**What is Inversion of Control?**

IoC is a design principle wgere the control of object creation and dependency management is transferred to a framework instead of being managed manually by the developer

**Before IoC ->** Developers manually create and manage object using keyword: **new**

**With IoC ->** The Spring framework automatically created and injects objects where needed

**How Does IoC work in Spring Boot?**

Spring uses IoC container to manage objects and dependencies

Spring Container functions:

* Create and manage objects (Inversion of Control)
* Inject object’s dependencies (Dependency Injection)

**IoC and Dependency Injection (DI)**

IoC is the concept, and Dependency Injection(DI) is how it’s implemented!

* IoC: The framework controls object creation
* DI: the framework injects dependencies into objects

**Spring supports 3 types of Dependency Injection:**

* Constructor Injection --> @Autowired on a constructor
* Setter Injection -> @Autowired on a setter method

What is Spring AutoWiring

* For dependency injection, Spring can use autowiring
* Spring will look for a class that matches (by class or interface)
* Spring will inject it automatically… hence it is autowired

Autowiring Example

* Injecting a Coach implementation
* Spring will scan for @Components (any class that have annotation )
  + @Component annotation marks the class as Spring Bean and makes it a candidate for dependency Injection. It makes a bean available for dependency injection
* Any one implements the Coach interface?
* If so, Lets inject them. For example: CricketCoach

**Conclusion**

* IoC moves object creation responsibility from developers to Spring
* Spring IoC container managers objects lifecycles and dependencies
* Implemented using Dependency Injection (@Autowired)
* IoC is a concept that shifts the responsibility of object creation and lifecycle mangaememnt to framework (Spring)
* DI is a way to Implement IoC by Injection dependencies into objects instead of manually creating them

**So**

* **IoC** is a concept and **DI**  is a way to achieve it
* **Spring boot** implements **IoC** through **Dependency Injection(DI)**