# **Name: Abdurrahman Qureshi**

# **Roll No: 210451**

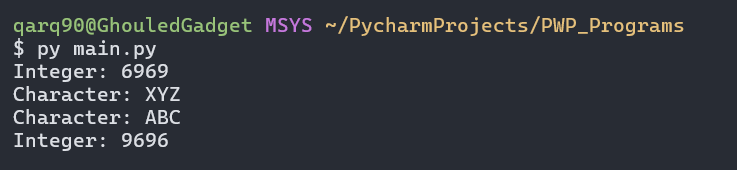
Practical No: 14

1) Create a class that prints an integer and a character with two methods having the same name but different sequences of integer and character parameters.

CODE:

class IntrospectionInPython:  
 def print\_karo(self, param1, param2):  
  
 if isinstance(param1, int):  
 print("Integer:", param1)  
 print("Character:", param2)  
 elif isinstance(param1, str):  
 print("Character:", param1)  
 print("Integer:", param2)  
  
  
obj = IntrospectionInPython()  
obj.print\_karo(6969, "XYZ")  
obj.print\_karo("ABC", 9696)

OUTPUT:

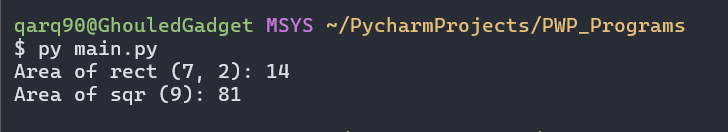
****

2) Create a class to print areas of squares and rectangles. The class has two methods with the same name but different numbers of parameters. One method has two parameters for the length and breadth of a rectangle, while the other has one parameter for the side of a square.

**CODE:**

class shape\_bolte:  
 def area\_karo\_print(self, \*args):  
  
 if len(args) == 2:  
 return args[0] \* args[1]  
  
 if len(args) == 1:  
 return args[0] \*\* 2  
  
  
areas = shape\_bolte()  
print("Area of rect (7, 2):", areas.area\_karo\_print(7, 2))  
print("Area of sqr (9):", areas.area\_karo\_print(9))

**OUTPUT:**

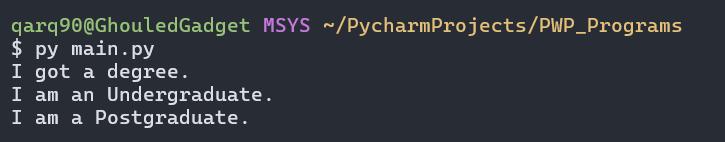
****

3) Create a class "Degree" having a "getDegree" method that prints "I got a degree". Two subclasses of the aforementioned class named "Undergraduate" and "Postgraduate" each have the same method of the base class, printing "l am an Undergraduate" and "l am a Postgraduate" respectively. Create an instance of each class.

**CODE:**

class Degree:  
 def getDegree(self):  
  
 print("I got a degree.")  
  
  
class Undergraduate(Degree):  
 def getDegree(self):  
  
 print("I am an Undergraduate.")  
  
  
class Postgraduate(Degree):  
 def getDegree(self):  
  
 print("I am a Postgraduate.")  
  
  
degree = Degree()  
under = Undergraduate()  
post = Postgraduate()  
degree.getDegree()  
under.getDegree()  
post.getDegree()

**OUTPUT:**

****

EXTRA QUESTIONS

4) Calculate the perimeter of squares and rectangles using method overloading.

**CODE:**

class shape\_ka\_area:  
 def area\_batao(self, \*args):  
  
 if len(args) == 2:  
 return 2 \* (args[0] + args[1])  
 if len(args) == 1:  
 return args[0] \* 4  
  
  
p = shape\_ka\_area()  
print("Area of rect (5, 3):", p.area\_batao(5, 3), "sq units")  
print("Area of sqr (4):", p.area\_batao(4), "sq units")

**OUTPUT:**

