第五章 函数和代码复用

5.2 实现 isOdd()函数，参数为整数，如果整数为奇数，返回 True，否则返回 False。
5.3 实现 isNum()函数，参数为一个字符串，如果这个字符串属于整数、浮点数或复数的表示，则返回 True，否则返回 False。

```
def isNum(num):
    if num.isnumeric():
        return True
    #下面为判断是否为复数的一种方法
    elif len(num)==len(set(num)):
        return True
    else:
        return False

num = input("请输入字符串：")
isnum = isNum(num)
if isnum:
    print("你输入的字符串是数字！")
else:
    print("你输入的字符串不是数字！")
```

5.4 实现 multi()函数，参数个数不限，返回所有参数的乘积。

```
def multi(nums):
    '''
    计算传入数字之积
    '''
    m=1
    for i in nums:
        m=m*i
    return m

item_num1 = input("请输入数字，以空格分隔，若输入完成请按回车：")
item_num = item_num1.split(' ')
ls =[]
for i in item_num:
    i = eval(i)
    ls.append(i)
ans = multi(ls)
print(ans)
```

5.5 实现 isPrime()函数，参数为整数，要有异常处理。如果整数是质数，返回 True，否则返回 False。

集中实现以上 5.2、5.5 的代码：

```
def isOdd(num):
```

```
    '''
    判断 num 数字是否是奇数，若是返回 True，否则返回 False
    '''
    if num%2 ==1:
        return True
    else:
        return False

def isPrime(num):
    '''
    判断是否是质数，若是则返回 True，否则返回 False
    '''
    if num ==1:
        return False
    elif num ==2 or num ==3:
        return True
    else:
        for i in range(3,num):
            if(num%i == 0):
                return False
        return True


def switFunction(item,num):
    '''
    选择所需要的功能并输出结果
    '''
    if(item == 1):
        ans = isOdd(num)
        if ans:
            print("{}是奇数".format(num))
        else:
            print("{}不是奇数".format(num))
    if(item == 2):
        ans = isPrime(num)
        if ans:
            print("{}是质数".format(num))
        else:
            print("{}不是质数".format(num))
    if(item!=1 and item!=2):
        print("请选择正确的功能")

func_num = input("请选择你所需要的功能，1 为判断奇数，2 为判断质数:")
func_num = eval(func_num)
```

```
item_num = input("请输入你需要判断的数字:")
item_num = eval(item_num)
switFunction(func_num,item_num)
```

5.7 汉诺塔问题。采用递归方法解决汉诺塔问题，要求输入汉诺塔的层数，输出整个移动流程。

```
def move(n,a,b,c):
    if n==1:
        print(a,'-->',c)
    else:
        move(n-1,a,c,b)    #将前 n-1 个盘子从 a 移动到 b 上
        move(1,a,b,c)      #将最底下的最后一个盘子从 a 移动到 c 上
        move(n-1,b,a,c)    #将 b 上的 n-1 个盘子移动到 c 上
move(3,'A','B','C')
```

第六章 组合数据类型

6.1 随机密码生成。编写程序，在 26 个字母大小写和 9 个数字组成的列表中随机生成 10 个 8 位密码。

```
#6.1 random password
from random import choice
def rand_pass(N):
    ls = []#create list
    #add alpha and number into list
    for i in range(97,123):
        ls.append(chr(i))
    for i in range(65,91):
        ls.append(chr(i))
    for i in range(48,58):
        ls.append(chr(i))
    #build one password
    return ''.join(choice(ls) for i in range(N))
#produce ten passwords
for i in range(10):
    ans_ls = rand_pass(8)
    print(ans_ls)
```

6.2 重复元素判定。编写一个函数，接受列表作为参数，如果一个元素在列表中出现了不止一次，则返回 True，但不要改变原来列表的值。同时编写调用这个函数和测试结果的程序。

6.3 利用集合的无重复性改编程序练习题 6.2 的程序，获得一个更快更简洁的版本。

6.2/6.3 代码如下：

```
def repDetermination_1(ls):
    '''
    Determinant of repeated elements.
    If a list has repeated elements, return True.
```

```
    If it does not has, return False
    This is the Method ONE.
    '''
    for i in range(len(ls)):
        for j in range(i+1,len(ls)):
            if ls[i] == ls[j]:
                return True
    return False

def repDetermination_2(ls):
    '''
    Determinant of repeated elements.
    If a list has repeated elements, return True.
    If it does not has, return False
    This is the Method TWO.
    '''
    len_or = len(ls)# original list
    ls_set = len(set(ls))#after being a set
    #if the original list has repeated elements, the length will be different
    if len_or == ls_set:
        return False
    else:
        return True

ls = input("请输入元素，以空格分隔。输入完毕，请按回车：\n")
ls = ls.split(" ")
ans_1 = repDetermination_1(ls)
ans_2 = repDetermination_2(ls)
print("方法一得到的结果是{}；\n 方法二得到的结果是：{}".format(ans_1,ans_2))
```

6.5 生日悖论分析。生日悖论指如果一个房间里有 23 人或以上，那么至少有两个人生日相同的概率大于 50%。编写程序，输出在不同随机样本数量下，23 个人中至少两个人生日相同的概率。

```
import random

def generate_birthday(stu_nums):
    '''
    Generate the birthday of 23 students.
    '''
    birth_date= []
    for i in range(stu_nums):
        birth_date.append(random.randint(1,365))
    return birth_date

def equal_birthday(birth_date):
```

```
    x = birth_date[:]
    x.sort()
    for i in range(len(x)-1):
        if x[i] == x[i+1]:
            return True
    return False

def birth_count(samples, stu_nums):
    count = 0
    m = []
    for i in range(samples):
        m = generate_birthday(stu_nums)
        if equal_birthday(m):
            count +=1
    return count

samples = input("请输入样本数：\n")
samples = eval(samples)
stu_nums = 0
while(stu_nums <23):
    stu_nums = input("请输入人数，不得小于 23，默认为 23：\n")
    stu_nums = eval(stu_nums)
count = birth_count(samples,stu_nums)
rate = count/samples*100
print("概率为{:.2f}%".format(rate))
```

第七章：文件和数据格式化

7.1 Python 源文件改写。编写一个程序，读取一个 Python 源程序文件，将文件中所有除保留字外的小写字母换成大写字母，生成后的文件要能够被 Python 解释器正确执行。

```
import keyword

excludes = keyword.kwlist
exclud = ['print','format']
fname = input("请输入要打开的文件（包含后缀名）：\n")
f = open(fname,'r').readlines()
print(f)
ls = []
for i in f:
    i = i.split(" ")
    ls.append(i)

print(ls)
```

```
#建立一个每行所有单词为一个元素的列表
new_name = input("请输入新建的文件的文件名（包含后缀名）：\n")
fo = open(new_name,'w+')
for i in range(len(ls)):
    if f[i].isspace():
        fo.write(" ")
    for j in range(len(ls[i])):
        x = ls[i][j]
        if x in excludes:
            x = x.lower()
        else:
            x = x.upper()
        if x ==ls[i][len(ls[i])-1]:
            fo.write(x +"\n")
        else:
            fo.write(x+" ")
fo.close()
print("处理结束！")
```

程序仍然存在问题，例如 range 或者 print 因为和其它字符组成一个字符串，暂时想不到如何分割然后让其仍然小写。

7.2 图像文件压缩。使用 PIL 库对图片进行等比例压缩，无论压缩前文件大小如何，压缩后文件小于 10KB。

```
from PIL import Image
def resizeImg(ori_name, des_name, dst_w, dst_h, save_q):
    im = Image.open(ori_name)
    ori_w,ori_h = im.size
    widthRatio = heightRatio = None
    ratio = 1
    if(ori_w and ori_w>dst_w) or (ori_h and ori_h >dst_h):
        if dst_w and ori_w>dst_w:
            widthRatio = float(dst_w/ori_w)
        if dst_h and ori_h>dst_h:
            heightRatio = float(dst_h/ori_h)

        if widthRatio and heightRatio:
            if heightRatio < widthRatio:
                ratio = heightRatio
            else:
                ratio = widthRatio
        if widthRatio and not heightRatio:
            ratio = widthRatio
        if heightRatio and not widthRatio:
            ratio = heightRatio
        newWidth = int(ori_w *ratio)
```

```
            newHeight = int(ori_h *ratio)
        else:
            newWidth = ori_w
            newHeight = ori_h

        im.resize((newWidth,newHeight),Image.ANTIALIAS).save(des_name, quality=save_q)

ori_name = '1.jpg'
des_name = '1_modify.jpg'
dst_w = 300
dst_h = 300
save_q = 50
resizeImg(ori_name, des_name, dst_w, dst_h, save_q)
```