**Q1. What is the relationship between classes and modules?**

**Ans:** A python class is like an outline/ blueprint/ mold for creating a new object . An object is anything that we wish to manipulate or change while working through the code. Every time a class object is instantiated, which is when we declare a variable, a new object is initiated form scratch.

Whereas in python, modules are simply files with the **.py** extension containing python code that can be imported inside another python program. In simple terms, we can consider a modules to be the same as a code library or a file that we want to include in your application

**Q2. How do you make instances and classes?**

**Ans**: For creating a class instance ,we call a class by its name and pass the arguments which its **\_init\_** method accepts.

**Example:** pinki = employee(‘Feamle’,2000), here pinki is an instance of class employee with attributes ‘Female’ and 2000.

Whereas for creating a class , we use the **class** keyword. Class keyword is followed by class name and semicolon.

**Example:** here employee is a class created with class keyword with arguments gender and salary.

Class Employee: def \_init\_ (self , gender , salary ): self. Gender = gender self . salary = salary

**Q3. Where and how should be class attributes created?**

Ans: class attributes or class level attributes belong to the class itself , these attributes will be shared by all the instances of the class . Hence these attributes are usually created/ defined in the top of class definition outside all methods.

**Example:** in the below code we are defining a class attribute called no \_ of \_wheels which will be shared by all the instances of the class car

Class car : no\_ of \_ wheels = 4; # this is a class attribute

def \_init\_(self, color, price , engine) : self . color = color # all this are instance attributes self . price = price self . engine = engine

**Q4. Where and how are instance attributes created?**

**Ans:** Instances attributes are passed to the class when an object of the class is created . unlike class attributes are not shared by all objects of the class , instead not shared by all objects of the class. Instance attributes are defined within the \_init\_ method of class.

Example: in the below sample code we are creating a class car with instance variables color , price , engine , which will be provided when an instance of class car is created.

**Class car :**

**def \_init\_(self , color, price, engine): self . color = color # all this are instance attributes self. Price = price self . engine = engine**

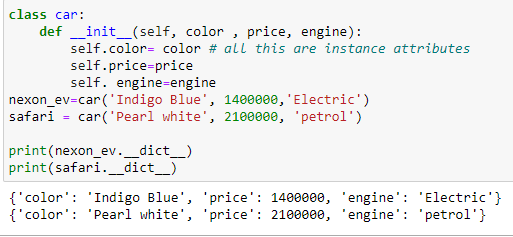
**nexon\_ev = car (‘Indigo Blue’,1400000, ‘electric’ )**

**safari = car (‘pearl white’, 2100000, ‘petrol’)**

nexon \_ ev , safari are both the instances of class car with different instance variables.

**Q5. What does the term "self" in a Python class mean?**

**Ans:self** represents the instance of the class (it represents the object itself ) . by using the “self “ keyword we can access the attributes and methods of the class with in the class in python . it binds the attributes with the given arguments.

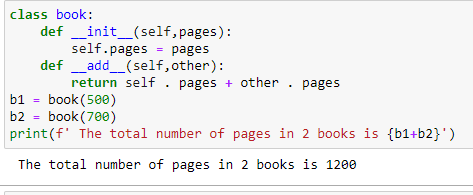


**Q6. How does a Python class handle operator overloading?**

Ans: python classes handle operator overloading by using special methods called **magic methods.** These special methods usually begin and end with \_\_(double underscore)

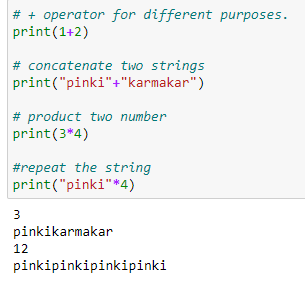
**Example**: magic methods for basic arithmetic operators are:

* + -> \_\_add\_\_( )
* - -> \_\_sub\_\_( )
* \* -> \_\_mul\_\_( )
* / -> \_\_div\_\_( )



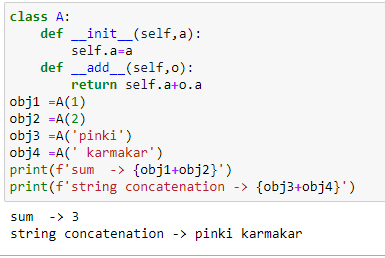
**Q7. When do you consider allowing operator overloading of your classes?**

**Ans:** We consider allowing operator overloading when we want to have different meaning for the same operator. For example operator + is used to add two integers as well as join two strings and merge two lists. It is achievable because ‘+’ operator is overloaded by int class and str class



**Q8. What is the most popular form of operator overloading?**

**Ans:** The most popular form of operator overloading in python is by special methods called magic methods. It usually begins and ends with double underscore **\_\_<method name>\_\_.**

****

**Q9. What are the two most important concepts to grasp in order to comprehend Python OOP code?**

**Ans**: **classes** and **objects** are the two most important concepts to grasp in order to comprehend python oop code as more formally objects are entities that represent instances of general abstract concept called class.

Along with classes and objects the important concepts to grasp are:

1. Inheritance
2. Abstraction
3. Polymorphism
4. Encapsulation