Certification Program in Data Analytics

Name: - **Sham Trimbakrao Mohite** Enrollment Number: **EBEON0223756169**

Topic: **Object Oriented Programming**

1. **What is OOPS? Explain its key features.**
   Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects". The object contains both data and code: Data in the form of properties (often known as attributes), and code, in the form of methods (actions object can perform).
   An object-oriented paradigm is to design the program using classes and objects. Python programming language supports different programming approaches like functional programming, modular programming. One of the popular approaches is object-oriented programming (OOP) to solve a programming problem is by creating objects
2. **What is a class, and how is it different from an object?**
   In Python, everything is an object. A **class is a blueprint for the object**. To create an object we require a model or plan or blueprint which is nothing but class.
3. **What is the purpose of an init method in a class, and how is it used?**
   In Python, the __init__ method is a special method that is called when an instance of a class is created. It is used to initialize the attributes of the object with values provided during object creation
4. **What is inheritance, and how is it implemented?**
   Inheritance is a mechanism in object-oriented programming that allows a new class to be based on an existing class, inheriting all of its attributes and methods. The existing class is called the base class, parent class or superclass, and the new class is called the derived class, child class, or subclass.
5. **What is polymorphism, and how is it implemented?**
   Object-Oriented Programming (OOP) has four essential characteristics: abstraction, encapsulation, inheritance, and polymorphism. This lesson will cover what polymorphism is and how to implement them in Python. Also, you'll learn how to implement polymorphism using function overloading, method overriding, and operator overloading. What is Polymorphism in Python? Polymorphism in Python is the ability of an object […]
6. **What is encapsulation, and how is it achieved?**
   Encapsulation is one of the fundamental concepts in object-oriented programming (OOP), including abstraction, inheritance, and polymorphism. This lesson will cover what encapsulation is and how to implement it in Python. After reading this article, you will learn: Encapsulation in Python Need for Encapsulation Data Hiding using public, protected, and private members Data Hiding vs. Encapsulation […]
7. **What is the difference between a private and a protected attribute or method in a class?**
   A private attribute or method is denoted by a double underscore prefix (__) in its name. It can only be accessed within the class and not from outside. In other words, it is hidden from the outside world.
   A protected attribute or method is denoted by a single underscore prefix (_) in its name. It is intended to be used as a convention to indicate that the attribute or method is intended for internal use within the class or its subclasses, but it is still accessible from outside the class.

8. **What is method overriding, and how is it implemented?**
   Method overriding is a feature of object-oriented programming (OOP) that allows a subclass to provide its implementation of a method that is already defined in its superclass. In other words, the subclass can change the behavior of the inherited method according to its needs.

   In Python, method overriding is implemented by defining a method in the subclass with the same name as the method in the superclass. When the method is called on an object of the subclass, the implementation in the subclass is executed instead of the implementation in the superclass.

9. **How does Python support multiple inheritance, and what are its benefits and drawbacks?**
   Multiple inheritance is a feature of object-oriented programming (OOP) that allows a class to inherit from multiple parent classes. In Python, multiple inheritance is supported by allowing a class to specify more than one parent class in its definition.
   The benefits of multiple inheritance include:

1. Reusability: Multiple inheritance allows a class to inherit behavior and attributes from multiple parent classes, which promotes code reuse and reduces code duplication.
2. Flexibility: Multiple inheritance allows a class to have multiple types of behavior and attributes, which can make it more flexible and adaptable to different situations.
3. Improved code organization: Multiple inheritance can help to organize code in a more logical and hierarchical manner, which can make it easier to understand and maintain.

   The drawbacks of multiple inheritance include:

1. Complexity: Multiple inheritance can make code more complex and difficult to understand, especially when there are multiple levels of inheritance and a large number of parent classes.
2. Name clashes: Multiple inheritance can lead to name clashes when multiple parent classes define methods or attributes with the same name. This can make it difficult to determine which method or attribute is being used.
3. Fragility: Changes to the parent classes can have unintended consequences on the child class, which can make the code more fragile and prone to errors.

10. **What are abstract classes and methods in Python OOPS, and how are they used?**
    Abstract classes and methods are a feature of object-oriented programming (OOP) that allows a class to define a template for other classes to follow. Abstract classes and methods cannot be instantiated, but they can be inherited and implemented by other classes.

    In Python, abstract classes and methods are implemented using the abc module, which stands for Abstract Base Classes. The abc module provides the ABC class and the abstractmethod decorator, which can be used to define abstract classes and methods, respectively.
11. **What is the difference between a static method and a class method, and when would you use each?**
    Both static methods and class methods are types of methods in Python that are defined inside a class. The main difference between them is how they are called and what they can access.

    A static method is a method that is bound to the class and not the instance of the class. This means that it can be called on the class itself, without needing to create an instance of the class. Static methods do not have access to the instance or class variables and can only access variables that are passed to them as arguments.

12. What are global, protected and private attributes?

i. **Global attributes:** *Global attributes are variables or data that can be accessed from any part of the program, including different modules or classes. They have a global scope and can be declared outside of any function or class.* Global attributes can be accessed and modified by any part of the program, which can make them susceptible to unintended changes and make code harder to maintain. It is generally recommended to use global attributes sparingly and only when necessary.

ii. **Protected attributes:** *Protected attributes are denoted by a single underscore (_) prefix in their name. They are intended to indicate that the attribute should be treated as "protected" or "internal" within the class or module, but they are still accessible from outside the class or module.*

iii. **Private attributes:** *Private attributes are denoted by a double underscore (__) prefix in their name. They are intended to indicate that the attribute should be treated as "private" or "internal" to the class and should not be accessed or modified from outside the class.* Private attributes are a way to encapsulate data and prevent direct access to it from other parts of the program.

13. What is the use of self in Python?

- Self-represent the current instance/ object of the class.
- Self can be used to access the variable and methods inside the same class.
- Self is the first parameter in the constructor.

14. Are access specifiers used in python?

- Yes, Access specifiers are used in the python. It is used to restrict the access of class variables and methods outside of class. This can be achieved by public, private and protected Keywords.

15. Is it possible to call parent class without its instance creation?

- Yes, it is possible. We can do by using static method.

16. How is an empty class created in python?

- In Python, you can create an empty class by using the pass statement inside the class definition. The pass statement acts as a placeholder that does nothing, allowing you to define an empty class without any attributes or methods**.**
- **Syntax : class EmptyClass:**
  pass

17. How will you check if a class is a child of another class?

- We can check if a class is a child of another class by using “ issubclass()” .

18. What is docstring in Python?

- In Python, docstring is used to provide documentation or a description of a module, function, class, or method. Docstrings serve as a way to document code, providing information about the purpose, usage, parameters, return values, and any other relevant details of the object. They are an essential part of good programming practice.

19. Is Python Object-oriented or Functional Programming?

- Python supports both object oriented and functional programming as well.
    i. **Object-oriented programming (OOP)** is a programming paradigm that organizes code around objects, which encapsulate data (attributes) and behavior (methods) together. In Python, you can define classes, create objects (instances), and leverage concepts like inheritance, encapsulation, and polymorphism to build modular and reusable code using the principles of OOP.
    ii. **Functional programming (FP)**, on the other hand, is a programming paradigm that focuses on writing code using pure functions, which have no side effects and produce the same output for the same input. FP promotes immutability, higher-order functions, and functions as first-class citizens. Python supports functional programming features such as higher-order functions, lambda functions, map, filter, and reduce functions, as well as support for immutable data types.

20. What does an object() do?

- The object() function in Python creates a new object of the base class object. The object class is the ultimate base class for all other classes in Python. It serves as a default class from which all other classes inherit.
- The object() function returns an empty object. You cannot add new properties or methods to this object. This object is the base for all classes, it holds the built-in properties and methods which are default for all classes.

21. What is the purpose of the super function in inheritance, and how is it used?

- The purpose of super function in inheritance is used to give access to methods and properties of a parent or sibling class. The super() function returns an object that represents the parent class.
- In short, Whenever you want to call parent constructor or method, we use super function.

22. What is data abstraction ?

- Data abstraction is a fundamental concept in object-oriented programming that's allows to hide unnecessary details and only exposes the necessary details. And it can be achieved by abstract class where we only declare the method, and don't define it.