**TUTORIAL 1**

1) What is the difference between procedural programming and object-oriented programming?

1. Data and Behavior: In procedural programming, data and behavior (functions) are often separate entities. Data is typically stored in data structures and operated upon by functions. In OOP, data and behavior are bundled together in objects. Objects have attributes (data) and methods (functions) that define their behavior and interaction with other objects.
2. Approach: Procedural programming focuses on writing procedures or functions that perform specific tasks or operations. The program is organized as a sequence of instructions that are executed in a linear fashion. On the other hand, object-oriented programming organizes code around objects, which are instances of classes that encapsulate data and behavior.
3. Approach: Procedural programming focuses on writing procedures or functions that perform specific tasks or operations. The program is organized as a sequence of instructions that are executed in a linear fashion. On the other hand, object-oriented programming organizes code around objects, which are instances of classes that encapsulate data and behavior.
4. Code Reusability: OOP promotes code reusability through the concept of classes and objects. Once a class is defined, multiple objects can be created from it, reducing code duplication. In procedural programming, code reuse is achieved through functions, but the level of reusability may be lower compared to OOP.
5. Inheritance and Polymorphism: OOP introduces the concepts of inheritance and polymorphism. Inheritance allows classes to inherit properties and behavior from parent classes, promoting code reuse and creating hierarchical relationships. Polymorphism enables objects of different classes to be treated as instances of a common superclass, providing flexibility and extensibility.
6. Encapsulation: OOP emphasizes encapsulation, which means bundling data and behavior together and hiding internal implementation details. Objects expose a public interface through which other objects can interact with them, while the internal workings remain hidden.

2)Define Class and Object

A class serves as a blueprint for creating individual instances of objects, providing a structure for their data and defining the operations they can perform.

Objects interact with each other by invoking methods and accessing each other's attributes. They represent the runtime entities that carry out the operations and hold the data within an object-oriented program.

3) Explain the difference between class and object using a simple example of an education institute.

this scenario, we can have a class called "Student" that represents the blueprint or template for all students in the education institute. The "Student" class would define the common attributes and behaviors that all students share. For example, the class could have attributes such as "name," "age," and "studentID" to represent the student's personal information. The class may also have methods like "enrollCourse()" or "submitAssignment()" to represent actions that students can perform.

class Student:

def \_\_init\_\_(self, name, age, studentID):

self.name = name

self.age = age

self.studentID = studentID

def enrollCourse(self, course):

# Code to enroll the student in a course

def submitAssignment(self, assignment):

# Code to submit an assignment

Each object will represent an individual student with their unique data. For instance, we can create two objects named "student1" and "student2" as instances of the "Student" class:

student1 = Student("John Doe", 20, "12345")

student2 = Student("Jane Smith", 22, "67890")

In this case, "student1" and "student2" are objects of the "Student" class. They have their own specific data, such as name, age, and student ID, which are defined by the attributes of the class. The objects can also invoke the methods defined in the class, such as enrolling in a course or submitting an assignment.

4) What are the advantages of using object-oriented concept in application development?

1. Modularity and Reusability: Object-oriented programming promotes modularity by organizing code into self-contained objects. These objects can be reused in different parts of the application or in other applications, saving development time and effort. Code reusability reduces redundancy, improves maintainability, and enhances productivity.
2. Encapsulation and Information Hiding: Object-oriented programming allows for encapsulation, which combines data and behavior within objects. Encapsulation provides information hiding, where the internal details of an object are hidden from other objects. This protects the integrity of the data and provides a clear interface for interacting with objects, improving security and code maintenance.
3. Flexibility and Extensibility: Object-oriented programming supports inheritance and polymorphism, enabling objects to inherit properties and behaviors from parent classes. Inheritance promotes code reuse and allows for the creation of hierarchical relationships between classes. Polymorphism allows objects of different classes to be treated uniformly, providing flexibility and extensibility to the codebase.
4. Maintainability and Scalability: Object-oriented programming emphasizes code organization and structure, making it easier to understand, maintain, and modify. Changes made to one part of the codebase have limited impact on other parts, improving code maintainability. Additionally, the modular nature of object-oriented programming enables scalability, as new objects and functionalities can be added without affecting existing code.

5) Explain the role and responsibilities of a Systems Analyst in SDLC?

1. Requirement gathering and analysis: System analysts interact with stakeholders, such as end-users, managers, and executives, to understand their requirements and needs. They conduct interviews, workshops, and surveys to gather information about existing processes, pain points, and desired improvements. They then analyze these requirements to identify the functionalities and features that the system should have.
2. System design and planning: System analysts contribute to the design and planning phase of a project. They create system models, diagrams, and specifications that outline the structure, components, interfaces, and functionalities of the proposed system. They collaborate with other stakeholders, such as developers and designers, to ensure the system design aligns with the business objectives and technical feasibility.
3. Feasibility assessment: System analysts evaluate the feasibility of proposed projects or system enhancements. They analyze technical, operational, economic, and scheduling factors to determine if the project is viable and worth pursuing. They assess potential risks, benefits, costs, and impacts to make recommendations on whether to proceed with the project.