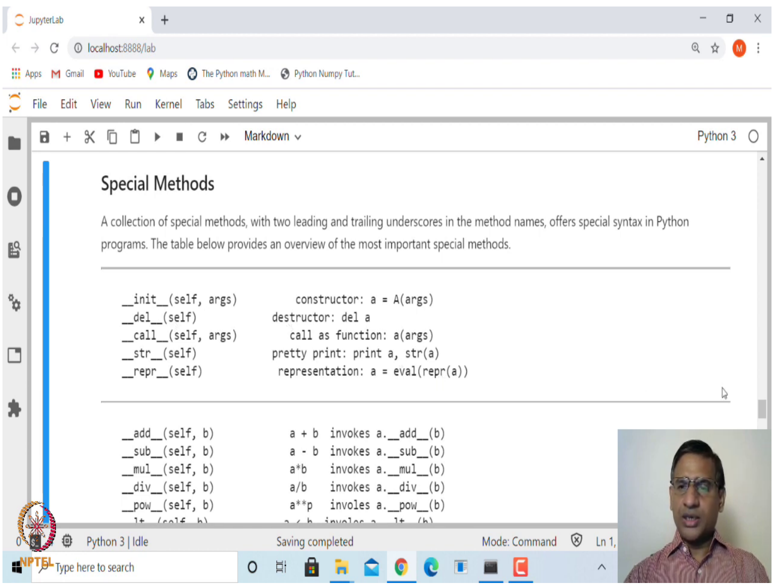
Computational Mathematics with SageMath

Prof. Ajit Kumar
Department of Mathematics
Institute of Chemical Technology, Mumbai

Lecture - 14
Classes in Python – Part 02

Welcome to the 13th lecture on Computational Mathematics with SageMath. In this lecture, we will continue exploring Classes in Python. In the last lecture we introduced classes, and also methods inside classes, including some of the special methods like init, and call. So, let us continue with the last lecture.

(Refer Slide Time: 00:41)



Special Methods

A collection of special methods, with two leading and trailing underscores in the method names, offers special syntax in Python programs. The table below provides an overview of the most important special methods.

```
__init__(self, args)         constructor: a = A(args)
__del__(self)             destructor: del a
__call__(self, args)         call as function: a(args)
__str__(self)             pretty print: print a, str(a)
__repr__(self)             representation: a = eval(repr(a))
```

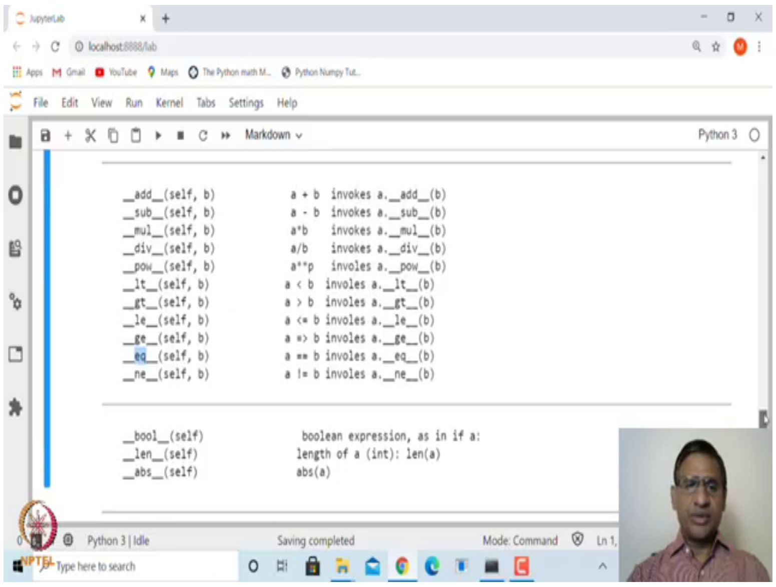
```
__add__(self, b)           a + b  invokes a.__add__(b)
__sub__(self, b)           a - b  invokes a.__sub__(b)
__mul__(self, b)           a*b    invokes a.__mul__(b)
__div__(self, b)           a/b    invokes a.__div__(b)
__pow__(self, b)           a**p   involes a.__pow__(b)
```

Now, let us look at some other special methods.

So, there are several special methods. So, we have already made use of init, we also saw del, and just now we saw call, str, and repr. These are the special methods which is used when you call print statement.

So, print, we saw in case of complex number z, we say, when we say print z, it printed that value, ok? Similarly, the repr also has similar behavior, but there is a small difference between str and repr methods, ok?

(Refer Slide Time: 01:37)



```
__add__(self, b)          a + b  invokes a.__add__(b)
__sub__(self, b)          a - b  invokes a.__sub__(b)
__mul__(self, b)          a*b    invokes a.__mul__(b)
__div__(self, b)          a/b    invokes a.__div__(b)
__pow__(self, b)          a**p   involes a.__pow__(b)
__lt__(self, b)          a < b  involes a.__lt__(b)
__gt__(self, b)          a > b  involes a.__gt__(b)
__le__(self, b)          a <= b involes a.__le__(b)
__ge__(self, b)          a => b involes a.__ge__(b)
__eq__(self, b)          a == b involes a.__eq__(b)
__ne__(self, b)          a != b involes a.__ne__(b)
```

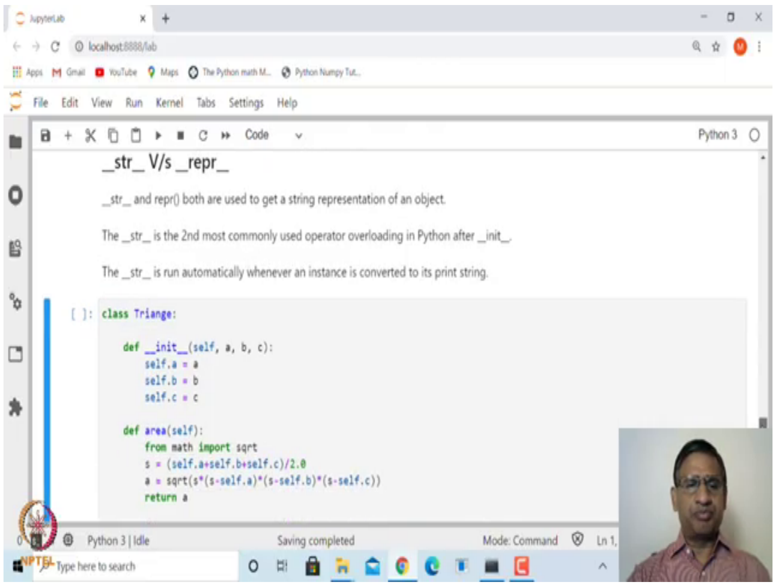
```
__bool__(self)            boolean expression, as in if a:
__len__(self)            length of a (int): len(a)
__abs__(self)            abs(a)
```

Next, let us look at, there are these special methods for adding. For example, if you say a plus b, it should add the two objects a and b, we saw that z and w in case of complex case. So, what it does? It, when you issue this command or syntax, it invokes a method called, class method called add, and it will say a dot add, and b.

Similarly, sub is for subtraction mul, mul for multiplication, div for division, pow for power, lt for checking less than, and gt for greater than, le for less than equal to, ge for greater than or equal to, eq is for equation equality actually, and ne for not equal to, ok? So, these are default methods, and there are some more, there are many more actually.

And if we look, see, there is one called Boolean, bool, this is boolean expression, whether something is in some object right, and length is also there len, and then you have absolute value, abs. So, these are some default methods. There are many more, but these are very commonly used special methods.

(Refer Slide Time: 03:07)



str V/s _repr_

str and repr() both are used to get a string representation of an object.

The _str_ is the 2nd most commonly used operator overloading in Python after _init_.

The _str_ is run automatically whenever an instance is converted to its print string.

```
[ ]: class Triange:

         def __init__(self, a, b, c):
             self.a = a
             self.b = b
             self.c = c

         def area(self):
             from math import sqrt
             s = (self.a+self.b+self.c)/2.0
             a = sqrt(s*(s-self.a)*(s-self.b)*(s-self.c))
             return a
```

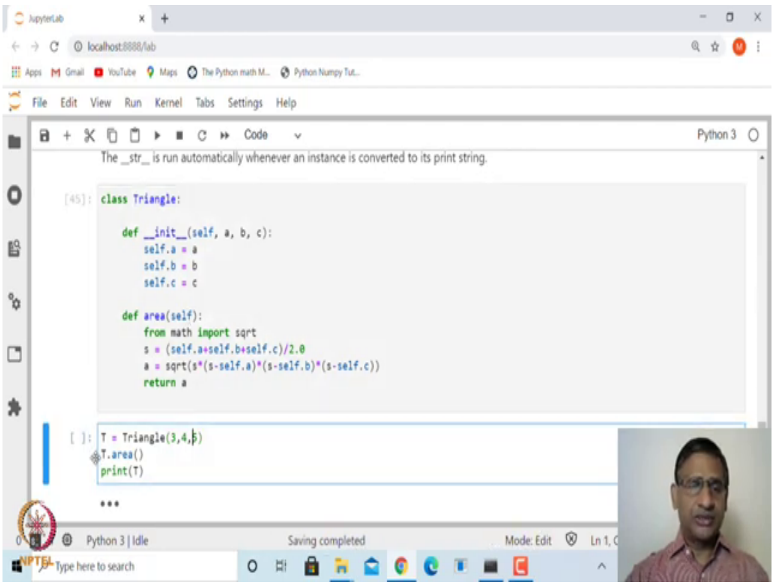
Let us create, for example, some class or type which makes use of these, these type of methods, ok? Before that, let us just briefly look at what is these str, and repr. So, for example, if I create a class called Triangle, with initial values as a, b, c, that is the side length right, and then suppose we want to print the area of this triangle. So, using Heron's formula.

So, we can create a method def area semicolon, from math import sqrt, find out this semi perimeter, which is a plus b plus c by 2, but here a,b,c are self dot a, self dot b, self dot c, and then create the, the area, which is square root of s into s minus a into s minus b into s minus c, and return s.

So, you can see here this method is exactly the same as you create any function outside the class, this we have seen it earlier, right? So, when it comes to creating method, there is no difference between creating method, and creating functions.

So, actually the class methods are nothing, but functions, right? User-defined functions.

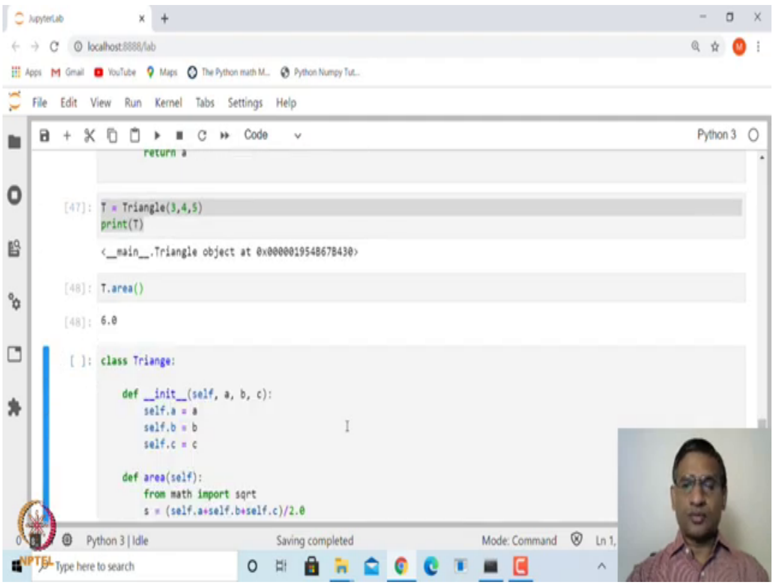
(Refer Slide Time: 04:46)


The __str__ is run automatically whenever an instance is converted to its print string.

```
[45]: class Triangle:

          def __init__(self, a, b, c):
              self.a = a
              self.b = b
              self.c = c

          def area(self):
              from math import sqrt
              s = (self.a+self.b+self.c)/2.0
              a = sqrt(s*(s-self.a)*(s-self.b)*(s-self.c))
              return a

[ ]: T = Triangle(3,4,5)
     T.area()
     print(T)
```

So, let us run this ok, and let us call this Triangle. So, Triangles, I am calling this as T, and the Triangle, Triangle spelling wrong ok, Triangle, it does not matter whatever name you want you can give, but let us, ok? So, this is Triangle, and then I am calling this, right?

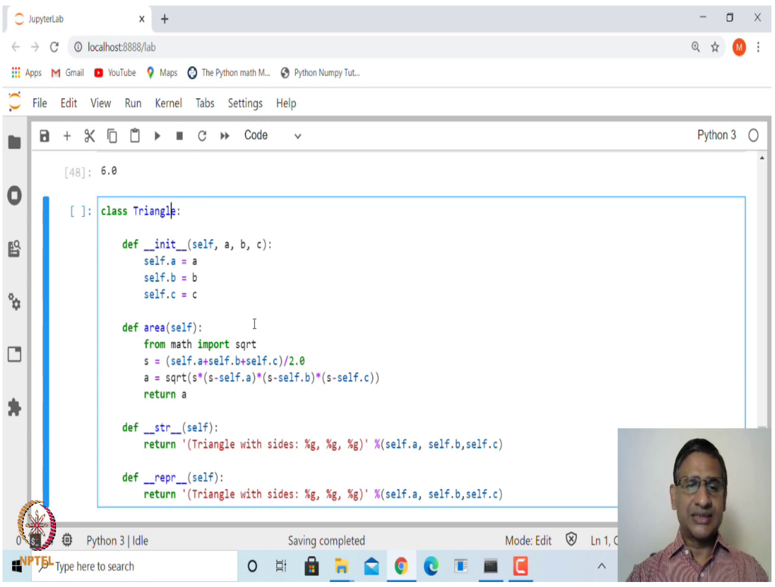
(Refer Slide Time: 05:18)


```
              return a

[47]: T = Triangle(3,4,5)
      print(T)

      <__main__.Triangle object at 0x0000019548678430>

[48]: T.area()

[48]: 6.0

[ ]: class Triange:

         def __init__(self, a, b, c):
             self.a = a
             self.b = b
             self.c = c

         def area(self):
             from math import sqrt
             s = (self.a+self.b+self.c)/2.0
```

Now, when I see, if I simply ask it to print T, it will give me, it will not print what I wanted actually, but what it has printed? It has printed the address, the location in which this T is stored in the memory, ok? So, that is what it is. So, if I want to print this as something, let us say, it is a triangle with sides this, this, this, this is not good. You, you this is not enough, ok?

Similarly, let us, let us look at, so, once we have passed this, created this Triangle, then we can get its area T dot area, and the area is 6.0, right, ok? So, we cannot use print on T because it does not understand, and that is where str comes into picture.

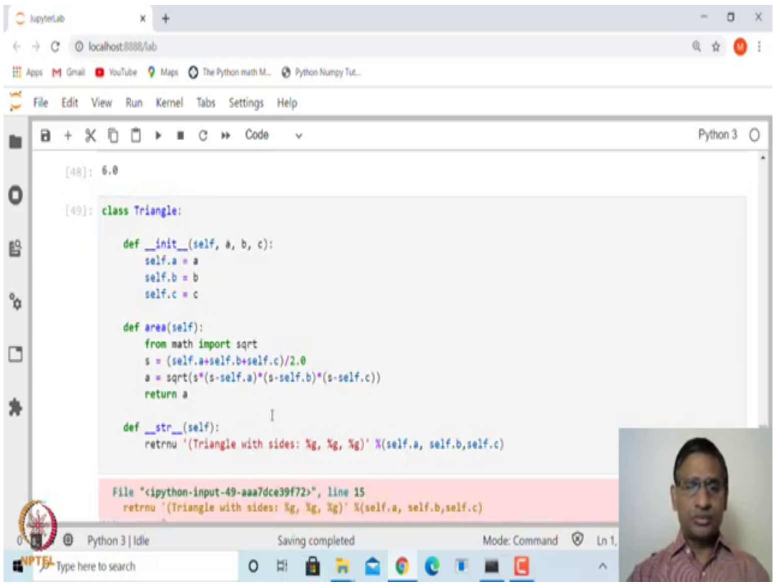
(Refer Slide Time: 06:19)


```
[48]: 6.0

[ ]: class Triangle:

         def __init__(self, a, b, c):
             self.a = a
             self.b = b
             self.c = c

         def area(self):
             from math import sqrt
             s = (self.a+self.b+self.c)/2.0
             a = sqrt(s*(s-self.a)*(s-self.b)*(s-self.c))
             return a

         def __str__(self):
             return '(Triangle with sides: %g, %g, %g)' %(self.a, self.b,self.c)

         def __repr__(self):
             return '(Triangle with sides: %g, %g, %g)' %(self.a, self.b,self.c)
```

So, you can create a, you can redefine this triangle class, and add state method called str, a method called str. Now, for the time being, let me just have str.

And what is that it returns? So, str should always return, actually a string. So, we are saying that it is a triangle with sides... So, this will get replaced by a, b, c, percentage g, percentage g, percentage g, these are the replacement operator for a,b,c, right?

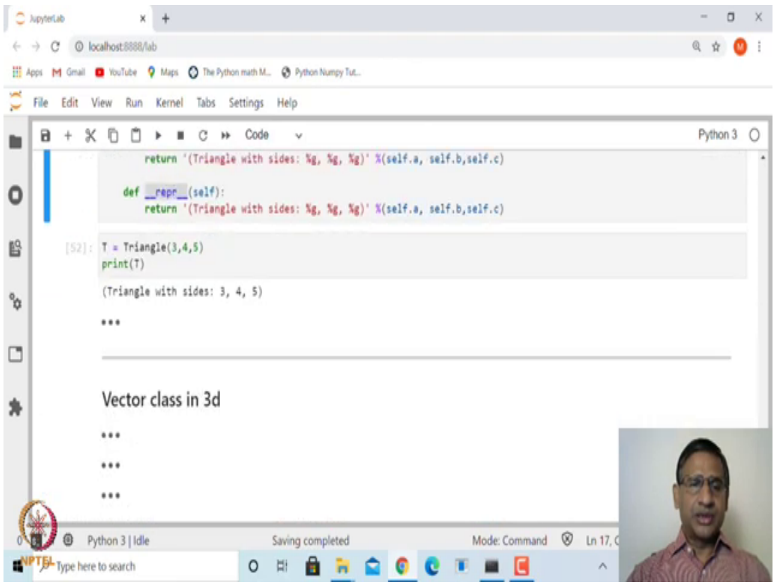
(Refer Slide Time: 06:54)



```
[48]: 6.0

[49]: class Triangle:

          def __init__(self, a, b, c):
              self.a = a
              self.b = b
              self.c = c

          def area(self):
              from math import sqrt
              s = (self.a+self.b+self.c)/2.0
              a = sqrt(s*(s-self.a)*(s-self.b)*(s-self.c))
              return a

          def __str__(self):
              retrnu '(Triangle with sides: %g, %g, %g)' %(self.a, self.b,self.c)

File "<ipython-input-49-aaa7dce39f72>", line 15
  retrnu '(Triangle with sides: %g, %g, %g)' %(self.a, self.b,self.c)
```

So, now if you return the spelling mistake return, ok?

(Refer Slide Time: 07:07)



```
              return '(Triangle with sides: %g, %g, %g)' %(self.a, self.b,self.c)

          def __repr__(self):
              return '(Triangle with sides: %g, %g, %g)' %(self.a, self.b,self.c)

[52]: T = Triangle(3,4,5)
      print(T)

      (Triangle with sides: 3, 4, 5)
```

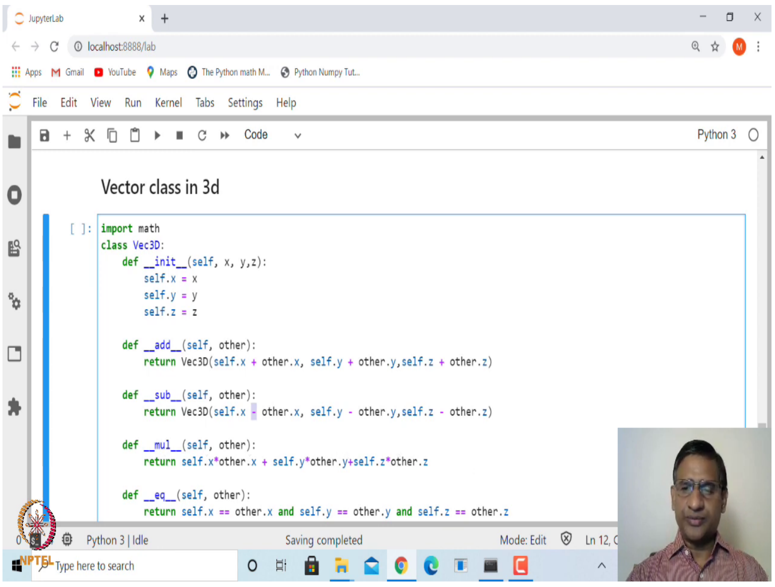
Vector class in 3d

Now, if you, if you create a triangle T with sides 3, 4, 5, and then ask it to print, it is a spelling again, this triangle, then if you ask it to print, it will print that triangle with sides this, this is what we wanted to return. So, this is, if you want to print something, some object which you have created, then you must have inside this str, ok?

You could have also used, you could have also used repr, right? So, I have just added in this. So, whenever actually when you call print, and in case it is not the first, it will look for str; if it is not there, it will look for repr method, and then it will give, but repr can also be used for evaluating the value. So, it can return actually anything whereas, str will return only the, the string variable, ok?

That is the difference, but the role of these two are quite similar, ok? Of course, these are also used by the developers. So, developer will, will know more about this classes whenever they see representation method ok?

(Refer Slide Time: 08:51)



Vector class in 3d

```
[ ]: import math
     class Vec3D:
         def __init__(self, x, y,z):
             self.x = x
             self.y = y
             self.z = z

         def __add__(self, other):
             return Vec3D(self.x + other.x, self.y + other.y,self.z + other.z)

         def __sub__(self, other):
             return Vec3D(self.x - other.x, self.y - other.y,self.z - other.z)

         def __mul__(self, other):
             return self.x*other.x + self.y*other.y+self.z*other.z

         def __eq__(self, other):
             return self.x == other.x and self.y == other.y and self.z == other.z
```

So, let us make use of these concepts, and these special methods, in order to create, and a class or a type for 3-dimensional vectors, right?

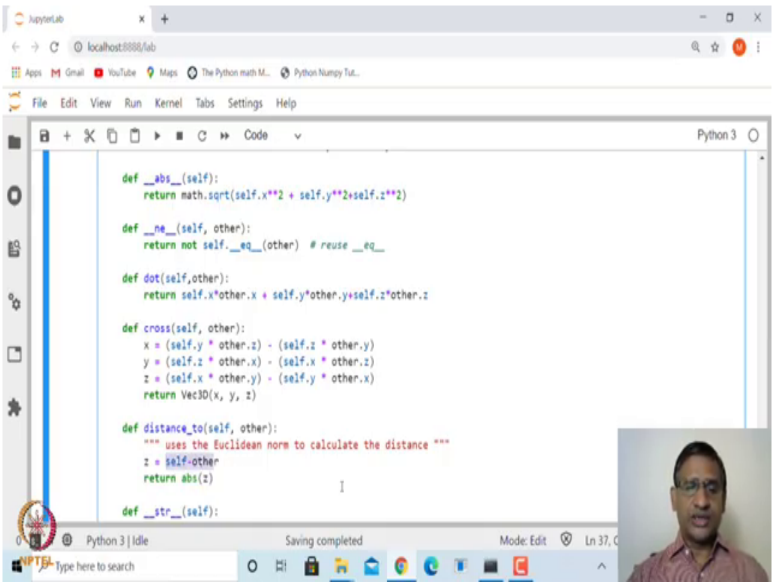
So, vectors in the space having 3 coordinates. So, how do we create this? So, let us look at this. So, we are giving the name Vec3D, and since the calculation, vector-calculation will require, maybe square root, and things like that, especially if we want to find the length of a vector, we will import this math module, and so this is the class name now. First you initialize this vector 3 D with the, each coordinate x, y, z.

So, self dot x equal to x, self dot y equal to y, self dot z equal to z, and then we want to add two vectors. So, how do I create? We will make use of the special method called add, and then the argument is self followed by the other second, right? So, once you have created a vector let us say v, and you want to add to w, then we have to say v dot v plus w, and w is other will be replaced by w, ok?

So, what should it return? It should return Vec3D, that is the, the, the class, the type, and self dot x plus other dot x, self dot y dot plus other dot y, self dot z plus other dot z right, x-coordinate is added x-coordinate of x is added to x-coordinate of others, and so on, y-coordinate of this vector is added to y-coordinate of the other, and so on. Similarly, subtraction, so instead of plus you have to say minus, here multiplication.

So, multiplication here, mult, we are replacing it by the dot product actually. So, multiply the first coordinates, second coordinate, and third coordinate, add all of them, add all of them, that is the multiplication.

(Refer Slide Time: 10:51)



```
    def __abs__(self):
        return math.sqrt(self.x**2 + self.y**2+self.z**2)

    def __ne__(self, other):
        return not self.__eq__(other)  # reuse __eq__

    def dot(self,other):
        return self.x*other.x + self.y*other.y+self.z*other.z

    def cross(self, other):
        x = (self.y * other.z) - (self.z * other.y)
        y = (self.z * other.x) - (self.x * other.z)
        z = (self.x * other.y) - (self.y * other.x)
        return Vec3D(x, y, z)

    def distance_to(self, other):
        """ uses the Euclidean norm to calculate the distance """
        z = self-other
        return abs(z)

    def __str__(self):
```

Similarly, if you want to check whether the two vectors are equal or not, we will use eq, and then in this case double equal to x-coordinate is equal to x-coordinate of other, x-coordinate of self is equal to other coordinate, y-coordinate of, sorry, y-coordinate of self should be equal to y-coordinate of other, and so on, ok?

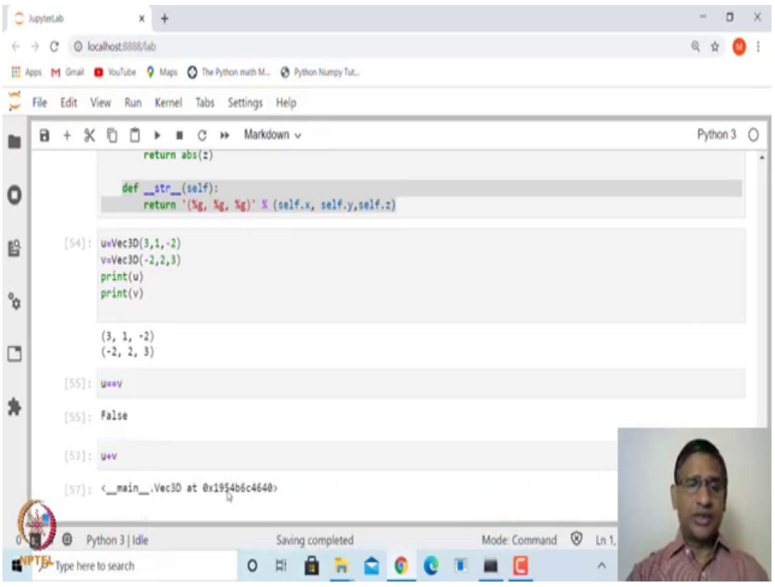
Similarly, absolute value is for the length. So, that is why math, sqrt of self dot x square plus self dot y square plus self dot z square, then you can also check if two vectors are equal or not. So, ne, not equal to self dot equal to, and other. So this here, this is just going to return true or false. So, in case it is true, it will return true, otherwise false. So, that is why it is self dot eq double underscore eq, and other.

So, it will check whether the self, that is one vector, let us say v, is equal to other, that is w, similarly dot product. So, dot product is very similar to mult right, but this is our own user-defined function dot, and. Similarly, another method is cross-product.

So, for cross-product what we are doing? We are defining x, y, z, variables, which is self dot. So, we, each coordinate we are defining. So, I am sure all of you know how to define cross-product of two vectors in the space. So, we are just writing each coordinate, and then it is returning as a vector x, y, z ok, and also distance between two vectors. So, self, and others.

So, this, this is z is equal to self minus other, and then return the absolute value of that, right? The distance between two vectors is nothing, but absolute value of, or the length of the difference between the two vectors ok?

(Refer Slide Time: 12:55)



```
        return abs(z)

    def __str__(self):
        return '(%g, %g, %g)' % (self.x, self.y,self.z)
```

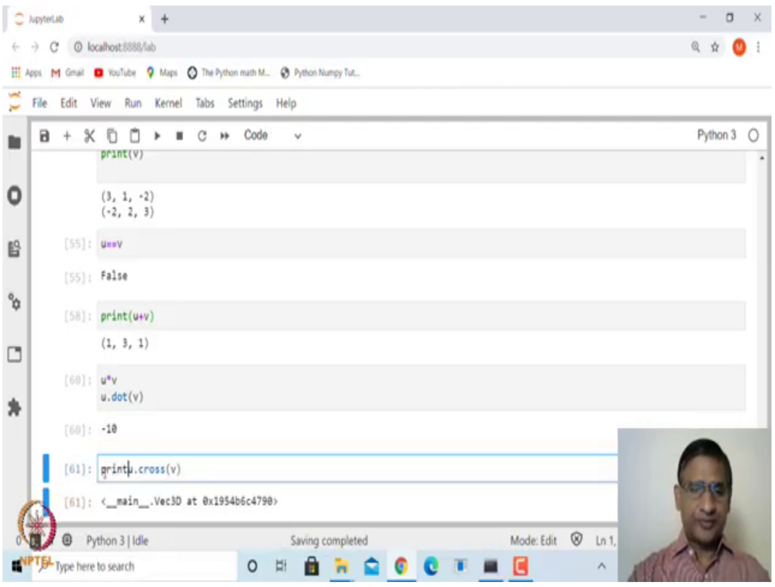
```
[54]: u=Vec3D(3,1,-2)
      v=Vec3D(-2,2,3)
      print(u)
      print(v)

      (3, 1, -2)
      (-2, 2, 3)

[55]: u==v

[55]: False

[57]: u+v

[57]: <__main__.Vec3D at 0x1954b6c4640>
```

And then you want to print the, the vector, whenever you are defining the vector, you also want to print.

So, we are invoking this method str, and it returns three coordinates x y z, ok? So, let us run this, and let us create a vector, let us create two vectors u, and v. So, u is 3, 1, minus 2, v is Vec 3D minus 2, 2, 3, and if I ask it to print u and v, it will, when I ask it to print u and v, it will invoke this last method, which is str, and it is printing this x-coordinate, y-coordinate, z-coordinate as a tuple.

If you want, you can add something inside the, this is a vector in 3-dimension with coordinate this, this, this all these things can be added, ok? Next, you can, you can check whether u is equal to v. In this case, answer is false; you can add two vectors u plus v, sorry, small v, then it will, but it is not printing.

(Refer Slide Time: 14:10)



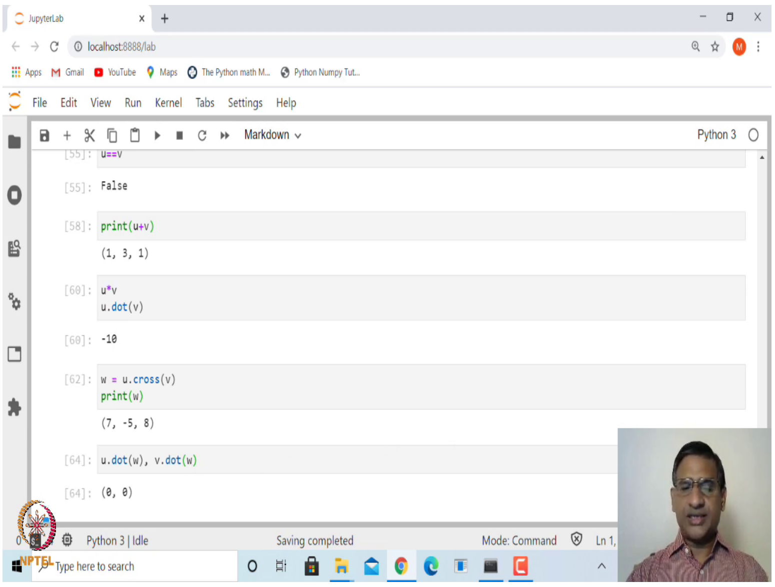
```
print(v)

(3, 1, -2)
(-2, 2, 3)

[55]: u==v

[55]: False

[58]: print(u+v)

(1, 3, 1)

[60]: u*v
      u.dot(v)

[60]: -10

[61]: printu.cross(v)

[61]: <__main__.Vec3D at 0x1954b6c4790>
```

So, I will have to say, print u plus v, we print u plus v.

Similarly, you can say, u star v, then it is giving you the dot product. Similarly, I can say u dot dot v, that is the again dot product right? So, the moment, if you say u star v, it will create, it will invoke this mult method, and so on, right? So, similarly, you can, you can do other operations.

So, for example, I can say find the cross-product. So, if I say u dot cross v, and since it is a vector, I have to ask you to print. So, I will say print, or let me say this is equal to w.

(Refer Slide Time: 15:10)

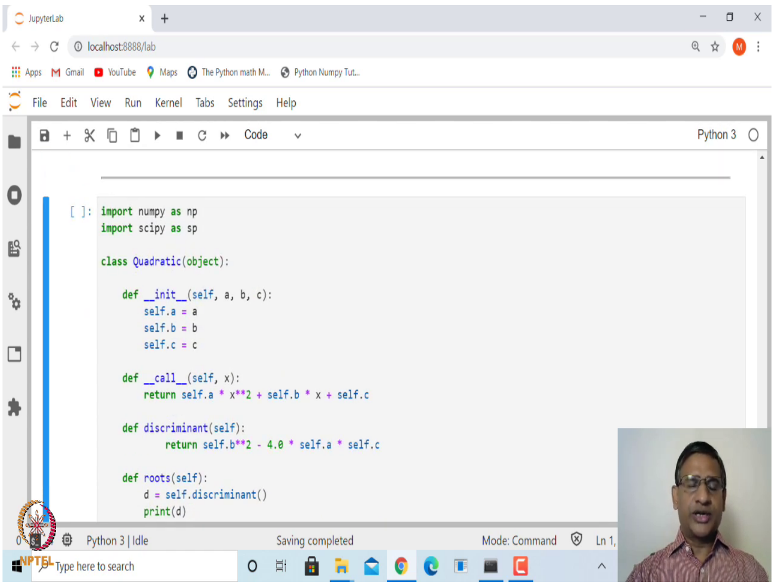


```
[55]: u==v

[55]: False

[58]: print(u+v)

      (1, 3, 1)

[60]: u*v
      u.dot(v)

[60]: -10

[62]: w = u.cross(v)
      print(w)

      (7, -5, 8)

[64]: u.dot(w), v.dot(w)

[64]: (0, 0)
```

And then say print w, it will print what is the cross-product, you can check since we, I am sure you know that when you create the cross-product or u and v, let us say that is w, the w is perpendicular to u and v.

So, if I asked for, if I asked for what is u dot w, this would be 0. Similarly, similarly, v dot w should also be 0, ok? So, both are 0 right, ok?

(Refer Slide Time: 15:51)

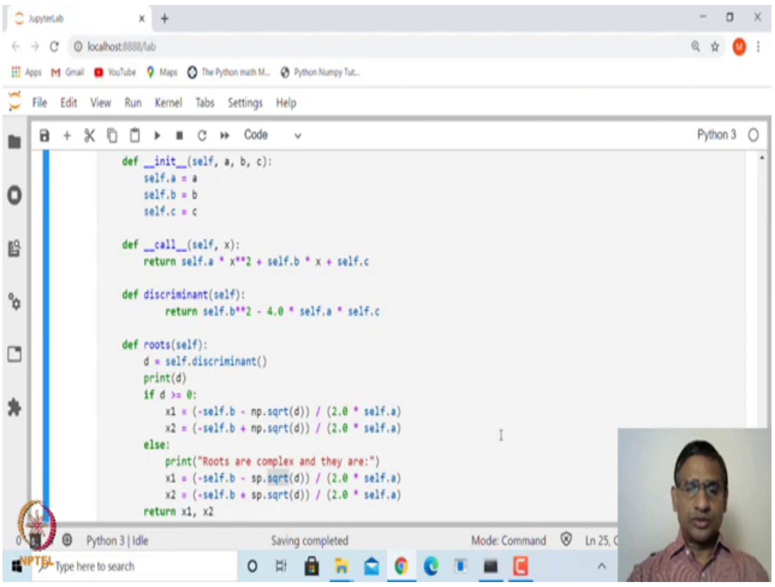


```
[ ]: import numpy as np
     import scipy as sp

     class Quadratic(object):

         def __init__(self, a, b, c):
             self.a = a
             self.b = b
             self.c = c

         def __call__(self, x):
             return self.a * x**2 + self.b * x + self.c

         def discriminant(self):
                 return self.b**2 - 4.0 * self.a * self.c

         def roots(self):
             d = self.discriminant()
             print(d)
```

Next, let us create another class called Quadratic, right, to solve a quadratic equation, and in this case, we also will print what is the discriminant, and what are the roots. So, these are the, we are creating one initialization, and then call function to print.

Then what is the value of the, the quadratic? What is the quadratic, how it looks like, a x square plus bx plus c that is the call function, and it gives you the discriminant method which will give you the discriminant, and then roots.

(Refer Slide Time: 16:29)

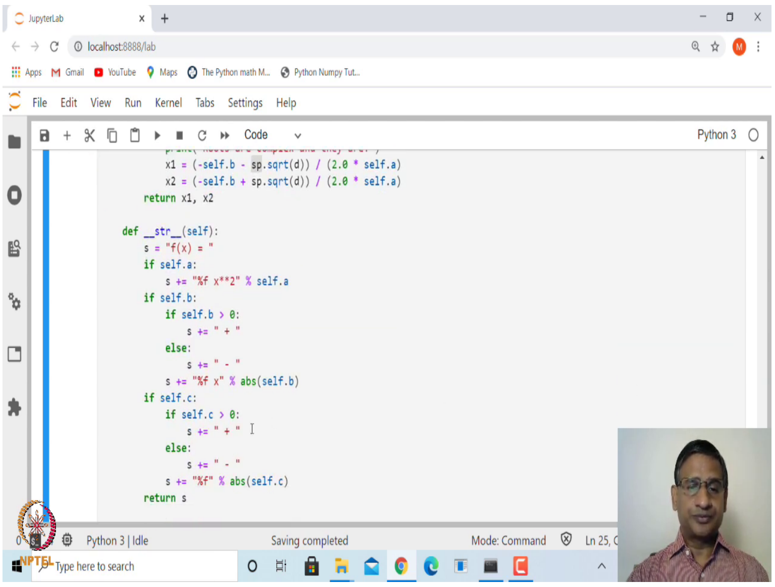


```
def __init__(self, a, b, c):
    self.a = a
    self.b = b
    self.c = c

def __call__(self, x):
    return self.a * x**2 + self.b * x + self.c

def discriminant(self):
        return self.b**2 - 4.0 * self.a * self.c

def roots(self):
    d = self.discriminant()
    print(d)
    if d >= 0:
        x1 = (-self.b - np.sqrt(d)) / (2.0 * self.a)
        x2 = (-self.b + np.sqrt(d)) / (2.0 * self.a)
    else:
        print("Roots are complex and they are:")
        x1 = (-self.b - sp.sqrt(d)) / (2.0 * self.a)
        x2 = (-self.b + sp.sqrt(d)) / (2.0 * self.a)
    return x1, x2
```

Again, it is exactly same as what we have done earlier. So, in case of real roots and complex root, it is just giving in case of complex root it is printing the roots are complex, and printing the value.

And we are making use of sqrt from NumPy library. So, therefore, it will, it will give you both real, and complex root, ok? So, here is actually this, in case of square root, we are using square root from SciPy library, in case of complex root, because SciPy sqrt has both real and imaginary part, ok?

(Refer Slide Time: 17:19)

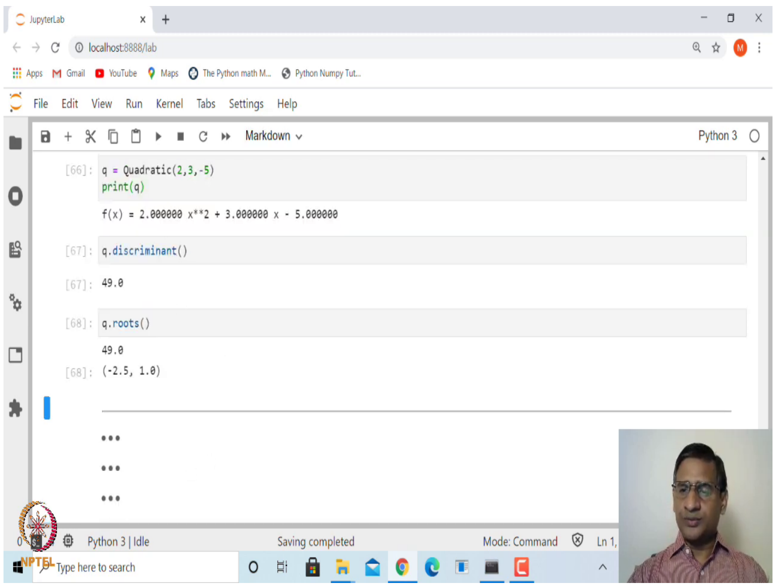


```
            x1 = (-self.b - sp.sqrt(d)) / (2.0 * self.a)
            x2 = (-self.b + sp.sqrt(d)) / (2.0 * self.a)
        return x1, x2

    def __str__(self):
        s = "f(x) = "
        if self.a:
            s += "%f x**2" % self.a
        if self.b:
            if self.b > 0:
                s += " + "
            else:
                s += " - "
            s += "%f x" % abs(self.b)
        if self.c:
            if self.c > 0:
                s += " + "
            else:
                s += " - "
            s += "%f" % abs(self.c)
        return s
```

And then at the end, you print. So, we have, though we have, we can call the, what is the value of the function. We can also create a printing string for printing, right? So, this will print, actually f(x), you can see here f(x) is equal to a x square plus b x plus c, and it will replace value of a, b, c by that value, that is what it is done.

So, it is saying that print f(x) equal to, so that is the initialization. In case a is present, it will add a into x square, and in case b is present, then it will check whether b is positive or negative, right? If it is positive, then it will put plus in between, if it is negative, it will put minus in between. Similarly for c; if c is present, then we will check whether c is positive or c is negative; if c is positive, after b x it will put plus, if it is negative, it, after b x it will put minus, and ultimately it will return s, right?

(Refer Slide Time: 18:33)



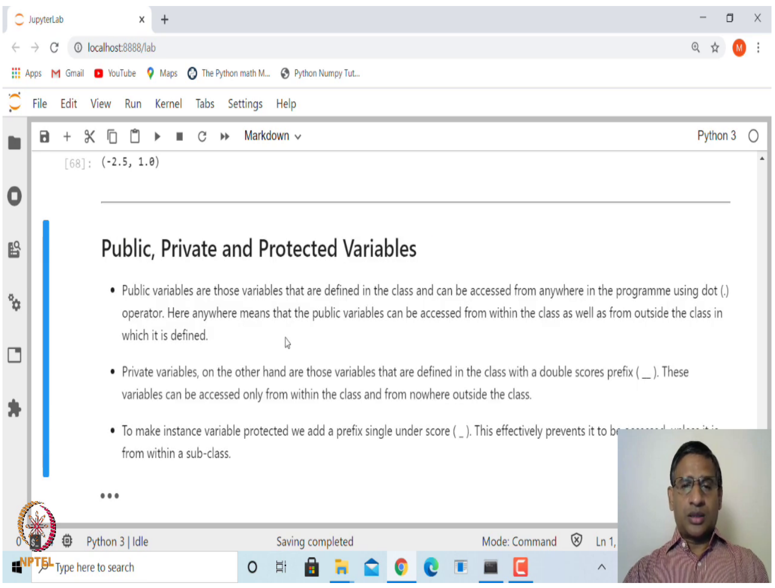
```
[66]: q = Quadratic(2,3,-5)
      print(q)

      f(x) = 2.000000 x**2 + 3.000000 x - 5.000000

[67]: q.discriminant()

[67]: 49.0

[68]: q.roots()

      49.0
[68]: (-2.5, 1.0)
```

So, let us run this, and let us call this. So, q is equal to quadratic 2, 3, minus 5, and if I say print q, and that is what you can see here; it is printing 2 x square plus 3 x minus 5, that was the role of this str, and we can ask for what is discriminant of this, we can ask for what are the roots. So, all these things are available.

So, see, you can see here, the advantage of creating class is that you can have different methods which will actually do tasks at a time.

So, if you have a big, big task, you can split this into small, small task, that is what exactly I meant by saying classes are blueprint ok, and so when you create, let us say, if you want to construct your house; first, you create a blueprint, and inside that, you keep on adding the bedroom will be, will have these features, bathroom will have these features, kitchen will have these features, and so on. So, that is exactly is the role of classes, right?

(Refer Slide Time: 19:39)



```
[68]: (-2.5, 1.0)
```

Public, Private and Protected Variables

- Public variables are those variables that are defined in the class and can be accessed from anywhere in the programme using dot (.) operator. Here anywhere means that the public variables can be accessed from within the class as well as from outside the class in which it is defined.
- Private variables, on the other hand are those variables that are defined in the class with a double scores prefix (__). These variables can be accessed only from within the class and from nowhere outside the class.
- To make instance variable protected we add a prefix single under score (_). This effectively prevents it to be
from within a sub-class.

We have created several attributes or variables, and which we accessed once we have created there the objects, but there are variables which, those variables which we have created, these are called public variables, but there are variable which are also private variables, and protected variables, ok?

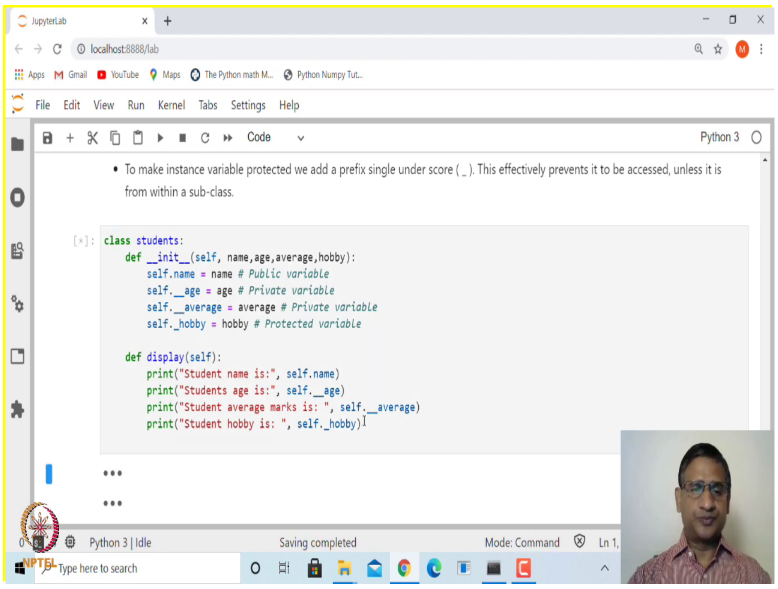
So, this, this public variables can be accessed from anywhere, ok? So, public variables are those variables that are defined in, in the class, and can be accessed from anywhere in the program using dot, that is what we have dot operator, that is what we have seen, and here anywhere means that, that it can be accessed from within the class as well as from the outside the class, that is what exactly we saw, right?

Whereas private variables are those variables that are defined in the class with double underscore, double underscore, we have to prefix it by double underscore, and these variables can be accessed only from within the class, and not from outside, ok?

Similarly, if you, there is another variable called protected variables, that is defined by a giving single prefix, that by single underscore, and actually, this effectively prevents it to be accessed, unless it is from within the the subclass, ok?

So, it again, it is, cannot be accessed; however, these variables can also be accessed if one wants. So, in, in some sense it is not that you cannot access at all, only thing is that good programmer will, will not try to change these variables, protected variables, and private variables, whenever they see this kind of variables, ok, because they would know that this is a private, and protected variables, so that is the difference, ok?

(Refer Slide Time: 21:45)



- To make instance variable protected we add a prefix single under score (_). This effectively prevents it to be accessed, unless it is from within a sub-class.

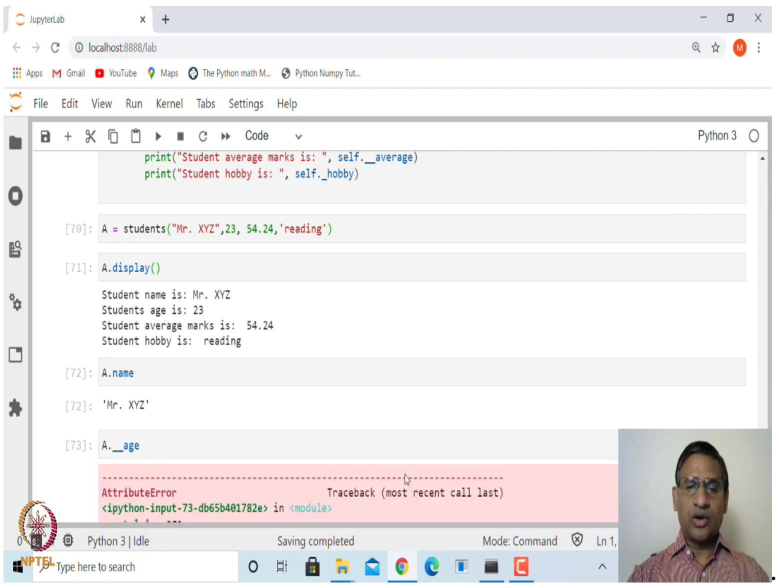
```
[*]: class students:
         def __init__(self, name,age,average,hobby):
             self.name = name # Public variable
             self.__age = age # Private variable
             self.__average = average # Private variable
             self._hobby = hobby # Protected variable

         def display(self):
             print("Student name is:", self.name)
             print("Students age is:", self.__age)
             print("Student average marks is: ", self.__average)
             print("Student hobby is: ", self._hobby)
```

Now, let us see how we can create these variables. So, let us create again this student class, initialize, has name, age, average, and hobby. So, first, name is actually a public variable. This we created as it is, which we have seen earlier, whereas, let us create a private variable called age.

So, how do we create? self dot two underscores, and age equal to age, and similarly another private variable as average, and let us create a hobby as a protected variable, and with this single underscore, right?

And then let us ask it to display. When you ask it to display, it should display this, all these variables; public variable, private variable, and also protected variable. So, within these, this class, this method, it can all, print all these things, ok? So, of course, it can print public variable, and it can also print the protected variable as well as private variable, but outside this class let us see whether it can do.

(Refer Slide Time: 22:51)

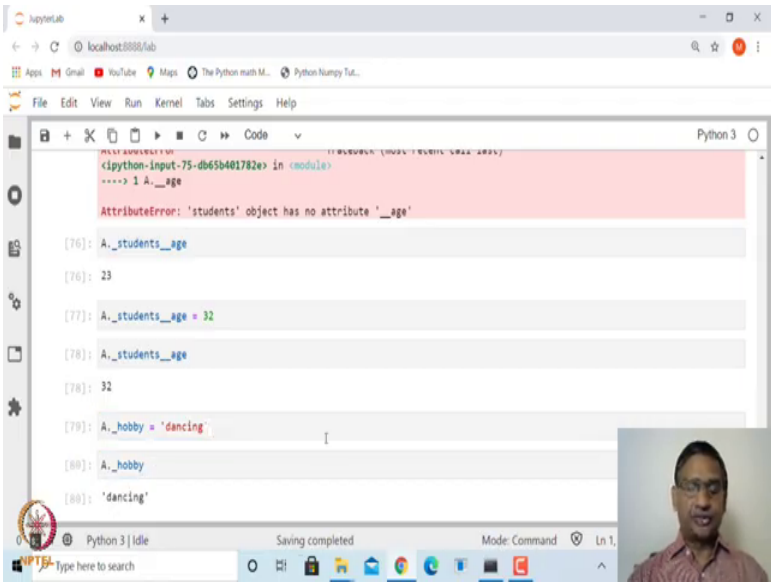


```
        print("Student average marks is: ", self.__average)
        print("Student hobby is: ", self._hobby)

[70]: A = students("Mr. XYZ",23, 54.24,'reading')

[71]: A.display()

      Student name is: Mr. XYZ
      Students age is: 23
      Student average marks is:  54.24
      Student hobby is:  reading

[72]: A.name

[72]: 'Mr. XYZ'

[73]: A.__age

      ---------------------------------------------------------------------------
      AttributeError                            Traceback (most recent call last)
      <ipython-input-73-db65b401782e> in <module>
```

So, let us run this, and now let us create a student "Mister XYZ", and its age is 23, average is 54.24, and his hobby is 'reading', ok? So, when we create this, and then let us see when we want to display it, when we call this method display, it will print all these things. So, all these variables whether public or private all these things are able to, we are able to access, ok, right.

Whereas, if I say A, if I say A dot name, name is a public variable. So, it will be able to print, but if I say A dot double underscore age, which was a private variable, it will not be available.

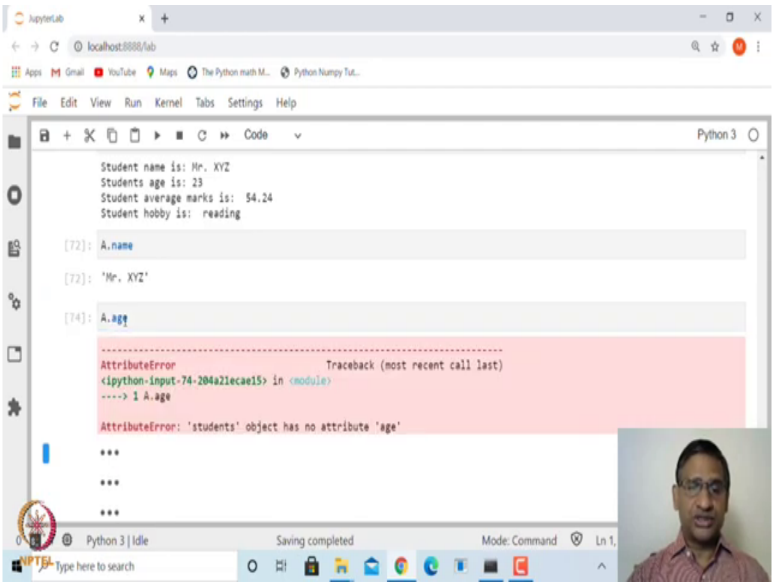
(Refer Slide Time: 23:37)



```
<ipython-input-75-db65b401782e> in <module>
----> 1 A.__age

AttributeError: 'students' object has no attribute '__age'

[76]: A._students__age

[76]: 23

[77]: A._students__age = 32

[78]: A._students__age

[78]: 32

[79]: A._hobby = 'dancing'

[80]: A._hobby

[80]: 'dancing'
```

It says that the 'student' object has no attribute this, and that is how we created dot double underscore.

(Refer Slide Time: 23:44)



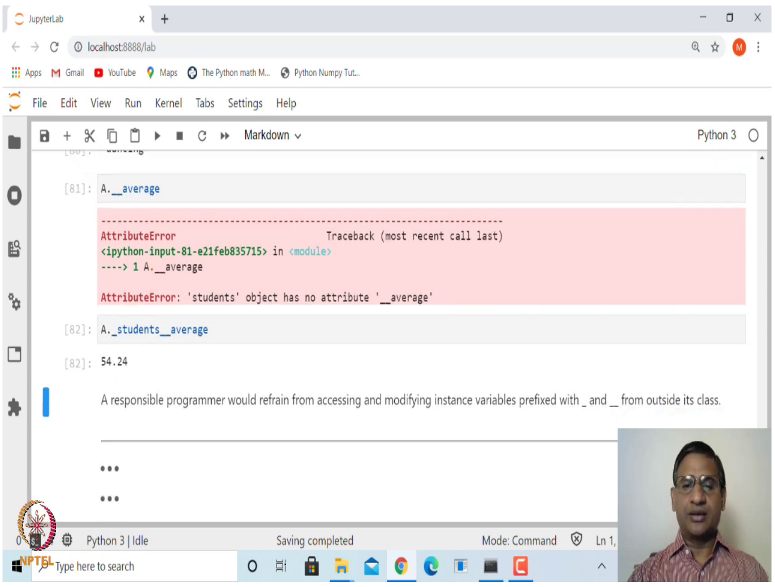
```
Student name is: Mr. XYZ
Students age is: 23
Student average marks is:  54.24
Student hobby is:  reading

[72]: A.name

[72]: 'Mr. XYZ'

[74]: A.age

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-74-204a21ecae15> in <module>
----> 1 A.age

AttributeError: 'students' object has no attribute 'age'
```

Of course, if I say dot age, it of course, it is not available, but even with dot double underscore you cannot access it, ok?

But how do we access, as I said it can still be accessed. So, in order to access, you have to see A dot underscore, the class name, which was students, and then double underscore age. Now you can access this. So, this is 23. Not only you can access, you can even change. So, for example, instead of age 23, if I change it to 32.

And next time when I ask it to, to print, when I asked it to print the age, it will show 32. So, as I said, though it is a private variable, but still it can be accessed with some efforts, and it can also be changed. Similarly, you can look at the, the hobby. So, hobby was private variable, sorry, it is a protected variable, you can, you can change the value of this.

(Refer Slide Time: 24:51)



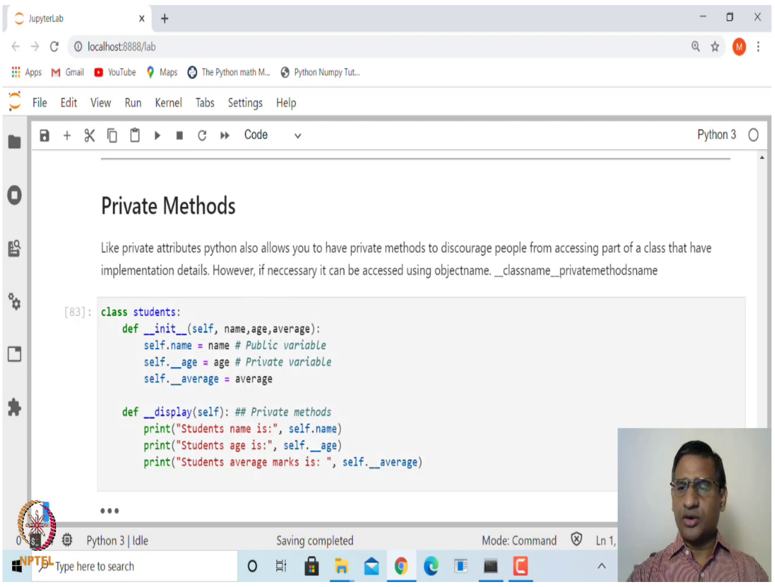
```
[81]: A.__average

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-81-e21feb835715> in <module>
----> 1 A.__average

AttributeError: 'students' object has no attribute '__average'

[82]: A._students__average

[82]: 54.24
```

A responsible programmer would refrain from accessing and modifying instance variables prefixed with _ and __ from outside its class.

So, again, though it is protected variable, still it can be changed. But as I said, whenever a good programmer looks at these type of variable, they will refrain from changing, because they will think that this is a protected variable, ok? Similarly, if I say A dot underscore underscore average, it will not be available, whereas, you, if you want to get access to this, you have to say A dot underscore student double underscore average, ok?

So, this is the, the meaning of protected variable, private variables, and public variables, ok? So, that is what I say. A responsible programmer would refrain from accessing and modifying instances variable, instance variables prefixed with underscore, and double underscore. They will know that they are private variables, and protected variables. So, they will not make use of, make, make any changes, right?

(Refer Slide Time: 25:43)



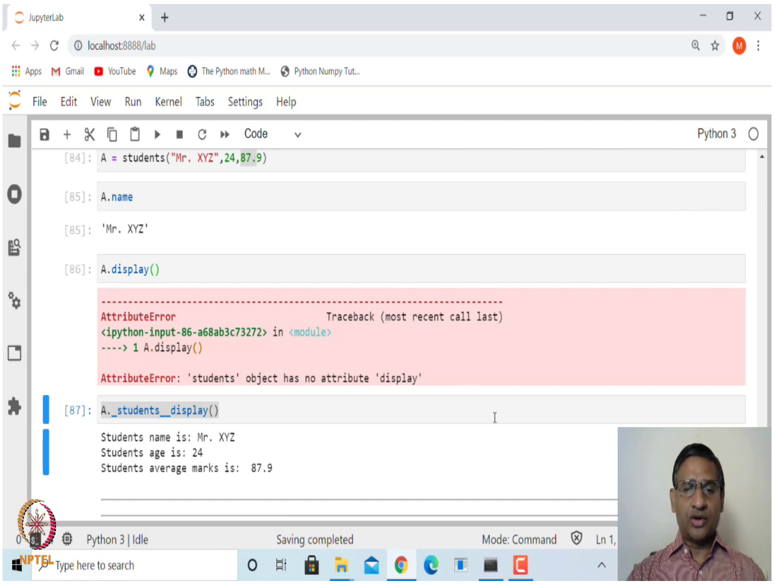
Private Methods

Like private attributes python also allows you to have private methods to discourage people from accessing part of a class that have implementation details. However, if neccessary it can be accessed using objectname._classname__privatemethodsname

```
[83]: class students:
          def __init__(self, name,age,average):
              self.name = name # Public variable
              self.__age = age # Private variable
              self.__average = average

          def __display(self): ## Private methods
              print("Students name is:", self.name)
              print("Students age is:", self.__age)
              print("Students average marks is: ", self.__average)
```

At the end, let me also tell you, you can also create what is called private methods, like private variables, and how do we do that? So, the way to do is, in the name of the method, you prefix double underscore, double underscore, this is double underscore, and then you define exactly similar to what we have defined earlier, ok?

(Refer Slide Time: 26:13)



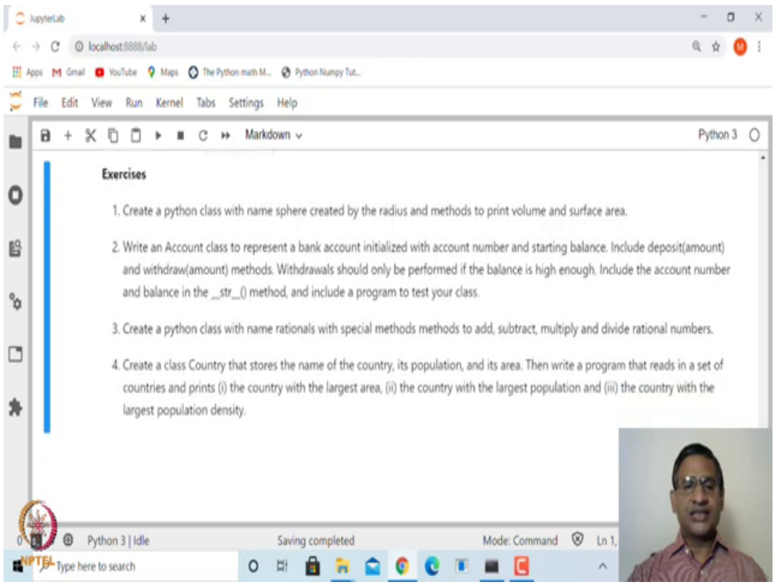
```
[84]: A = students("Mr. XYZ",24,87.9)

[85]: A.name

[85]: 'Mr. XYZ'

[86]: A.display()

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-86-a68ab3c73272> in <module>
----> 1 A.display()

AttributeError: 'students' object has no attribute 'display'

[87]: A._students__display()

Students name is: Mr. XYZ
Students age is: 24
Students average marks is:  87.9
```

Now, in order to, to define an object, let us again create an object A, student A, with name Mister XYZ, with age 24, the, the average value is this, and when we ask what is the name, it will be of course, give you, if I say A dot display. A dot display, they say that it does not have attribute, because we have created as a private method.

So, if I want to access that, I have to say A dot underscore student double underscore display, exactly similar to how you accessed private variable, ok? So, again this is what you have. So, these are the private methods. Of course, there are many more things when it comes to dealing with classes in Python. This is very important concept as far as object-oriented programming language is concerned.

But we are not going to get into all the details, because our focus is to learn mathematics using SageMath, and since SageMath is object-oriented, SageMath is based on Python, which is object-oriented programming language, I thought it is a good idea to introduce briefly, classes, so that you will be able to, to understand it much better.

(Refer Slide Time: 27:39)



Exercises

1. Create a python class with name sphere created by the radius and methods to print volume and surface area.
2. Write an Account class to represent a bank account initialized with account number and starting balance. Include deposit(amount) and withdraw(amount) methods. Withdrawals should only be performed if the balance is high enough. Include the account number and balance in the __str__() method, and include a program to test your class.
3. Create a python class with name rationals with special methods methods to add, subtract, multiply and divide rational numbers.
4. Create a class Country that stores the name of the country, its population, and its area. Then write a program that reads in a set of countries and prints (i) the country with the largest area, (ii) the country with the largest population and (iii) the country with the largest population density.

At the end, let me leave you with some simple exercises.

So, there are four exercises of creating classes. The first exercise is, create a Python class with name sphere, sphere is the name of the class, and it should have input radius. So, you create the object with some radius, and then print its volume, and surface area. So, these are the two methods volume, and surface area.

Similarly, create a class called Account, for account details of some customers in a bank, right, and which, as initialization, has the account number, and starting balance, and inside that you create two, let us say method called deposit of an amount, and withdrawal of an amount. So, when you deposit an amount, the initial, it should add to initial amount, when you withdraw, of course, withdrawals should be possible only when there are enough balance available.

And then, when you withdraw, the balance should get deducted, and of course, for printing you can use str method, ok?

Create another class called rational for doing computation with rational numbers, and use special methods like add, subtract, multiply, as we have done for vector 3D. So that you should be able to add, multiply, divide, subtract rational numbers, even equate, ok?

And the last one is to create a class called country, and that stores name of the country and its population, and the surface area, the land area, and then write program to, to read in a set of countries which reads a set of countries, and prints the country name with largest area, and the country name with largest population, and also the country name with largest population density.

So, these are the four simple exercises. Of course, we will be posting the solution of these exercises, but you should try to do it, ok? So, in the next class onwards, we will be actually focusing on, we will start learning SageMath, and explore mathematics with SageMath, but going forward, you should also look at some more packages in Python.

For example, you can look at a package called pandas, which is very important for data analysis; it deals with handling data, and doing various kinds of data cleaning, and things like that, including the plotting, and things like that.

So, if you, if you have, if you want to learn more detailed on Python as a mathematics student, these are the things which you should concentrate on; learn basic Pythons, looping, then learn how to create your own functions, make use of NumPy library, SciPy library, library look at these classes, and if you want to deal with data analysis, maybe look at the pandas, ok?

So, I hope these two weeks of introductory lecture on Python will actually make you well equipped in order to make use of SageMath in a much better way.

Thank you very much.