**用面向对象的思想实现一个有理数以及有理数的加减乘除运算——Python版本**

2018年08月08日 19:08:42 [Lockeyi](https://me.csdn.net/Lockey23) 阅读数：302

 版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/Lockey23/article/details/81513854

class Rational(object):

def \_\_init\_\_(self,a,b=1):

if(b ==0 ):*#对于分母为0的情况直接举异常*

raise Exception("Denominator must not be 0!")

else:

g = self.gcd(abs(a), abs(b))*#求分子分母的最大公约数然后进行约分*

self.a = a/g

self.b = b/g

def gcd(self,a,b):*#求最大公约数*

if(b==0):return a

else:return self.gcd(b,a%b)

*#以下为运算符重载*

def \_\_add\_\_(self, another):

try:

return Rational(self.a\*another.b +another.a\*self.b,self.b\*another.b)

except TypeError:

return NotImplemented

def \_\_mul\_\_(self, another):

try:

return Rational(self.a\*another.a,self.b\*another.b)

except TypeError:

return NotImplemented

def \_\_sub\_\_(self, another):

try:

return Rational(self.a \* another.b - self.b \* another.a, self.b \* another.b)

except TypeError:

return NotImplemented

def \_\_div\_\_(self, another):

try:

return Rational(self.a \* another.b, self.b \* another.a)

except TypeError:

return NotImplemented

def \_\_str\_\_(self):

return '%s/%s' % (self.a, self.b) if self.b != 1 else str(self.a)

*# Below are tests:*

a = Rational(3,6)

b = Rational(4,5)

print a,b

print a+b

print a\*b

print b-a

print b/a

a = Rational(3,7)

b = Rational(4)

print a,b

print a+b

print a\*b

print b-a

print b/a

a = Rational(3)

b = Rational(4,7)

print a,b

print a+b

print a\*b

print b-a

print b/a

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52
* 53
* 54
* 55
* 56
* 57
* 58
* 59
* 60
* 61
* 62
* 63
* 64
* 65
* 66
* 67
* 68
* 69
* 70

**注意**

在运算符重载的时候Python2和Python3有细微的不同，如果没有实际经验会报以下错误

**TypeError: unsupported operand type(s) for /: ‘Rational’ and ‘Rational’**

问题主要出在重载除法运算符的时候2和3有以下差异：

Python3 uses special division names: \_\_truediv\_\_ and \_\_floordiv\_\_ for the / and // operators, respectively.

python2 uses \_\_div\_\_