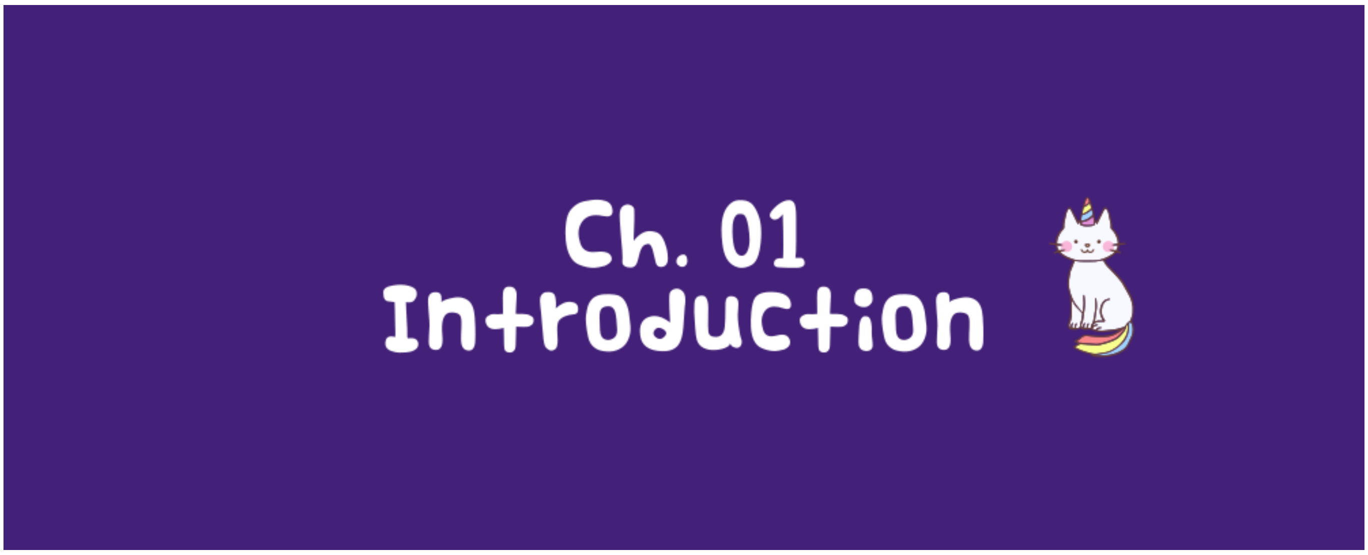
# Chapter 01



## Requirements

Before starting the course make sure you have all the pre-requisites -

```
1. Python 3.6 or above
2. Git
3. Any code editor, preferably VS Code
4. For Windows - Please install Git Bash that ships along with Git
```

## 01 - Introduction to Course

The course is divided into chapters, each chapter is built-up gradually on previous content, however we have tried to keep them as standalone as possible.

The recommended flow of approaching the course is to follow all the chapters sequentially, you can skip a chapter only if you are well versed with the topic.

### Github Repo -

Before starting the course please download all the content from our Github Repo -

Github Repo

The github repo contains all the starter and final files wherever required.

### Notes -

Each chapter is accompanied with notes in pdf format, you can reference these notes along with the videos.

## 02 - Introduction to Django

Django is a python based web framework which provides high level access to various components for rapid web development. In short, with django you can easily and rapidly create complex web applications! Django is known for its scalablity, stability, security and ease of development.

Django is a batteries included framework, it has everything built-in to create complete applications, we will explore these built-in components as we progress with the course.

Now, since you know what is Django, it would make sense to know what it is not -

- If you come from Wordpress or Drupal, they are Content Management Systems (CMS), while Django is a framework, you can create applications such as Wordpress using Django.
- Django vs Node JS, lot of people try to compare these two, however the right comparison would be Django vs Express JS, Node is not a framework, rather a javascript runtime based on Chrome V8 engine.

Django is based on shared nothing architecture, which helps create highly scalable apps. Django is also a great choice for handling relational databases, it ships with ORM (Object Relation Mapping) which can interface between your application and database.

## 03 - MVC Pattern

MVC (Model-View-Controller) is a very popular pattern for developing applications, Django makes it easy to follow MVC architecture. MVC allows for separation of layers, however there is a small difference.

Django itself is a 'Controller', so Django is essentially a MVT (Model-View-Template) architecture.

Don't worry if all this doesn't make sense, its a lot of stuff to absorb, eventually everything will be clear as we start developing.

## 04 - Virtual Environment

Scenario No 1 -

Consider you are working on a Django project and are using Django 2.2, but now you need to handle development for an older project which is using version 1.11, how would you handle it on your local machine ?

Scenario No 2 -

Often development takes place across teams, how can you ensure everyone are using the same Django version ? In addition to this, suppose you have some dependencies installed, how do you manage consistencies across all the teams ?

The answer to all these issues is 'Virtual Environments', virtual environment creates an isolated environment where you can install the correct dependencies relevant to your project. Each time while creating a project you can simply create a virtual environment and start your development process.

Python ships with two packages -

1. Virtualenv - For creating virtual environment.
2. pip - For installing dependencies.

While you can use both of them separately for creating and managing your virtual environment, the official recommended way is by using 'Pipenv'. Pipenv integrates virtualenv and pip, so that you no longer need to use them separately.

Since, Pipenv is the recommended method, we will be using it for creating all our virtual environments. Creating virtual environment using Pipenv is very simple, but first make sure you have Pipenv installed.

### Pipenv Installation -

Open your terminal and type (Window users please use Git Bash) -

For Mac/Linux with brew installed -

```
$ brew install pipenv
```

or without brew including for Windows -

```
$ pip install pipenv
```

### Create Virtual Environment -

Using the terminal navigate to the folder where you want to install virtual environment

```
$ pipenv install
```

To activate virtual environment

```
$ pipenv shell
```

To de-activate

```
$ deactivate
```

To remove

```
$ pipenv --rm
```

# 05 - Installing Django

## Installation

We will first create a virtual environment and then install django

Steps -

1. Install virtual environment

   ```
   $ pipenv install
   ```

2. Activate shell

   ```
   $ pipenv shell
   ```

3. Install Django

   ```
   $ pipenv install django
   ```

Pipenv fetches the latest release and installs it in the virtual environment.

**OR**

You can create virtual environment and install django at the same time

```
$ pipenv install django
$ pipenv shell
```

## Checking Django installation

Make sure you have installed django and activated your shell

```
$ pipenv install django
$ pipenv shell
$ django-admin version
```