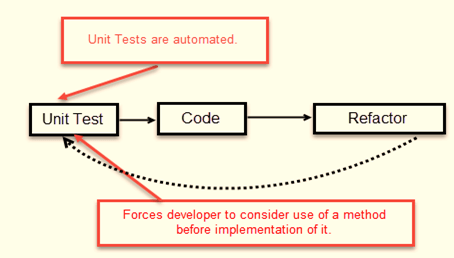
**Test Driven Development (TDD)**

Test Driven Development (TDD) is a programming practice that instructs developers to write new code only if an automated test has failed. This avoids duplication of code. The primary goal of TDD is to make the code clearer, simple and bug free.

### What is Test Driven Development (TDD)?

Test-Driven Development starts with designing and developing tests for every small functionality of an application. In TDD approach, first the test is developed which specifies and validates what the code will do.

In normal Testing process, we first generate the code and then test [To know more about software testing refer [Software Testing](http://www.guru99.com/software-testing.html) help]. Tests might fail since tests are developed even before the development. In order to pass the test, the development team has to develop and refactors the code. Refactoring a code means changing some code without affecting its behavior.

[](http://cdn.guru99.com/images/8-2016/081216_0811_TestDrivenD1.png)

The simple concept of TDD is to write and correct the failed tests before writing new code (before development). This helps to avoid duplication of code as we write a small amount of code at a time in order to pass tests. (Tests are nothing but requirement conditions that we need to test to fulfill them).

## **Why test-driven development?**

When creating a new application, at first you may not need tests. Tests can be difficult to write at first and they take time, but they can save an enormous amount of manual troubleshooting time.

As your application grows, it becomes more difficult to grow and to refactor your code. There’s always the risk that a change in one part of your application will break another part. A good collection of automated tests that go along with an application can verify that changes you make to one part of the software do not break another.

### Benefits of TDD

* **Early bug notification.**

Developers tests their code but in the database world, this often consists of manual tests or one-off scripts. Using TDD you build up, over time, a suite of automated tests that you and any other developer can rerun at will.

* **Better Designed, cleaner and more extensible code.**
  + It helps to understand how the code will be used and how it interacts with other modules.
  + It results in better design decision and more maintainable code.
  + TDD allows writing smaller code having single responsibility rather than monolithic procedures with multiple responsibilities. This makes the code simpler to understand.
  + TDD also forces to write only production code to pass tests based on user requirements.
* **Confidence to Refactor**
  + If you refactor code, there can be possibilities of breaks in the code. So having a set of automated tests you can fix those breaks before release. Proper warning will be given if breaks found when automated tests are used.
  + Using TDD, should results in faster, more extensible code with fewer bugs that can be updated with minimal risks.
* **Good for teamwork**

In the absence of any team member, other team member can easily pick up and work on the code. It also aids knowledge sharing, thereby making the team more effective overall.

* **Good for Developers**

Though developers have to spend more time in writing TDD test cases, it takes a lot less time for debugging and developing new features. You will write cleaner, less complicated code.

**Summary:**

* Test driven development is a process of modifying code in order to pass a test designed previously.
* It more emphasis on production code rather than test case design.
* It is sometimes known as **"Test First Development."**
* TDD includes refactoring a code i.e. changing/adding some amount of code to the existing code without affecting the behavior of the code.
* TDD when used, the code becomes clearer and simple to understand.

**Note: Gives error when module is not imported in the file.**

# Testing tools[¶](https://googleweblight.com/?lite_url=https://docs.djangoproject.com/en/1.7/topics/testing/tools/%23module-django.test.client&lc=en-IN&s=1&m=404&host=www.google.co.in&ts=1481784036&sig=AF9Nedkk0B0KoUyKNYGE-mbHNUy0-e9rPA#a-testing-tools)

Django provides a small set of tools that come in handy when writing tests.

## **The test client**[¶](https://googleweblight.com/?lite_url=https://docs.djangoproject.com/en/1.7/topics/testing/tools/%23module-django.test.client&lc=en-IN&s=1&m=404&host=www.google.co.in&ts=1481784036&sig=AF9Nedkk0B0KoUyKNYGE-mbHNUy0-e9rPA#a-the-test-client)

The test client is a Python class that acts as a dummy Web browser, allowing you to test your views and interact with your Django-powered application programmatically.

Some of the things you can do with the test client are:

* Simulate GET and POST requests on a URL and observe the response – everything from low-level HTTP (result headers and status codes) to page content.
* See the chain of redirects (if any) and check the URL and status code at each step.
* Test that a given request is rendered by a given Django template, with a template context that contains certain values.