As per the assignment I have created three python files named **Q2.py, Q3.py** and **Q4.py** . This report contains detailed explanation of my code.

Description of Q2.py:

Q2.py file a contains **Polygon** class, which has 2 class attributes named **numSides** and **area.**

Properties of Polygon class:

- It has 2 class attributes named **numSides** and **area.**
- It should have at-least 3 sides and non-zero area. Else, an exception should be thrown and handled.

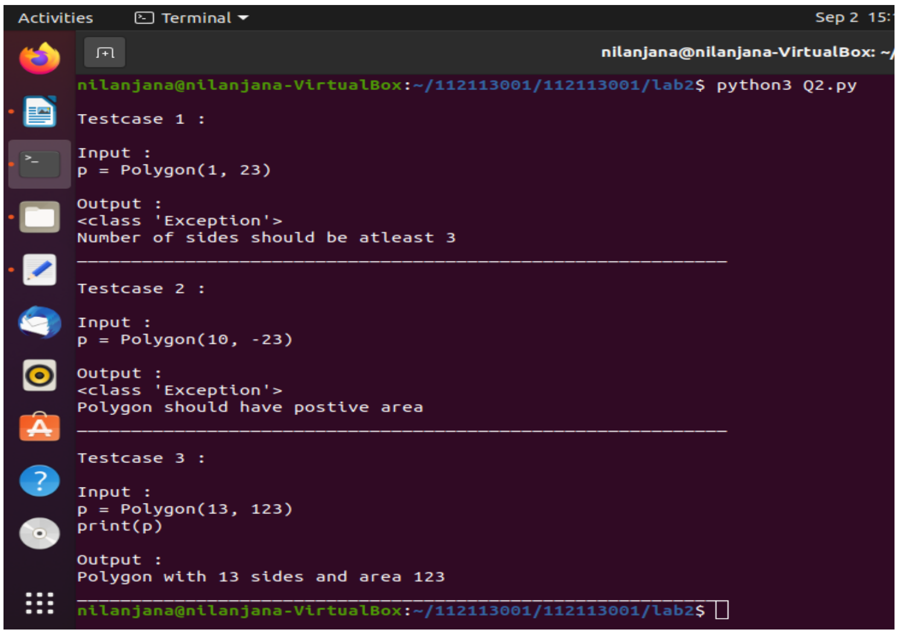
__Init__(self, numSides, area) method or constructor of Polygon class:

- **__init__()** method is the first method that will be invoked at the time of **object creation**
- In the **Polygon** class **__init__()** function is used to initialize the 2 class attricutes **numSides** and **area.**
- For handling exceptions, t**ry except** block is used. A Polygon object can be created only if defined with a **area > 0 and numSides > 2** otherwisse it will raise **Exception.**
- Here **self** is used to assign values to the attributes of the **Polygon** object.
- **raise Exception(“**message to user **”)** is used to raise custom exceptions , when **numSides < 3 or area < 0.**
- **except** block will catch the exception raised in **try** block and it will print the **type of exception and the message to user for that exception.**

__str__(self) method of Polygon class:

- **__str()** method is used to return an object as a string . We can use it to print a object as per our customization.
- **format()** method is used to avoid any string formatting error and for converting float to string.
- When we want to print an object of **Polygon** class the object will be formatted as a string by using **__str__()** method

Snapshots of Output:



```
nilanjana@nilanjana-VirtualBox:~/112113001/112113001/lab2$ python3 Q2.py
Testcase 1 :

Input :
p = Polygon(1, 23)

Output :
<class 'Exception'>
Number of sides should be atleast 3
------------------------------------------------------------
Testcase 2 :

Input :
p = Polygon(10, -23)

Output :
<class 'Exception'>
Polygon should have postive area
------------------------------------------------------------
Testcase 3 :

Input :
p = Polygon(13, 123)
print(p)

Output :
Polygon with 13 sides and area 123
------------------------------------------------------------
nilanjana@nilanjana-VirtualBox:~/112113001/112113001/lab2$
```

Description of Q3.py:

Q3.py file a contains **Triangle** class, which is a **derived class** of **Polygon** class.

Properties of Triangle class:

- It **Inherits** the properties of **Polygon** class.
- It has **no additional** attributes of its own but **2 inherited** attributes named **numSides** and **area** from its base class **Polygon.**
- It takes 3 attributes **a, b, c** as its 3 side-length but don't stores them.
- All 3 sides-lengths of the Triangle should be Positive number.
- For a **Triangle** object **sum of any 2 sides is greater than the side-length of 3rd side.**
- **Exceptions** will be **raised and and Handled** in the class itself, if triangle properties are not satisfied.

__Init__(self, a, b, c) method or constructor of Triangle class:

- **__init__()** method is the first method that will be invoked at the time of **object creation**
- In the **Triangle** class **__init__()** method is used to initialize the 2 inherited attricutes **numSides** and **area** of its **Base class Polygon.**
- For handling exceptions, t**ry except** block is used.
- **__init__()** will take **a, b, c** as argument and checks if **a <= 0 or b <= 0 or c <= 0** or not. If any of the side-lengths is **-ve** it will **raise an exception.**
- It will check if **(a < b+c) and (b < a+c) and (c < b+a)** or not . If the condition satisfies, it will call the **__init__()** method of the **base class Polygon** with the values as **numsides = 3** and **area = area of the Triangle** computed using semiperimeter formula **.**

  (code snippet:

  ```
  super(Triangle,self).__init__(3,(((a+b+c)/2)*(((a+b+c)/2)-a)*(((a+b+c)/2)-b)*(((a+b+c)/2)-c))**(1/2))
  ```

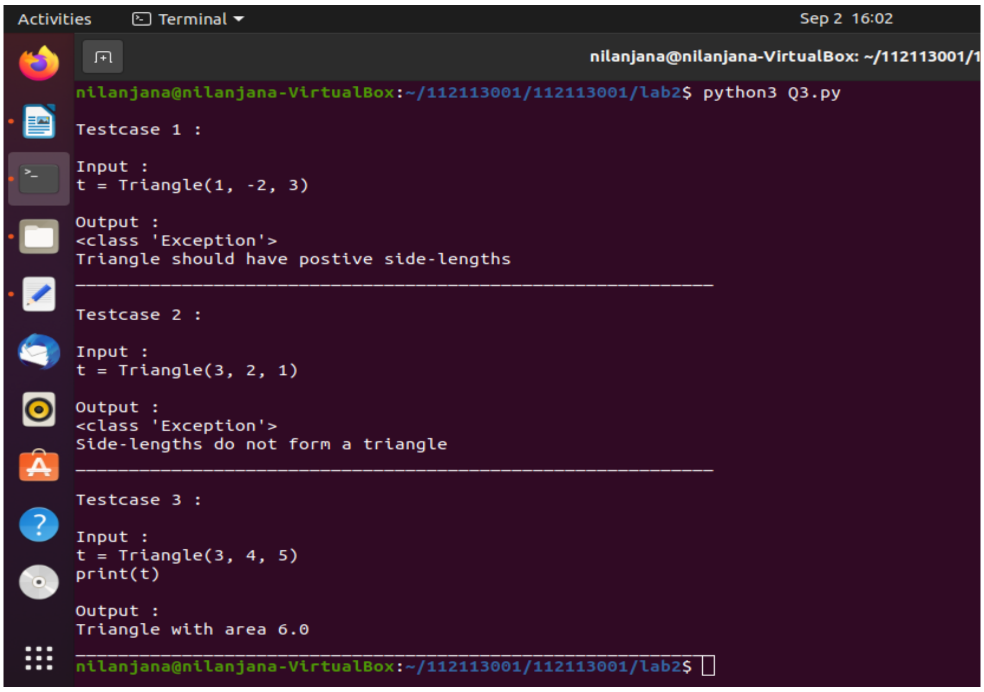
)

  If the above condition if false , an exception will be raised as the side-lengths does not form a triangle.
- **super()** method is used to know the Base class .
- **raise Exception("**message to user **")** is used to raise custom exceptions .
- **except** block will catch the exception raised in **try** block and it will print the **type of exception and the message to user for that exception.**

__str__(self) method of Triangle class:

- **__str()** method is used to return an object as a string . We can use it to print a object as per our customization.
- **format()** method is used to avoid any string formatting error and for converting float to string.
- When we want to print an object of **Triangle** class the object will be formatted as a string by using **__str__()** method

Snapshots of Output:



```
nilanjana@nilanjana-VirtualBox:~/112113001/112113001/lab2$ python3 Q3.py

Testcase 1 :

Input :
t = Triangle(1, -2, 3)

Output :
<class 'Exception'>
Triangle should have postive side-lengths
------------------------------------------------------------

Testcase 2 :

Input :
t = Triangle(3, 2, 1)

Output :
<class 'Exception'>
Side-lengths do not form a triangle
------------------------------------------------------------

Testcase 3 :

Input :
t = Triangle(3, 4, 5)
print(t)

Output :
Triangle with area 6.0
------------------------------------------------------------
nilanjana@nilanjana-VirtualBox:~/112113001/112113001/lab2$ 
```

Description of Q4.py:

Q4.py file a contains **Paper** class, which has 3 attributes **area, full_area, objList[]**.

area represents the available area where other objects of Triangle and Polygon class can be added
full_area represents the full area of the Paper object .
objList[] represents a list of objects that are already added to the Paper object.

Properties of Paper class:

- It will take value for only **area** attribute to create a Paper object.
- It should have non-zero area. Else, an exception should be thrown and handled.
- Any object of class Polygon or Triangle will only be added to a Paper object only if it has enough space to fit that object.
- **Exceptions** will be **raised and and Handled** in the class itself .

__Init__(self, area) method or constructor of Paper class:

- **__init__()** method is the first method that will be invoked at the time of **object creation.**
- In the **Paper** class **__init__()** function is used to initialize the 3 class attricutes **area, full_area** and **objList[]. area** and **full_area** will be initiated by the value of area parameter passed for creating Paper object and **objList[]** will be initialized by an empty list , as at the time of list creation, it doesnot have any object added to it.
- For handling exceptions, t**ry except** block is used. A Paper object can be created only if defined with a **area** > **0** otherwise it will raise **Exception.**
- Here **self** is used to assign values to the attributes of the **Paper** object.

- **raise Exception("**message to user **")** is used to raise custom exceptions , when **area < 0.**
- **except** block will catch the exception raised in **try** block and it will print the **type of exception and the message to user for that exception.**

__str__(self) method of Paper class:

- **__str()** method is used to return an object as a string . We can use it to print a object as per our customization.
- **format()** method is used to avoid any string formatting error and for converting float to string.
- While printing the Paper object, it should also show us the objects it holds along with its free area and total area.
- Used for loop to i**terate objList[]** list and added the details of each object with the string representation of Paper object.
- When we want to print an object of **Paper** class the object will be formatted as a string by using **__str__()** method

__add__(self, other) method of Paper class to Overload '+' operator :

- Operator Overloading is used to give extra ability to an operator to perform more than its predefined operation.
- **'+'** is overloaded using **__add__()** method overloading.
- When an object of **Polygon** or **Triangle** is added to a Paper object , it will check first, if the paper has enough space to fit that object. If it has enough space it will add that object to the paper by appending the object in **objList[]** list. Otherwise, an exception will be raised .
- **try except** blockis used to handle exceptions .
- **raise Exception("**message to user **")** is used to raise custom exceptions .
- **except** block will catch the exception raised in **try** block and it will print the **type of exception and the message to user for that exception.**

merge (self) method of Paper class to merge objects with same number of sides:

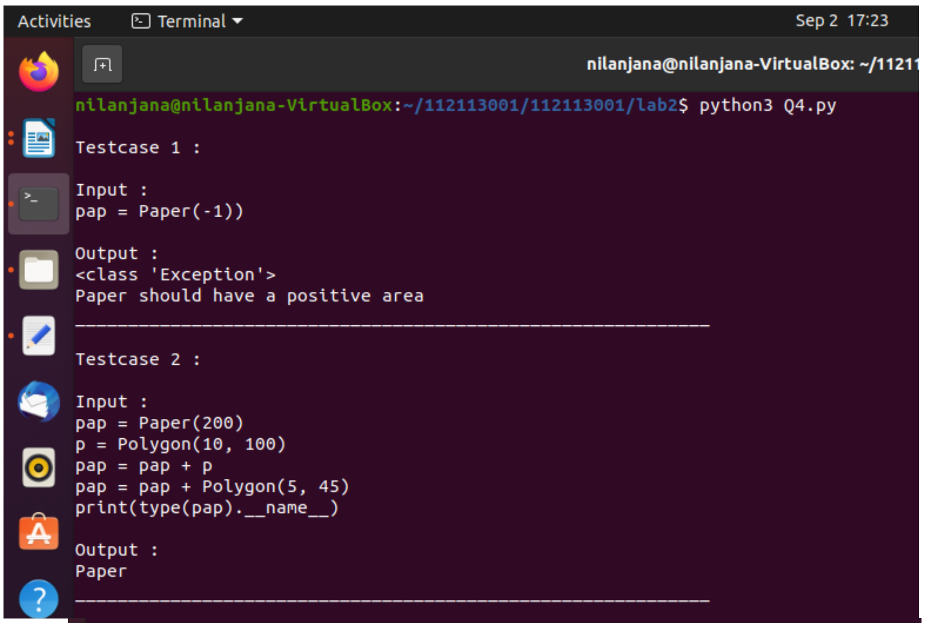
- when **merge()** method is called , It will iterate through **objList[]** list and check if any other object in objList[] is having same no of sides or not, if yes, it will increase the area of the object by adding the area value of other one and then delete that other object from objList[] (lets say , obj1 , obj2 are having numSides 10 , so obj1. area will be increased to obj1.area + obj2.area and obj2 will be removed from objlist).
- **Code snippet:**

```
(
    i=0
    while i < len(self.objList):
        j = i + 1
        while j < len(self.objList):
            if(self.objList[i].numSides == self.objList[j].numSides):
             self.objList[i].area += self.objList[j].area
             self.objList.remove(self.objList[j])
             j += 1
        i += 1
)
```

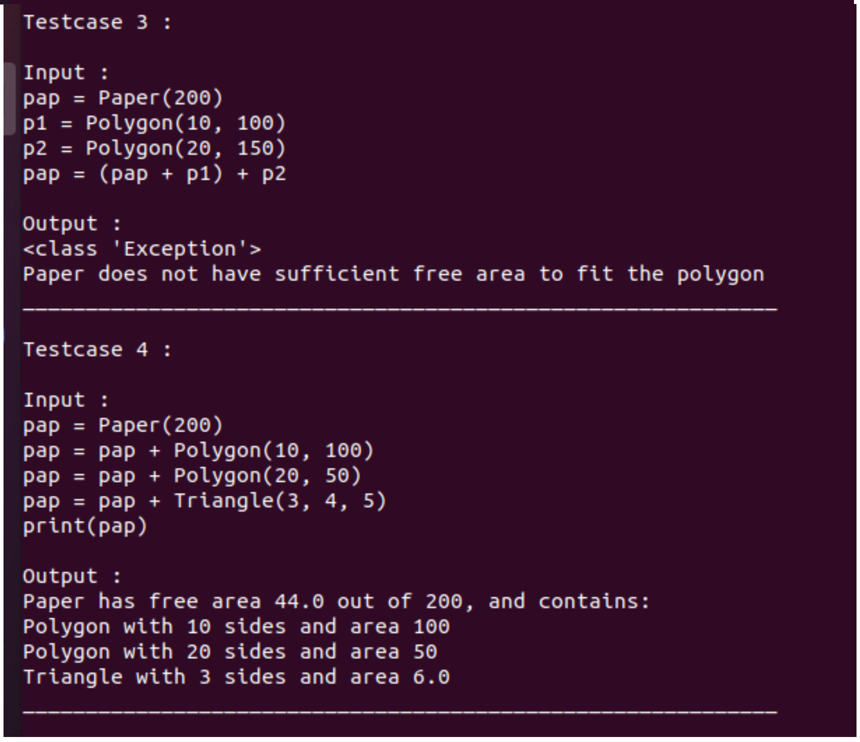
erase(self) method of Paper class to erase previously added objects from Paper object :

- When **erase()** function is called it erases previously added objects fron Paper object and after erasing the paper will have all its srea available for further operations.
- **clear()** is used here to clear the **objList[]** list . As the objList is empty , it represents that Paper object has no other objects added to it .
- **area** attribute is assigned with the value of **full_area** to get the papaer object area as its initial value at the time of Paper object creation.

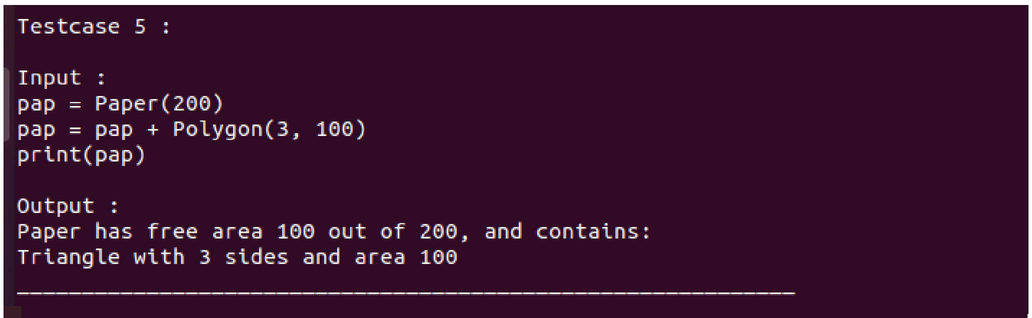
Snapshot of Output:



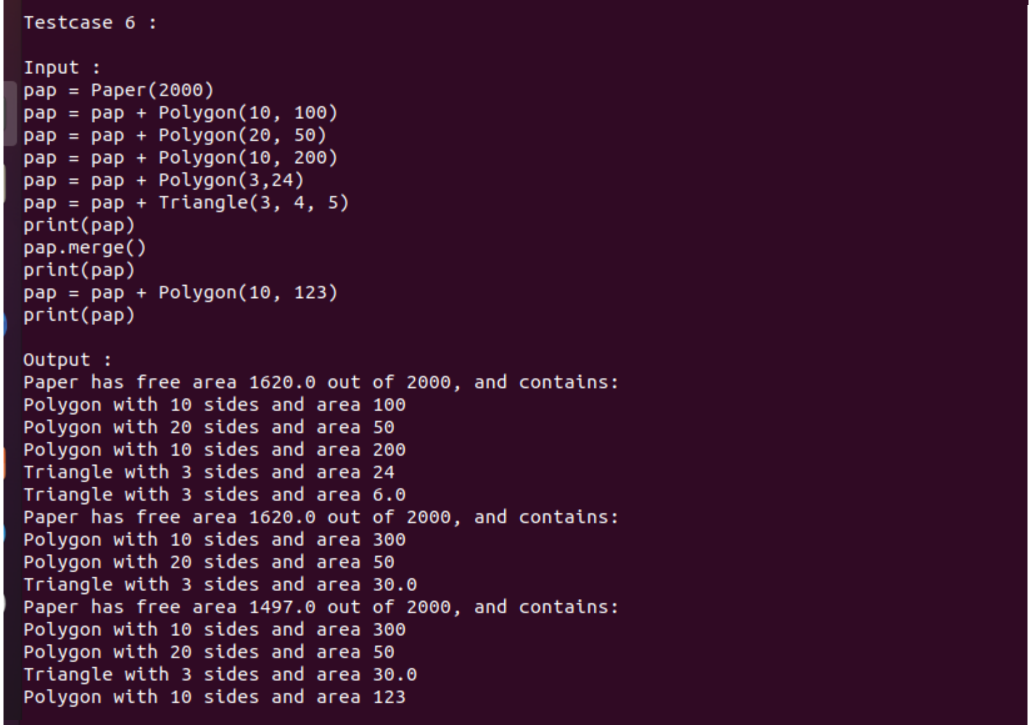
```
nilanjana@nilanjana-VirtualBox:~/112113001/112113001/lab2$ python3 Q4.py

Testcase 1 :

Input :
pap = Paper(-1))

Output :
<class 'Exception'>
Paper should have a positive area
------------------------------------------------------------

Testcase 2 :

Input :
pap = Paper(200)
p = Polygon(10, 100)
pap = pap + p
pap = pap + Polygon(5, 45)
print(type(pap).__name__)

Output :
Paper
------------------------------------------------------------
```



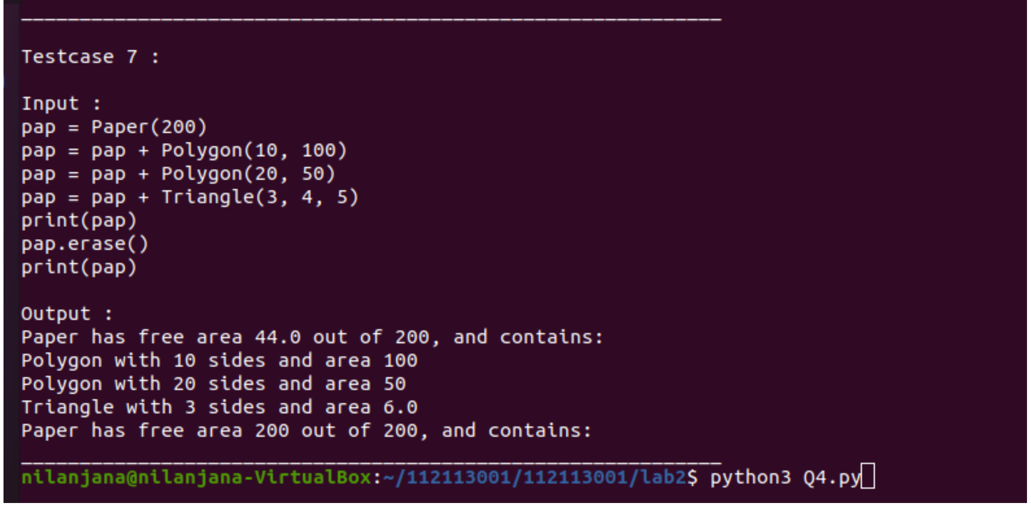
```
Testcase 3 :

Input :
pap = Paper(200)
p1 = Polygon(10, 100)
p2 = Polygon(20, 150)
pap = (pap + p1) + p2

Output :
<class 'Exception'>
Paper does not have sufficient free area to fit the polygon
------------------------------------------------------------

Testcase 4 :

Input :
pap = Paper(200)
pap = pap + Polygon(10, 100)
pap = pap + Polygon(20, 50)
pap = pap + Triangle(3, 4, 5)
print(pap)

Output :
Paper has free area 44.0 out of 200, and contains:
Polygon with 10 sides and area 100
Polygon with 20 sides and area 50
Triangle with 3 sides and area 6.0
------------------------------------------------------------
```



```
Testcase 5 :

Input :
pap = Paper(200)
pap = pap + Polygon(3, 100)
print(pap)

Output :
Paper has free area 100 out of 200, and contains:
Triangle with 3 sides and area 100

------------------------------------------------------------
```



```
Testcase 6 :

Input :
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)

Output :
Paper has free area 1620.0 out of 2000, and contains:
Polygon with 10 sides and area 100
Polygon with 20 sides and area 50
Polygon with 10 sides and area 200
Triangle with 3 sides and area 24
Triangle with 3 sides and 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 3 sides and area 30.0
Paper has free area 1497.0 out of 2000, and contains:
Polygon with 10 sides and area 300
Polygon with 20 sides and area 50
Triangle with 3 sides and area 30.0
Polygon with 10 sides and area 123

------------------------------------------------------------
```



```
Testcase 7 :

Input :
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)

Output :
Paper has free area 44.0 out of 200, and contains:
Polygon with 10 sides and area 100
Polygon with 20 sides and area 50
Triangle with 3 sides and area 6.0
Paper has free area 200 out of 200, and contains:
------------------------------------------------------------
nilanjana@nilanjana-VirtualBox:~/112113001/112113001/lab2$ python3 Q4.py
```