1. **What are the benefits of Databases versus scripting languages**

*Modularity for easy problem solving*

When you work with object-oriented programming languages, you know exactly where to look when something goes wrong. "Oh, did the object of the car break down? The problem must be in the car class!" You don't have to go line by line through all your code.

That's the beauty of encapsulation. Objects are autonomous, and each bit of functionality does its own thing while leaving the other bits alone. In addition, this modularity allows an IT team to work on multiple objects simultaneously while minimizing the possibility that one person can duplicate the functionality of another.

1. **¿Cuales son los beneficios de las bases de datos contra los lenguajes de script?**

*La modularidad para resolver los problemas con facilidad*

Cuando trabajas con lenguajes OOP, tu sabes exactamente donde mirar cuando las cosas van mal. “¿Oh, se rompió el objeto coche? El problema puede ser mayor en la clase coche” No tiens que ir línea por línea viendo donde está el error.

Esa es la belleza de la encapsulación. Los objetos son autónomos, y cada bit de funcionalidad hace lo suyo mientras deja los otros bits solos. Además, esta modularidad permite que un equipo de TI trabaje en múltiples objetos simultáneamente mientras minimiza la posibilidad de que una persona pueda duplicar la funcionalidad de otra.

1. **What are the benefits of object-oriented programming?**

*Reuse of code through inheritance*

Suppose that in addition to your Car object, one colleague needs a RaceCar object and another needs a Limousine object. They all build their objects separately but discover commonalities between them. In fact, each object is just a different type of car. This is where the inheritance technique saves time: Create a generic class (Car) and then define the subclasses (Race and Limousine) that will inherit the traits of the generic class.

Flexibility through polymorphism

Based on this example, you now only need a few drivers or functions, such as "driveCar", "driveRaceCar" and "DriveLimousine". RaceCarDrivers shares some traits with LimousineDrivers, but other things, such as RaceHelmets and BeverageSponsorships, are unique.

This is where the polymorphism of object-oriented programming comes into play. A single function can change shape to suit whatever class it is in.

Effective problem solving

Many people avoid learning OOP because the learning curve seems steeper than that of top-down programming. But take the time to master object-oriented programming and you'll find that it's the easiest and most intuitive approach to developing large projects.

Object-oriented programming is ultimately about taking a big problem and breaking it down into solvable parts. For each mini-issue, write a class that does what you need. And then, best of all, you can reuse those classes, which makes it even faster to solve the next problem.

1. **¿Cuales son los beneficios de OOP?**

*Reutilización de código mediante herencia*Suponga que, además de su objeto coche, un colega necesita un objeto Carrera y otro necesita un objeto Limusina. Todos construyen sus objetos por separado pero descubren puntos en común entre ellos. De hecho, cada objeto es solo un tipo diferente de automóvil. Aquí es donde la técnica de herencia ahorra tiempo: Cree una clase genérica (Coche) y luego defina las subclases (Carrera y Limusina) que heredarán los rasgos de la clase genérica.*Flexibilidad a través del polimorfismo*Basándose en este ejemplo, ahora solo necesita unos pocos controladores o funciones, como "manejoCoche", manejoCarrera" y "ManejoLimusina". CarreraConductores comparte algunos rasgos con LimusinaConductores, pero otras cosas, como CarreraCascos y Bebidas Campeonatos, son únicas.Aquí es donde entra en juego el polimorfismo de la programación orientada a objetos. Una sola función puede cambiar de forma para adaptarse a cualquier clase en la que se encuentre.

*Resolución efectiva de problemas*Muchas personas evitan aprender OOP porque la curva de aprendizaje parece más pronunciada que la de la programación de arriba hacia abajo. Pero tómese el tiempo para dominar la programación orientada a objetos y descubrirá que es el enfoque más fácil e intuitivo para desarrollar grandes proyectos.La programación orientada a objetos se trata, en última instancia, de tomar un gran problema y dividirlo en partes solucionables. Para cada miniproblema, escribe una clase que hace lo que necesita. Y luego, lo mejor de todo, puede reutilizar esas clases, lo que hace que sea aún más rápido resolver el siguiente problema.

**3.** **What are the benefits of declarative and procedural languages?**

*A) Declarative languages*

One of the main advantages of declarative programming is that maintenance can be performed without interrupting application development. Teams can innovate and focus on solutions without getting caught up in the how of what they want to achieve. In addition, declarative language tends to use short and efficient code for faster and more convenient programming.

*B) Procedural or imperative languages*

Procedural programming is best for general programming projects because the code is relatively simple and easy to implement. In addition, since the language has been around for so long, there is a vast collection of resources available. This makes it easy for programmers to improve their skills, learn new solutions and run tests. Finally, procedural language makes it easy to track the workflow and record changes.

**3. ¿Cuáles son los beneficios de los lenguajes declarativos y procedimentales o imperativos?**

*A) Lenguajes declarativos*

Una de las principales ventajas de la programación declarativa es que el mantenimiento se puede realizar sin interrumpir el desarrollo de la aplicación. Los equipos pueden innovar y centrarse en soluciones sin quedar atrapados en el cómo de lo que quieren lograr. Además, el lenguaje declarativo tiende a usar código corto y eficiente para una programación más rápida y conveniente.

*B) Lenguajes de procedimiento o imperativos*

La programación procedimental es mejor para proyectos de programación general porque el código es relativamente simple y fácil de implementar. Además, dado que el lenguaje ha existido durante tanto tiempo, hay una vasta colección de recursos disponibles. Esto hace que sea fácil para los programadores mejorar sus habilidades, aprender nuevas soluciones y ejecutar pruebas. Finalmente, el lenguaje de procedimiento facilita el seguimiento del flujo de trabajo y registra los cambios.