M.Tech : System on Chip Design

Indian Institute of Technology, Palakkad

CS5107: Programming Lab

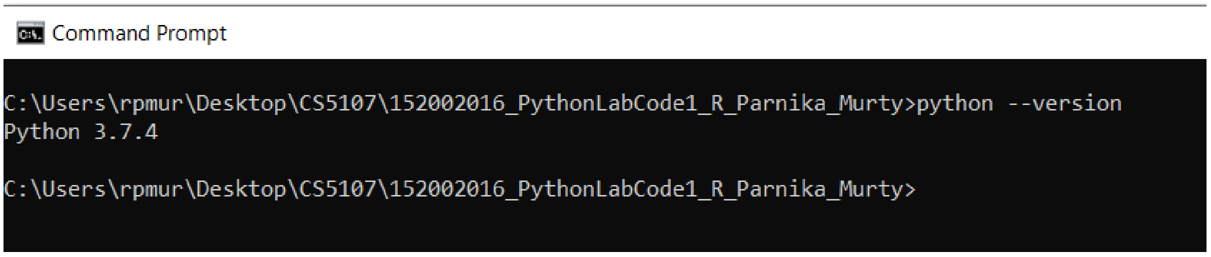
Lab 1: All about classes

Name: R. Parnika Murty

Roll No.: 152002016

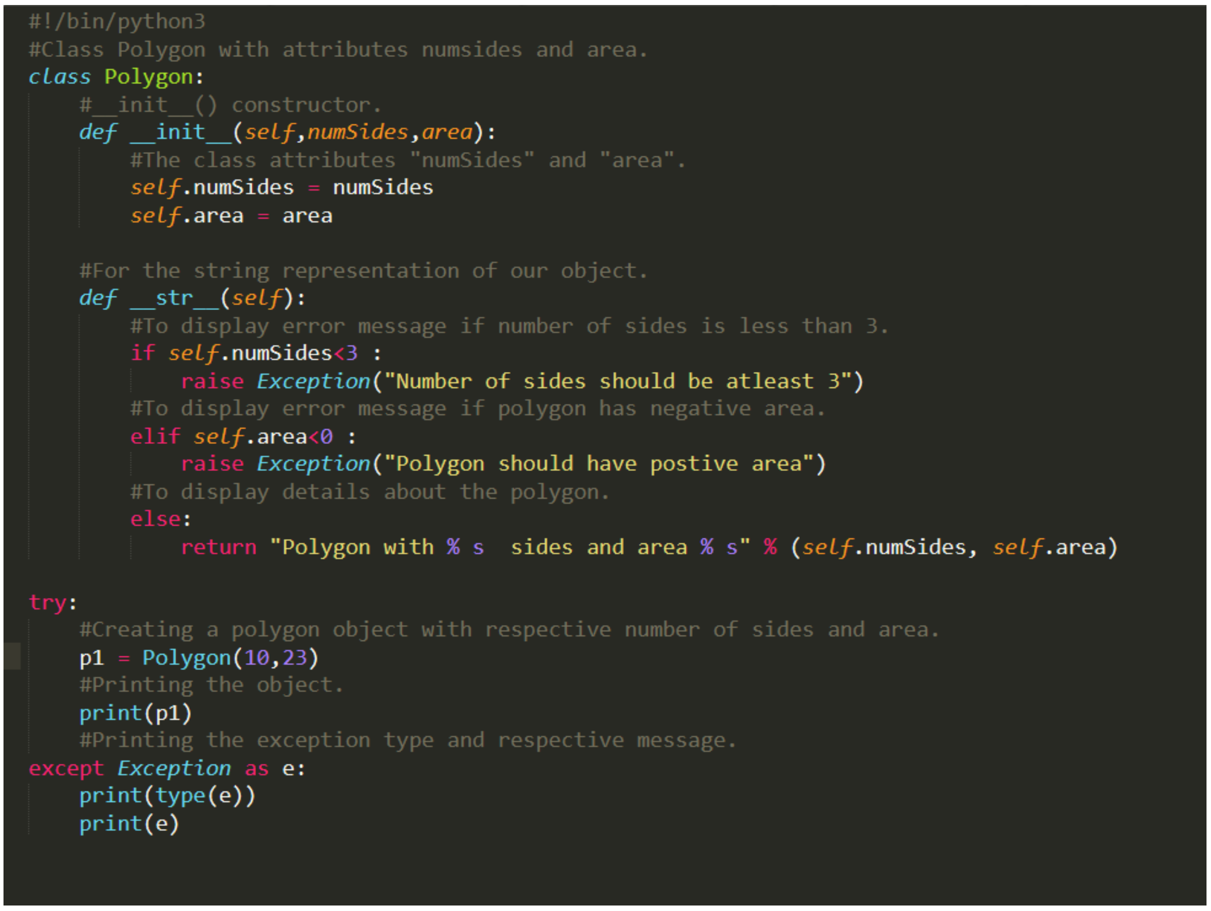
(The assignment is done using Python3)

Version of python used:



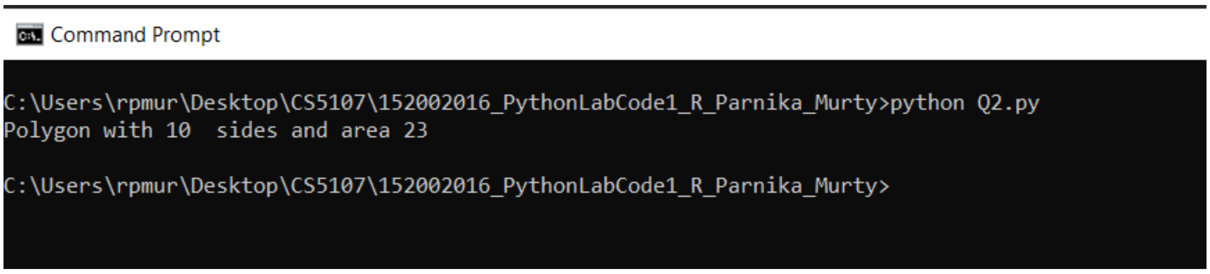
```
C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>python --version
Python 3.7.4

C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>
```

1) The following image is a snippet from python program named as **Q2.py**. We create a class **Polygon** as shown and try to create an object of type **Polygon** by specifying number of sides as “numSides” and area as “area”. We then proceed to printing the object by using “print(p1)”:



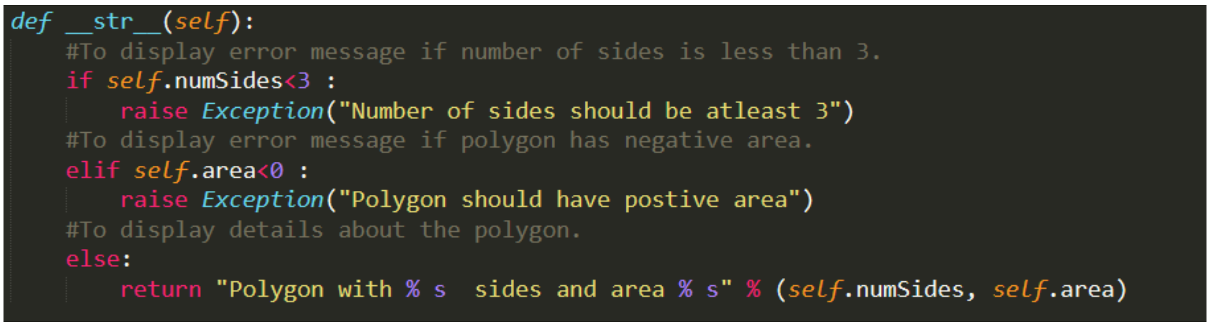
```
#!/bin/python3
#Class Polygon with attributes numsides and area.
class Polygon:
    #__init__() constructor.
    def __init__(self,numSides,area):
        #The class attributes "numSides" and "area".
        self.numSides = numSides
        self.area = area

    #For the string representation of our object.
    def __str__(self):
        #To display error message if number of sides is less than 3.
        if self.numSides<3 :
            raise Exception("Number of sides should be atleast 3")
        #To display error message if polygon has negative area.
        elif self.area<0 :
            raise Exception("Polygon should have postive area")
        #To display details about the polygon.
        else:
            return "Polygon with % s  sides and area % s" % (self.numSides, self.area)

try:
    #Creating a polygon object with respective number of sides and area.
    p1 = Polygon(10,23)
    #Printing the object.
    print(p1)
    #Printing the exception type and respective message.
except Exception as e:
    print(type(e))
    print(e)
```

The output can be seen as:

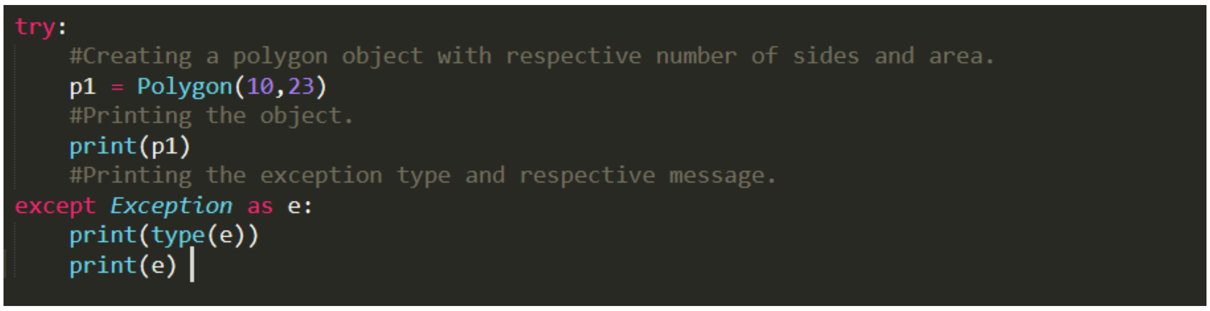


```
Command Prompt
C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>python Q2.py
Polygon with 10  sides and area 23

C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>
```

Since the polygon should have positive area and the number of sides atleast be 3, we have done some exception handling to display our error messages as required.:



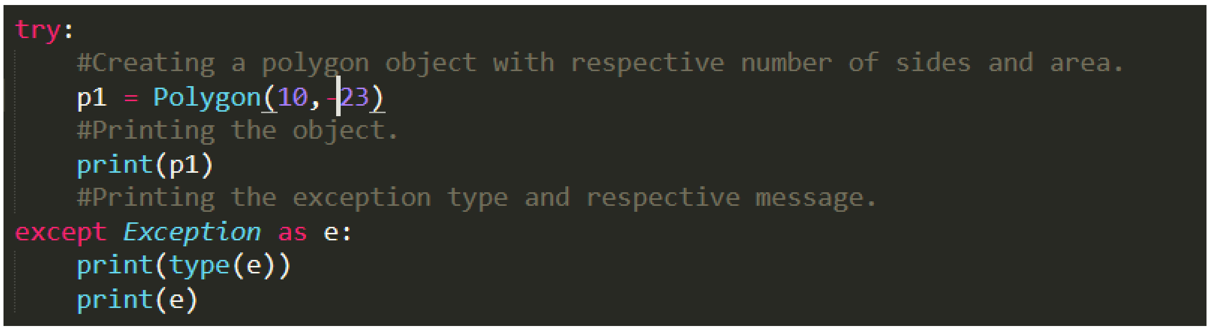
```python
def __str__(self):
    #To display error message if number of sides is less than 3.
    if self.numSides<3 :
        raise Exception("Number of sides should be atleast 3")
    #To display error message if polygon has negative area.
    elif self.area<0 :
        raise Exception("Polygon should have postive area")
    #To display details about the polygon.
    else:
        return "Polygon with % s  sides and area % s" % (self.numSides, self.area)
```



```python
try:
    #Creating a polygon object with respective number of sides and area.
    p1 = Polygon(10,23)
    #Printing the object.
    print(p1)
    #Printing the exception type and respective message.
except Exception as e:
    print(type(e))
    print(e)
```

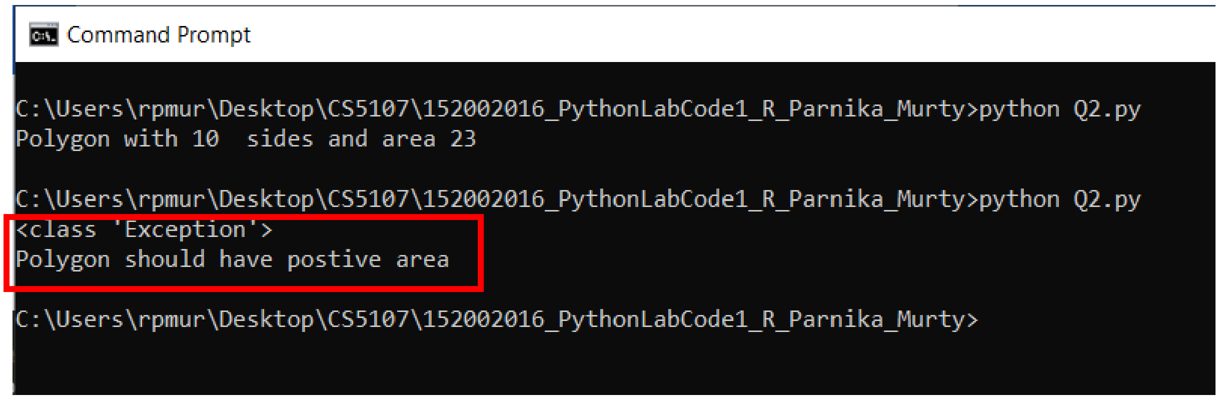
We used some test-cases to test this as follows:

For negative area:



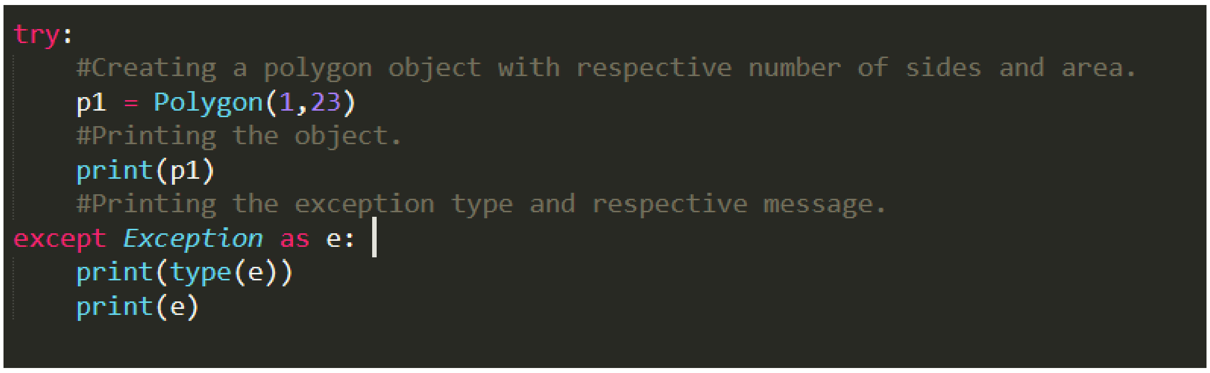
```python
try:
    #Creating a polygon object with respective number of sides and area.
    p1 = Polygon(10,-23)
    #Printing the object.
    print(p1)
    #Printing the exception type and respective message.
except Exception as e:
    print(type(e))
    print(e)
```

The output is:



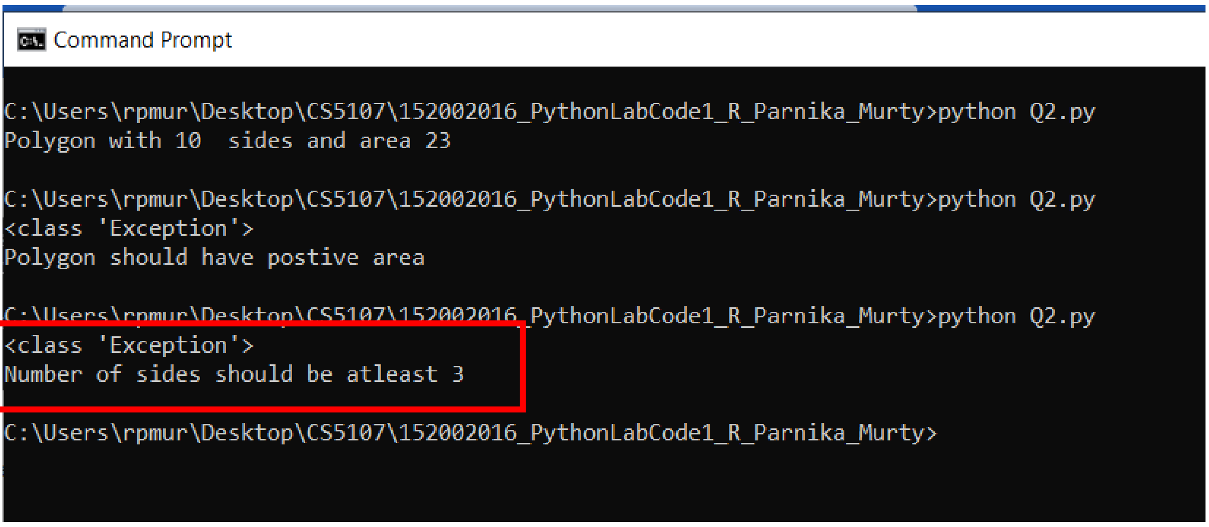
```
C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>python Q2.py
Polygon with 10  sides and area 23

C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>python Q2.py
<class 'Exception'>
Polygon should have postive area

C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>
```

For sides less than 3:



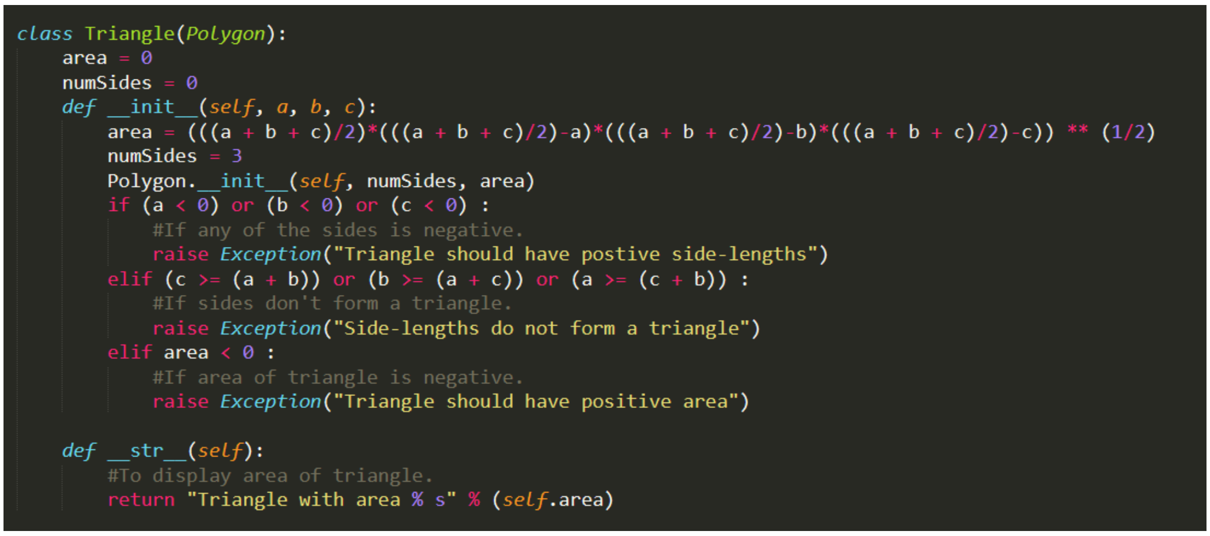
```python
try:
    #Creating a polygon object with respective number of sides and area.
    p1 = Polygon(1,23)
    #Printing the object.
    print(p1)
    #Printing the exception type and respective message.
except Exception as e:
    print(type(e))
    print(e)
```

The output is:



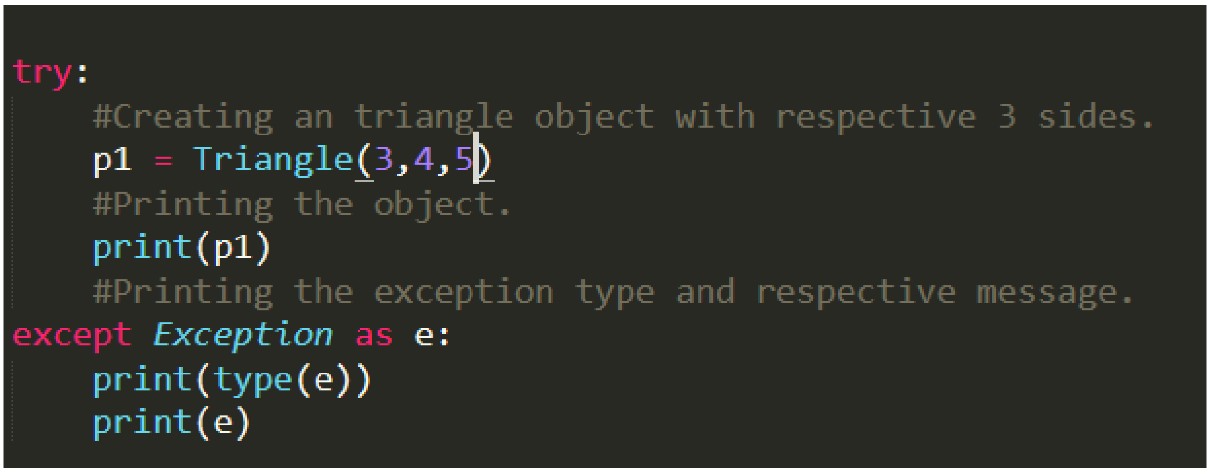
```
C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>python Q2.py
Polygon with 10  sides and area 23

C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>python Q2.py
<class 'Exception'>
Polygon should have postive area

C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>python Q2.py
<class 'Exception'>
Number of sides should be atleast 3

C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>
```

2) **Q3.py**:- We then created a derived class **Triangle** from the base class **Polygon** with no additional attributes as follows:



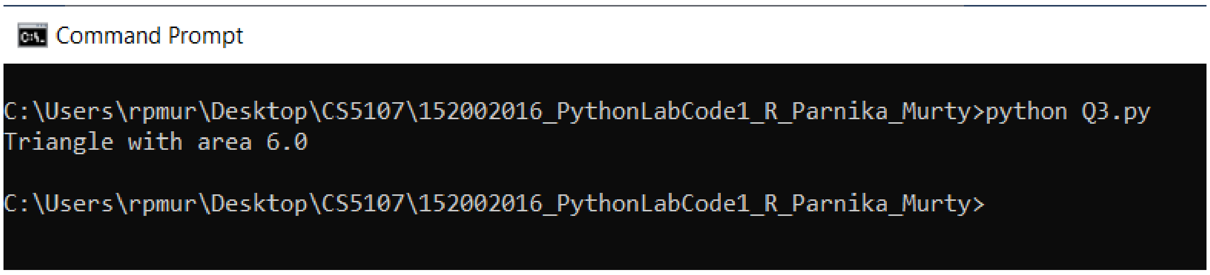
```python
class Triangle(Polygon):
    area = 0
    numSides = 0
    def __init__(self, a, b, c):
        area = (((a + b + c)/2)*(((a + b + c)/2)-a)*(((a + b + c)/2)-b)*(((a + b + c)/2)-c)) ** (1/2)
        numSides = 3
        Polygon.__init__(self, numSides, area)
        if (a < 0) or (b < 0) or (c < 0) :
            #If any of the sides is negative.
            raise Exception("Triangle should have postive side-lengths")
        elif (c >= (a + b)) or (b >= (a + c)) or (a >= (c + b)) :
            #If sides don't form a triangle.
            raise Exception("Side-lengths do not form a triangle")
        elif area < 0 :
            #If area of triangle is negative.
            raise Exception("Triangle should have positive area")

    def __str__(self):
        #To display area of triangle.
        return "Triangle with area % s" % (self.area)
```

We create a Triangle object by specifying its side lengths and print its area using “print(p1)” statement as below:



```python
try:
    #Creating an triangle object with respective 3 sides.
    p1 = Triangle(3,4,5)
    #Printing the object.
    print(p1)
    #Printing the exception type and respective message.
except Exception as e:
    print(type(e))
    print(e)
```

The output is:

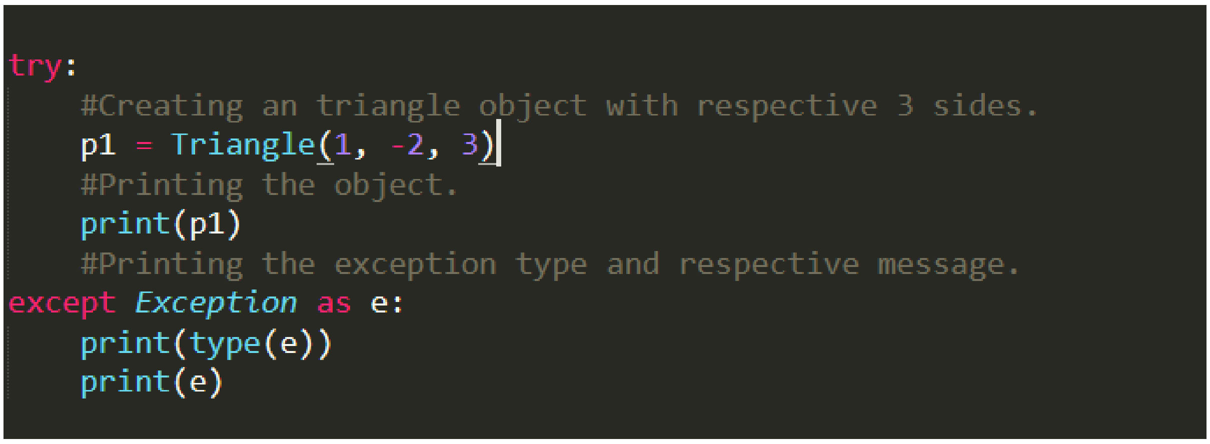


Command Prompt

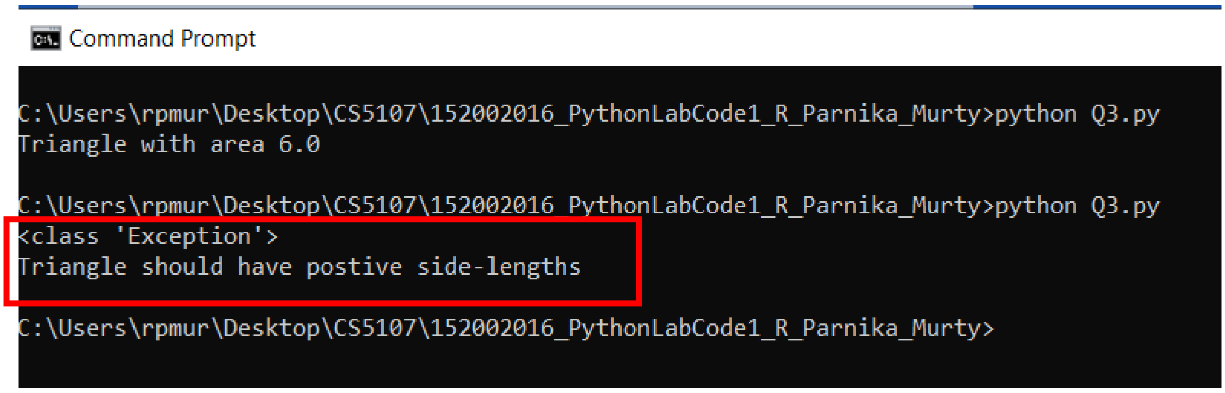
```
C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>python Q3.py
Triangle with area 6.0

C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>
```

Since the sides need to follow a property that the third side should never be greater than or equal to the sum of the other two sides, we do some exception handling for this. Moreover, we also throw exception and handle it whenever any of the sides of the triangle is not positive. We used some test-cases as follows:

For negative sides:

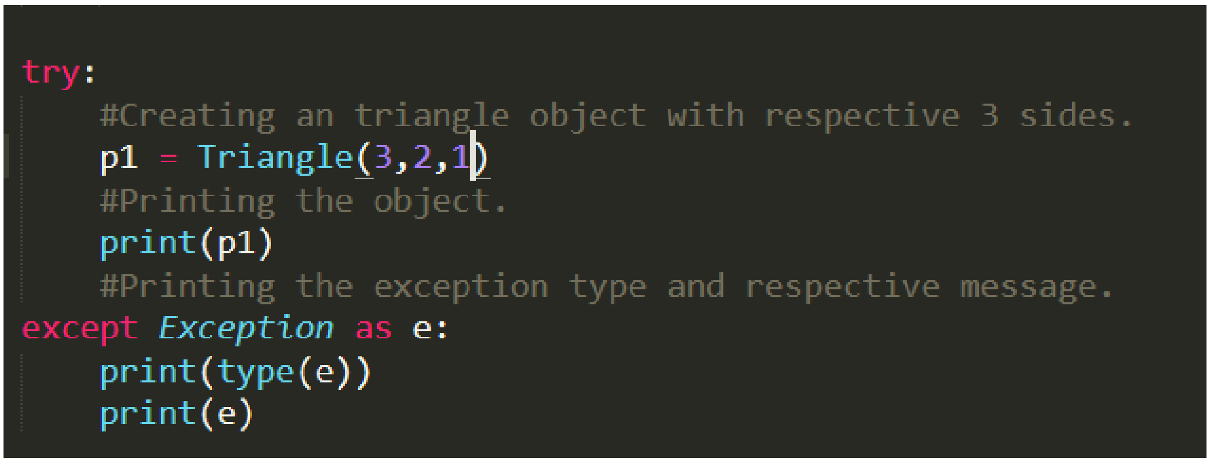

```python
try:
    #Creating an triangle object with respective 3 sides.
    p1 = Triangle(1, -2, 3)
    #Printing the object.
    print(p1)
    #Printing the exception type and respective message.
except Exception as e:
    print(type(e))
    print(e)
```

The output is:

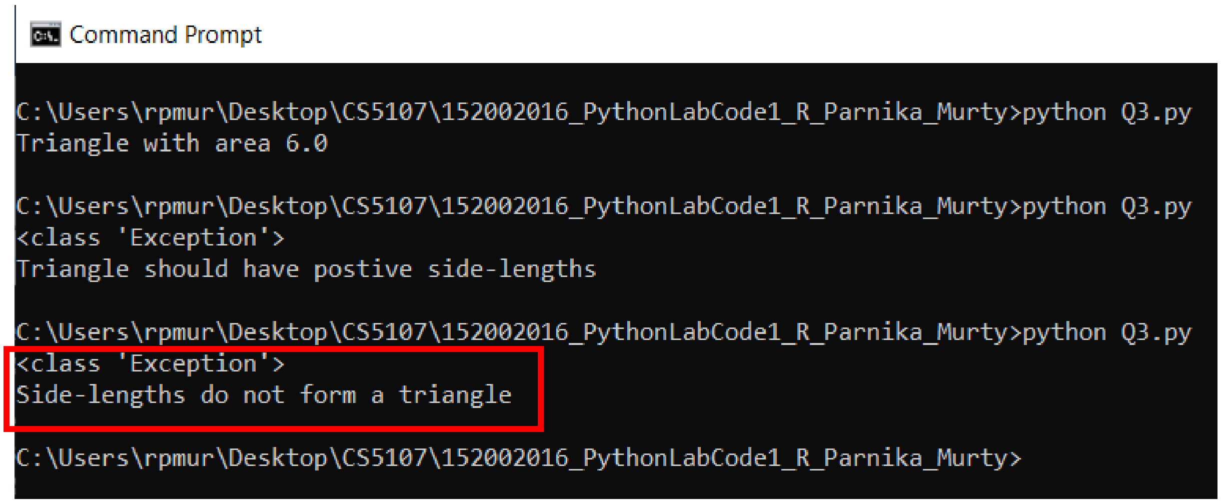


```
C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>python Q3.py
Triangle with area 6.0

C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>python Q3.py
<class 'Exception'>
Triangle should have postive side-lengths

C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>
```

When side lengths do not form a triangle:


```python
try:
    #Creating an triangle object with respective 3 sides.
    p1 = Triangle(3,2,1)
    #Printing the object.
    print(p1)
    #Printing the exception type and respective message.
except Exception as e:
    print(type(e))
    print(e)
```

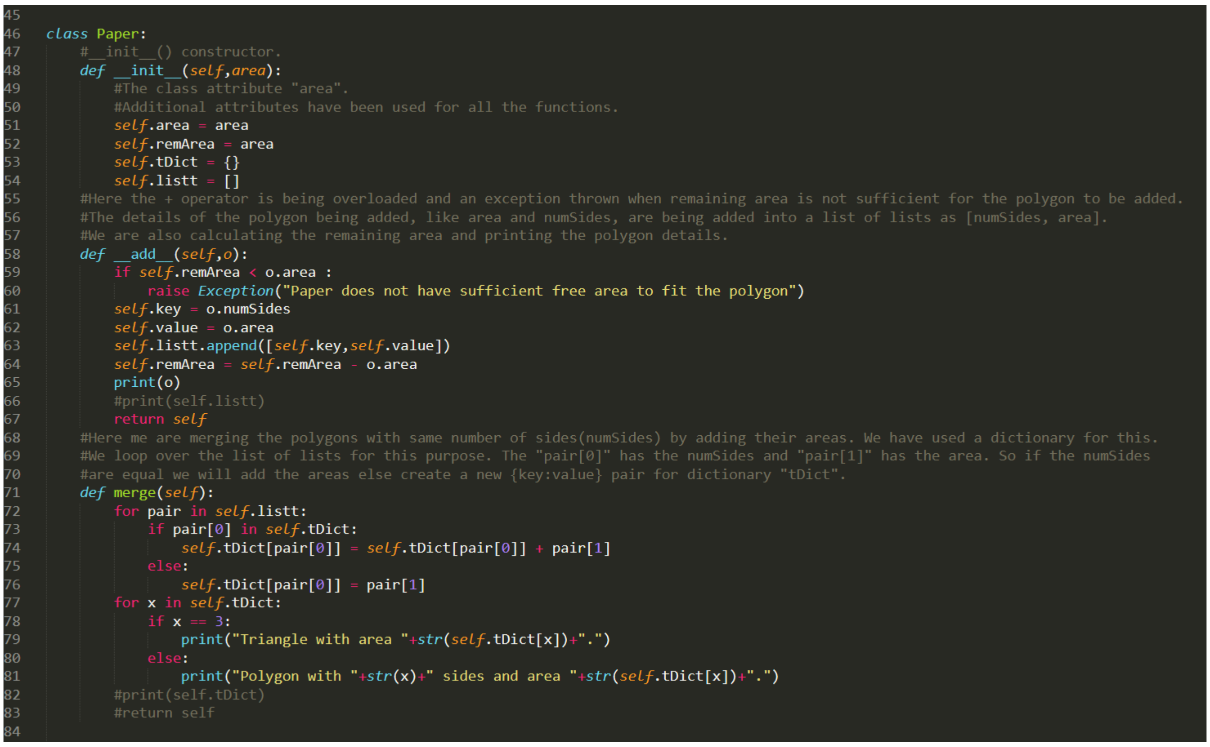
The output is:



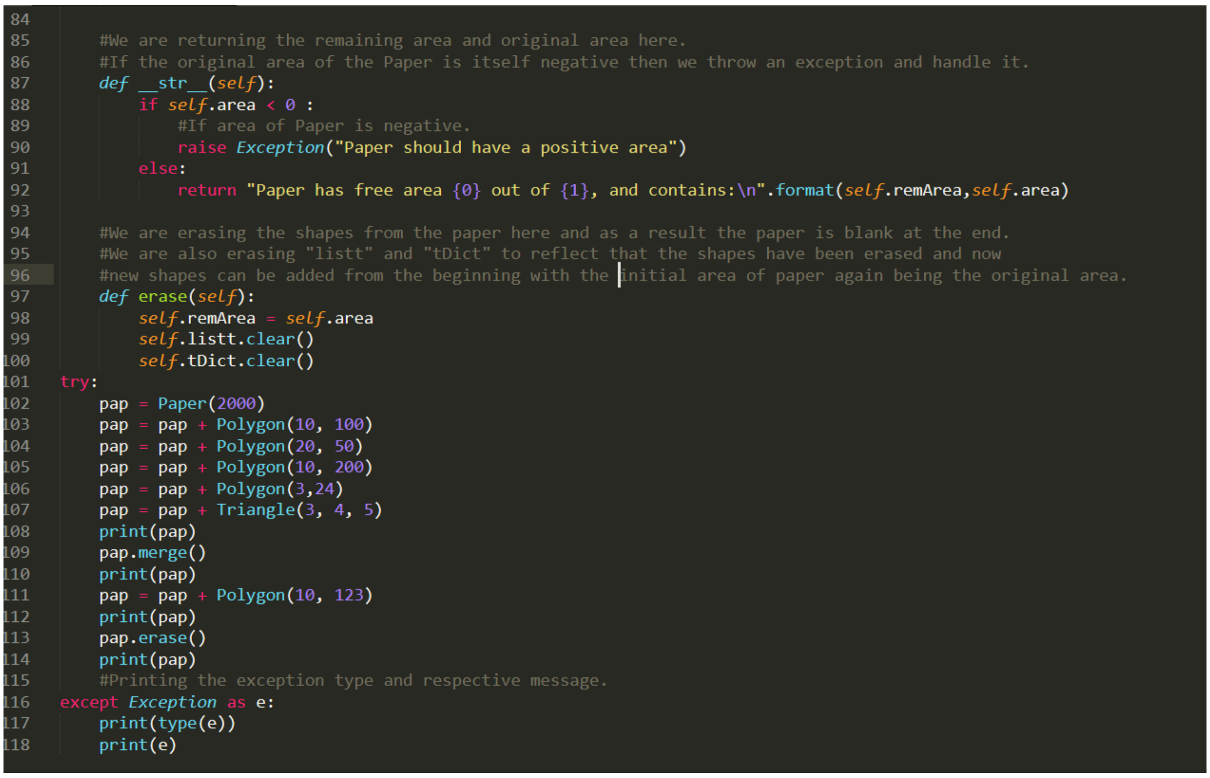
Command Prompt

```
C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>python Q3.py
Triangle with area 6.0

C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>python Q3.py
<class 'Exception'>
Triangle should have postive side-lengths

C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>python Q3.py
<class 'Exception'>
Side-lengths do not form a triangle

C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>
```

3) **Q4.py**:-We create a class **Paper** that has an attribute area and some additional attributes as follows:

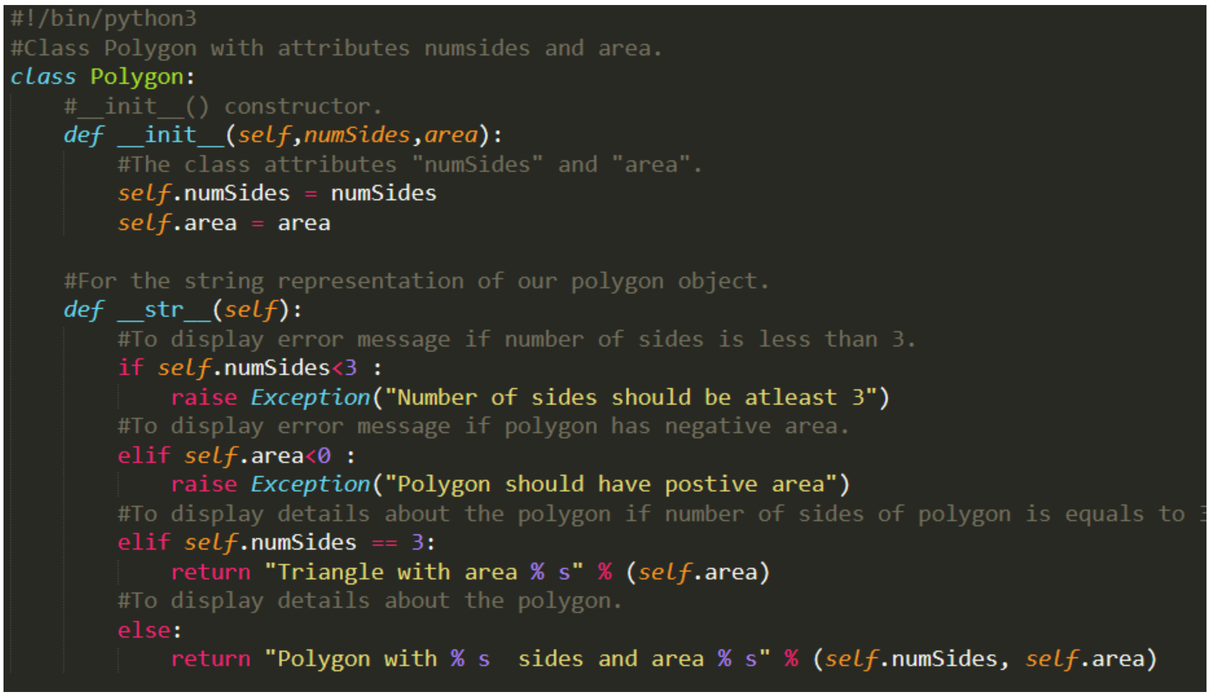


```python

class Paper:
    #__init__() constructor.
    def __init__(self,area):
        #The class attribute "area".
        #Additional attributes have been used for all the functions.
        self.area = area
        self.remArea = area
        self.tDict = {}
        self.listt = []
    #Here the + operator is being overloaded and an exception thrown when remaining area is not sufficient for the polygon to be added.
    #The details of the polygon being added, like area and numSides, are being added into a list of lists as [numSides, area].
    #We are also calculating the remaining area and printing the polygon details.
    def __add__(self,o):
        if self.remArea < o.area :
            raise Exception("Paper does not have sufficient free area to fit the polygon")
        self.key = o.numSides
        self.value = o.area
        self.listt.append([self.key,self.value])
        self.remArea = self.remArea - o.area
        print(o)
        #print(self.listt)
        return self
    #Here me are merging the polygons with same number of sides(numSides) by adding their areas. We have used a dictionary for this.
    #We loop over the list of lists for this purpose. The "pair[0]" has the numSides and "pair[1]" has the area. So if the numSides
    #are equal we will add the areas else create a new {key:value} pair for dictionary "tDict".
    def merge(self):
        for pair in self.listt:
            if pair[0] in self.tDict:
                self.tDict[pair[0]] = self.tDict[pair[0]] + pair[1]
            else:
                self.tDict[pair[0]] = pair[1]
        for x in self.tDict:
            if x == 3:
                print("Triangle with area "+str(self.tDict[x])+".")
            else:
                print("Polygon with "+str(x)+" sides and area "+str(self.tDict[x])+".")
        #print(self.tDict)
        #return self

```



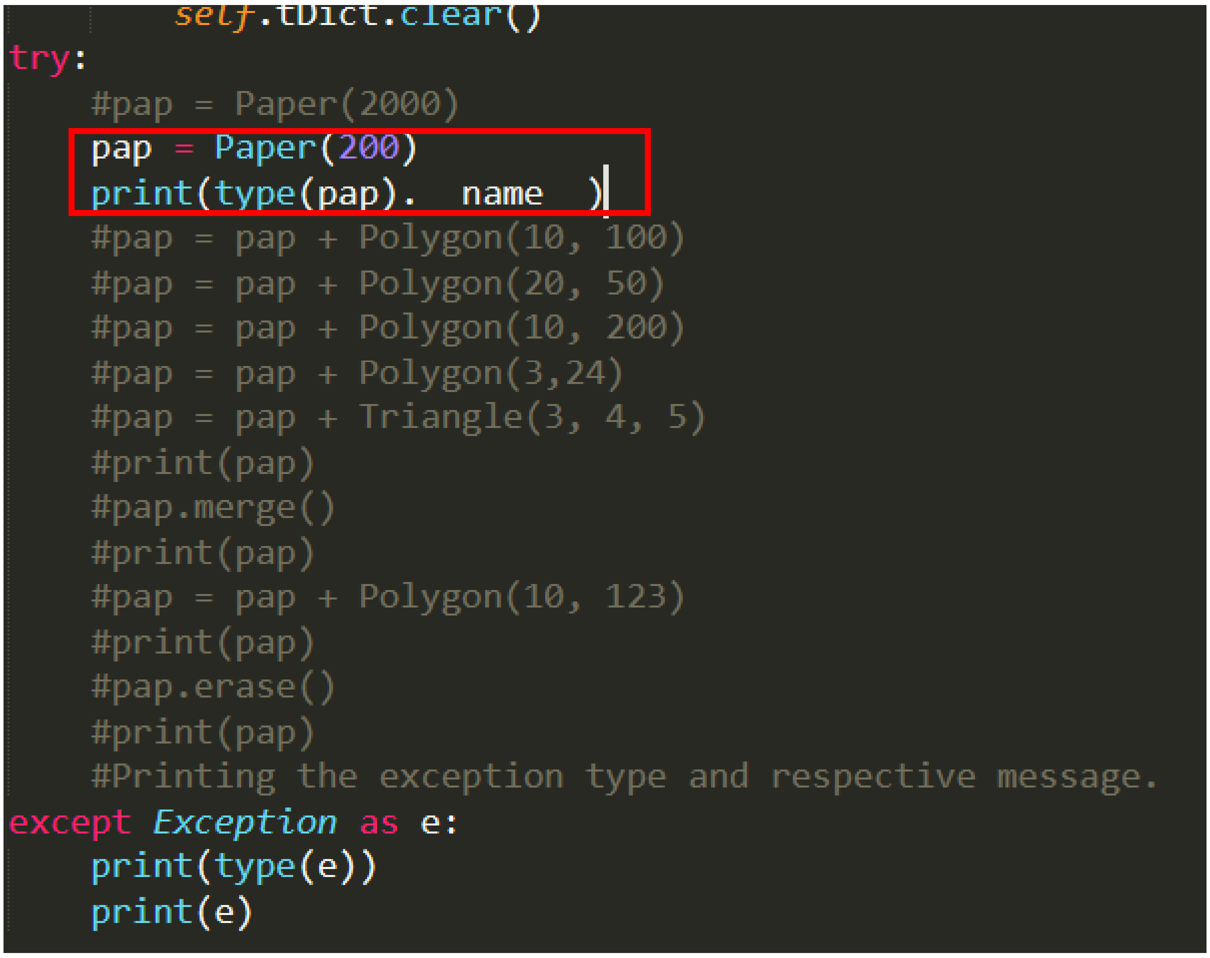
```python

    #We are returning the remaining area and original area here.
    #If the original area of the Paper is itself negative then we throw an exception and handle it.
    def __str__(self):
        if self.area < 0 :
            #If area of Paper is negative.
            raise Exception("Paper should have a positive area")
        else:
            return "Paper has free area {0} out of {1}, and contains:\n".format(self.remArea,self.area)

    #We are erasing the shapes from the paper here and as a result the paper is blank at the end.
    #We are also erasing "listt" and "tDict" to reflect that the shapes have been erased and now
    #new shapes can be added from the beginning with the initial area of paper again being the original area.
    def erase(self):
        self.remArea = self.area
        self.listt.clear()
        self.tDict.clear()
try:
    pap = Paper(2000)
    pap = pap + Polygon(10, 100)
    pap = pap + Polygon(20, 50)
    pap = pap + Polygon(10, 200)
    pap = pap + Polygon(3,24)
    pap = pap + Triangle(3, 4, 5)
    print(pap)
    pap.merge()
    print(pap)
    pap = pap + Polygon(10, 123)
    print(pap)
    pap.erase()
    print(pap)
    #Printing the exception type and respective message.
except Exception as e:
    print(type(e))
    print(e)
```

The Polygon class was also modified for this part of the assignment as follows:



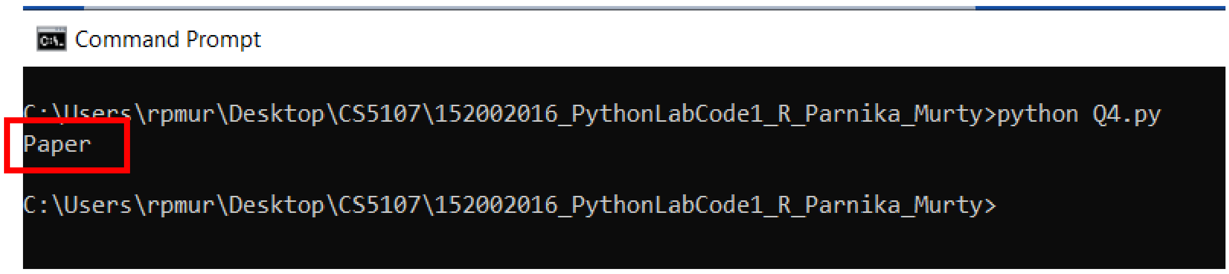
```python
#!/bin/python3
#Class Polygon with attributes numsides and area.
class Polygon:
    #__init__() constructor.
    def __init__(self,numSides,area):
        #The class attributes "numSides" and "area".
        self.numSides = numSides
        self.area = area

    #For the string representation of our polygon object.
    def __str__(self):
        #To display error message if number of sides is less than 3.
        if self.numSides<3 :
            raise Exception("Number of sides should be atleast 3")
        #To display error message if polygon has negative area.
        elif self.area<0 :
            raise Exception("Polygon should have postive area")
        #To display details about the polygon if number of sides of polygon is equals to
        elif self.numSides == 3:
            return "Triangle with area % s" % (self.area)
        #To display details about the polygon.
        else:
            return "Polygon with % s  sides and area % s" % (self.numSides, self.area)
```

It should be possible to create an object of type Paper by specifying area of the paper.



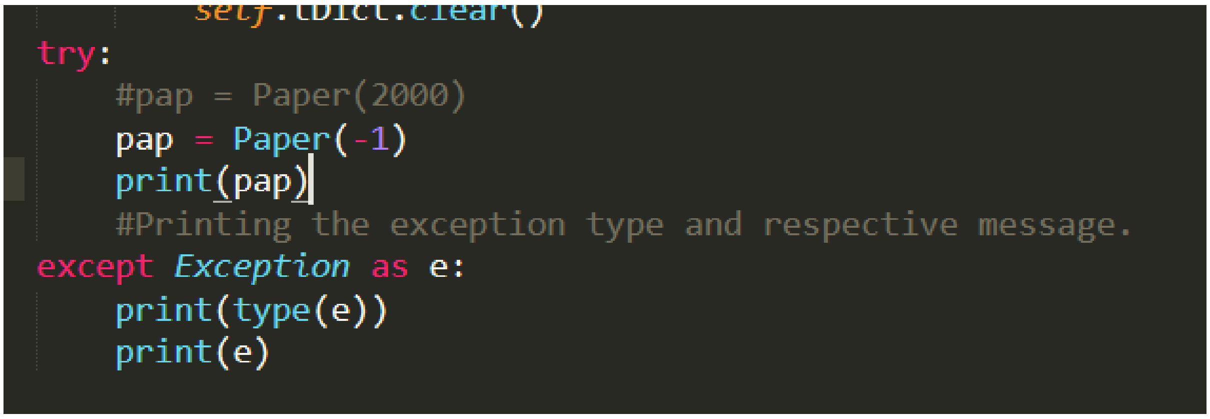
```
        self.tDict.clear()
try:
    #pap = Paper(2000)
    pap = Paper(200)
    print(type(pap).__name__)
    #pap = pap + Polygon(10, 100)
    #pap = pap + Polygon(20, 50)
    #pap = pap + Polygon(10, 200)
    #pap = pap + Polygon(3,24)
    #pap = pap + Triangle(3, 4, 5)
    #print(pap)
    #pap.merge()
    #print(pap)
    #pap = pap + Polygon(10, 123)
    #print(pap)
    #pap.erase()
    #print(pap)
    #Printing the exception type and respective message.
except Exception as e:
    print(type(e))
    print(e)
```

The output is:



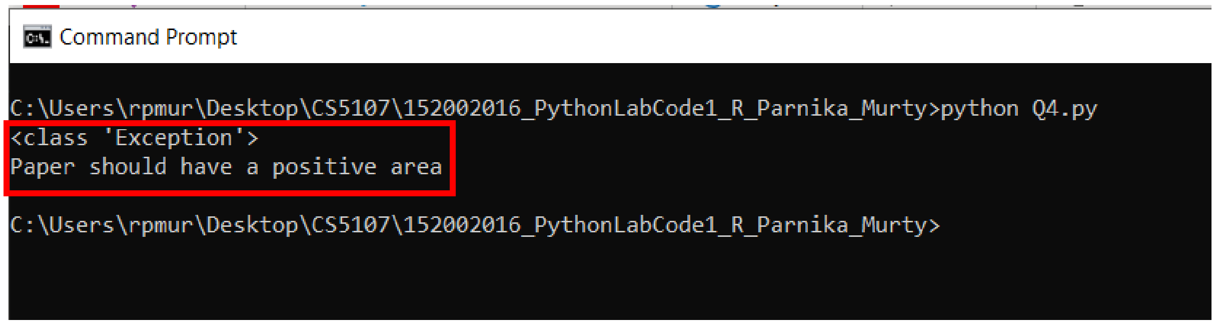
```
Command Prompt

C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>python Q4.py
Paper

C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>
```

Paper should have a positive area. Else, throw an exception and handle it.



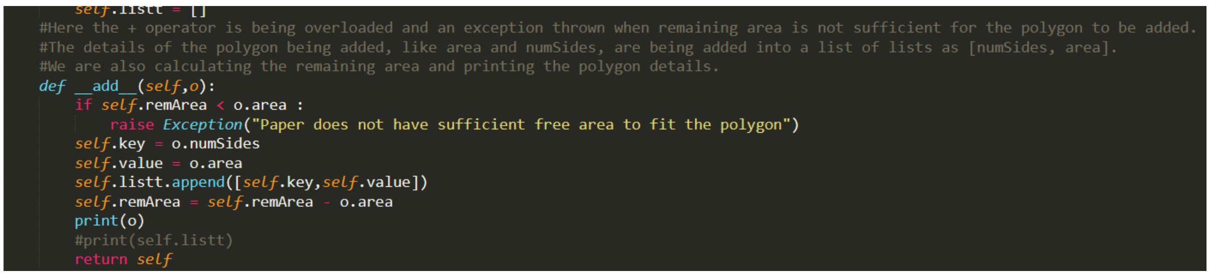
```
        self.tDict.clear()
try:
    #pap = Paper(2000)
    pap = Paper(-1)
    print(pap)
    #Printing the exception type and respective message.
except Exception as e:
    print(type(e))
    print(e)
```

The output is:

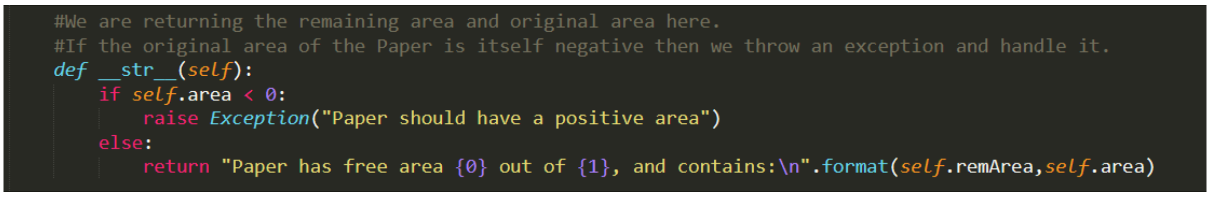

Command Prompt

```
C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>python Q4.py
<class 'Exception'>
Paper should have a positive area

C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>
```

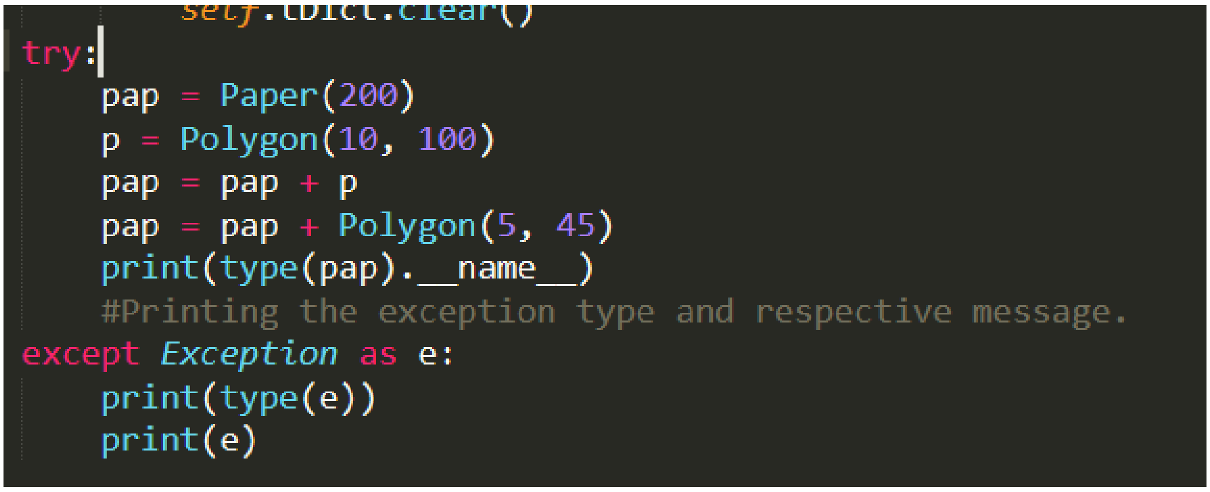
Overload the + operator so that you will be able add a polygon or triangle object to the paper object. Each time an object is added to the paper, available area of the paper decreases by the area attribute of the object.



```
        self.listt = []
    #Here the + operator is being overloaded and an exception thrown when remaining area is not sufficient for the polygon to be added.
    #The details of the polygon being added, like area and numSides, are being added into a list of lists as [numSides, area].
    #We are also calculating the remaining area and printing the polygon details.
    def __add__(self,o):
        if self.remArea < o.area :
            raise Exception("Paper does not have sufficient free area to fit the polygon")
        self.key = o.numSides
        self.value = o.area
        self.listt.append([self.key,self.value])
        self.remArea = self.remArea - o.area
        print(o)
        #print(self.listt)
        return self
```

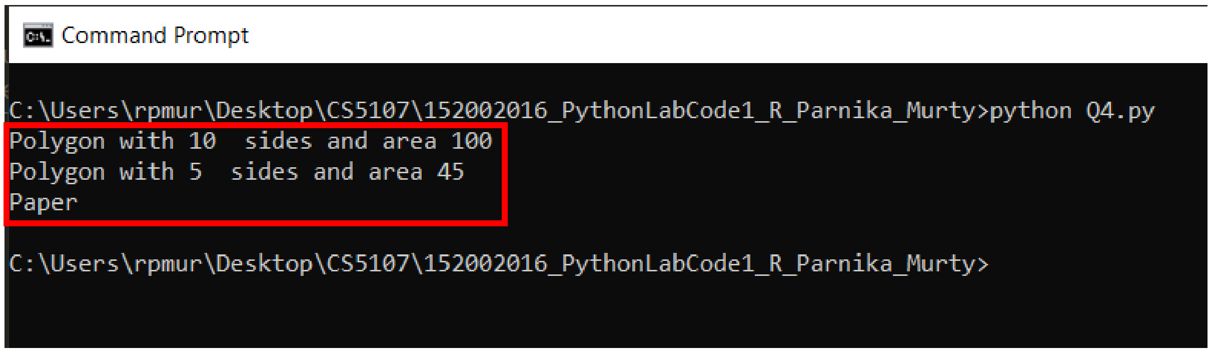


```
    #We are returning the remaining area and original area here.
    #If the original area of the Paper is itself negative then we throw an exception and handle it.
    def __str__(self):
        if self.area < 0:
            raise Exception("Paper should have a positive area")
        else:
            return "Paper has free area {0} out of {1}, and contains:\n".format(self.remArea,self.area)
```



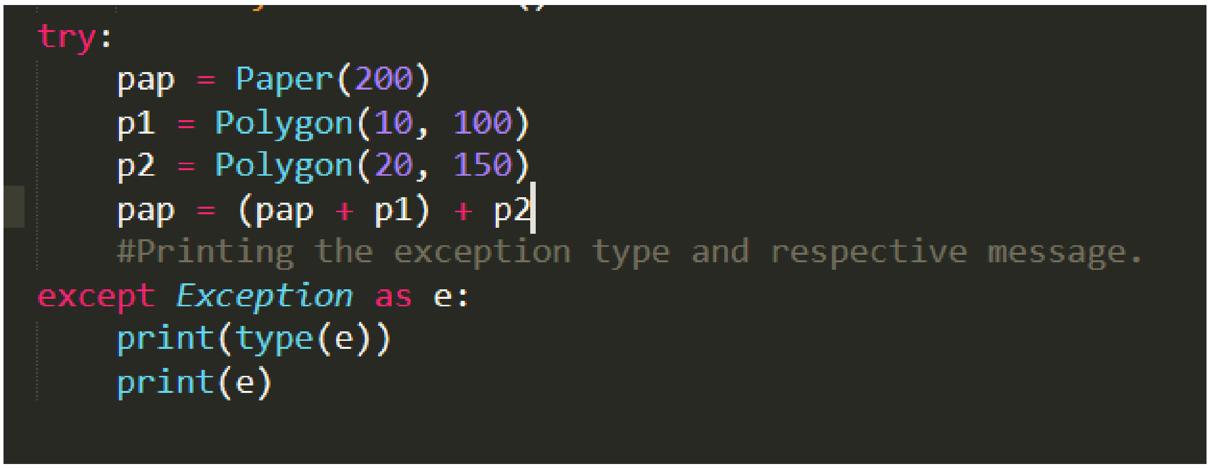
```
        self.tDict.clear()
try:
    pap = Paper(200)
    p = Polygon(10, 100)
    pap = pap + p
    pap = pap + Polygon(5, 45)
    print(type(pap).__name__)
    #Printing the exception type and respective message.
except Exception as e:
    print(type(e))
    print(e)
```

The output is:



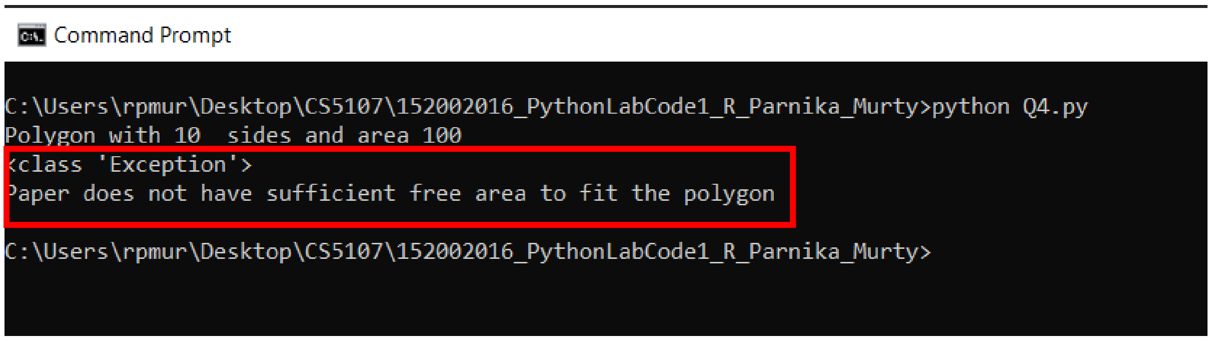
```
C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>python Q4.py
Polygon with 10  sides and area 100
Polygon with 5  sides and area 45
Paper

C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>
```

If the available area is less than the area of the polygon being added, an exception should be thrown and handled.



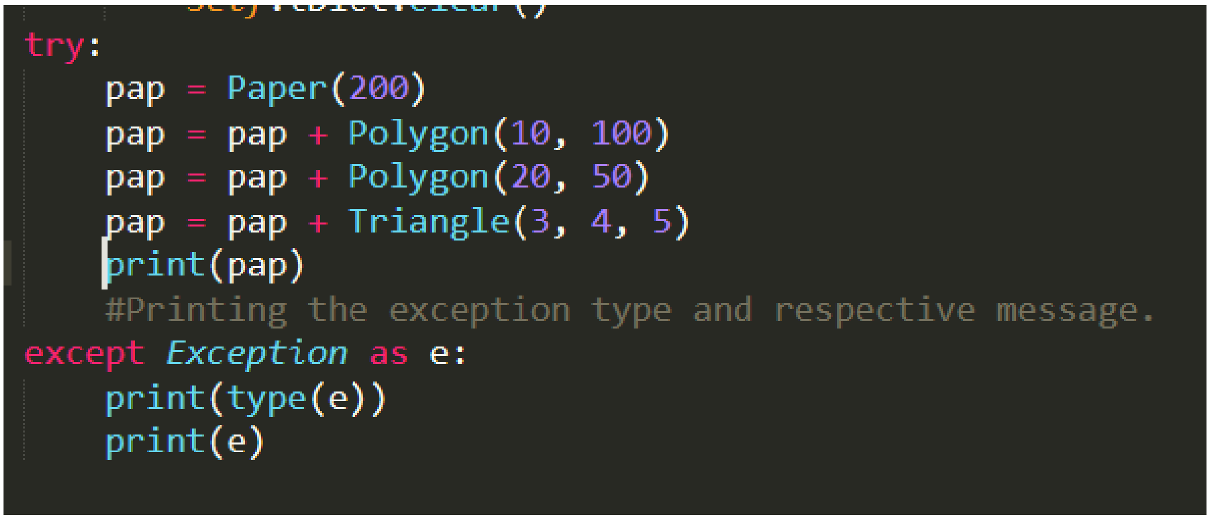
```python
try:
    pap = Paper(200)
    p1 = Polygon(10, 100)
    p2 = Polygon(20, 150)
    pap = (pap + p1) + p2
    #Printing the exception type and respective message.
except Exception as e:
    print(type(e))
    print(e)
```

The output is:

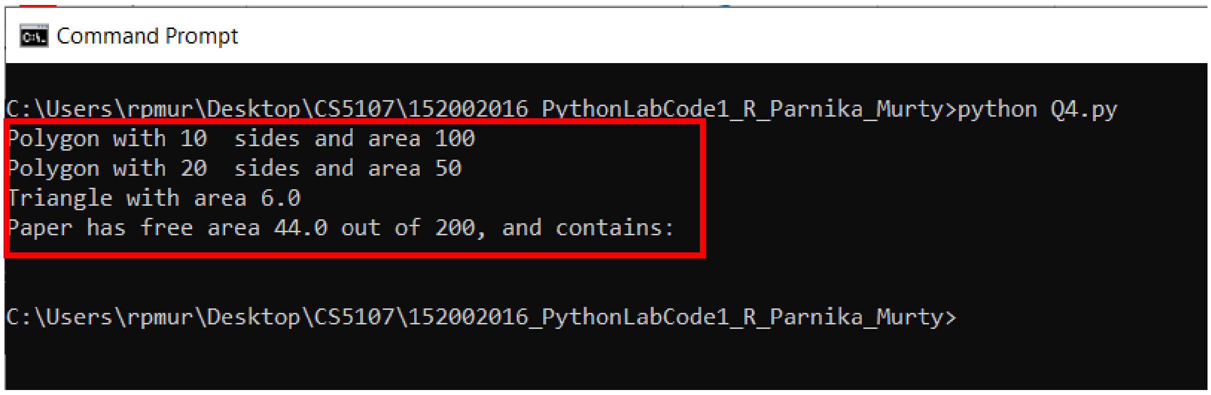


```
C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>python Q4.py
Polygon with 10  sides and area 100
<class 'Exception'>
Paper does not have sufficient free area to fit the polygon

C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>
```

It should be possible to print an object of type Paper; outputs free area in the paper and details of all polygons and triangles added to the paper. My code prints the remaining area statement after printing the polygons' details.

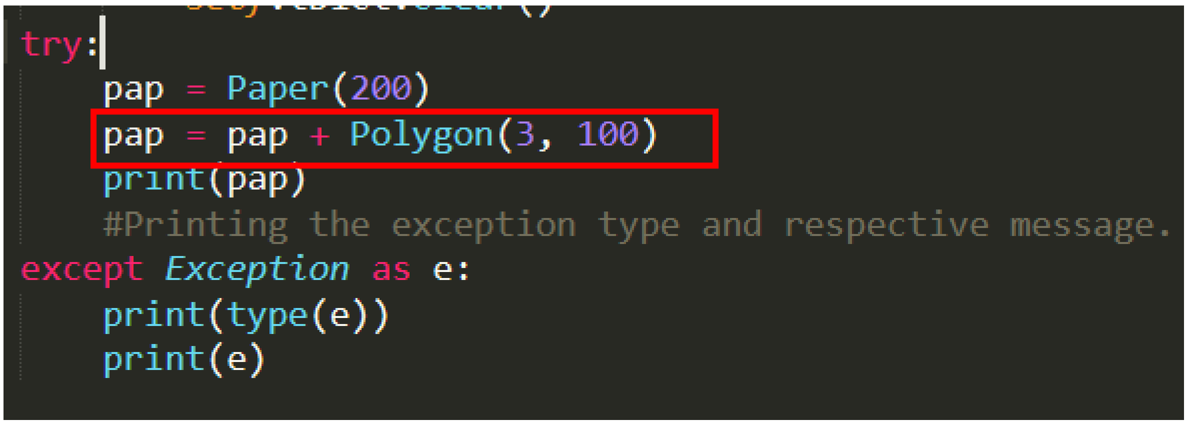

```
try:
    pap = Paper(200)
    pap = pap + Polygon(10, 100)
    pap = pap + Polygon(20, 50)
    pap = pap + Triangle(3, 4, 5)
    print(pap)
    #Printing the exception type and respective message.
except Exception as e:
    print(type(e))
    print(e)
```

The output is:

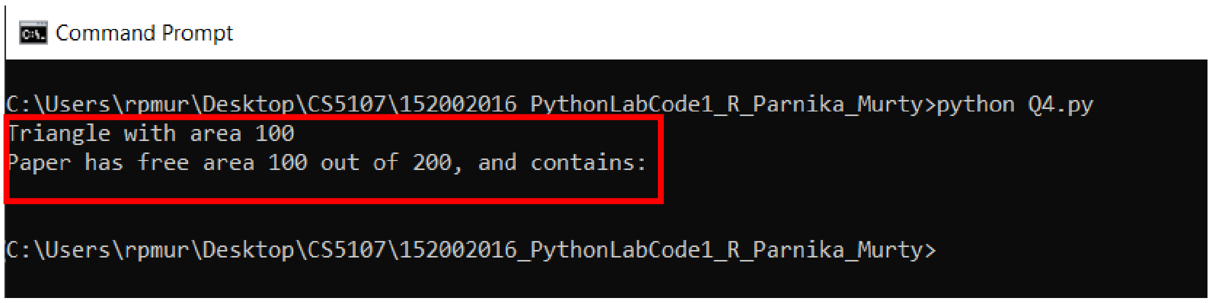

Command Prompt

```
C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>python Q4.py
Polygon with 10  sides and area 100
Polygon with 20  sides and area 50
Triangle with area 6.0
Paper has free area 44.0 out of 200, and contains:

C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>
```

Polygons with 3 sides added to the paper are always stored as triangles.

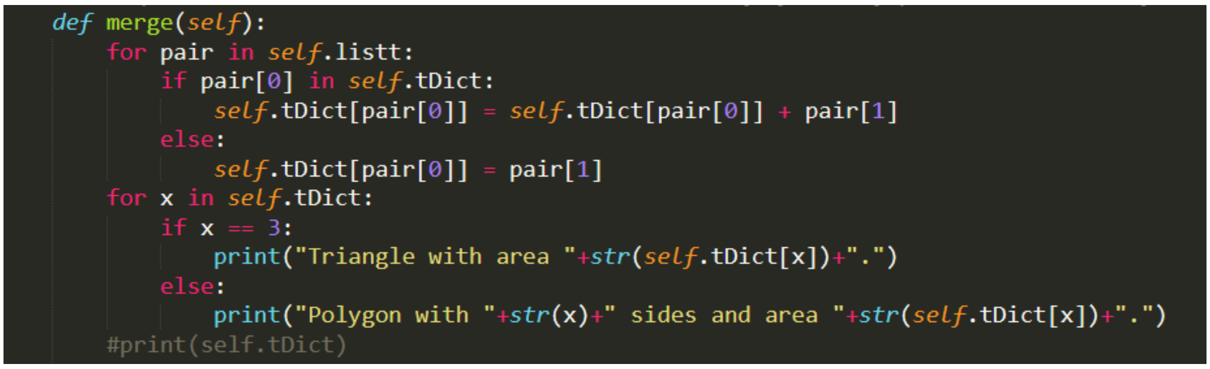

```
try:
    pap = Paper(200)
    pap = pap + Polygon(3, 100)
    print(pap)
    #Printing the exception type and respective message.
except Exception as e:
    print(type(e))
    print(e)
```

The output is:

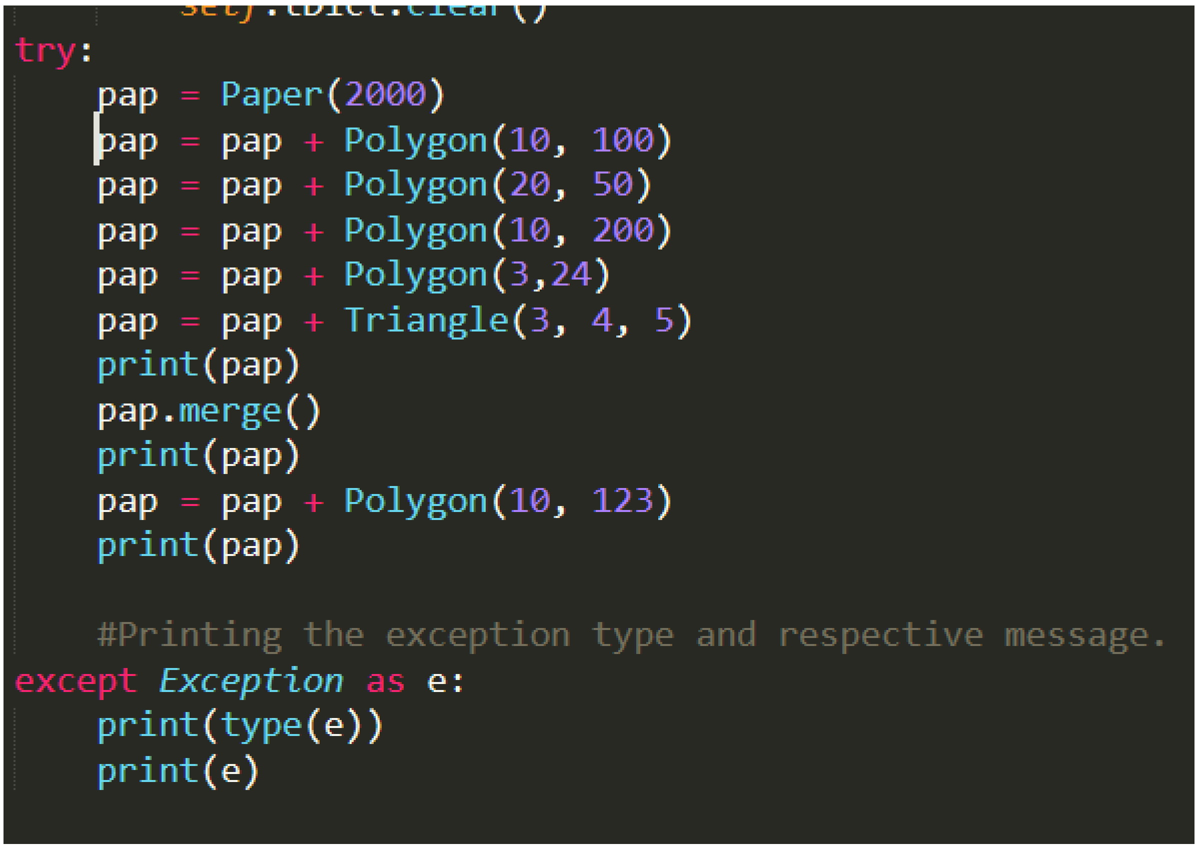

Command Prompt

```
C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>python Q4.py
Triangle with area 100
Paper has free area 100 out of 200, and contains:

C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>
```

It should have a method merge that will combine polygons with same number of sides together. Area of the resultant polygon is sum of areas of the individual polygons.

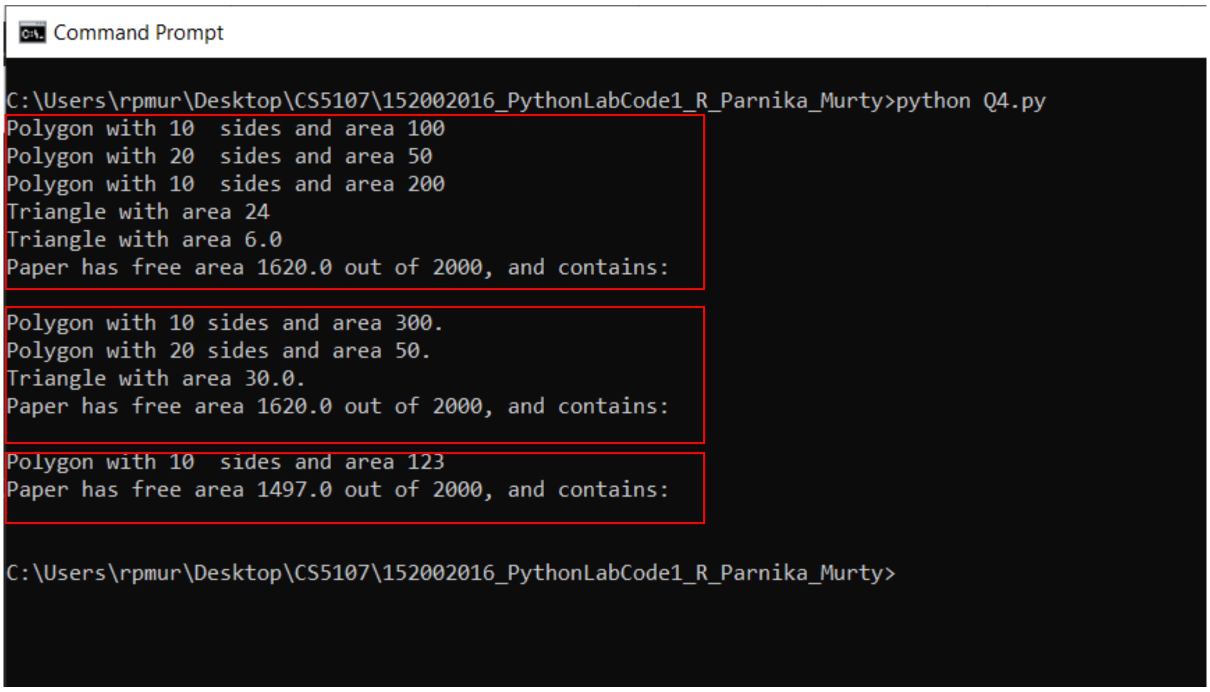


```python
    def merge(self):
        for pair in self.listt:
            if pair[0] in self.tDict:
                self.tDict[pair[0]] = self.tDict[pair[0]] + pair[1]
            else:
                self.tDict[pair[0]] = pair[1]
        for x in self.tDict:
            if x == 3:
                print("Triangle with area "+str(self.tDict[x])+".")
            else:
                print("Polygon with "+str(x)+" sides and area "+str(self.tDict[x])+".")
        #print(self.tDict)
```



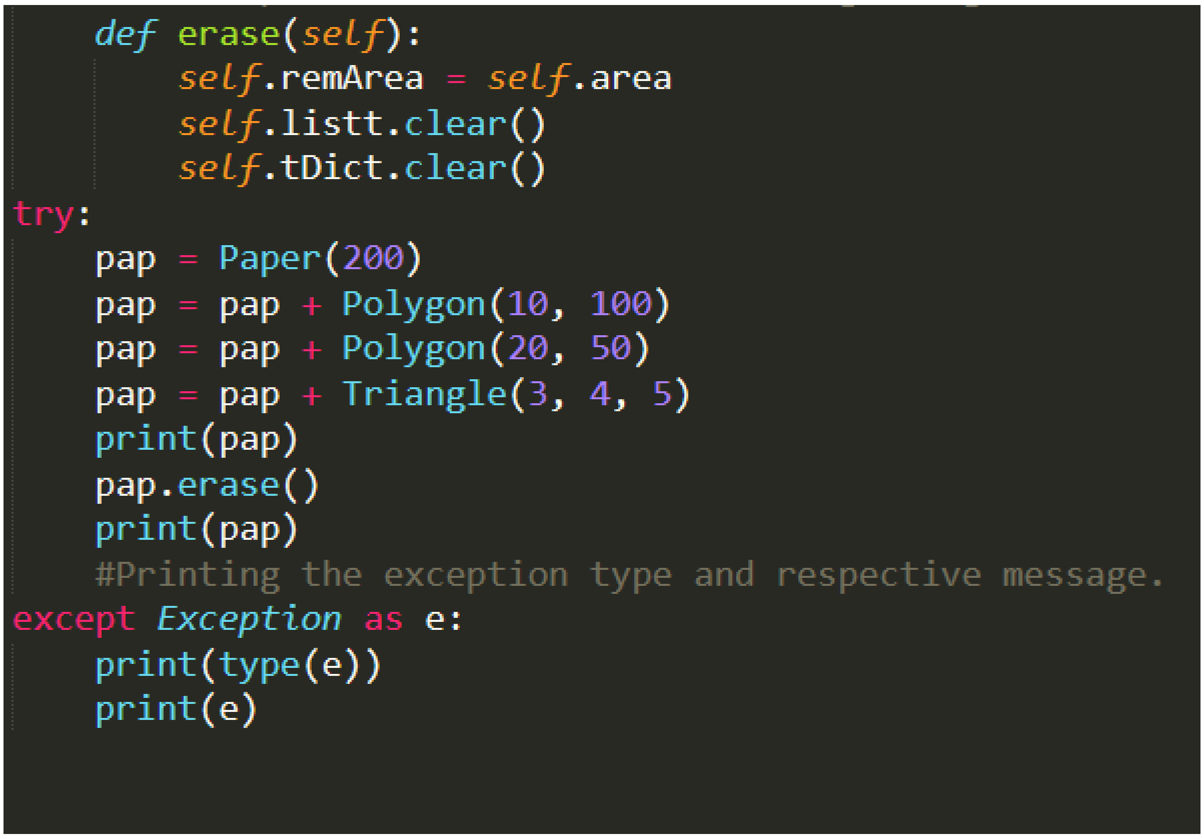
```python
        self.tDict.clear()
try:
    pap = Paper(2000)
    pap = pap + Polygon(10, 100)
    pap = pap + Polygon(20, 50)
    pap = pap + Polygon(10, 200)
    pap = pap + Polygon(3,24)
    pap = pap + Triangle(3, 4, 5)
    print(pap)
    pap.merge()
    print(pap)
    pap = pap + Polygon(10, 123)
    print(pap)

    #Printing the exception type and respective message.
except Exception as e:
    print(type(e))
    print(e)
```

The output is:



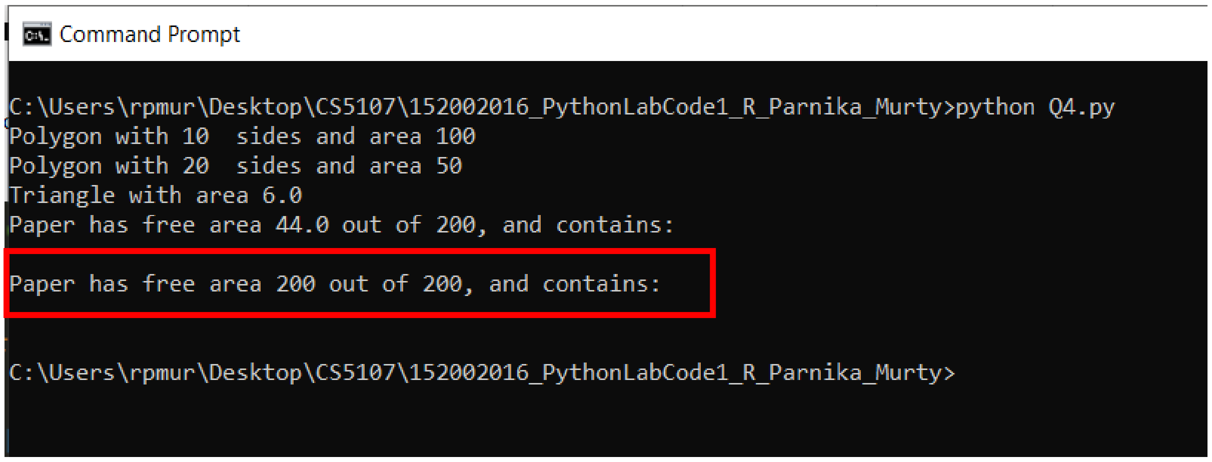
```
C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>python Q4.py
Polygon with 10  sides and area 100
Polygon with 20  sides and area 50
Polygon with 10  sides and area 200
Triangle with area 24
Triangle with area 6.0
Paper has free area 1620.0 out of 2000, and contains:

Polygon with 10 sides and area 300.
Polygon with 20 sides and area 50.
Triangle with area 30.0.
Paper has free area 1620.0 out of 2000, and contains:

Polygon with 10  sides and area 123
Paper has free area 1497.0 out of 2000, and contains:


C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>
```

It should have a method erase that will remove everything from the paper.:



```python
    def erase(self):
        self.remArea = self.area
        self.listt.clear()
        self.tDict.clear()
try:
    pap = Paper(200)
    pap = pap + Polygon(10, 100)
    pap = pap + Polygon(20, 50)
    pap = pap + Triangle(3, 4, 5)
    print(pap)
    pap.erase()
    print(pap)
    #Printing the exception type and respective message.
except Exception as e:
    print(type(e))
    print(e)
```

The output is:


Command Prompt

```
C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>python Q4.py
Polygon with 10  sides and area 100
Polygon with 20  sides and area 50
Triangle with area 6.0
Paper has free area 44.0 out of 200, and contains:

Paper has free area 200 out of 200, and contains:


C:\Users\rpmur\Desktop\CS5107\152002016_PythonLabCode1_R_Parnika_Murty>
```

Thank you