        rt(t,180)
```

第二题我对 koch 曲线的代码做了一些小的改动，画出了 2 个大的二次型 1 曲线，与 koch 曲线的思路一样，利用 向前->左转 90 度->向前->右转 90 度->向前->右转 90 度->向前->左转 90 度->向前 的基本步骤递归似的画出了由 2 个二次型 1 曲线组成的正方形。

3.

```
x=input('请输入 M 和 N:')
import re
x=list(map(int,re.split(' ',x)))
m,n=x[0],x[1]
print(m,n,end='\n')
import random
a={1:31,2:28,3:31,4:30,5:31,6:30,7:31,8:31,9:30,10:31,11:30,12:31}
b=[[(y:=random.randint(1,12),random.randint(1,a[y])) for i in range(n)] for j in range(m)]
print(b)
q=0
for item in b:
    len1=len(item)
    len2=len(set(item))
    if len1!=len2:
        q=q+1
p=q/m
print(p)
```

第三题我利用字典存储每一个月最大的天数，并利用 random 模块以及列表推导式的嵌套创造了各有 N 个随机生日的 M 个班级，之后我利用了集合计算出含有相同生日的班级的个数。

4.

```
def is_reducible(word, word_dict, reducible_dict):
    # 判断一个单词是否可缩减
    if word in reducible_dict:
        return reducible_dict[word]
    if word in ['i', 'a']:
        reducible_dict[word] = True
        return True
    # 遍历单词中的每个字母，在单词列表中查找子单词是否可缩减
    for i in range(len(word)):
```

```
            child_word = word[:i] + word[i+1:]
            if child_word in word_dict and is_reducible(child_word, word_dict, reducible_dict):
                reducible_dict[word] = True
                return True
        reducible_dict[word] = False
        return False

def find_longest_reducible_word(word_list):
        # 在单词列表中查找最长可缩减单词
        word_dict = set(word_list)
        reducible_dict = {}
        longest_reducible_word = ''
        # 遍历所有单词，判断是否可缩减，更新最长可缩减单词
        for word in word_dict:
            if is_reducible(word, word_dict, reducible_dict):
                if len(word) > len(longest_reducible_word):
                    longest_reducible_word = word
        return longest_reducible_word

if __name__ == '__main__':
        with open('words.txt', 'r') as fin:
            word_list = [line.strip() for line in fin]
        longest_reducible_word = find_longest_reducible_word(word_list)
        print('The longest reducible word is:', longest_reducible_word)
```

第四题我先写了一个判断单词是否为可缩减单词的函数 is_reducible,此函数里循环处写了一个递归，以此判断删减了单词某个字母的所有情况，并会将已经判断是可缩减且为被记录过的单词记录在字典中，之后写了一个调用 is_reducible 函数，用来查找最长可缩减单词的函数 find_longest_reducible_word,最后在主函数中利用文件对象打开文件调用上面定义的函数即可。

五、测试用例

Words.txt

六、收获与体会

在本次实验中，我感觉到难度的提升，这使得我们小组更懂得了交流

的意义，虽然我们小组本次又没能为大家上台分享，但我们还是在努力跟上节奏，并力求把每一道题弄懂，争取下次上台与大家分享。