Write about the following built-in attributes in OOP

1. **\_\_dict\_\_** :

* It is a dictionary or other mapping object used to store an object’s (writable) attributes.
* if you would like for your class to use dynamic attribute creation or weak references. In those cases, you can add '\_\_dict\_\_'
* \_\_dict\_\_ is a specific dictionary that exists for each Python object, and contains the attributes of that object and their values. The double underscore is simply a Python convention for marking a variable as "special," but you're free to modify it like any other variable if you want.
* When you define a Python class the execution engine creates a *\_\_dict\_\_* to store all of its methods and any class attributes you define as parts of it.
* Normal class instantiation in Python entails creating an object with another “dunder dict” (*\_\_dict\_\_*) attribute. This then links to all of the instance’s other attributes (as initialized by the class’ *\_\_init\_\_*method, and/or by its *\_\_new\_\_*code.

1. **\_\_doc\_\_**

* **Python objects have an attribute called \_\_doc\_\_ that provides a documentation of the object. For example, you simply call Dog.\_\_doc\_\_ on your class Dog to retrieve its documentation as a string.**
* You can define the docstring using a string surrounded by triple quotes as shown in the example:

**class Dog:**

**"""Your best friend."""**

**def do\_nothing(self):**

**pass**

**print(Dog.\_\_doc\_\_)**

**# Your best friend.**

1. **\_\_name\_\_  :**

* Unlike other programming languages, python is not designed to start execution of the code from a main function explicitly. A special variable called \_\_name\_\_ provides the functionality of the main function. As it is an in-built variable in python language, we can write a program just to see the value of this variable as below.
* The \_\_name\_\_ variable (two underscores before and after) is a special Python variable. It gets its value depending on how we execute the containing script.
* When you run your script, the \_\_name\_\_ variable equals \_\_main\_\_. When you import the containing script, it will contain the name of the script.

1. **\_\_module\_\_  :**

* The \_\_module\_\_ property is intended for retrieving the module where the function was defined, either to read the source code or sometimes to re-import it in a script.

class A(object): pass

class B(A): pass

b = B()

print B.\_\_module\_\_

\_\_main\_\_

1. **\_\_bases\_\_**

* This built-in class attribute when called prints the tuple of base classes of a class object.
* The following code shows how the \_\_bases\_\_ works. B is a child class of the parent/base class A.
* class A(object): pass
* class B(A): pass
* b = B()
* print B.\_\_bases\_\_
* (<class '\_\_main\_\_.A'>,)