Operating System

OS

Table of Contents

OOPs (Object Oriented Programming) 2

How OOPs is better? .... **Error! Bookmark not defined.**

Main features of OOPs .... **Error! Bookmark not defined.**

Basic Terminologies:- .... **Error! Bookmark not defined.**

Encapsulation .... **Error! Bookmark not defined.**

Access Modifiers in Python .... **Error! Bookmark not defined.**

Abstraction .... **Error! Bookmark not defined.**

Ways to achieve abstraction in python .... **Error! Bookmark not defined.**

Abstract Classes .... **Error! Bookmark not defined.**

Inheritance .... **Error! Bookmark not defined.**

Super() keyword .... **Error! Bookmark not defined.**

Method Overriding .... **Error! Bookmark not defined.**

Types of inheritance .... **Error! Bookmark not defined.**

Polymorphism .... **Error! Bookmark not defined.**

Method overloading .... **Error! Bookmark not defined.**

Operator overloading .... **Error! Bookmark not defined.**

Advance OOPs concept .... **Error! Bookmark not defined.**

Method Resolution Order (MRO's) .... **Error! Bookmark not defined.**

Mixins .... **Error! Bookmark not defined.**

Interview Questions 4

OS (Operating System)

An **operating system** is a system software that manages all the resources of a computer, both hardware and software. It provides interaction between users and computer hardware and is responsible for managing and controlling all the activities and sharing of computer resources, thus acting as resource manager.

Example of OS: Windows, Linux, Mac OS, Android (for mobile) etc.

Important functions of OS

Note[pb1]

1. **Resource Management**: The operating system manages and allocates memory, CPU time, and other hardware resources among the various programs and processes running on the computer.
2. **Process Management**: OS manages processes, which are instances of running programs. It allocates resources, schedules tasks, and facilitates communication between processes.
3. **Memory Management**: OS allocates and deallocates memory as needed, ensuring efficient utilization and preventing conflicts.
4. **File System Management**: It organizes and controls files on storage devices, allowing users to store, retrieve, and manipulate data.
5. **Device Management**: OS controls peripheral devices like printers, disks, and input/output devices, facilitating communication with the computer.
6. **Security**: The operating system provides a secure environment for the user, applications, and data by implementing security policies and mechanisms such as access controls and encryption.
7. **User Interface**: The operating system provides a user interface that enables users to interact with the computer system.

Components of OS

1. **Kernel**: The kernel is the core component of the operating system. It interacts directly with hardware, managing resources and providing essential services.
2. **Shell**: The shell is the interface between the user and the operating system. It interprets user commands and executes them. Examples include the Bash shell in UNIX-like systems.
3. **Device Driver**: Device drivers are software components that enable communication between the operating system and hardware devices like printers, graphics cards, and network interfaces.

4. **Utilities**: Utilities are additional tools provided by the operating system for system management and user convenience. Examples include file management tools, disk utilities, and system monitoring tools.

Types of Operating System

1. **Encapsulation**: Bundling data (attributes) and methods that operate on the data into a single unit (object) while hiding internal details.
   Eg: In a smartphone, there are various components such as the processor, memory, camera, and battery. These components are encapsulated within the device's outer shell, which serves as a protective barrier. Users interact with the smartphone through a limited set of well-defined interfaces, such as the touchscreen, buttons, and ports.

2. **Abstraction**: Abstraction is a OOPs concept which “shows” only essential attributes and “hides” unnecessary information from the outside.
   Eg: Abstraction is like using a TV remote without knowing its inner workings; you interact with its buttons (interface) to control the TV, without needing to understand the technical details inside.

3. **Inheritance**: Creating new classes by inheriting attributes and methods from existing ones, promoting code reuse and hierarchy.
   Eg: Think of a vehicle hierarchy: all vehicles share common traits like having wheels and engines. Inheritance is like having a "Vehicle" class as the base, and then creating subclasses like "Car," "Bike," and "Truck," inheriting the basic attributes from the "Vehicle" class while adding specific features for each type of vehicle.

4. **Polymorphism**: Treating different objects through a common interface, allowing flexibility and dynamic behaviour in code.
   Eg: Imagine a music player: different types of devices like phones, tablets, and laptops can all play music. Polymorphism allows you to control the music playback using the same play, pause, and stop buttons on these devices, even though the underlying mechanisms are different, making it easy to interact with different objects in a unified way.
   Eg: A person at the same time can have different characteristics. Like a man at the same time is a father, a husband, and an employee.

Interview Questions

Q1. Does multi core laptop is multi processing or threading? Or what

Classes do not consume any memory. They are just a blueprint based on which objects are created. Now when objects are created, they actually initialize the class members and methods and therefore consume memory.

Multi tasking VS Multi P