TCP/IP 网络编程
（基于 Python 的实现）
实验

张慧翔编

Version 0.1

2017 年 10 月 15 日

目录

介绍

以班级为单位上交是实验代码文件，层次结构如下：

班级名
|----------学号
|-----------题目类别
|-------------题目 id 编号

1、Number 类

题目 1 二进制与十进制转换器（id：number1）

描述：代码实现用户输入二进制数据，输出其对应的十进制的值，或者用户输入十进制，输出其对应二进制的值。提示：使用一些简单的二进制及十进制进行测试，确保输出结果正确。例如使用那些很容易进行计算的值 3,4,5。尝试讲一个数字（以整除的方式）除以 2，并且记录其余数。注意，N%2 的余数总是 1 或 0。将除 2 得到的结果再次除 2，直到不能被 2 整除为止。增大难度：将数据转化为八进制或者其它进制。

输入：十进制数和二进制数，以英文逗号分开，如：101,101011

输出：（1）十进制对应的二进制值，二进制对应的十进制值，中间以英文逗号分开，如：1100101,43 ；（2）当输入格式不正确时，输出：“格式错误”

示例：

输入

101,101011

输出

1100101,43

输入

asd,sd

输出

“格式错误”

代码方法定义：

def d2b_b2d(decimal_int, binary_string)

题目 2 类似找零钱的操作（id：number2）

描述：编写程序，该程序显示用户物品列表及其金额，然后请用户输入要购买的物品及其支付金额。该程序实现找给用户找零的操作，最大面值为 100 元。找寻的零钱有以下几种：50 元，20 元，10 元，5 元，1 元，5 角， 1 角。比如，物品：12.3 元，支付 100 元，程序应找寻：1 个 50 元，1 个 20 元，1 个 10 元，一个 5 元，2 个 1 元，1 个 0.5 元，2 个 0.1 元。提示：首先计算出差额，然后用整除的方式计算。商品列表及对应金额如下：

"item01": 2.3,

"item02": 35.8,

"item03": 16.3,

"item04": 12,

"item05": 13.6,

"item06": 29,

"item07": 17.4,

"item08": 63.9,

"item09": 56.7,

"item10": 23.8,

输入：要购买的商品及支付的金额，以英文逗号分开，如：item01,5

输出：（1）当商品存在且支付金额大于等于商品金额时，输出各个面值的数目及对应个数，如：50*0,20*0,10*0,5*0,1*2,0.5*1,0.1*2 ；（2）无此商品时，输出“无此商品,请重新选择...” ；（3）支付金额不足时，输出：“支付金额不足,请重新支付...”

示例：

输入：item01,5

输出：50*0,20*0,10*0,5*0,1*2,0.5*1,0.1*2

输入：item00,8

输出：无此商品,请重新选择...

输入：item02,6

输出：支付金额不足,请重新支付...

代码方法定义：

def get_changes(goods,pay):

题目 3 根据两地坐标，计算两地之间的距离（id：number3）

描述：用户以十进制形式输入两地经纬度的坐标，程序计算两地之间的距离，并以千米为计量单位的形式进行输出。提示：地球并不是一个平面。为简单起见，不考虑椭球形的影响，因此我们假定地球是圆的。第一个帮助解决这个问题的是 WiKi 百科上的 Haversine 公式，这个公式可以计算两个点之间的距离，其输入参数包括两点的经纬度，地球的半径（6371km）。使用你知道的点进行测试，这将有助你发现你程序是否能够输出正确结果。提高难度：使用余弦定理来进行举例的计算，这样精确度就能下降到 1 米左右。

参考网站：https://en.wikipedia.org/wiki/Haversine_formula

输入：坐标 1 的经度纬度，坐标 2 的经度纬度。中间以英文逗号分隔，如：112.3,25.3,152.6,36.4

输出：（1）两地之间的距离，以 km 为单位：如 4007.65756018；（2）输入数据格式或者范围不合法时，输出“输入有误”的提示。

示例

输入：56.3,114.5,56.9,23.4

输出：输入有误

输入：112.3,25.3,152.6,36.4

输出：4007.65756018

代码方法定义：

def haversine_formula(longitude01, latitude01, longitude02, latitude02):

题目 4 计算 Fibonacci 序列的值（id：number4）

描述：编写代码，使用递归和常规循环操作计算 Fibonacci 序列第 N 项的值。用户输入 N，程序输出该序列计算得到的第 N 项的值。提示：递归实现：该题是学习递归实现很好的例子，但是采用递归实现效率很低；常规的循环操作：使用之前两个值的和作为该序列的下一个值，第二种方法因为无需进行多次调用，因此性能较高，本题要求两种方法均实现。增加难度：通过图形的形式说明递归实现为什么比常规的循环要效率低。

输入：N 的值，即 Fibonacci 序列的项数，如：12

输出：（1）采用常规循环和递归实现的 Fibonacci 序列第 N 项的对应的值，中间以英文逗号分隔，如：144,144 ；（2）输入参数异常时，提示“参数异常”

示例

输入：12

输出：144,144

输入：fibo

输出：参数异常

代码方法定义：

def fibonacci_recursion(term_int):

def fibonacci_loop(term_int):

题目 5 输出既是回文又是素数的数字的阶乘（id：number5）

描述：分别写出判断回文的函数、判断素数的函数、实现阶乘的函数，实现对用户输入的十进制值进行判断，如果通过，输出该数字的阶乘。

输入：将要判断的十进制的数字，如：11

输出：（1）如果该数字即是回文又是素数，则输出其阶乘，中间以英文逗号

分隔；如 11,39916800；（2）如果该数字非回文或者非素数，则输出“非回文或者非素数”；（3）如果输入非十进制，则输出“参数异常”

示例：

输入：11

输出：11,39916800

输入：100

输出：非回文或者非素数

输入：dsfd

输出：参数异常

代码方法定义：

def is_palin(num_int):

def is_prime(num_int):

def get_num_factorial(num_int):

题目 6 输出 9*9 乘法表（id：number6）

描述：编写程序输出九九乘法表。

输出格式：

1 * 1 = 1

1 * 2 = 2 2 * 2 = 4

1 * 3 = 3 2 * 3 = 6 3 * 3 = 9

...

题目 7 打印输出日历（id：number7）

描述：输入年份和月份，输出对应的日历。

输入：年份和月份，中间以英文逗号分隔，如：2012,11

输出：（1）对应的日历；（2）输入年份（1900-2100）或者月份（1-12）区间异常时，输出“日期有误”；（3）输入参数异常时，输出“参数异常”。

示例：

输入：2013,15

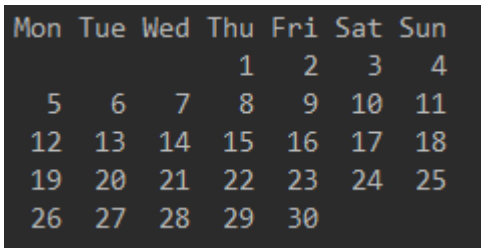
输出：日期有误

输入：calendar,7

输出：参数异常

输入：2012,11

输出：



代码方法定义：

```
# 判断年份是否为闰年
def is_leap_year(year):
# 获取一个月份的天数
def get_num_of_days_in_month(year, month):
# 获取从 1800 年 1 月 1 日到现在一共有多少天
def get_total_num_of_day(year, month):
# 返回该月的第一天从该周的第几天开始
def get_start_day(year, month):
```

2、Text 类

题目 1 判断一个字符串是不是回文（id：text1）

描述：回文即是无论顺着读还是倒着读，都一样。比如 “race car”.编写代码判断输入字符串是不是回文。如果是，打印 “The string '_____' is a palindrome" 。提示：简单方法，将字符串反转，然后与先前的进行对比。增加难度：如果你还没做过此类题，自己编写反转方法。不要使用语言自带的反转方法。

输入：判断是否为回文的字符串，如“race car”

输出：（1）是回文时，输出“The string ' race car ' is a palindrome”；（2）不是回文时，输出“The string ' race car ' is not a palindrome”

示例：

adbdcgs

输出：

The string 'adbdcgs' is not a palindrome

输入：

abdc dba

输出：

The string 'abdc dba' is a palindrome

代码方法定义：

def judge_if_parlindrome(input_string):

题目 2 摩斯码生成器（id：text2）

描述：编写代码实现从用户输入一段字符串，输出其对应的摩斯码（@代表点，#代表破折号）。提示：该题主要是将字符转换为其对应的摩斯电码。字符串可以被看作是一个字符序列，因此从开始的位置开始循环每一个字符，并且同时从摩斯码表中找到其对应的摩斯码。这种方式下，你可以迅速将字母与其对应的摩斯码进行对应。提高难度：将摩斯码反转为原始的字符串，这需要你确定哪些空格是原始的，那些是转化生成的。

参考网站：https://en.wikipedia.org/wiki/Morse_code

输入：字符串，将要转换为 Morse 码的字符串；

输出：转化为 Morse 码之后的字符串

示例

输入

i am morse 258

输出

@@ -@# ## -## ### @#@ @@@ @ -@@###@@@@@###@@

代码方法定义：

@ means dot ; # means dash

def to_morse_code(ori_string):

题目 3 重建 grep 命令工具（id：text3）

描述：在 Linux/Unix 系统中，grep 是一个长久支持的命令工具，创建你自己的 grep 工具。提示：此题是关于正则表达式的。从简单的开始，逐步深入。如果最简单的测试允许查找一个单词或者短语，甚至是一段文件列表，然后打印包含这个单词或者短语的所有行的行数。提高难度：查看你所创建的这个工具，能支持多少标志和格式。你能否指定输出管道？（打印至文件或者屏幕）

输入：文件的文件名，以及即将要查询的单词，中间以英文逗号分隔，如：grep.txt,session

输出：匹配该字符串所在的文本文件的行数

示例：

输入：

grep.txt,session

输出：

8,69,70,90,96,97,98,99,144,157,

代码方法定义：

def grep_pattern_line(file_path, pattern):

题目 4 获取出现频率最高的 10 个单词（id：text4）

描述：在数据分析中，提取热度最高的但此时经常要进行的操作，请编写代码，实现从 TXT 文本文件中提取出现频率最高的前十个单词。提示：逐行读取文本内容，并进行切分，并逐个统计该行单词的数目信息，存储于字典中，最终对字典中的数据进行排序，可转化为列表之后排序，并输出前 10 个出现频率最高的单词及其出现的次数。提高难度：查看你所创建的这个工具，能支持多少标志和格式。你能否指定输出管道？（打印至文件或者屏幕）

输入：文本文件的文件名，如：GoneWithTheWind.txt

输出：前 10 个高频词汇及其出现的次数，单词及出现次数以英文冒号分隔，不同单词信息之间以空格分隔。

示例：

输入：

GoneWithTheWind.txt

输出：

the:376 and:258 to:191 of:188 a:131 in:119 was:83 had:81 she:72 that:69

代码方法定义：

def get_ten_popular_words(filename):