OOPS ASSIGNMENT

Question1 . What is Object-Oriented Programming (OOP)?

Answer- accutally it is a programming paradigm that organizes software design around objects, which are data-driven entities with unique attributes and behaviors.

Question 2. What is a class in OOP?

-a class is a fundamental construct that serves as a blueprint for creating objects. It encapsulates data for the object and methods to manipulate that data, promoting modularity and code reuse

Question 3. What I object in class?

Answer- an object is an instance of a class that encapsulates data and functionality including that data.

Question 4. Abstraction and encapsulation difference.

Answer. Abstraction - Abstraction involves hiding complex implementation details and showing only the essential features of an object. It focuses on what an object does rather than how it does it

Encapsulation- Encapsulation is the bundling of data (variables) and methods that operate on that data into a single unit, typically a class. It restricts direct access to some of an object's components, which is a means of preventing unintended interference and misuse.

Question 5. What are dunder methods in Python?

Answer . Dunder methods are not typically called directly by you. Instead, Python invokes them automatically in response to certain operations.

__init__: Called when a new instance of a class is created.

__str__: Called by the str() function and by the print() function to compute the "informal" string representation of an object.

__repr__: Called by the repr() function and by backticks in Python 2 to compute the "official" string representation of an object.

__add__: Called to implement the addition operator +.

__len__: Called by the built-in len() function to return the length of the object.

__eq__: Called to implement the equality operator ==.

Questiom 6. explain the concept of inheritance in oop?

Answer.6 In object-oriented programming (OOP), inheritance is a mechanism that allows a class to inherit properties and behaviors from another class. Inheritance in oops is based on a hierarchical relationship between classes, where a derived class (also known as a subclass or child class) inherits the characteristics of a base class (also known as a superclass or parent class). The derived class extends the functionality of the base class by adding new features or overriding existing ones

Question 7. What is polymorphism in OO?

Answer. It actually allows objects of different classes to be treated as objects of a common type, enabling flexible and reusable code.

Question 8. How is encapsulation achieved in Python?

Answer. class MyClass:

```
    def __init__(self, value):
        self.value = value  # Public attribute
    def show_value(self):  # Public method
        print(self.value)
# Use case
obj = MyClass(10)
obj.show_value()  # Accessing public method
print(obj.value)  # Accessing public attribute
```

Answer will be 10, 10

Question. 9 What is a constructor in Python?

Answer. Youb can say that a special class method for creating and initializing an object instance at that class.

Question. 10 What are class ad static methods in Python?

Answer.

A static method is bound to the class and does not take self or cls as its first parameter. It behaves like a regular function but resides in the class's namespace.

# Use Cases:

- Utility functions that perform a task in isolation, without needing class or instance data.
- Organizing functions that have a logical connection to the class.

A class is bound to the class and not the instance. It takes cls as its first parameter, which refers to the class itself.

# Use Cases:

- Creating factory methods that return class instances with specific configurations.
- Modifying class-level attributes that should reflect across all instances.

Question 11- What is method overloading in Python?

Answer. Method overloading in Python provides a way to handle different types or numbers of inputs using a single method name, making your code more flexible and easier to maintain. While Python does not directly support method overloading like other languages, you can simulate it using default arguments, *args, or **kwargs.

Question 12 - What is method overriding in OOP?

Answer. is a language feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its superclasses or parent classes.

Question 13 What is a property decorator in Python?

Answer The decorator is a built-in Python decorator that allows you to turn class methods into properties in a way that's both elegant and user-friendly.

Question 14 Why is polymorphism important in OOP?

Answer . Polymorphism is a feature of object-oriented programming languages that allows a specific routine to use variables of different types at different times. Polymorphism in programming gives a program the ability to redefine methods for derived classes

Question 15. What is abstract in python?

Answer. An abstract class is like a template for other classes. It defines methods that must be included in any class that inherits from it, but it doesn't provide the actual code for those methods. Think of it as a recipe without specific ingredients—it tells you what steps to follow, but the details depend on the subclass .

Question 16 What are the advantages of OOP?

Answer. We can build the programs from standard working modules that communicate with one another, rather than having to start writing the code from scratch which leads to saving of development time and higher productivity,

- OOP language allows to break the program into the bit-sized problems that can be solved easily (one object at a time).
- The new technology promises greater programmer productivity, better quality of software and lesser maintenance cost.
- OOP systems can be easily upgraded from small to large systems.
- It is possible that multiple instances of objects co-exist without any interference,
- It is very easy to partition the work in a project based on objects.
- It is possible to map the objects in problem domain to those in the program.
- The principle of data hiding helps the programmer to build secure programs which cannot be invaded by the code in other parts of the program.
- By using inheritance, we can eliminate redundant code and extend the use of existing classes.
- Message passing techniques is used for communication between objects which makes the interface descriptions with external systems much simpler.
- The data-centered design approach enables us to capture more details of model in an implementable form.

Question 17 –

Ansswer

Instance variables are declared in a class, but outside a method, constructor or any block.	Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block.
Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.	Static variables are created when the program starts and destroyed when the program stops.
Instance variables can be accessed directly by calling the variable name inside the class. However, within static methods (when instance variables are given accessibility), they should be called using the fully qualified name. ObjectReference.VariableName.	Static variables can be accessed by calling with the class name ClassName.VariableName.
Instance variables hold values that must be referenced by more than one method, constructor or block, or essential parts of an object's state that must be present throughout the class.	There would only be one copy of each class variable per class, regardless of how many objects are created from it.

Question 18 - What is multiple inheritance in Python?

Answer. multiple inheritance allows a class to inherit attributes and methods from more than one parent class. This feature enables the creation of classes that combine functionalities from multiple sources

```
class Parent1:
    def method1(self):
        print("Method from Parent1")


class Parent2:
    def method2(self):
        print("Method from Parent2")


class Child(Parent1, Parent2):
    pass


obj = Child()
obj.method1()  # Output: Method from Parent1
obj.method2()  # Output: Method from Parent2
```

Question 19 - Explain the purpose of ‘’__str__’ and ‘__repr__’ ‘ methods in Python ?

Answer. the __str__ and __repr__ methods are special functions that define how objects are represented as strings. While they might appear similar, they serve distinct purposes and are used in different contexts.

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


    def __repr__(self):
        return f"Person(name='{self.name}', age={self.age})"


person = Person("Alice", 30)
```

```
print(repr(person))  # Output: Person(name='Alice', age=30)
```

Question 20 - What is the significance of the ‘super()’ function in Python?

Answer.

Accessing Parent Class Methods: super() enables you to call methods from a parent class, which is essential when you want to extend or customize inherited methods.

Avoiding Hardcoding Parent Class Names: By using super(), you don't need to explicitly name the parent class, making your code more maintainable and flexible, especially in complex hierarchies.

Supporting Multiple Inheritance: In scenarios involving multiple inheritance, super() ensures that the method resolution order (MRO) is followed correctly, preventing issues like the diamond problem.

Question 21- What is the significance of the __del__ method in Python?

Answer . In Python, the __del__ method is a special method known as a destructor. It's invoked when an object is about to be destroyed, allowing for the execution of cleanup actions such as releasing external resources (e.g., file handles, network connections, or database connections) that the object may hold. This method is automatically called when Python's garbage collector determines that there are no more references to the object.

Purpose of __del__

The primary purpose of the __del__ method is to define specific cleanup actions that should be taken when an object is garbage collected. This can include releasing external resources such as files or database connections associated with the object.

Question 22. What is the difference between @staticmethod and @classmethod in Python?

Answer @staticmethod

- Definition: A static method does not receive any implicit first argument (neither self nor cls).
- Usage: It behaves like a regular function but resides within the class's namespace.
- Use Cases: Utility functions that have a logical connection to the class but don't need to access class or instance data
- class MathOperations:
- `    @staticmethod`
- `    def add(a, b):`
- `        return a + b`
-
- # Usage

- result = MathOperations.add(5, 3) # Output: 8

@classmethod

- Definition: A class method receives the class (cls) as its first argument.
- Usage: It can access and modify class state that applies across all instances
- Access: Can access and modify class variables and other class methods.
- Use Cases: Factory methods, alternative constructors, or methods that need to modify class state.

```
class Person:
    species = 'Homo sapiens'

    def __init__(self, name):
        self.name = name

    @classmethod
    def create_anonymous(cls):
        return cls('Anonymous')

# Usage
anonymous_person = Person.create_anonymous()
print(anonymous_person.name)
```

Question- 23. How does polymorphism work in Python with inheritance?

Answer. Inheritance could be a feature of object-oriented programming that permits one class to acquire characteristics from another class. In other words, inheritance permits a class to be characterized in terms of another class, which makes it simpler to make and keep up an application.

Polymorphism allows objects of different types to be treated similarly. In other words, polymorphism allows objects to be treated as a single type of object, even if they are of different types. This means that a single set of code can handle any object, even if the objects are of different types.

Question 24. What is method chaining in Python OOP?

Answer. Method chaining is a powerful technique in Python programming that allows us to call multiple methods on an object in a single, continuous line of code. This approach makes the code cleaner, more readable, and often easier to maintain.

Question 25 What is the purpose of the __call__ method in Python?

Answer The __call__ method in Python enables instances of a class to be called like regular functions. When a class defines the __call__ method, it makes its instances "callable." This means you can apply the function call syntax () directly to an object of that class. The primary purpose of __call__ is to encapsulate functionality within an object, allowing for more organized and flexible code, especially when dealing with stateful operations or function-like behavior that needs to maintain context.