1) Explain the importance of Functions?

Usages: Reusability , Clean Code , organised code , collaborative

3)What is the difference between print and return statements

Print statement only print the values. Whereas return statement is used return the value from the funtion and use it for other purpose;

What are \*args and \*\*kwargs ?

\*args is object which is used to take variable number of non dictionary parameter in function.

\*\*kwargs is used to take the variable number of input in dictionary format in function.

Explain the iterator function0

An object for list, tuple ,dictionary & sets. It is done by using iter method and next method, next method return the next element.

What is encapsulation in OOP?

Encapsulation is a Python technique for combining data and functions into a

single object. A class, for instance, contains all the data (methods and variables). Encapsulation refers to the

broad concealment of an object's internal representation from areas outside of its specification

Explain the use of access modifiers in Python classes?

: A class's data members and methods can be made private or protected in

order to achieve encapsulation. Direct access modifiers like public, private, and protected don't exist in Python,

though. Single and double underscores can be used to accomplish this.

Public Member : from outside of class, anywhere accessible.

Private Member : Within the class, accessible.

Protected Member : Within the class and its subclasses, accessible.

Public Member : Both inside and outside of a class, public data members are accessible. By default, the

class's member variables are all public

What is inheritance in OOP%?

Inheritence play a significant role in oop language. Inheritence in python refer to the process of a child receiving the parent class’s properties.

The reuse of code is inheritance’s main goal. Instaed of starting fron wratch when developing a new class. We can use the existing class instead of re\_creating it form scractch.

Define polymorphism in OOP?

Polymorphism in Object-Oriented Programming (OOP) refers to the ability of different objects to respond to the same function or method call in different ways. It allows objects of different classes to be treated as objects of a common super class, with each object responding in a way that is appropriate to its specific class.

Explain method overriding in Python?

Method overriding in Python is a feature that allows a subclass to provide a specific implementation of a method that is already defined in its superclass. When a method in a subclass has the same name, parameters, and return type as a method in its superclass, the method in the subclass overrides the one in the superclass.

What is abstraction in Python? How is it implemented?

Abstraction refers to hiding the Implementation part and showing the user what they are meaningful to them.

This is implemented using abstract class which can’t be instantiated. The abstract class is the blueprint for implementation.

Explain the importance of abstraction in object-oriented programming

Abstraction help in hiding the implementation part and showing the part which is meaningfull to user and make the implemntation more easier and faster.

How are abstract methods different from regular methods in Python?

Abstract Methods: Definition: Abstract methods are declared in an abstract class but do not have an implementation. They are meant to be overridden in derived classes. Instantiation: You cannot instantiate an abstract class directly. It must be inherited by a subclass that provides implementations for all abstract methods. Purpose: They define a common interface for all subclasses, ensuring that certain methods are implemented in every subclass. Regular Methods: Definition: Regular methods have an implementation and can be called on instances of the class. Instantiation: You can instantiate a class with regular methods directly. Purpose: They provide specific functionality that can be used by instances of the class.

How can you achieve abstraction using interfaces in Python?

abstraction can be achieved using abstract base classes (ABCs) provided by the abc module. An abstract base class can define abstract methods that must be implemented by any subclass.

Can you provide an example of how abstraction can be utilized to create a common interface for a group of related classes in Python

an abstract base class Vehicle with abstract methods that must be implemented by any subclass. We’ll then create specific vehicle classes like Car and Motorcycle that implement these methods.

How does Python achieve polymorphism through method overriding

Polymorphism in Python is achieved through method overriding, which allows a subclass to provide a specific implementation of a method that is already defined in its superclass. This enables dynamic behavior depending on the object’s actual type.

Define a base class with a method and a subclass that overrides the method

class Animal:

def make\_sound(self):

print("Generic animal sound")

class Dog(Animal):

def make\_sound(self):

print("Woof!")

# Creating instances

generic\_animal = Animal()

dog = Dog()

# Calling the make\_sound method

generic\_animal.make\_sound() # Output: Generic animal sound

dog.make\_sound() # Output: Woof!

Base Class (Animal): Defines a method make\_sound that prints “Generic animal sound”. Subclass (Dog): Inherits from Animal. Overrides the make\_sound method to print “Woof!”.

Dynamic Behavior: When make\_sound is called on an instance of Dog, the overridden method in the Dog class is executed. When make\_sound is called on an instance of Animal, the method in the Animal class is executed