Python is an object oriented programming language

Almost everything in Python is an object, with its properties and methods.

A Class is like an object constructor, or a "blueprint" for creating objects.

Create a Class

class MyClass:
  x = 5
class MyClass:
This defines a class named MyClass.
Classes are used to create objects and encapsulate data and behavior.

  x = 5
This is a **class variable**. It means that any instance of MyClass (or the class itself) can access this variable.
The value of x is 5
print(MyClass)
This prints a reference to the **class itself**, not an instance of it.
The output will look something like this:
Print result <class '__main__.MyClass'>

It tells you that MyClass is a class defined in the main module.
If you want to print the value of x, you'd need to do something like:
print(MyClass.x) result 5

Create Object

```
class MyClass:
x = 5
p1 = MyClass()
print(p1.x)
```

p1 = MyClass()

This creates a **new object** (or instance) of the class MyClass and stores it in the variable p1.

The constructor (__init__) is not explicitly defined, so Python uses the default constructor.

print(p1.x)

This prints the value of x **from the object p1**.

Since x is defined at the class level and not overridden in p1, it outputs:

You're creating a class MyClass with a variable x, then creating an object p1 of that class, and accessing the class variable x through that object.

The __init__() Function

The examples above are classes and objects in their simplest form, and are not really useful in real life applications.
To understand the meaning of classes we have to understand the built-in __init__() function.
All classes have a function called __init__(), which is always executed when the class is being initiated.
Use the __init__() function to assign values to object properties, or other operations that are necessary to do when the object is being created:
Create a class named Person, use the __init__() function to assign values for name and age:
Coding

```
1. class Person:
2.   def __init__(self, name, age):
3.     self.name = name
4.     self.age = age
5. p1 = Person("Ali Raza", 36)
6. print(p1.name)
7. print(p1.age)
```

result John, 36

1. class Person:
This defines a new class named Person. A class is like a blueprint for creating objects.

2. def __init__(self, name, age):
This is the **constructor method**. It is automatically called when you create a new object from the class.

- self refers to the instance being created.
- name and age are parameters you pass when creating the person.

3. self.name = name
This line assigns the passed name to the instance variable self.name.

4. self.age = age
Similarly, this sets the instance variable self.age using the value passed to the constructor.

5. p1 = Person("Ali Raza", 36)
This creates an **object** p1 of class Person with:

- name = "Ali Raza"
- age = 36

The __init__ method is called automatically.
6. print(p1.name)
Prints the name attribute of object p1, which is "Ali Raza".
7. print(p1.age)
Prints the age attribute of object p1, which is 36.

Object Methods

Objects can also contain methods. Methods in objects are functions that belong to the object.

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def myfunc(self):
    print("Hello my name is " + self.name)

p1 = Person("John", 36)
p1.myfunc()
```

Python Inheritance

Inheritance allows us to define a class that inherits all the methods and properties from another class.

Parent class is the class being inherited from, also called base class.

Child class is the class that inherits from another class, also called derived class.

Create a Parent Class

Any class can be a parent class, so the syntax is the same as creating any other class:

Create a class named Person, with firstname and lastname properties, and a printname method:

```
class Person:
  def __init__(self, fname, lname):
    self.firstname = fname
    self.lastname = lname

  def printname(self):
    print(self.firstname, self.lastname)

#Use the Person class to create an object, and then execute the printname method:
x = Person("John", "Doe")
x.printname()
```

Create a Child Class

To create a class that inherits the functionality from another class, send the parent class as a parameter when creating the child class: