QUESTION#01

Object-oriented programming (OOP) refers to a type of computer programming (software design) in which programmers define the data type of a data structure, and also the types of operations (functions) that can be applied to the data structure.

In this way, the data structure becomes an object that includes both data and functions. In addition, programmers can create relationships between one object and another. For example, objects can inherit characteristics from other objects.

The Basic OOP Concepts

1-Abstraction: The process of picking out (abstracting) common features of objects and procedures.

2-Class: A category of objects. The class defines all the common properties of the different objects that belong to it.

3-Encapsulation: The process of combining elements to create a new entity. A procedure is a type of encapsulation because it combines a series of computer instructions.

4-Information hiding: The process of hiding details of an object or function. Information hiding is a powerful programming technique because it reduces complexity.

5-Inheritance: a feature that represents the "is a" relationship between different classes.

6-Interface: the languages and codes that the applications use to communicate with each other and with the hardware.

7-Messaging: Message passing is a form of communication used in parallel programming and object-oriented programming.

8-Object: a self-contained entity that consists of both data and procedures to manipulate the data.

9-Polymorphism: A programming language's ability to process objects differently depending on their data type or class.

10-Procedure: a section of a program that performs a specific task.

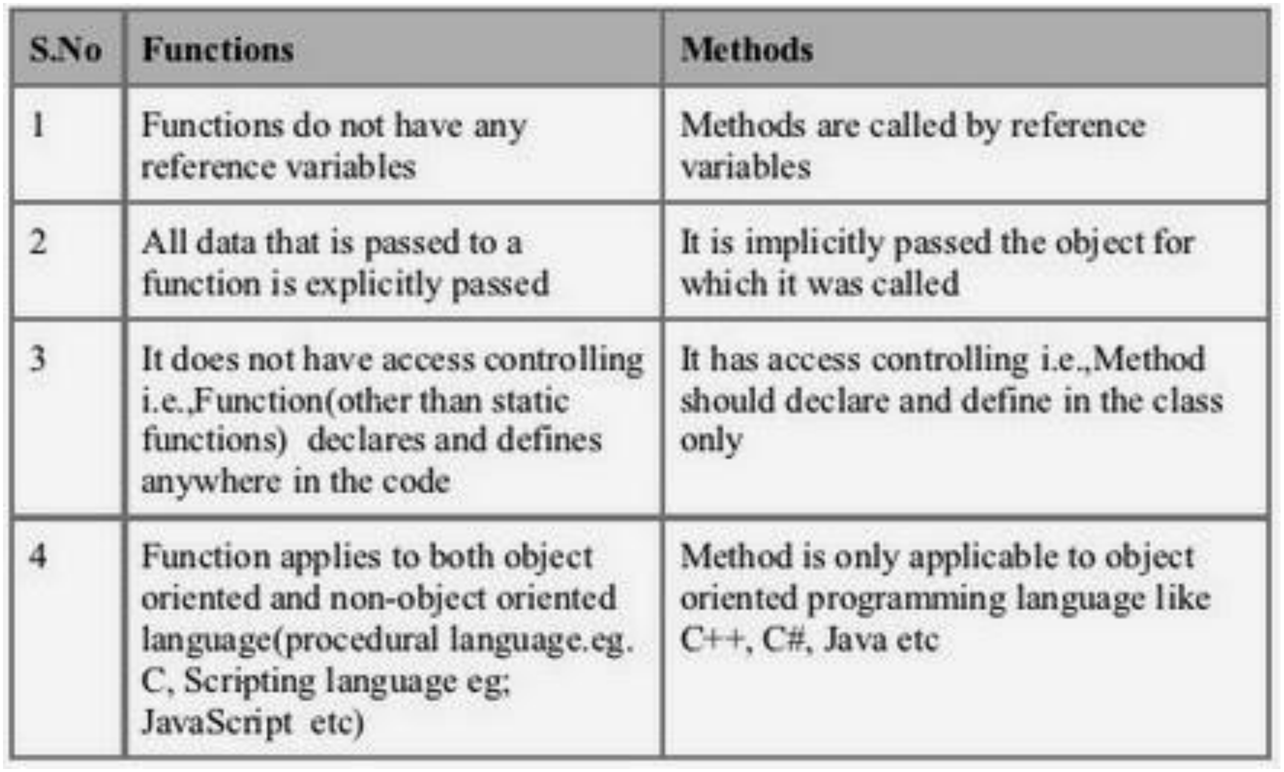
QUESTION # 02

Benefits Of OOP:

The four basic concepts of OOP are listed below:

- Modularity for easier troubleshooting. Something has gone wrong, and you have no idea where to look. ...
- Reuse of code through inheritance. ...
- Flexibility through polymorphism. ...
- Effective problem solving.

Question#03



S.No	Functions	Methods
1	Functions do not have any reference variables	Methods are called by reference variables
2	All data that is passed to a function is explicitly passed	It is implicitly passed the object for which it was called
3	It does not have access controlling i.e.,Function(other than static functions) declares and defines anywhere in the code	It has access controlling i.e.,Method should declare and define in the class only
4	Function applies to both object oriented and non-object oriented language(procedural language.eg. C, Scripting language eg; JavaScript etc)	Method is only applicable to object oriented programming language like C++, C#, Java etc

QUESTION#04

Class:

A category of objects. The class defines all the common properties of the different objects that belong to it.

Object:

A self-contained entity that consists of both data and procedures to manipulate the data.

Attribute:

Belong to the class itself they will be shared by all the instances. Such attributes are defined in the class body parts usually at the top, for legibility.

Attributes are the individual things that differentiate one object from another and determine the appearance, state, or other qualities of that object. Let's create a theoretical class called Motorcycle. A motorcycle class might include the following attributes and have these typical values:

Color: red, green, silver, brown

Style: cruiser, sport bike, standard

Make: Honda, BMW, Bultaco.

Behavior:

A class's behavior determines how an instance of that class operates; for example, how it will "react" if asked to do something by another class or object or if its internal state changes. Behavior is the only way objects can do anything to themselves or have anything done to them. For example, to go back to the theoretical `Motorcycle` class, here are some behaviors that the `Motorcycle` class might have:

- Start the engine
- Stop the engine
- Speed up
- Change gear
- Stall

Question#05

```
class Car:
    def __init__(self,name,make,model,color,price):
        self.name=name
        self.make=make
        self.model=model
        self.color=color
        self.price=price
    def start(self):
         print("Started")
    def stop(self):
        print("stopped")
    def accelerate(self):
        print("accelerating")
car_a = Car(“Honda City”, ”Honda” , ”City”, ”Black”, “14.5 lac”)
car_b= Car(“AudiQ5”,”Honda” ,”Civic”, ”Blue”, “233 lac”)
```

```
car_c= Car(“Honda Civic”, ”Honda” ,”Civic”,” Blue”, “15.5 lac”)

car_d= Car(“FordMustan”, ”Ford” , ”Mustang”, ”Grey”, “$26,260 ”)

car_e= Car(“Honda Accord”, ”Honda” , ”Accord”, ”Red”, “43.5 lac”)
```