INDIAN INSTITUTE OF TECHNOLOGY MADRAS

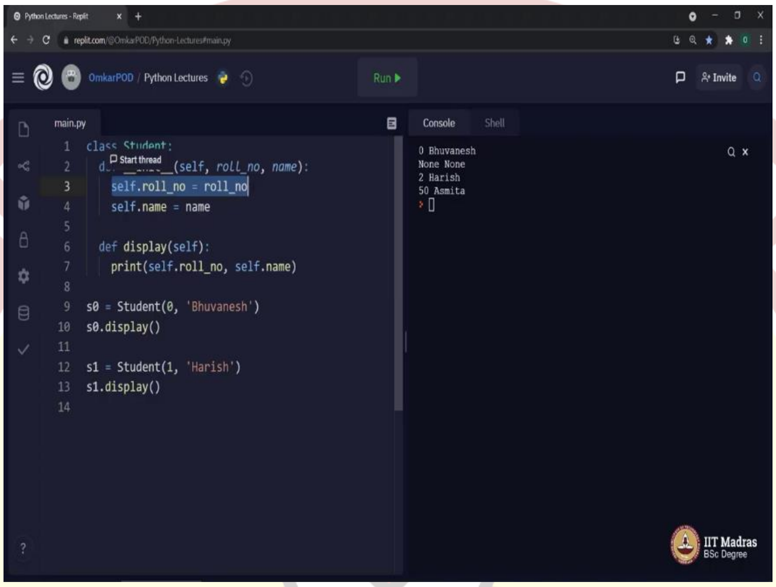
सिद्धिर्भवति कर्मजा

IIT Madras

ONLINE DEGREE

Programming in Python
Professor. Sudarshan Iyengar
Department of Computer Science and Engineering
Indian Institute of Technology, Ropar
Mr. Omkar Joshi
Course Instructor
Attributes and Methods

(Refer Slide Time: 0:16)



```python
class Student:
    def __init__(self, roll_no, name):
        self.roll_no = roll_no
        self.name = name

    def display(self):
        print(self.roll_no, self.name)

s0 = Student(0, 'Bhuvanesh')
s0.display()

s1 = Student(1, 'Harish')
s1.display()
```

```
0 Bhuvanesh
None None
2 Harish
50 Asmita
```

Hello python, students. This is the program where we left in our last lecture. And if you remember, I said, even though this program executes fine, still, this is not a ideal way to write python program using object oriented programming concepts. And at the same time, so far, we have only considered attributes part of the object, but we have not talked about behavior part of an object.

So first, let us modify this code in order to make it more appropriate for OOP concepts, and then we will think about behavior part of object. First, I will modify the entire code and then I will try to explain every aspect of that code. Alright, this is the most accurate way of using OOP concepts in Python.

Once again class, followed by class name, which is student and after that, we have one special method this method or this function is referred as init, two underscores then this word init again followed by two underscores, and it has three parameters. First is self, second is roll number, and third is name. We already know, why we have roll number and name, but this self is a new term, we will come to that.

After that, we are doing something like self dot roll number is equal to roll number self dot name is equal to name. Then we have a regular function called display. Once again in this function as well we have something called as self. And if you come to this particular object part, then S0 object, instead of just calling that constructor, which we had earlier, now in that parenthesis we are passing two parameters. First parameter will represent roll number, whereas second parameter will represent name.

Similarly, we are doing it for second object. S1, where first parameter is 1 and second parameter is name Harish. So, the first question is, what is this init? It looks like a function, but we have never called that function anywhere in our code. Then, what is the use of such function? And the answer is, if you remember, last time, I said, this is the constructor of the class.

But if we have these kinds of parameters, which we want to initialize against some variables, which are available in the class, then we have to use a constructor with parameters, and whenever we call such a constructor, internally python executes this particular function called init. Therefore, we do not have to explicitly call this function like other functions, like this display. Whenever we say student in bracket these two parameters, python executes this particular function.

So, whenever we said this is S0 is equal to student and these values in parentheses, python executed these three lines. Once again, whenever we said S1 is equal to student again, in bracket these two parameters, python executed these three lines. If we do same thing, again, with let us say S3 is equal to again, student with two parameters, python will execute these three lines.

So, every time we try to create an object using these two parameters, python will execute this particular function, which is called init. And because of that, the name of this particular function has to be exactly like this, two underscores followed by init followed by two underscores. This is a special function, which exists inside every single class in python programming language. Alright, that part is sorted.

Now, the next question, what is this self? Self is a new keyword, which is used along with OOP concepts. We are initializing this role number 0 and named Bhuvanesh to these two variables roll number and name. Similarly, this roll number 1 and name Harish is also getting initialize these two same variables, roll number and name. Then how python will know whether these two variables belong to S0 or S1? And this confusion is resolved using this keyword, self.

Every time any function from the class is executed python knows which object initiated that execution. Because of that, whenever we execute this line number 9, the value for this self is 0. And because of that, it comes over here it says S0 dot roll number S0 dot name. Whereas, in second case, as we execute this line number 12, the value of self is now S1. Hence, it becomes S1 dot roll number, S1 dot name.

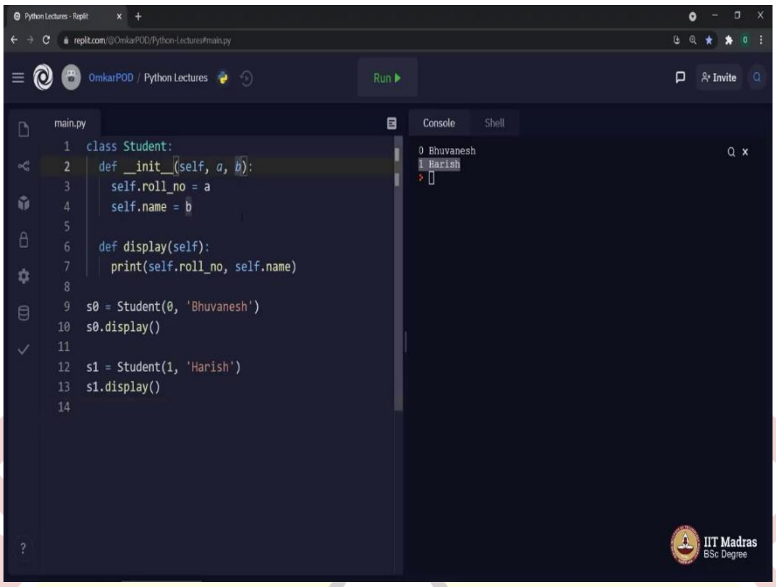
So, in order to generalize this self is a container which holds the current object, and because of that we have to use this self, as the first parameter in every single function, which is there inside a class. Similarly, whenever we say S0 dot display, computer will execute this display function. And because of this S0, the value which is associated with self is S0, the object S0. And because of that, when this particular print statement executes it says S0 dot roll number and then S0 dot name.

Whereas in second case, S1 dot display will make this self as S1 and then we will get S1 dot roll number, and S1 dot name. Alright. Now init is done, self is done. What is this part? Why we are seeing roll number is equal to roll number, and name is equal to name? This does not make any sense. So, the answer is, if you remember from previous lecture whenever we create a object, it has its own copy of every single variable inside that class, which means, this S2 has roll number and name.

At the same time, S1 also has its own roll number and name, and these two variables, roll number and name can have different values for S0 as well as S1, whereas, these two variables roll number and name, which is on the righthand side of this equal to sign are nothing but our parameters of this function. So, this value 0 will be passed to this variable roll number, this value Bhuvanesh will be passed to this variable name.

These two are normal variables, which we always use along with functions. These two variables cannot be accessed outside this function definition because these are function variables, whereas, these two variables; self dot name and self dot roll number are object variables, which are nothing but attributes of these two objects S0 minus S1. And that is why we are saying self dot roll number is equal to roll number.

(Refer Slide Time: 11:42)

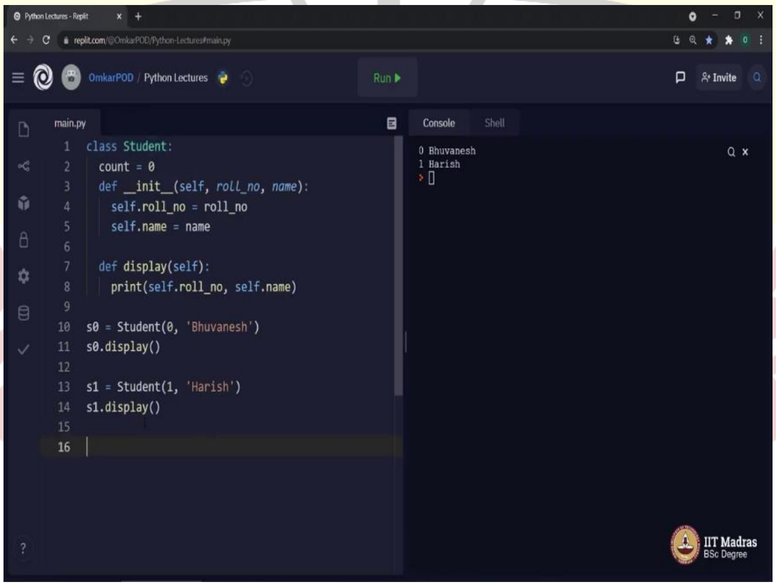


```
class Student:
    def __init__(self, a, b):
        self.roll_no = a
        self.name = b

    def display(self):
        print(self.roll_no, self.name)

s0 = Student(0, 'Bhuvanesh')
s0.display()

s1 = Student(1, 'Harish')
s1.display()
```

```
0 Bhuvanesh
1 Harish
> 
```

If this is too confusing, we can always replace it with something simple, let us say a, and we can make this as a, and this adds b and b. This is exactly similar to what we had earlier. Let us first execute both the versions and then we will continue our discussion. As expected, we got output 0 Bhuvanesh and 1 Harish.

(Refer Slide Time: 12:17)



```
class Student:
    count = 0
    def __init__(self, roll_no, name):
        self.roll_no = roll_no
        self.name = name

    def display(self):
        print(self.roll_no, self.name)

s0 = Student(0, 'Bhuvanesh')
s0.display()

s1 = Student(1, 'Harish')
s1.display()

```

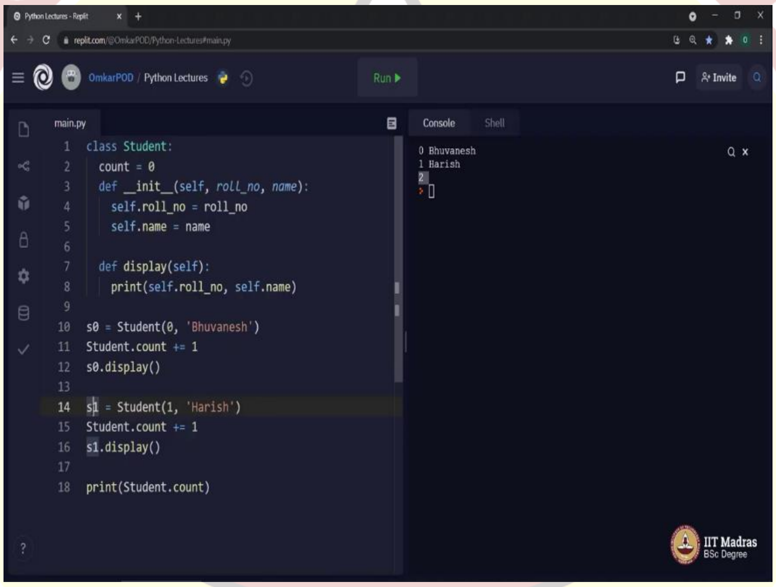
```
0 Bhuvanesh
1 Harish
> 
```

Let us go back to our original code, roll number, name, roll number and name. Let us execute it again, and the output is same, 0 Bhuvanesh, 1 Harish. So, now, the next question is what was the difference between previous code and this code both are giving same output, and the differences in earlier code we use these variables roll number and name directly inside a class. But in this case, we are using these variables inside a specific function called init.

So, you must be wondering, so what, still we are getting same output then why does this make any difference. So, the difference between having variable inside a class and inside init is like this. Whenever we have something inside init, it is owned by or it belongs to a specific object either S0 or S1.

But if we have something over here in the class itself, then it is owned by class, not its object. Which means, let us say if we have something like count over here then this count variable is not owned by S0 or S1, we will not have two separate copies of count inside S0 and S1, there will be only one copy of this count variable, which is owned by a class student. And it will be shared by every single object of this class student.

(Refer Slide Time: 14:25)



```python
class Student:
    count = 0
    def __init__(self, roll_no, name):
        self.roll_no = roll_no
        self.name = name

    def display(self):
        print(self.roll_no, self.name)

s0 = Student(0, 'Bhuvanesh')
Student.count += 1
s0.display()

s1 = Student(1, 'Harish')
Student.count += 1
s1.display()

print(Student.count)
```

```
0 Bhuvanesh
1 Harish
2
```

So, for example, at the end, if we say print, now as we know count is owned by student, so we cannot say S0 dot count or S1 dot count, we should say student dot count. And then we will see what happens, as expected output is 0 because we have not modified this value. So, let us write, so let us update this value, student dot count plus equal to 1. Same when we create second object. Let us execute this code.

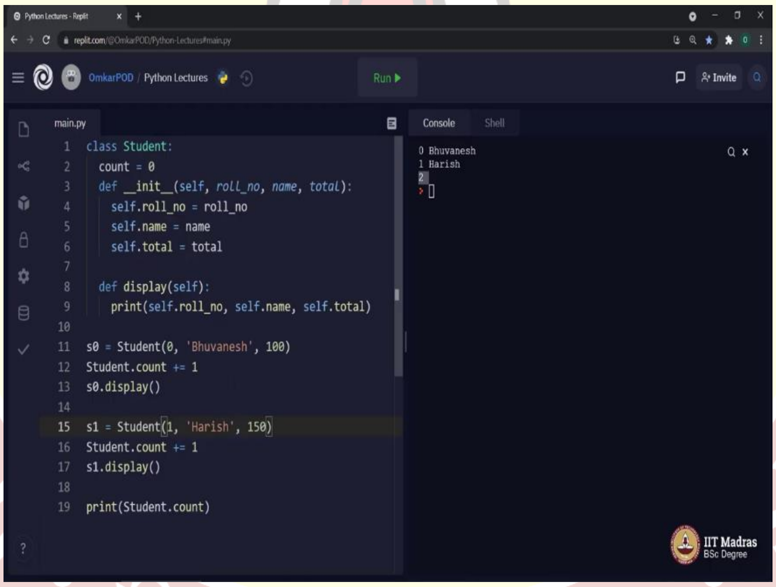
Now it says 2. Because, as I said this count variable does not belong to S0 or S1, this count variable belongs to class. And because of that, there is only single copy of this particular variable, which is shared by every single object. And this is the reason this variable is used always with the class name. And this is the difference between having a variable inside a class and a variable inside init.

So, let me repeat variables inside init belong to individual object, whereas, variables inside class belong to class and all objects will share that variable, they will not have their own copy of that variable. So, this count is not an object attribute, this is a class attribute, whereas roll number and name are object attributes. Alright.

Now, we have done with attributes part of the object, still the behavior part is missing. You might say that these two functions are already there, and we said functions are considered as behavior of objects, which is true. But if you consider this init or display are not actually adding any value to the existing objects because init is simply initializing the values, whereas display is simply printing the values. They are not doing anything specific to these two objects.

They are not adding any special behavior to these objects. So, let us make few changes into this code to add the behavior part of object.

(Refer Slide Time: 17:31)

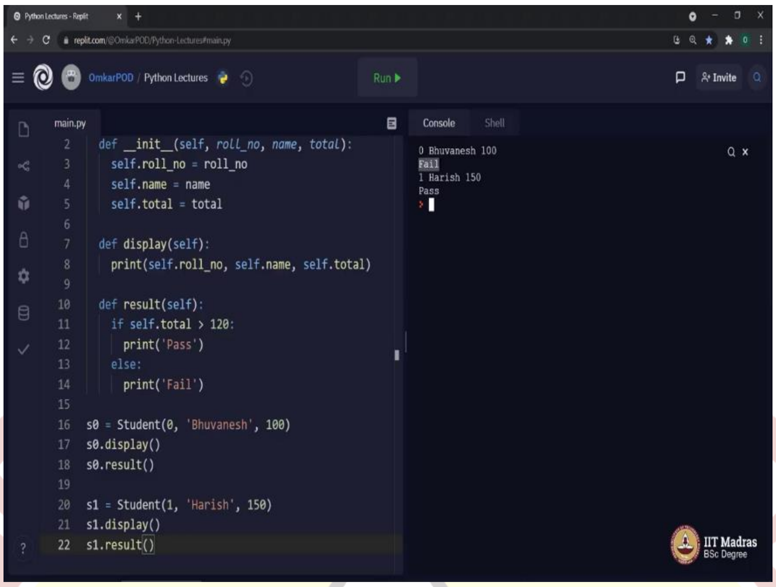


```python
class Student:
  count = 0
  def __init__(self, roll_no, name, total):
    self.roll_no = roll_no
    self.name = name
    self.total = total

  def display(self):
    print(self.roll_no, self.name, self.total)

s0 = Student(0, 'Bhuvanesh', 100)
Student.count += 1
s0.display()

s1 = Student(1, 'Harish', 150)
Student.count += 1
s1.display()

print(Student.count)
```

```
0 Bhuvanesh
1 Harish
2
>
```

Let us say, we have one more parameter from scores data set it is total. Let us initialize it inside init method. We will print it as well self dot total and we have to pass some value here, let us say 100 and 150.

(Refer Slide Time: 18:01)



```python
    def __init__(self, roll_no, name, total):
        self.roll_no = roll_no
        self.name = name
        self.total = total

    def display(self):
        print(self.roll_no, self.name, self.total)

    def result(self):
        if self.total > 120:
            print('Pass')
        else:
            print('Fail')

s0 = Student(0, 'Bhuvanesh', 100)
s0.display()
s0.result()

s1 = Student(1, 'Harish', 150)
s1.display()
s1.result()
```

```
0 Bhuvanesh 100
Fail
1 Harish 150
Pass
>
```

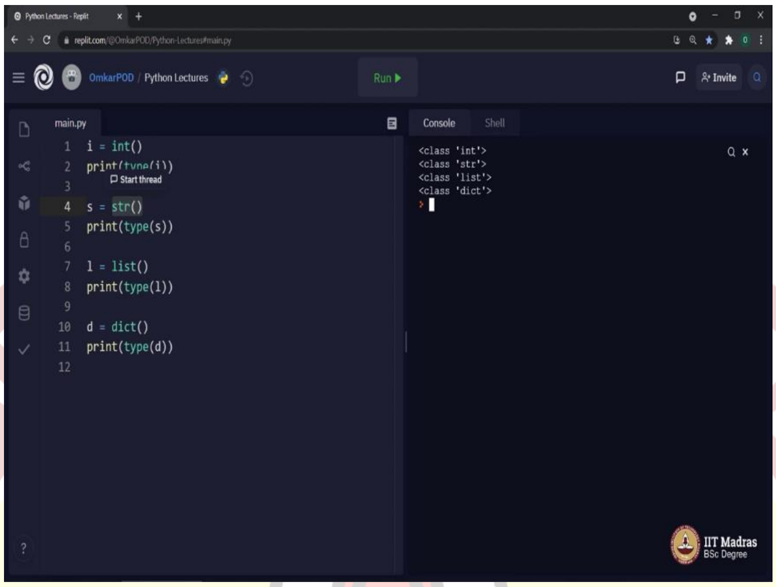
Now, we do not want this count, so let us remove this and this as well. First, let us execute. We are getting 0 Bhuvanesh 100, 1 Harish 150. Now the behavior for any student is his result, whether Bhuvanesh and Harish passed the exam or they failed the exam. So, let us add that behavior def result. Once again, this particular function is inside class. So, we have to write self. If self dot total greater than, let us say 120, print pass else print fail. Now after display, we will call S0 dot result and S1 dot result.

Now, let us execute this code. 0 Bhuvanesh, 100 fail, 1 Harish, 150 Pass. Now this fail or pass is a behavior for individual objects based on their total marks. So, these first three values represent individual attributes, whereas, this particular value represents behavior of that particular student, similarly for second student, and this is how we write object-oriented programs using classes and objects, including its behavior.

So, before closing this lecture a small note. I have been using two different names for these functions inside class. Sometimes, I called it function or sometimes I called it method. But if you remember, we have seen this earlier when we discussed about string methods. And the difference between function and method is, whenever any function is inside a class or when any function belongs to a class, then that particular function is referred as method. And that is why init, display, result all are referred as methods, methods of class student.

And because of the same reason, earlier, we used to call all those string functions as methods, because there is a class called string, and all those functions are inside that class. That is why they are methods. I know this is just a technical term, which differentiates two different types of functions, but is always a good habit to prefer these terms with their appropriate names, functions and methods. Now, it is time to give you one small exercise. I will write one code and you have to figure out on your own what exactly happening with that code.

(Refer Slide Time: 22:00)



```python
i = int()
print(type(i))

s = str()
print(type(s))

l = list()
print(type(l))

d = dict()
print(type(d))
```

```
<class 'int'>
<class 'str'>
<class 'list'>
<class 'dict'>
>
```

Consider this particular code block, let me execute it first. It has some output. Now, it is your responsibility to relate this to the concepts of object-oriented programming, which we discussed just now.

So, the first question is, what is this int, str, list? And the second why always it says class int, class string, class list, class dictionary. So, this is small homework where you have to figure out what exactly these functions are and why it always says class? Thank you for watching this lecture. Happy learning.