|  |  |
| --- | --- |
| **Name:** | Muhammad Arslan Raza |
| **Roll no.** | 2020-EE-403 |

**Lab # 13: Classes and Object-Oriented Programming**

**Objective:**

The objective of this problem set is how to write different programs on compiler

**Task # 1:**

Create a class called RationalNumber for performing arithmetic with fractions. Write a driver program to test your class Use integer variables to represent the data of the class—the numerator and the denominator. Provide a constructor that enables an object of this class to be initialized when it is declared. The constructor should contain default values, in case no initializers are provided, and should store the fraction in reduced form (i.e., the fraction

2/4

would be stored in the object as 1 in the numerator and 2 in the denominator).

Provide methods for each of the following:

a) Adding two RationalNumbers. The result should be stored in reduced form.

b) Subtracting two RationalNumbers. The result should be stored in reduced form.

c) Multiplying two RationalNumbers. The result should be stored in reduced form.

d) Dividing two RationalNumbers. The result should be stored in reduced form.

e) Printing RationalNumbers in the form a/b, where a is the numerator and b is the denominator.

f) Printing RationalNumbers in floating-point format.

**Code:**

#Creating Objects

class Rational():

def \_\_init\_\_(self,n,d):

self.n = n

self.d = d

print('Fraction = ',self.n," / ",self.d)

def addition(self,other):

sn = (self.n\*(other.d) + other.n\*(self.d))

sd = (self.d \* other.d)

r = reduce(sn,sd)

return "Sum = " + str(r[0]) + " / " + str(r[1])

def subtraction(self,other):

sn = (self.n\*(other.d) - other.n\*(self.d))

sd = (self.d \* other.d)

r = reduce(sn,sd)

return "Difference = " + str(r[0]) + " / " + str(r[1])

def multiplication(self,other):

sn = self.n \* other.n

sd = self.d \* other.d

r = reduce(sn,sd)

return "Product = " + str(r[0]) + " / " + str(r[1])

def division(self,other):

sn = self.n\*(other.d)

sd = self.d \* other.n

r = reduce(sn,sd)

return "Division = " + str(r[0]) + " / " + str(r[1])

def get\_frac(self):

return "Fraction Value = " + str(self.n) + ' / ' + str(self.d)

def get\_float(self):

return "Floating Value = " + str(self.n/self.d)

def reduce(n,d):

if n < d:

for k in range(abs(n),0,-1):

if n % k == 0 and d % k == 0:

n = int(n/k)

d = int(d/k)

else:

for k in range(abs(d),0,-1):

if n % k == 0 and d % k == 0:

n = int(n/k)

d = int(d/k)

return(n,d)

#Driver Code

frac1 = Rational(10,7)

frac2 = Rational(15,7)

print(frac1.addition(frac2))

print(frac1.subtraction(frac2))

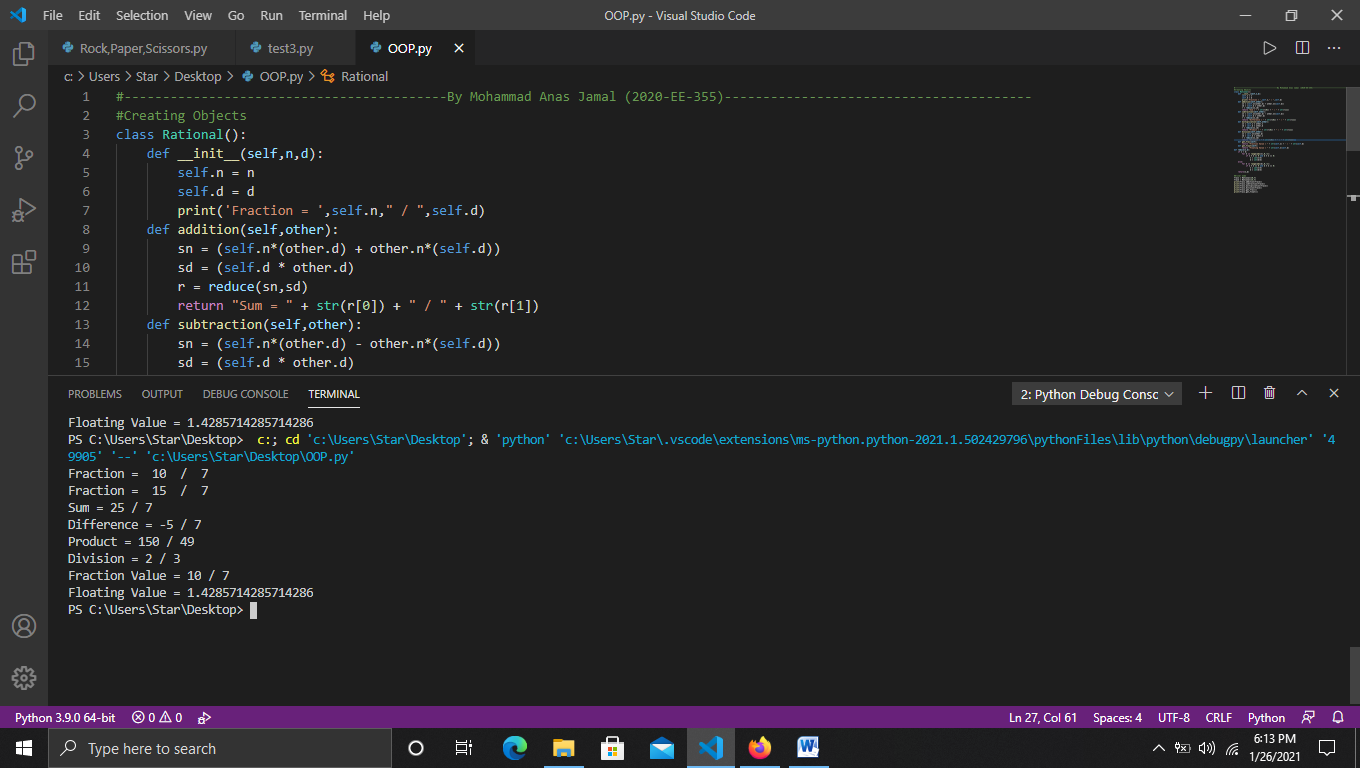
print(frac1.multiplication(frac2))

print(frac1.division(frac2))

print(frac1.get\_frac())

print(frac1.get\_float())

**Output:**



**Task # 2:**

Write an inheritance hierarchy for class Quadrilateral, Trapezoid, Parallelogram, Rectangle and Square. Use Quadrilateral as the base class of the hierarchy. Make the hierarchy as deep (i.e., as many levels) as possible. The data of Quadrilateral should be the (x, y) coordinate pairs for the four endpoints of the Quadrilateral. Write a driver program that creates and displays objects of each of these classes.

**Code:**

#Parent class

class Quadilateral(object):

def \_\_init\_\_(self,p1,p2,p3,p4):

self.p1 = p1

self.p2 = p2

self.p3 = p3

self.p4 = p4

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

s = str(self.p1) + "\n" + str(self.p2) + "\n" + str(self.p3) + "\n" + str(self.p4)

return s

#Child classes

class Trapezoid(Quadilateral):

pass

class Square(Quadilateral):

pass

class Rectangle(Quadilateral):

pass

class Parallelogram(Quadilateral):

pass

#Driver code

trapezoid = Trapezoid((10,15),(20,24),(50,10),(10,20))

sqruare = Square((20,20),(40,10),(50,20),(10,20))

rectangle = Rectangle((15,10),(40,20),(10,10),(60,50))

parallelogram = Parallelogram((50,20),(10,20),(20,24),(50,10))

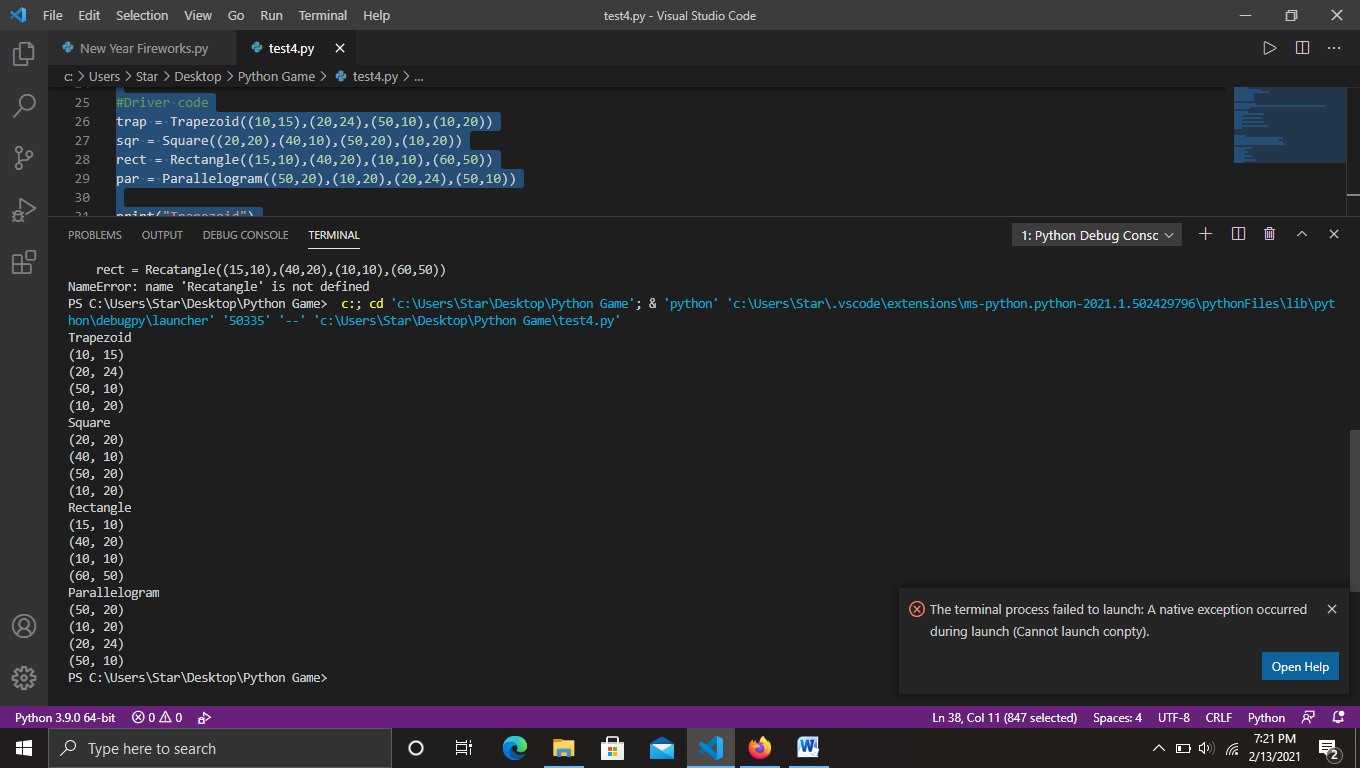
print("Trapezoid \n",trapezoid)

print("Square \n",sqr)

print("Rectangle \n",rectangle)

print("Parallelogram \n",parallelogram)

**Output:**



**Conclusion:**

. Today in this lab I learn about Classes and Object-Oriented Programming.