源代码已上传至Github,https://github.com/akino-liuxing/arithmetic,中的sizeyunsuan.py文件

## 题目要求：

　　（1）能自动生成小学四则运算题目，其中不能出现负数；

　　（2）除了整数外，还能支持真分数的四则运算；

## 解题思路描述：

　　（1）四则运算加减乘除，采用两个随机数，由于不能出现负数，则对两个随机数进行比较大小再进行减法运算，除法一向特殊，所以在随机数的取值范围中设置不包括0。

　　（2）真分数运算在pycharm中导入fractions库，代码：

                import random

                from fractions import Fraction

## 设计实现过程：

　　设计出以下三个函数：

　　　　def newint() 生成整数四则运算

　　　　def newfra() 生成真分数四则运算

　　　　def newtest() 生成制定指定数量的四则运算题目

　　三个函数的函数关系：

　　　　newint()与newfra()为独立的函数，负责生成随机四则运算，newtest()则随机调用上述两个函数生成题目。

## 代码说明：

　　首先说明整数的四则运算，生成两个随机数并随机运算，在减法中比较大小防止出现负数，在除法中比较大小并循环取整除的随机数组合。最后输出算式并返回正确答案。

def newint():  
opr = ['＋', '－', '×', '÷']  
fh = random.randint(0, 3)  
n1 = random.randint(1, 20)  
n2 = random.randint(1, 20)  
rjg = 0  
if fh == 0:  
rjg = n1 + n2  
elif fh == 1:  
n1, n2 = max(n1, n2), min(n1, n2)  
rjg = n1 - n2  
elif fh == 2:  
rjg = n1 \* n2  
elif fh == 3:  
n1, n2 = max(n1, n2), min(n1, n2)  
while n1 % n2 != 0:  
n1 = random.randint(1, 10)  
n2 = random.randint(1, 10)  
n1, n2 = max(n1, n2), min(n1, n2)  
rjg = int(n1 / n2)  
print(n1, opr[fh], n2, '= ', end='')  
return rjg  
'''jg = input()  
sr = int(jg)  
if int(sr) == rjg:  
print('right')  
else:  
print('error. the Tight answer is', rjg)'''

　　真分数四则运算类似。

def newfra():  
opr = ['＋', '－', '×', '÷']  
fh = random.randint(0, 3)  
t1 = random.randint(1, 10)  
t2 = random.randint(t1, 10)  
n1 = Fraction(t1, t2)  
t1 = random.randint(1, 10)  
t2 = random.randint(t1, 10)  
n2 = Fraction(t1, t2)  
rjg = 0  
if fh == 0:  
rjg = n1 + n2  
elif fh == 1:  
n1, n2 = max(n1, n2), min(n1, n2)  
rjg = n1 - n2  
elif fh == 2:  
rjg = n1 \* n2  
elif fh == 3:  
n1, n2 = max(n1, n2), min(n1, n2)  
rjg = n1 / n2  
print(n1, opr[fh], n2, '= ', end='')  
return rjg  
'''jg = input()  
sr = Fraction(jg)  
if sr == rjg:  
print('right')  
else:  
print('error. the Tight answer is', rjg)'''

　　newtest()函数是要求用户输入一个整数来输出算式的数量，采用while循环随机生成整数或者真分数运算，将答案保存在rjg列表的同时输出算式直到算式数量达到要求。最后输出rjg列表即输出答案。

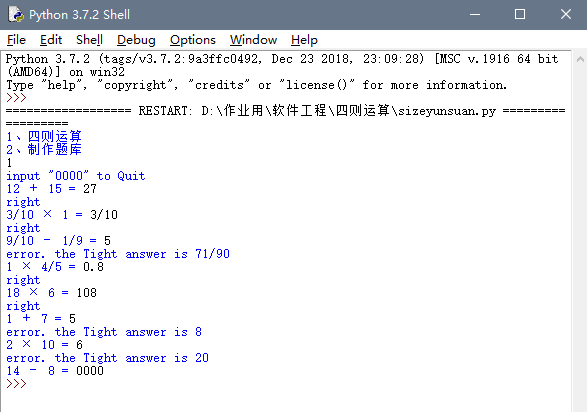
def newtest():  
opr = ['＋', '－', '×', '÷']  
print('输入题库所需要的题目数量')  
n=int(input())  
rjg=[]  
m=0  
while m<=(n-1):  
fh = random.randint(0, 4)  
if fh==0:  
print(m+1,end='、')  
rjg.append(newfra())  
print(' ')  
else:  
print(m+1,end='、')  
rjg.append(newint())  
print(' ')  
m=m+1  
m=0  
print('答案：')  
while m<=(n-1):  
print(m+1,'、',rjg[m])  
m=m+1

　　下列为主函数，第一个模式负责调用上述newint()、new函数，并获得函数返回值即算式答案，与用户输入值进行比较。第二个模式则是生成算式题目。

print('1、四则运算')  
print('2、制作题库')  
n=int(input())  
if n==1:  
print('input "0000" to Quit')  
while True:  
fh = random.randint(0, 4)  
if fh == 0:  
rjg = newfra()  
jg = input()  
if jg == '0000':  
break;  
sr = Fraction(jg)  
if sr == rjg:  
print('right')  
else:  
print('error. the Tight answer is', rjg)  
else:  
rjg = newint()  
jg = input()  
if jg == '0000':  
break;  
sr = int(jg)  
if sr == rjg:  
print('right')  
else:  
print('error. the Tight answer is', rjg)  
if n==2:  
newtest()

## 测试运行：

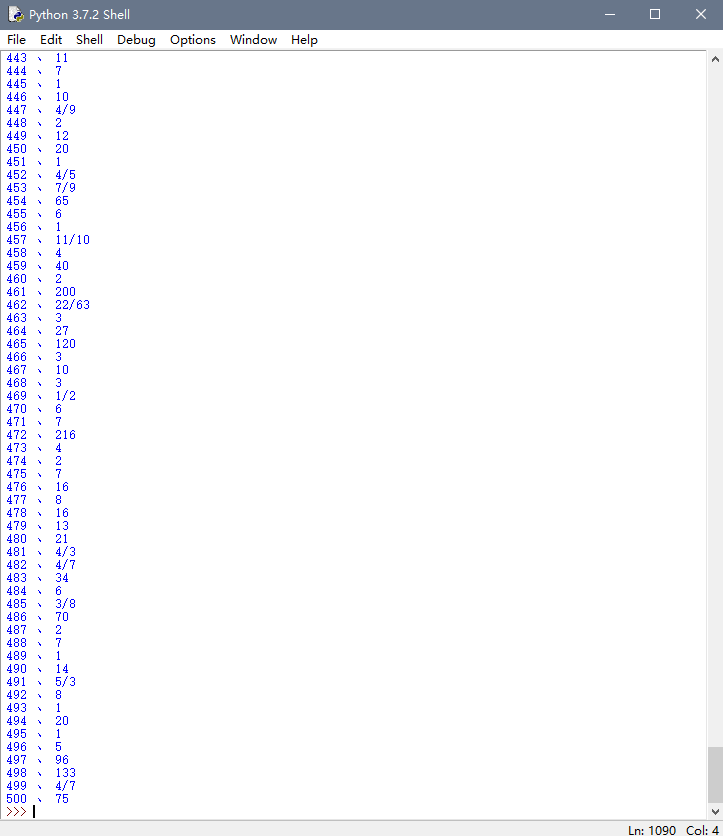
　　先测试运行第一个模式，如下图：



　　第二个模式，先输出25个算式数量：



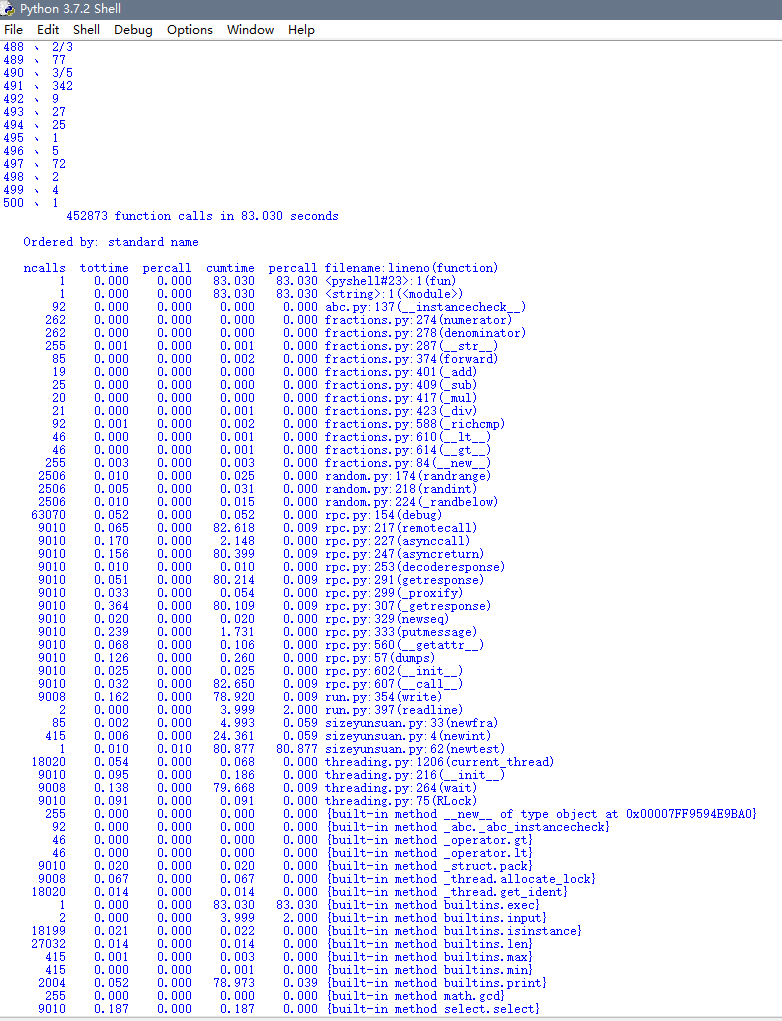
　　输出500个算式数量，运行完成且无报错，部分截图如下：

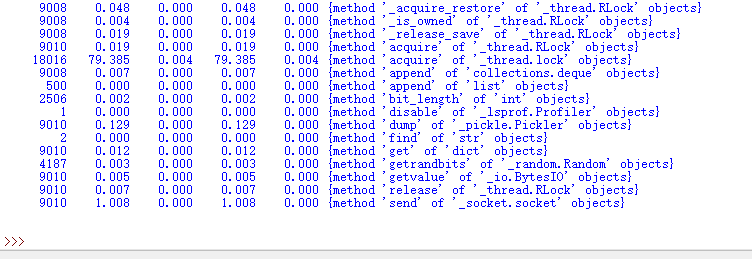


## 改进程序性能：

　　由于本人从未尝试对程序进行性能分析和改进，初次尝试使用cProfile方式，如有错误还请包涵。

       输出500个算式结果：





## PSP表格：

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 预计耗时（分钟） | 是实际耗时（分钟） |
| Planning | 计划 | 5 | 15 |
| Estimate | 估计这个任务需要多少时间 | 5 | 10 |
| Development | 开发 | 100 | 150 |
| Analysis | 需求分析 | 2 | 15 |
| Design Spec | 生成设计文档 | 20 | 30 |
| Design Review | 设计复审（和同事审核设计文档） | 10 | 20 |
| Coding Standerd | 代码规范（为目前的开发制定合适的规范） | 30 | 50 |
| Design | 具体设计 | 10 | 30 |
| Coding | 具体编码 | 40 | 100 |
| Code Review | 代码复审 | 3 | 15 |
| Text | 测试（自测，修改代码，提交修改） | 15 | 40 |
| Reporting | 报告 | 20 | 40 |
| Text Report | 测试报告 | 5 | 15 |
| Size Measurement | 计算工作量 | 10 | 15 |
| Postmortem & Process Improvement Plan | 事后总结，并提出过程改进计划 | 10 | 20 |
| Sum | 合计 | 280 | 565 |

## 附：

　　代码参考资料出处：https://www.cnblogs.com/chaigee/p/8877263.html