INDIAN INSTITUTE OF TECHNOLOGY MADRAS
सिद्धिर्भवति कर्मजा

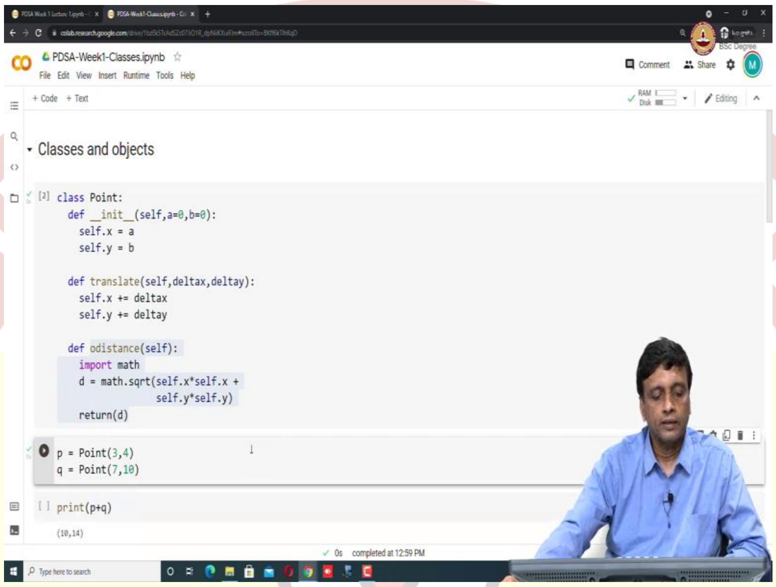
IIT Madras

ONLINE DEGREE

Programming, Data Structures and Algorithms using Python
Professor Madhavan Mukund
Implementation of Python Codes (Part 2)

So, as we discussed classes and objects, we looked at this example of this point. So, let us see how it works here.

(Refer Slide Time: 0:16)

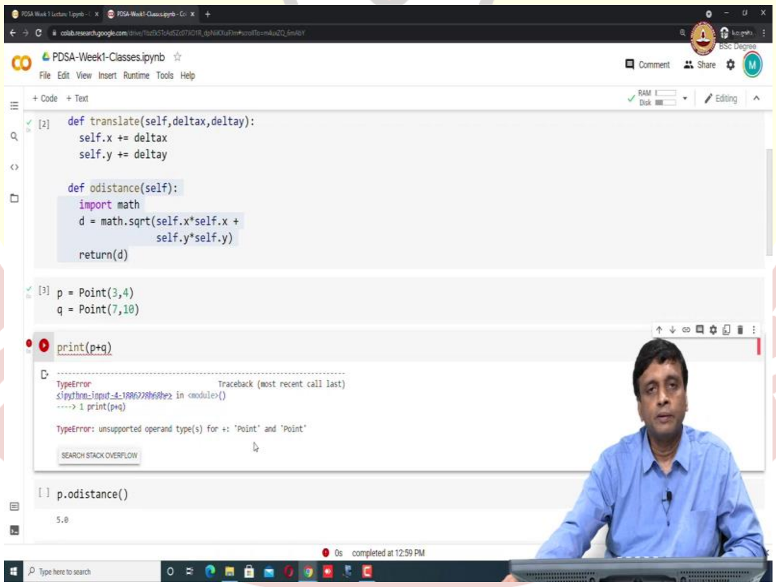


Classes and objects

```
[2] class Point:
      def __init__(self,a=0,b=0):
        self.x = a
        self.y = b

      def translate(self,deltax,deltay):
        self.x += deltax
        self.y += deltay

      def odistance(self):
        import math
        d = math.sqrt(self.x*self.x +
                      self.y*self.y)
        return(d)
```

```
p = Point(3,4)
q = Point(7,10)
```

```
[ ] print(p+q)

(10,14)
```

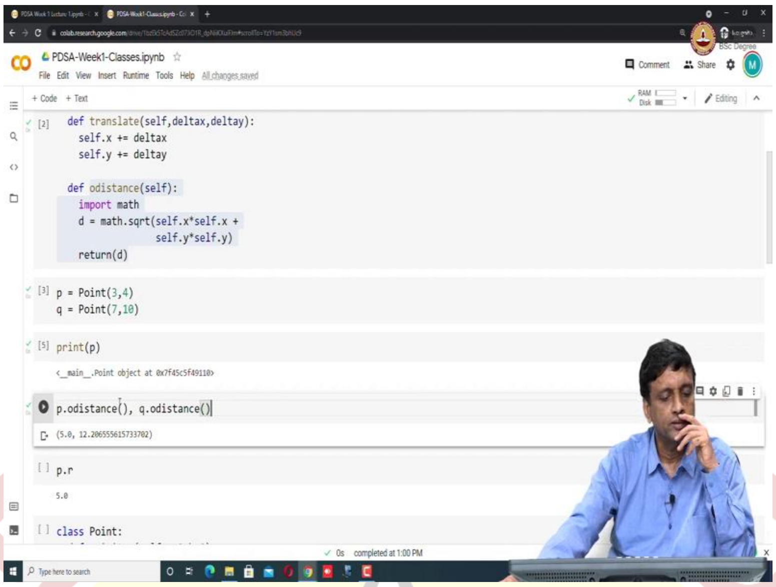


```
[2]   def translate(self,deltax,deltay):
        self.x += deltax
        self.y += deltay

      def odistance(self):
        import math
        d = math.sqrt(self.x*self.x +
                      self.y*self.y)
        return(d)
```

```
[3] p = Point(3,4)
    q = Point(7,10)
```

```
print(p+q)

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-4-1886228b68be> in <module>()
----> 1 print(p+q)

TypeError: unsupported operand type(s) for +: 'Point' and 'Point'
```

```
[ ] p.odistance()

5.0
```



```
[2]   def translate(self,deltax,deltay):
        self.x += deltax
        self.y += deltay

      def odistance(self):
        import math
        d = math.sqrt(self.x*self.x +
                      self.y*self.y)
        return(d)

[3] p = Point(3,4)
    q = Point(7,10)

[5] print(p)

    <__main__.Point object at 0x7f45c5f49110>

    p.odistance(), q.odistance()

    (5.0, 12.206555615733702)

[ ] p.r

    5.0

[ ] class Point:
```

So, here is the basic definition of point with the x y coordinates. So, we have a class point. And this has this initialization, where it takes the values a and b as the initial coordinates of the point. And then we have this function translate, which shifts x by delta x and y by delta y. And we have this function odistance, which computes the Euclidean distance, as is called by Pythagoras formula, x squared plus y squared, whole square root.

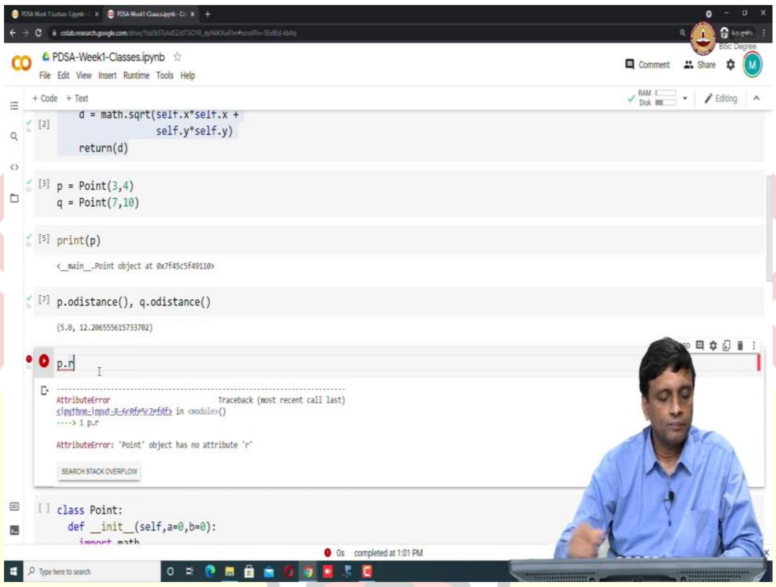
So, now I can create these points by using this. So, I, but of course, I must, nothing has run, so I must first run the code there. So, I run the code to create the point class. And now I run this and now it has created two points for me called p and q at 3, 4 and 7, 10. Now at this point, if I try to print out something like p plus q, then it is going to say that there is no remember we said that you have these ad functions and string functions.

So, even if I just try to print p, for instance, if I just try to print p, I will get some kind of useless output, which says that this is a point object. It does not tell me the coordinates or anything, it just tells me that this is a point object is, the same thing happens if you try to print for example, a list or a dictionary sometimes without, you know getting the values out properly.

So, for instance, if you try to print range of something, which is not a list, it will just say this is the range function? So, that is the kind of useless output. But on the other hand, I can compute the distance. So now, P is 3, 4. So, if you remember your Pythagoras theorem, then 3, 4, 5 is the right angled triangle. So, the distance of this point 3, 4 is 5 from origin.

So, if I compute the distance of p from the origin, it is 5. And if I compute the distance of Q from the origin, so notice that this kind of gives you this helpful hint as to what function to complete it, but let us not worry about that. So, in a Python notebook, I can do this kind of thing, I can implicitly call two things, and that will give me a pair. So, it says that P is that distance 5, q is a distance 12.2 because that is 7 squared plus 10 squared, which is 184 square root, which is something like 12.2, sorry, 149 square root, which is something like.

(Refer Slide Time: 2:43)



```
        d = math.sqrt(self.x*self.x +
                      self.y*self.y)
        return(d)
```

```
p = Point(3,4)
q = Point(7,10)
```

```
print(p)
```

```
<__main__.Point object at 0x7f45c5f49110>
```

```
p.odistance(), q.odistance()
```

```
(5.0, 12.206555615733702)
```

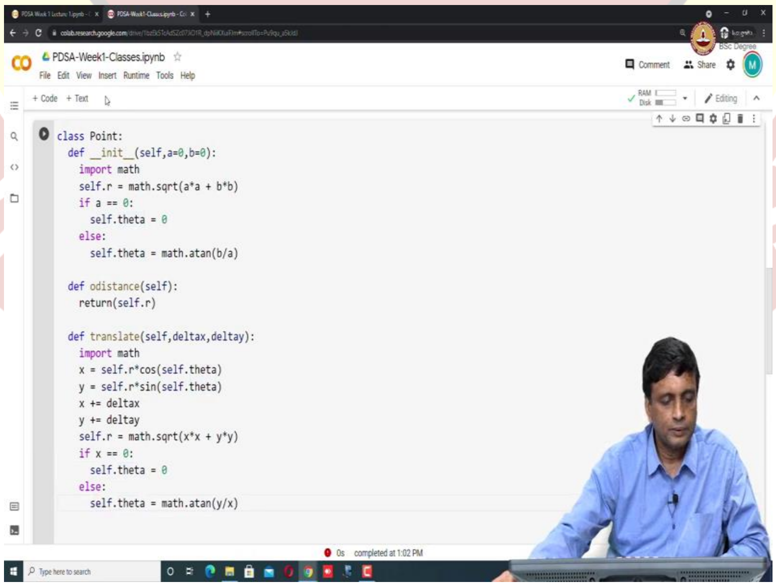
```
p.r
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-8-6c0fe5c7efdf> in <module>()
----> 1 p.r

AttributeError: 'Point' object has no attribute 'r'
```

```
class Point:
  def __init__(self,a=0,b=0):
    import math
```

Now, if I ask what is p dot r, then it will say that this is not defined, because there is no internal value called r.

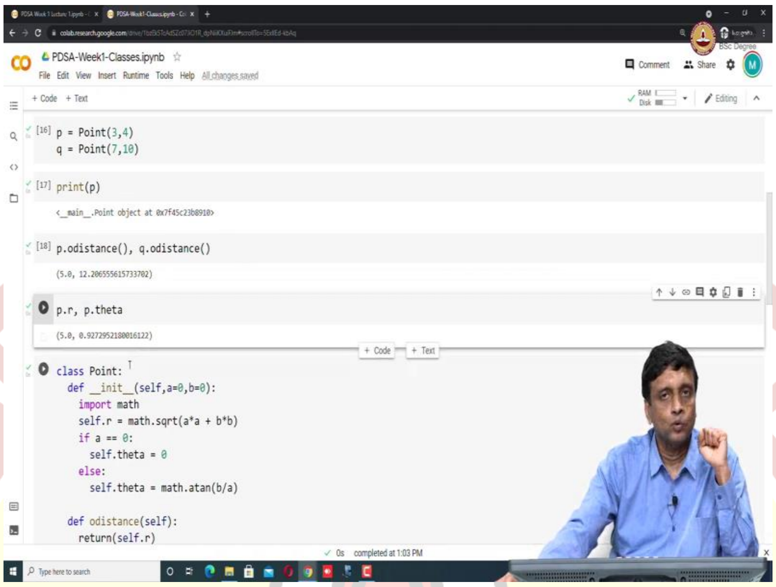
(Refer Slide Time: 2:53)



```
class Point:
  def __init__(self,a=0,b=0):
    import math
    self.r = math.sqrt(a*a + b*b)
    if a == 0:
      self.theta = 0
    else:
      self.theta = math.atan(b/a)

  def odistance(self):
    return(self.r)

  def translate(self,deltax,deltay):
    import math
    x = self.r*cos(self.theta)
    y = self.r*sin(self.theta)
    x += deltax
    y += deltay
    self.r = math.sqrt(x*x + y*y)
    if x == 0:
      self.theta = 0
    else:
      self.theta = math.atan(y/x)
```

Then the next thing we did was we defined this r theta form. So, now I have a new definition of point in which I use this r theta forms. So, remember that we convert x y to r by using the distance formula and we use this tan inverse to construct the angle and so on. So, now, if I take this definition and run it, remember in the Jupyter Notebook, the last thing that ran works, so this is now my new definition of point.

(Refer Slide Time: 3:20)



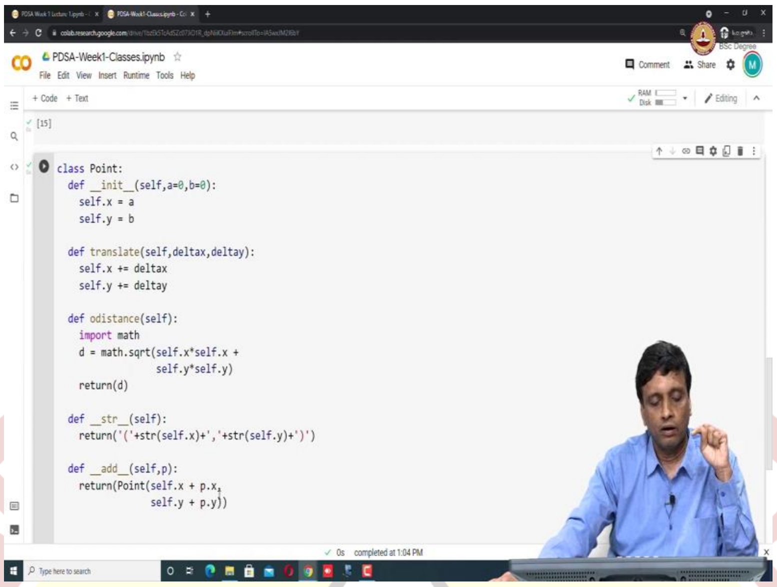
```
[16] p = Point(3,4)
     q = Point(7,10)

[17] print(p)

     <__main__.Point object at 0x7f45c23b8910>

[18] p.odistance(), q.odistance()

     (5.0, 12.206555615733702)

     p.r, p.theta

     (5.0, 0.9272952180016122)

     class Point:
       def __init__(self,a=0,b=0):
         import math
         self.r = math.sqrt(a*a + b*b)
         if a == 0:
           self.theta = 0
         else:
           self.theta = math.atan(b/a)

       def odistance(self):
         return(self.r)
```

So, now, if I take this and if I again execute this, I get two new points, p and q, but these are now defined in terms of r and theta. So, again, if I try to print p, I will still get some meaningless thing. If I try to compute the distance, okay I get the same distance, but this time is coming from r it is not coming from a calculation involving p.x and p.y.

And I can check that I am using the new format by asking now what is P.r and it tells me p.r is 5.2. So, similarly, I can ask what is p.theta? So, it tells me that, this mistake here, this should be self-supposed to be safe. This is what happens in Python if you forget the self in an object, then it does not and so it takes theta as an arbitrary value. So, everywhere this should be self. So, let me run this again.

So, now I have updated to self.r and self.theta, which I had forgotten the self part of it. So, now let me create these points again, and I can print the point. So, notice theta was not being used for the distance, that is why old distance worked. Because I was just looking up r and getting it, but now if I say what is p.r and p.theta, I get that p.r is 5 and p.theta is 0.972. But remember that this is not in degrees but in radians. So, just remember the unit. So that is the two different formats.

(Refer Slide Time: 5:04)



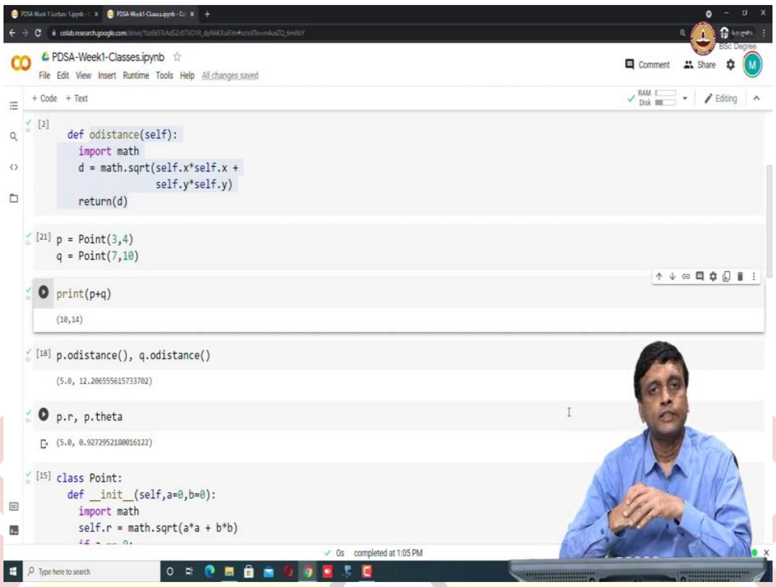
```
class Point:
  def __init__(self,a=0,b=0):
    self.x = a
    self.y = b

  def translate(self,deltax,deltay):
    self.x += deltax
    self.y += deltay

  def odistance(self):
    import math
    d = math.sqrt(self.x*self.x +
                  self.y*self.y)
    return(d)

  def __str__(self):
    return('('+str(self.x)+','+str(self.y)+')')

  def __add__(self,p):
    return(Point(self.x + p.x,
                 self.y + p.y))
```

Now, we also said that you could have these special functions, string and add to convert. So, let us compile these things. So notice that this thing which was obscured in the slide, so this is how we convert to a string, I take self dot x self dot y, convert each of them to a string, and then embedded inside this open bracket comma and close bracket.

So, I create a new string, it has an open bracket, the string version of self.x, comma, the string version of self .y, close bracket. So, this is my output string that I create. And this is what str does. And add if you remember, take self.x, a new point p.x adds them up and then creates a new point.

So, now let me replace this is my third definition of point is the, I have gone back to the x comma y definition but I have added this self.str and self.add.

(Refer Slide Time: 5:56)



```
[2]
    def odistance(self):
      import math
      d = math.sqrt(self.x*self.x +
                    self.y*self.y)
      return(d)

[21] p = Point(3,4)
     q = Point(7,10)

print(p+q)

(10,14)

[18] p.odistance(), q.odistance()

(5.0, 12.206555615733702)

p.r, p.theta

(5.0, 0.9272952180016122)

[15] class Point:
       def __init__(self,a=0,b=0):
         import math
         self.r = math.sqrt(a*a + b*b)
```

So, now let me go back up. So, this is again the advantage of Jupiter note I do not have to type anything again I can go back and reexecute it. So, now I reexecute this code. So, again, now it has created two points, but also a new definition. And as proof that is a new definition I can now say, tell me what is the value of p plus q?

Now, this has two things; one is p plus q has to be executed it has to give me a point, what is that point, it should have 3 plus 7 as the x coordinate 10; 4 plus 10, 14 as the y coordinate and it should then separately, because print should convert it to a string it should not give me this kind of strange output which says it is a point so and so hexadecimal, it should give me the values.

So, if I execute this, indeed, I get the output 10 comma 14. So, two things have happened, it has added and it is converted to a string. So, this is just to illustrate how this class and object definition work. And again, an illustration of what how in this notebook format, you can keep updating the code very naturally and keep rerunning the code without typing a lot of stuff back and forth. So, it is very convenient for this kind of incremental creation of code.