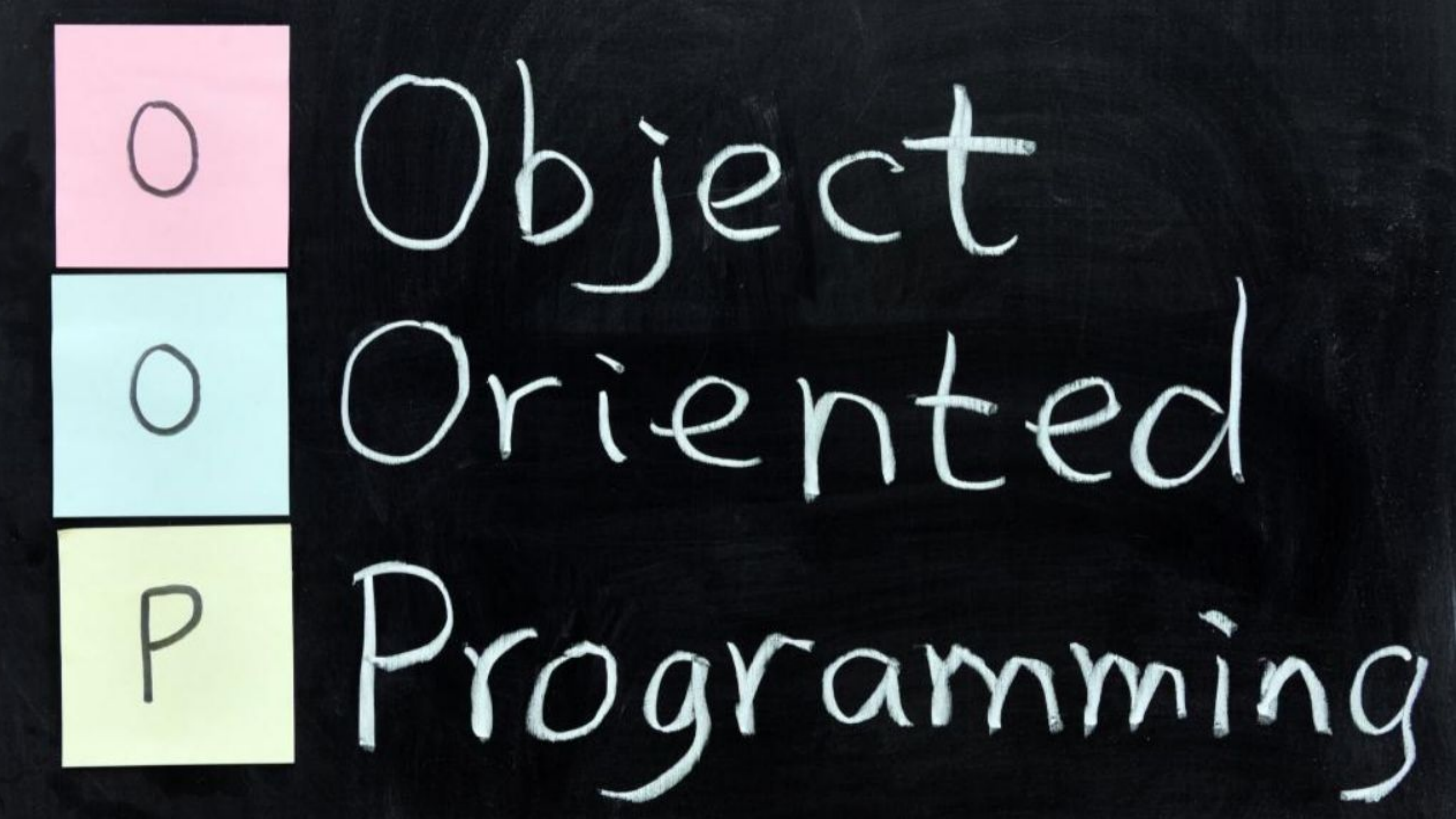
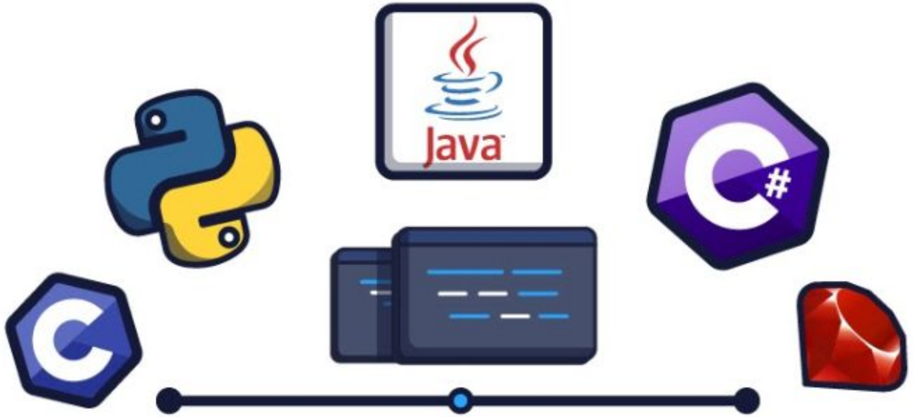

O
O
P
Object
Oriented
Programming

Object Oriented Programming?

Object Oriented Programming concept is a way of programming in which every possible agent is broken down into elements called “object”. i.e. It is oriented around entity called objects which are basically models of the real world.

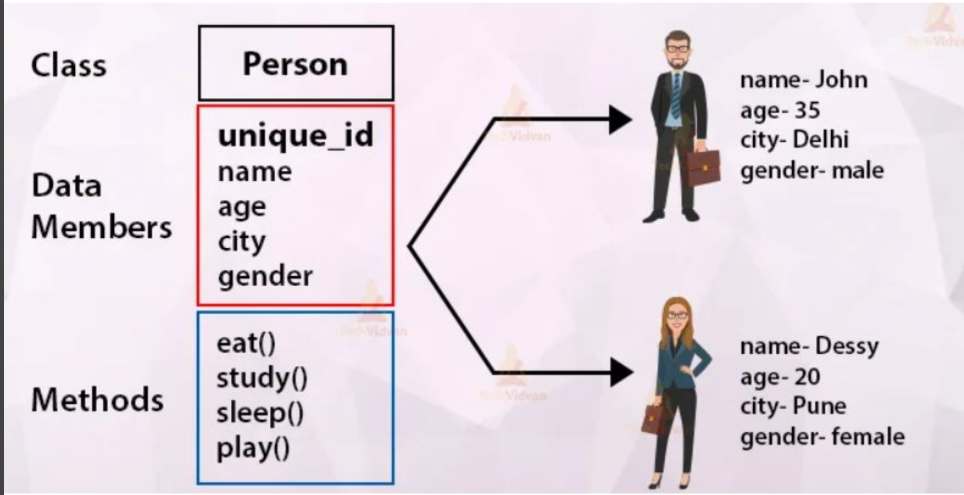
Some examples of OOPL are Python, Ruby, C# etc.



Basic Element of OOP

The most basic element in OOP are classes. They are the blueprint for objects. Whenever we need to create objects, we invoke classes.

Class consist of “attributes” that define characteristics of objects and “functions” that define what task can the object perform.



Object & Class Intuition

Waiter

Here, Waiter is a class

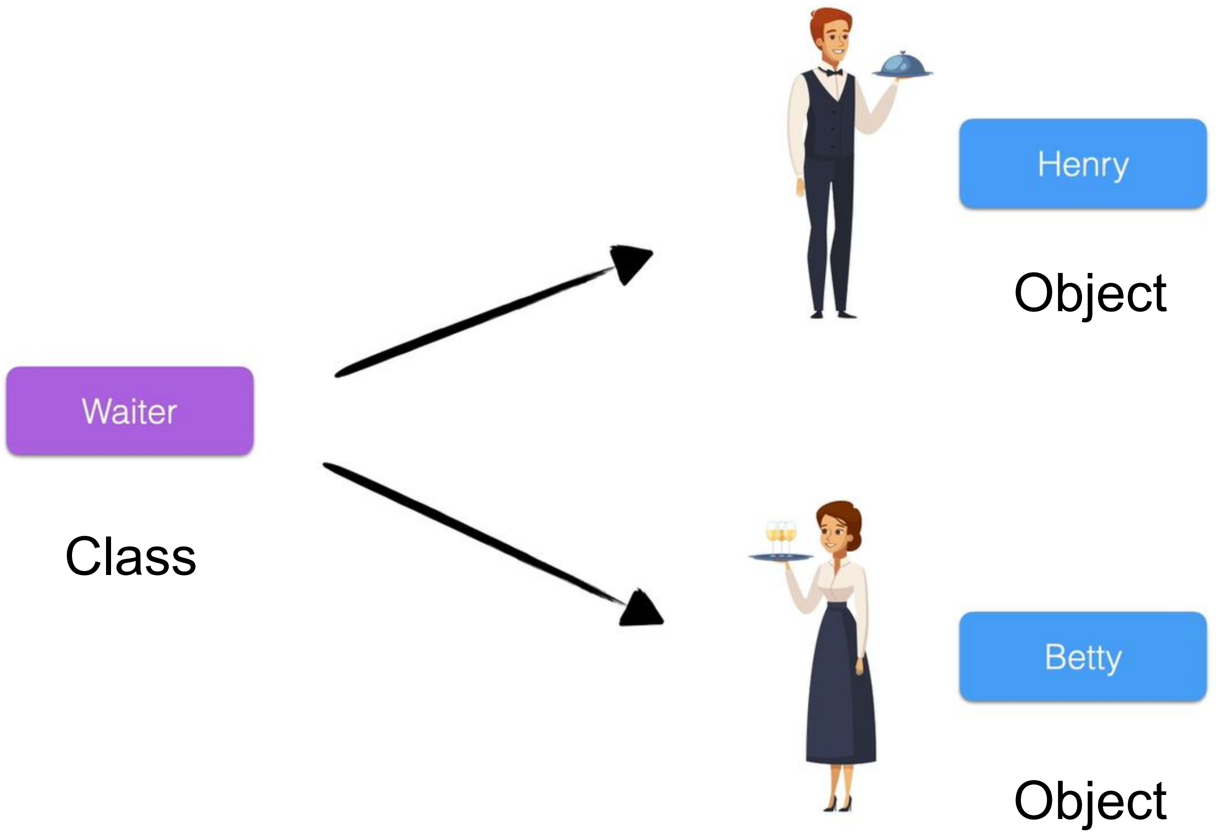
has:

```
is_holding_plate = True
tables_responsible = [4, 5, 6]
```

does:

```
def take_order(table, order):
    #takes order to chef

def take_payment(amount):
    #add money to restaurant
```

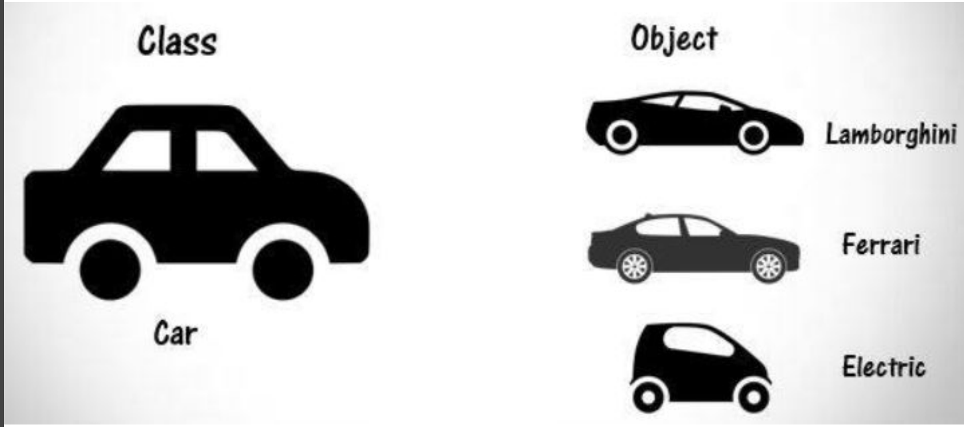
Object & Class Intuition



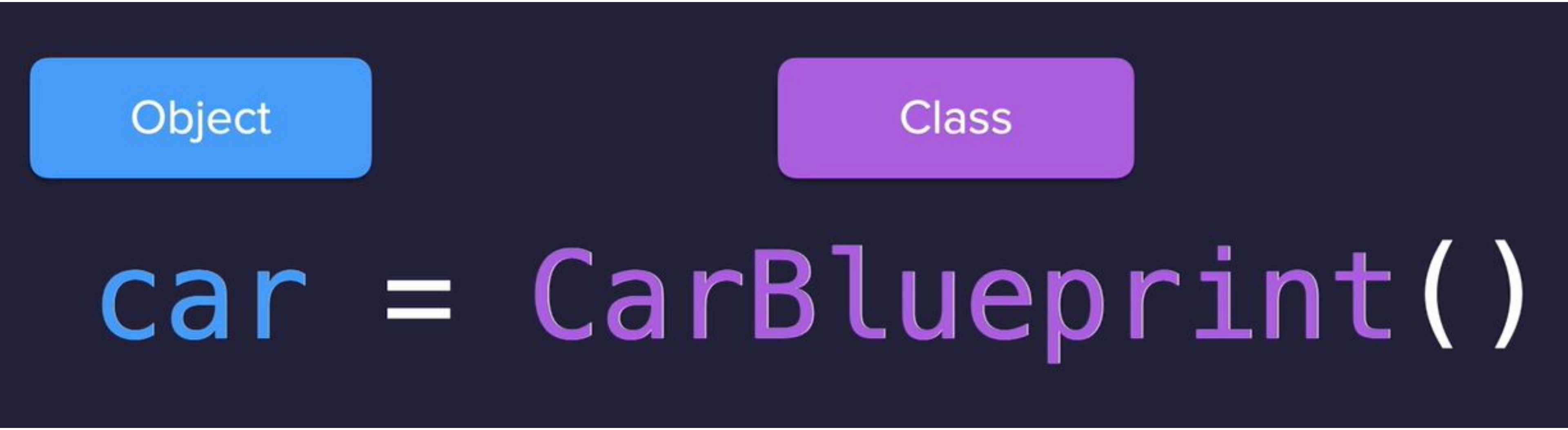
Object and Class Intuition

The blueprint is called class.

The entities created using class are called objects.



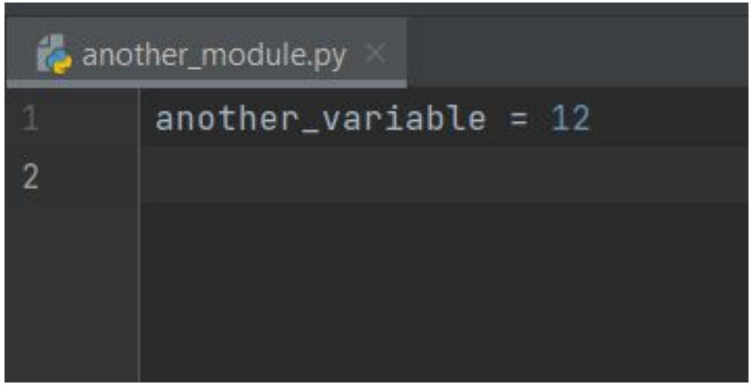
Example of how classes are invoked in Python



```python
car = CarBlueprint()
```

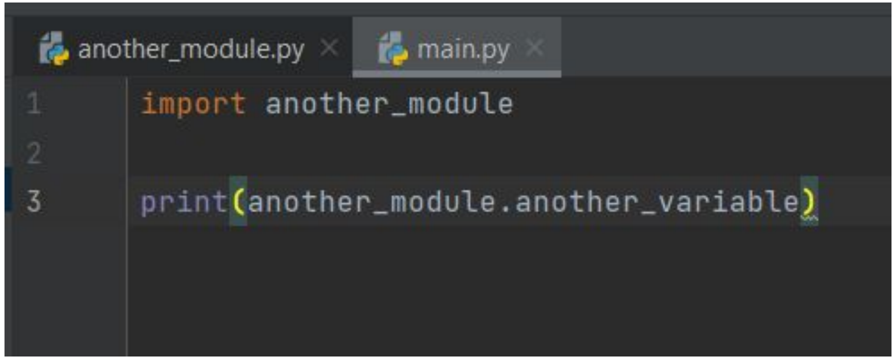
Let’s have a look at a basic example:

Initially we created a file and named it “**another_module.py**” and create a variable and give it a value.


another_module.py

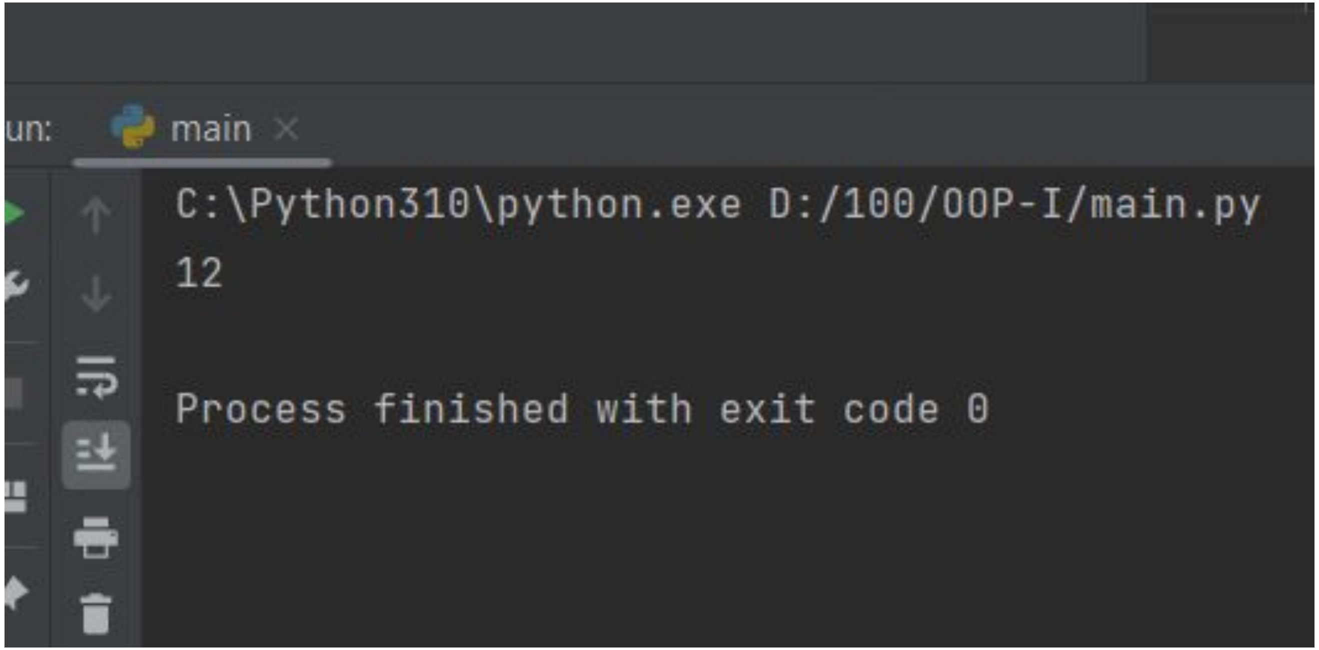
```
another_variable = 12
```

Now, we created a new file called “**main.py**” and imported “**another_module.py**” using “**import**” command.
After importing, we now can use variables inside “**another_module.py**” in “**main.py**” as shown below:


another_module.py main.py

```
import another_module

print(another_module.another_variable)
```

The output we get is :



```
C:\Python310\python.exe D:/100/OOP-I/main.py
12

Process finished with exit code 0
```

What are python packages?

Python packages are the modules that other developer wrote. We can import that and use the functions to perform desired work.

How can we import those packages into our project?

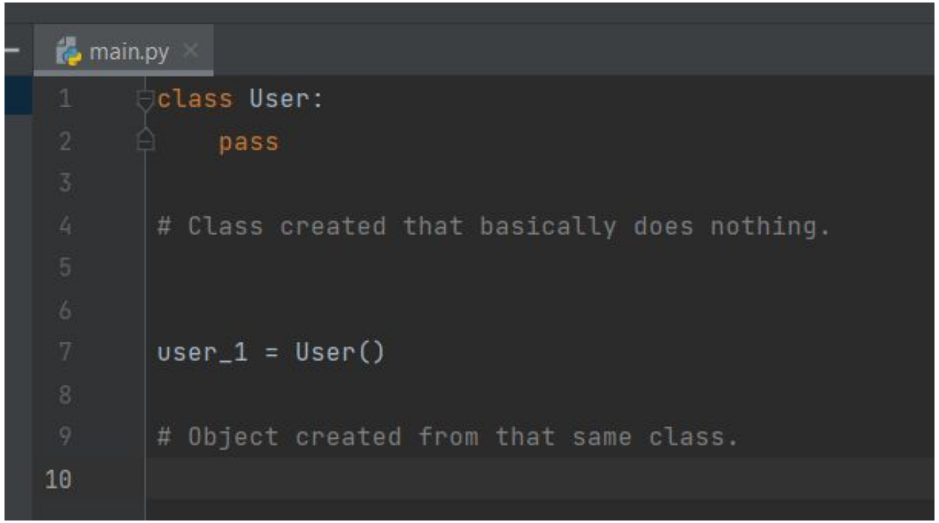
First, we need to install them into our PC.

And then we need to import them into our project.

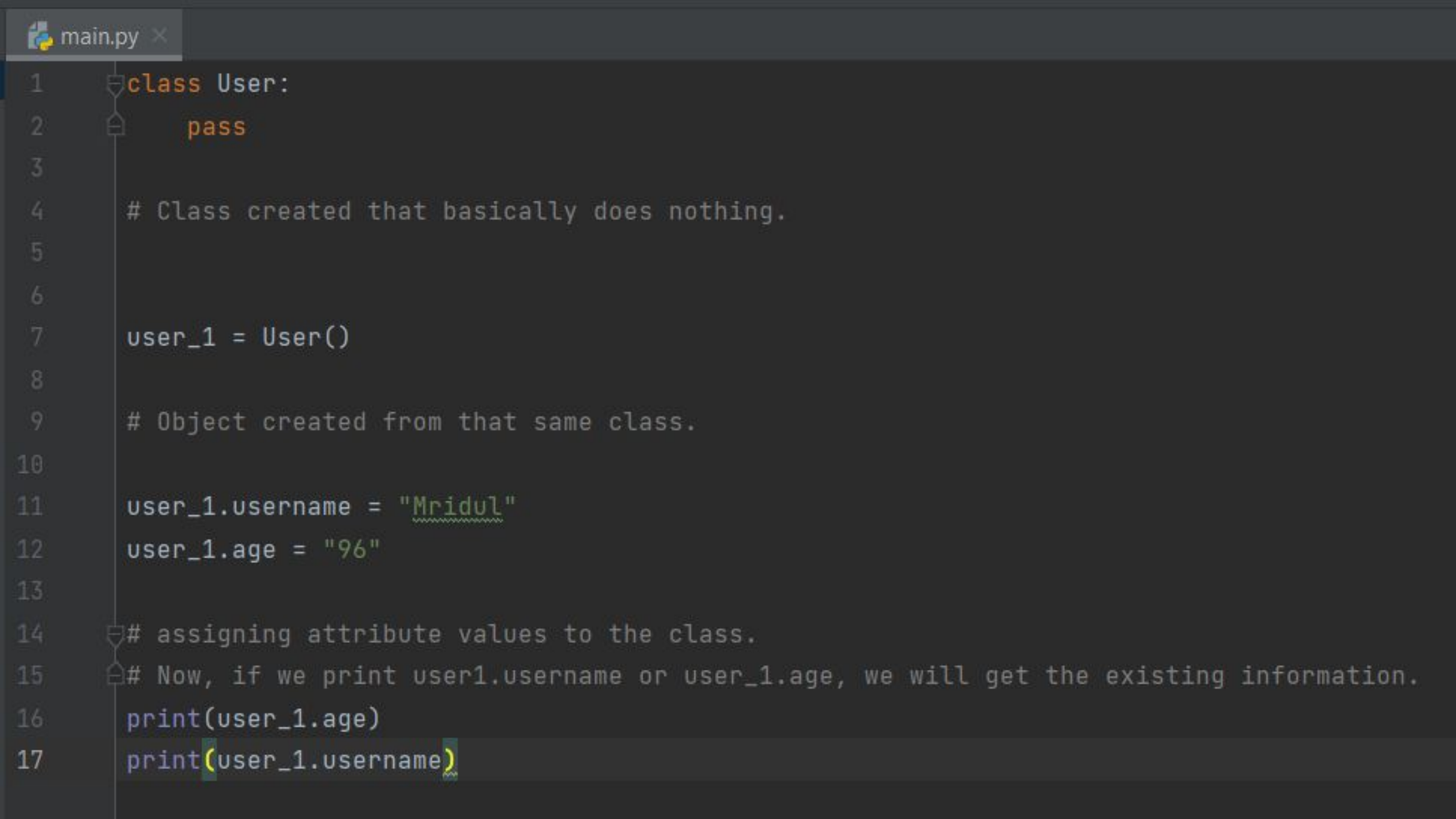
But, how can we create our own python class??

When have our own class, we can build our own objects and work with them.

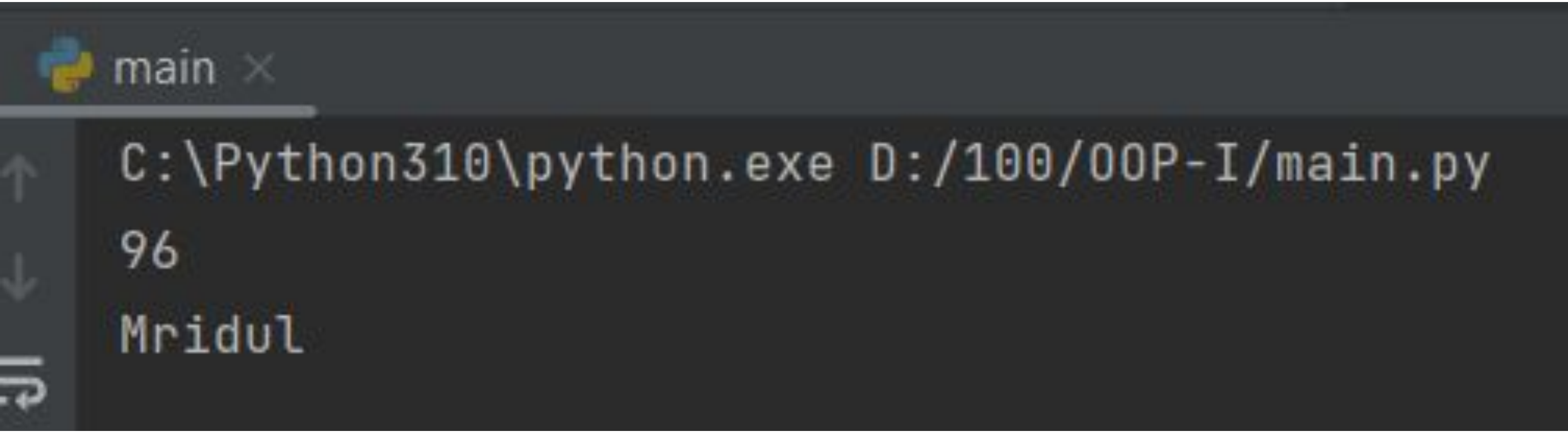
So, How do we do it?
First, we start with class keyword and give our class the name. i.e. “**class <*classname*>:**”



```python
class User:
    pass

# Class created that basically does nothing.


user_1 = User()

# Object created from that same class.

```


main.py

```python
class User:
    pass

# Class created that basically does nothing.


user_1 = User()

# Object created from that same class.

user_1.username = "Mridul"
user_1.age = "96"

# assigning attribute values to the class.
# Now, if we print user1.username or user_1.age, we will get the existing information.
print(user_1.age)
print(user_1.username)
```


main

```
C:\Python310\python.exe D:/100/OOP-I/main.py
96
Mridul
```

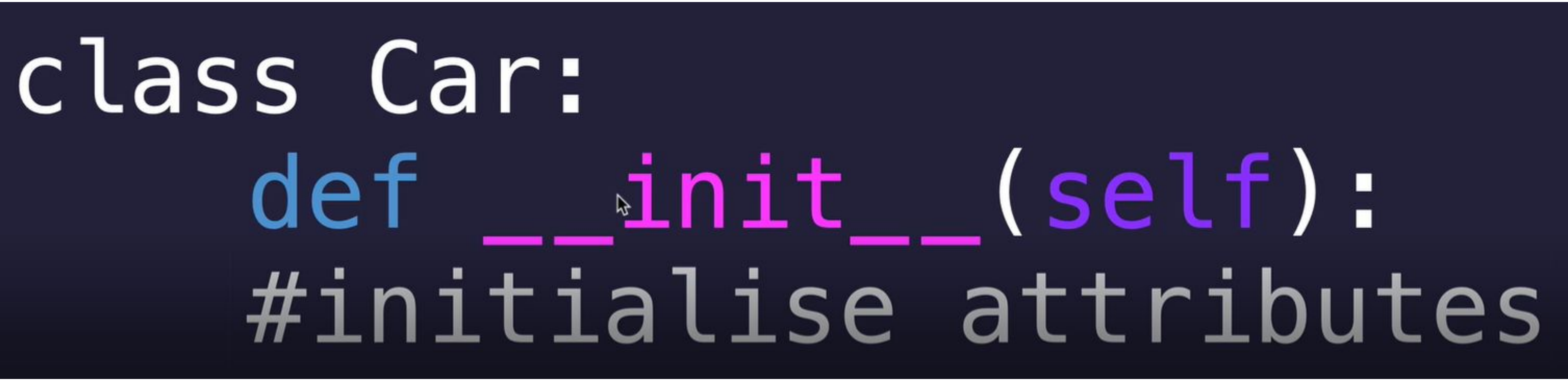
Now, the method we used previously is very error prone and there is high chance that we will make basic errors which will slow down the development process.

So, how can we minimize the error???
We can achieve that by something called Constructor.

so, what exactly is this constructor???
Constructor is basically the part of the blueprint that allows us to define what should happen when our object is being created.

In programming languages, this is also called initialization of object. i.e. While initializing the object, we can set the starting values for attributes.

In python we create constructor by using a special function called
"**__init__**"

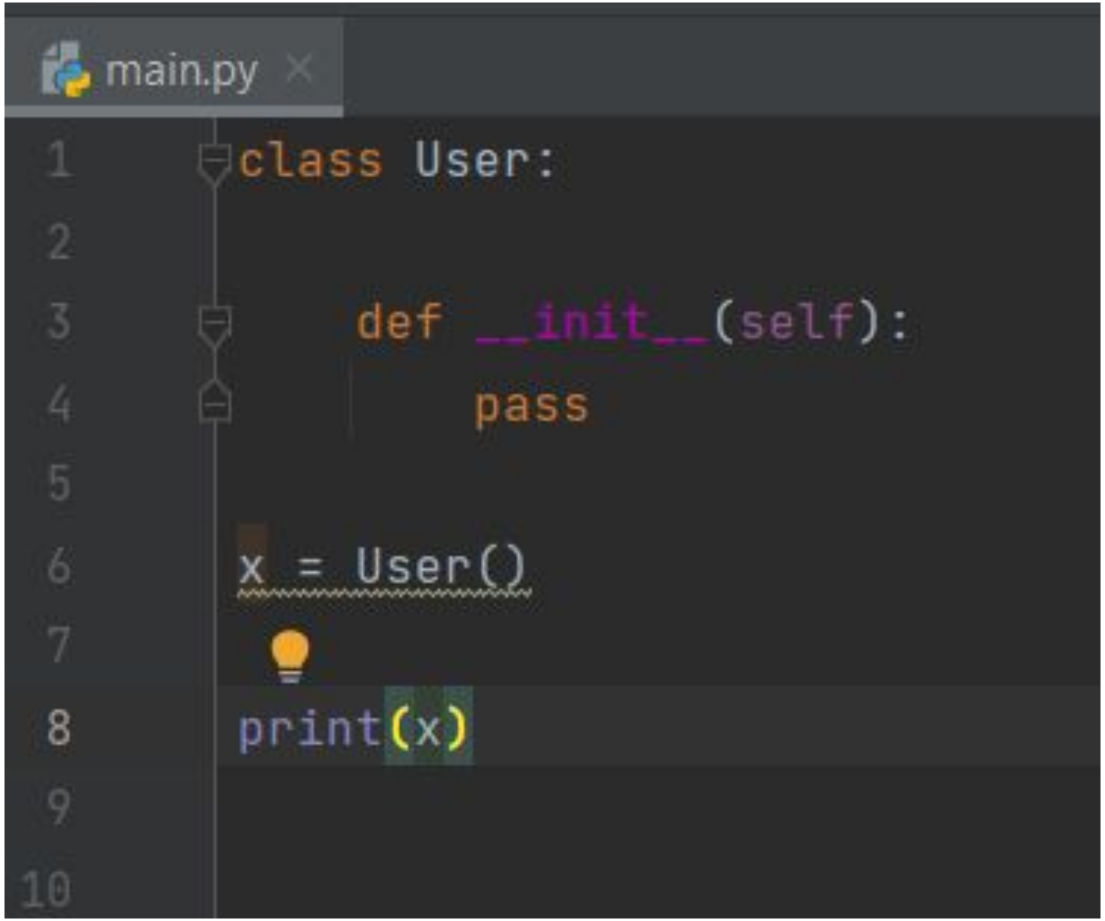


```
class Car:
    def __init__(self):
    #initialise attributes
```

Note :

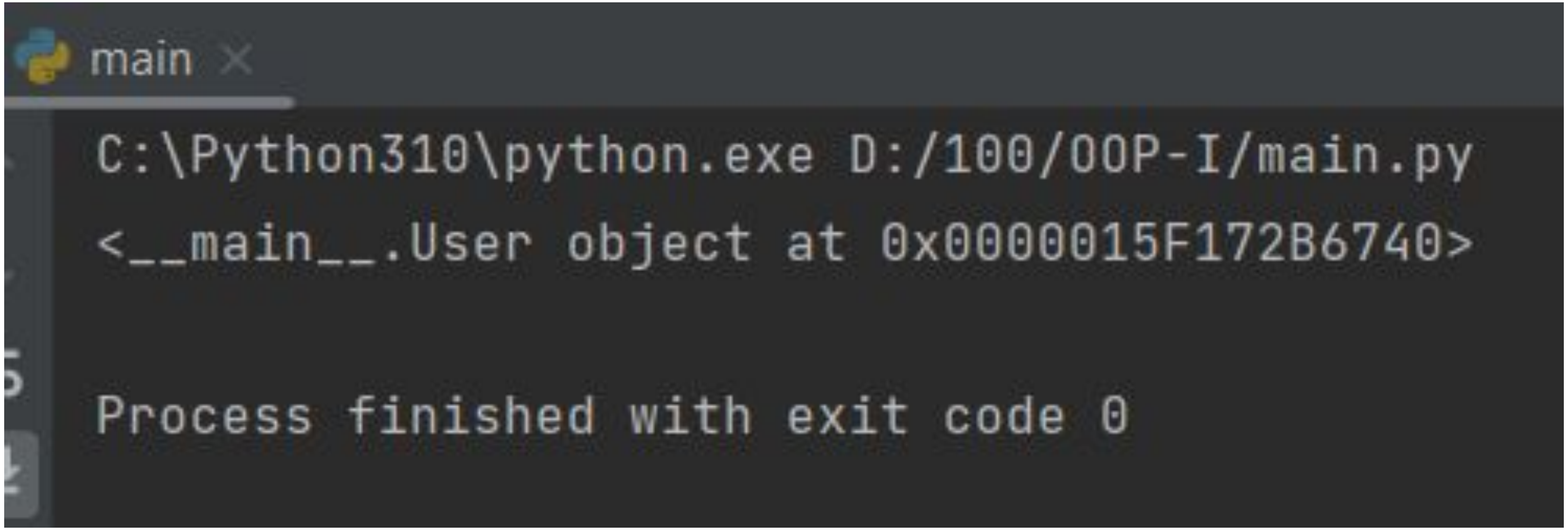
"__init__" function is going to be called everytime we call a class.

Let’s look at example of constructor:



```python
class User:

    def __init__(self):
        pass

x = User()

print(x)
```

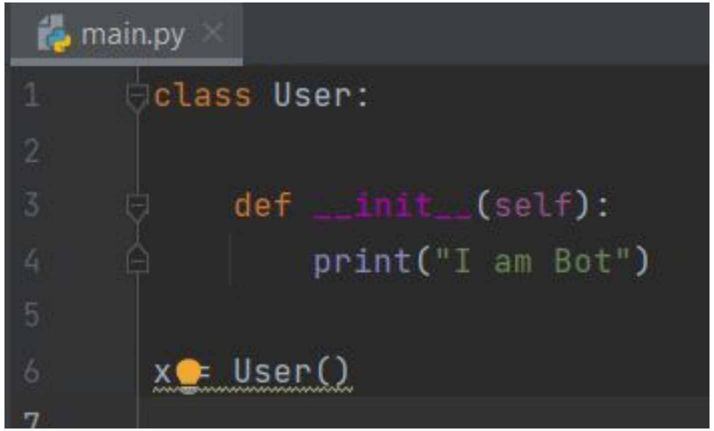
Output of previous program is :



```
C:\Python310\python.exe D:/100/OOP-I/main.py
<__main__.User object at 0x0000015F172B6740>

Process finished with exit code 0
```

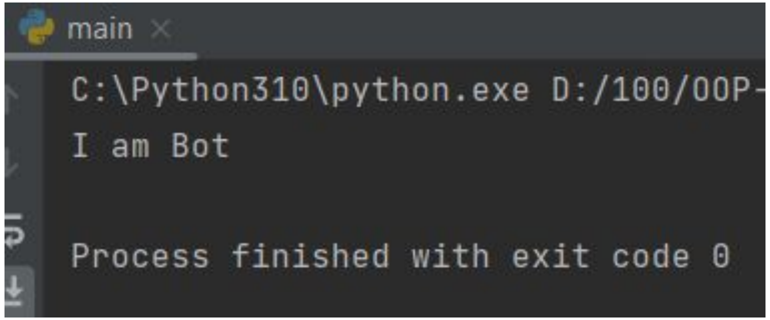
“__main__.User object at 0x0000015F172B6740” means that the object is created at that specific location from class User

Previously we have said that the constructor is executed everytime we create an object no matter what. Let's look that in practice.


main.py

```
class User:

    def __init__(self):
        print("I am Bot")

x = User()
```

Output for this is:

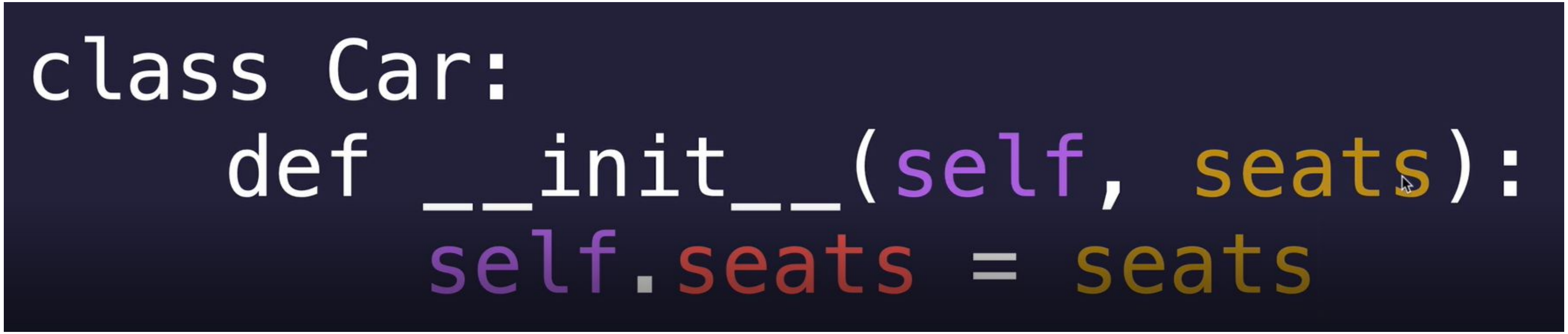

main

```
C:\Python310\python.exe D:/100/OOP-
I am Bot

Process finished with exit code 0
```

We can see that our object is created and the constructor is executed as well.

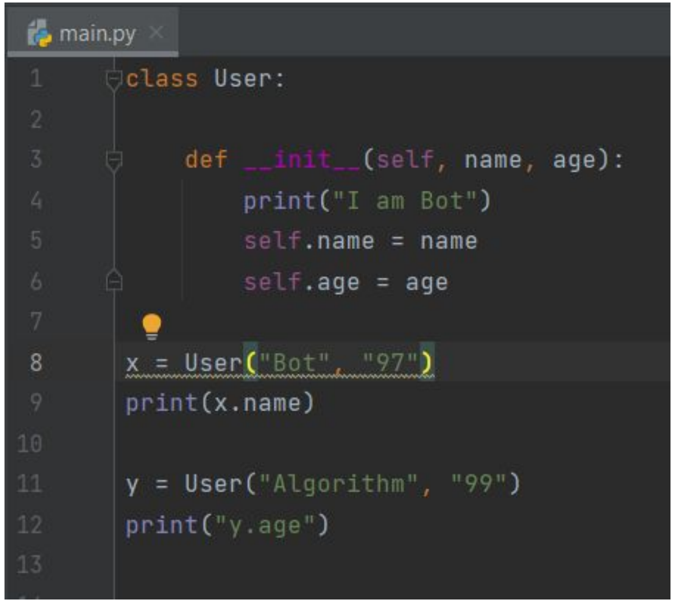
We can also make the class to take parameters while creating object so that we can create many object with different attributes while initializing an object. We can add as many parameters as per need.

We can achieve this by just adding parameters in “__init__” function and using self.<variable_name>=<parameter_name>



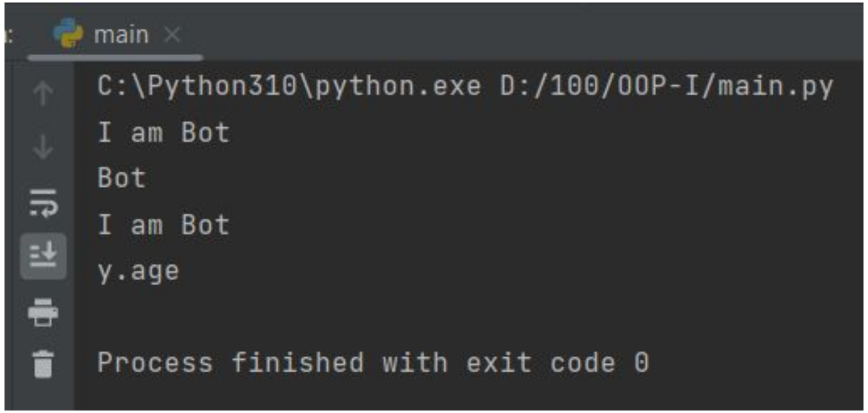
```
class Car:
    def __init__(self, seats):
        self.seats = seats
```

Let’s look at the example:



```
main.py
class User:

    def __init__(self, name, age):
        print("I am Bot")
        self.name = name
        self.age = age

x = User("Bot", "97")
print(x.name)

y = User("Algorithm", "99")
print("y.age")

```

It’s output is:



```
main
C:\Python310\python.exe D:/100/OOP-I/main.py
I am Bot
Bot
I am Bot
y.age

Process finished with exit code 0
```

We can see that the constructor is called and executed everytime we create a new object.

Thank You!!!