



# IIT Madras

## ONLINE DEGREE

**Programming in Python**  
**Professor Sudarshan Iyengar**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Ropar**  
**Classes and Objects**

(Refer Slide Time: 0:17)

## Classes and Objects

Classes and Objects



Hello Python students. In last lecture, we discussed about attributes which are nothing but variables. And then we talked about behavior, which refers to functions. We also introduced a new entity called object, which is a combination of attributes and behavior. We also said every object or in that case every student must have own identity. This means every student will have own set of variables and functions.

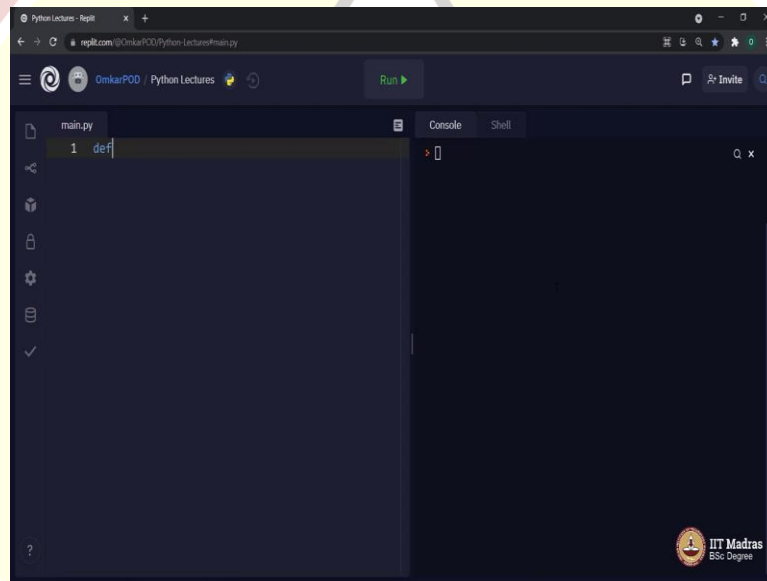
But then there is a problem because now there is no way to maintain the uniformity across all students. What if first student has all those attributes from scores dataset. Second student also has all those attributes except date of birth, which means one attribute is missing. Whereas, third student has one extra attribute, let us say email address.

This situation of missing or extra parameter will cause a lot of problems when we start doing some computations using that data. This means we want every student to have own identity, but at the same time, in order to maintain the uniformity, we also want these students to express their identity through some common predefined attributes, like what we have in scores dataset.

So, the question is, how to achieve this? And the answer is using class. For example, architects or civil engineers use blueprints to construct houses, if they use same blueprint to build 100 houses, then they all will have same structure, but then they can have different names, different colors, different interior, exterior and so on.

If you relate this to the concept of classes and objects, then that blueprint is a class and all those 100 houses are objects which have some uniformity, but at the same time, they have their own identity. Now, I hope this idea of classes and objects is clear to you. So enough of this theory, and now let us jump to our Python editor and start coding using classes and objects.

(Refer Slide Time: 3:26)



```
Python Lectures - Replit
replit.com/@OmkarPOD/Python-Lectures/main.py

main.py
1 class Student:
2     roll_no = None
3     name = None
4
5 s0 = Student()
6 s0.roll_no = 0
7 s0.name = 'Bhuvanesh'
8 print(s0.roll_no, s0.name)
9

Console
0 Bhuvanesh
> []

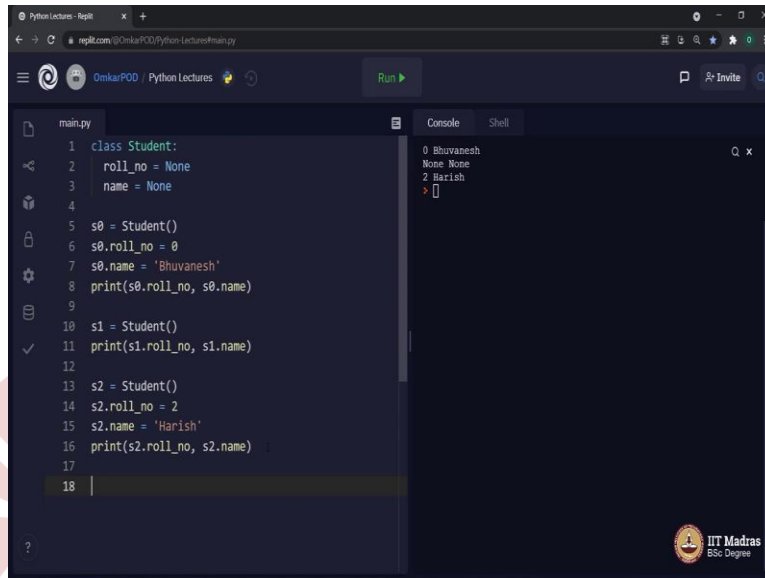
IIT Madras
BSc Degree
```

```
Python Lectures - Replit
replit.com/@OmkarPOD/Python-Lectures/main.py

main.py
1 class Student:
2     roll_no = None
3     name = None
4
5 s0 = Student()
6 s0.roll_no = 0
7 s0.name = 'Bhuvanesh'
8 print(s0.roll_no, s0.name)
9
10 s1 = Student()
11 print(s1.roll_no, s1.name)
12
13 s2 = Student()
14 s2.roll_no = 2
15 s2.name = 'Harish'
16 print(s2.roll_no, s2.name)

Console
0 Bhuvanesh
None None
> []

IIT Madras
BSc Degree
```

A screenshot of a web-based Python IDE interface. The left pane shows a file named 'main.py' with the following code:

```
1 class Student:
2     roll_no = None
3     name = None
4
5 s0 = Student()
6 s0.roll_no = 0
7 s0.name = 'Bhuvanesh'
8 print(s0.roll_no, s0.name)
9
10 s1 = Student()
11 print(s1.roll_no, s1.name)
12
13 s2 = Student()
14 s2.roll_no = 2
15 s2.name = 'Harish'
16 print(s2.roll_no, s2.name)
17
18
```

The right pane shows the 'Console' output:

```
0 Bhuvanesh
None None
2 Harish
```

The interface includes a 'Run' button at the top and a search bar on the right. A watermark of the IIT Madras logo is visible in the background.

Please observe this small and simple piece of code very carefully, as I type it slowly. We use `def`, as in `define` to write a function. Similarly, we use another keyword called `class` to define a class. So, `class` followed by class name let us say, `Student` and if you can observe the first letter of class name is capital. This is not a mandatory thing. But it is a standard practice followed by all the programmers, which says the first letter of class name is always capital.

Let us say roll number is equal to default value, let us say it is `none`, because it will vary from student to student. Similarly, name is equal to `none`. And just like that, our class is ready. As you can see, creating a class is very simple. It just defines what will be the blueprint, what will be the template for the object.

Now, here onwards, we can create as many objects as we want, which refer to this particular class `Student`, and each of those objects will have two different variables, roll number, and name. And the default value will be `none`. So, now, as the class is ready, let us see how object is created.

Let us pick the first student from our score's dataset. As the ID starts from 0, let us say `S0` is equal to student and that is it, that is how we create object. Now, this `S0` is an object of class student. But if you notice, this particular highlighted command, it looks like a function student followed by parenthesis and we know we call functions using such parenthesis.

So, you might be wondering, is it a class, is it a function or is it something else. The answer is, this is a special type of function only. And I call it special because this particular function name is exactly identical to the name of the class. And we do not have to define this function anywhere in the given class.

As you can see, we have not returned `def student` anywhere in our program. Still, it will work because whenever we create a class with a specific name, Python on its own, add one function inside it, and the name of that function is always same, which is given to the class. And in programming terms, this special function is referred as constructor.

Why constructor? Let us go back to that same example. We use blueprint to construct houses. Same, we use class to construct objects. So, this is that constructor, which is used to construct objects. Alright, so far, we got the class, we got object. Now, somehow for this student `S0`, we have to specifically mention what is the roll number and what is name.

`S0` dot roll number is equal to 0. As we all know, in scores dataset, the numbers start from 0. Similarly, we can write `S0` dot name is equal to and the name of first student is Bhuvnesh. Now, let us print and check whether it is working or not. Print `S0` dot roll number. Similarly, `S0` dot name let us execute, we got the output as 0 and name Bhuvnesh.

Now, how exactly it works? As I said, using this constructor we constructed a object called `S0`. Then inside this particular object, we have two variables which are owned by this particular object `S0`. So, for that specific object, we set the value to 0 for roll number and Bhuvnesh for name using this dot symbol over here.

This particular symbol is referred as dot operator in Python language. Whenever we want to access any entity which belongs to an object, we use this dot operator, object name, dot operator, and then variable name; object, dot operator and then function name, and so on. This is how we access attributes and behavior of a specific object.

If I create one more object, let us say `S1` is equal to `student`, we got the object. And instead of writing these two steps, where I initialize roll number and name for student `S0`, I decided not to do this for `S1`. Instead, I directly wrote something like this `print S1 dot roll number`, and `S1 dot name` and executed the program.

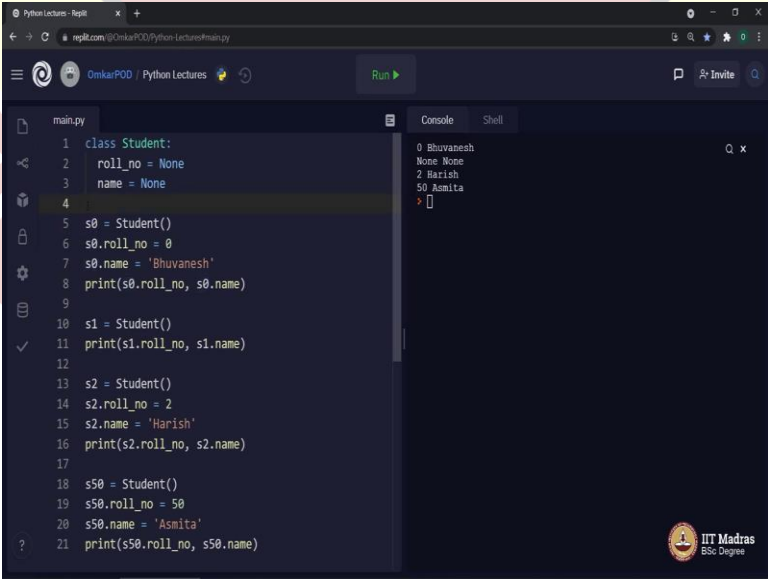
It shows output 0 Bhuvnesh for S0 object. And then for S1 object, it says none, and none. Because those were the default value initialized inside the class. And this is how we make sure each student or each object will have its own identity. These two variables roll number and name are same for S0 as well as S1.

But still, because of this class and object structure, we make sure, we modify these values only for S0, and it has no effect on S1. Similarly, if we create one more object, let us say S2 is equal to student. Now, S2 dot roll number is equal to let us say next roll number 2 and S2 dot name is Harish.

Print S2 dot roll number, comma S2 dot name. Let us execute. As you can see, once again, this S2 object has its own identity with roll number 2, and name Harish. So, in this case, we are reusing these two same variables. But instead of modifying the value of original variables over here in class, we are modifying its values specific to an object, either it is S0 or S2.

And in case of S1, we are not modifying it at all. This particular feature of classes and objects provides us that flexibility, which we discussed in our previous lecture. For example, after this, let us say a totally new student join this college, a student which is not there in our score's dataset.

(Refer Slide Time: 13:25)



```
main.py
1 class Student:
2     roll_no = None
3     name = None
4
5 s0 = Student()
6 s0.roll_no = 0
7 s0.name = 'Bhuvnesh'
8 print(s0.roll_no, s0.name)
9
10 s1 = Student()
11 print(s1.roll_no, s1.name)
12
13 s2 = Student()
14 s2.roll_no = 2
15 s2.name = 'Harish'
16 print(s2.roll_no, s2.name)
17
18 s50 = Student()
19 s50.roll_no = 50
20 s50.name = 'Asmita'
21 print(s50.roll_no, s50.name)
```

Console

```
0 Bhuvnesh
None None
2 Harish
50 Asmita
```



Let us say something like S50. Again, S50 is equal to student, as it is a new student should not have a roll number allotted to her. So, we cannot say S50 dot roll number equal to something because still, a roll number is not allotted to this particular student, because she is a new student in the college.

But irrespective of that, still she will have a name. So, we can do something like S50 dot Name is equal to let us say Asmita. Print S50 dot roll number S50 dot name. Let us execute. And we have output for this fourth student where roll number is none. But the name is Asmita. Now, what she will say?

She will say at this point of time, I do not have roll number. That is why I am saying the roll number is none. As soon as college allots me a roll number, I will simply say S50 dot roll number is equal to let us say 50. And now she also has a roll number. So, the point over here is, this division of class and object provides us this flexibility where we are using same variable names with different values for each object.

In this case, the name might be shared across all objects, but the values which are stored for each object are different. And that is exactly how things happen in real world. Every person has a name. So, this particular attribute called name is there for every single person, but the value against it will vary.

And this is how we bring the real world into programming through object-oriented programming, or through classes and objects. Before closing this lecture, a small instruction for everyone, even though this is working fine, this is not the ideal way to write a program using objects and classes. There is small change which is required in order to accommodate the second part of object concept, which is behavior.

So far, we are working only with attributes. We have not discussed about that second part, which is behavior, which is nothing but functions. So, remember this code. In our next lecture, we will try to modify this code in order to accommodate behavior, along with these attributes. Thank you for watching this lecture. Happy learning.