**테스트의 필요성**

* **개발 과정에서의 필요성**: 개발을 하다 보면 내 코드가 잘 동작하고 있는지, 버그가 없는지 테스트하고 싶을 때가 있다.
* **테스트에 대한 고민**: "어떻게 테스트를 하면 좋을까?"라는 질문이 생길 수 있다.
* **테스트의 목적**: 코드의 신뢰성을 높이고, 버그를 사전에 방지하기 위한 것이다.

**테스트의 범위 분류**

* **테스트의 종류**:
  1. **유닛 테스트**: 코드의 작은 부분을 테스트하는 것
  2. **통합 테스트**: 서로 다른 시스템들의 상호작용을 테스트하는 것
  3. **기능 테스트**: 사용자와 어플리케이션의 상호작용을 테스트하는 것
* **테스트의 피라미드**: 유닛 테스트가 가장 많고, 통합 테스트가 중간, 기능 테스트가 가장 적은 형태로 나타난다.

**유닛 테스트의 정의**

* **정의**: 유닛 테스트는 전체 코드 중 작은 부분을 테스트하는 것이다.
* **예시**: 함수 하나하나 개별로 테스트 코드를 작성하는 것.
* **외부 리소스**: 네트워크나 데이터베이스와 같은 외부 리소스가 포함되면 유닛 테스트가 아니다.
* **특징**: 유닛 테스트는 간단하고 명확해야 하며, 입력 값에 대한 함수의 출력 값을 판단하는 것이 핵심이다.

**유닛 테스트의 중요성**

* **코드 설계 개선**: 유닛 테스트는 좋은 코드를 만들도록 도와준다.
* **안전성 보장**: 함수에 변화가 생겼을 때 안전하게 수행되는지를 보장해준다.
* **버그 예방**: 빈번히 발생하는 버그를 막는 데 뛰어난 역할을 한다.
* **문제 해결**: 코드의 일부분에 문제가 있을 경우, 유닛 테스트를 통해 쉽게 문제를 해결할 수 있다.

**통합 테스트의 정의**

* **정의**: 통합 테스트는 각각의 시스템들이 서로 어떻게 상호작용하고 제대로 작동하는지를 테스트하는 것이다.
* **유닛 테스트와의 차이**: 유닛 테스트는 독립적이지만, 통합 테스트는 다른 컴포넌트와의 상호작용을 포함한다.
* **예시**: 데이터베이스와의 실제 통신을 포함하는 테스트.
* **필요성**: 유닛 테스트만으로 충분하지 않을 때 사용된다.

**통합 테스트의 필요성**

* **상호작용 검증**: 두 개의 다른 시스템이 잘 통신하고 있는지를 증명하고 싶을 때 사용된다.
* **복잡성**: 통합 테스트는 유닛 테스트보다 복잡하고 시간이 더 걸린다.
* **설정 과정**: 데이터베이스 설정, 설정 파일 읽기 등의 과정이 포함된다.
* **우선순위**: 통합 테스트가 꼭 필요하지 않으면 유닛 테스트에 집중하는 것이 좋다.

**기능 테스트의 정의**

* **정의**: 기능 테스트는 E2E 테스트(E2E Test) 또는 브라우저 테스트(Browser Test)라고 불리며, 어플리케이션의 완전한 기능을 테스트하는 것이다.
* **예시**: 웹 어플리케이션에서 특정 페이지를 클릭하는 자동화 테스트.
* **유닛 테스트와의 관계**: 유닛 테스트는 개별 함수의 동작을 확인하고, 통합 테스트는 시스템 간의 상호작용을 확인한다.
* **레벨 차이**: 기능 테스트는 유닛 테스트와는 완전히 다른 레벨에 있다.

**기능 테스트의 복잡성**

* **작성 난이도**: 기능 테스트는 작성하기 매우 어렵고 높은 복잡성을 가진다.
* **시간 소요**: 실제 사용자 상호작용을 시뮬레이션하므로 많은 시간이 걸린다.
* **세밀한 나눔**: 기능 테스트를 너무 세밀하게 나누는 것은 좋지 않다.
* **사용자 상호작용**: 기능 테스트는 사용자와 앱의 상호작용을 테스트하는 데 유용하다.

**테스트의 요약**

* **테스트 중심 개발**: 테스트 중심 개발(test-driven development)에서는 유닛 테스트를 꼭 작성해야 한다.
* **코드 디자인 개선**: 유닛 테스트는 코드의 디자인을 개선해준다.
* **리팩토링 지원**: 나중에 코드의 리팩토링이 필요할 때, 어떤 부분을 완전히 분리할 필요 없이 깔끔하게 해준다.
* **종합적 이해**: 각 테스트의 종류와 필요성을 이해함으로써, 개발 과정에서의 테스트의 중요성을 인식할 수 있다.