Installing Django + Setup

pip install django

Creating a project

django-admin startproject projectName

Starting a server

python manage.py runserver

Django MVT

Django follows MVT(Model, View, Template) architecture.

Sample Django Model

The model represents the schema of the database.

```
from django.db import models

class Product(models.Model):   //Product is the name of our model
product_id=models.AutoField
```

Sample views.py

View decides what data gets delivered to the template.

```
from django.http import HttpResponse
def index(request):
return HttpResponse("Django Cheatsheet")
```

Sample HTML Template

A sample .html file that contains HTML, CSS and Javascript.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>Cheat Sheet</title>
</head>
<body>
<h1>This is a sample template file.</h1>
</body>
</html>
```

Views in Django

Sample Function-Based Views

A python function that takes a web request and returns a web response.

```
from django.http import HttpResponse
def index(request):
return HttpResponse("This is a function based view.')
```

Sample Class-Based Views

Django's class-based views provide an object-oriented (OO) way of organizing your view code.

```
from django.views import View

class SimpleClassBasedView(View):
def get(self, request):
... # code to process a GET request
```

URLs in Django

set of URL patterns to be matched against the requested URL.

Sample urls.py file

```
from django.contrib import admin
from django.urls import path
from . import views


urlpatterns = [
path('admin/', admin.site.urls),
path('', views.index, name='index'),
path('about/', views.about, name='about'),

]
```

Forms in Django

Similar to HTML forms but are created by Django using the form field.

Sample Django form

```
from django import forms

# creating a form
class SampleForm(forms.Form):
Name = forms.CharField()
description = forms.CharField()
```

Apps in Django

Apps in Django are like independent modules for different functionalities.

Creating an app

```
python manage.py startapp AppName
```

Listing app in the settings.py

After creating an app, we need to list the app name in INSTALLED_APPS

```
INSTALLED_APPS = [
  'django.contrib.admin',
  'django.contrib.auth',
  'django.contrib.contenttypes',
  'django.contrib.sessions',
  'django.contrib.messages',
  'django.contrib.staticfiles',
  'AppName'
]
```

Templates in Django

Used to handle dynamic HTML files separately.

Configuring templates in settings.py

```
TEMPLATES = [
{
'BACKEND': 'django.template.backends.django.DjangoTemplates',
'DIRS': ["templates"],
'APP_DIRS': True,
'OPTIONS': {
# ... some options here ...
},
},
]
```

Changing the views.py file

A view is associated with every template file. This view is responsible for displaying the content from the template.

```
def index(request):
return render(request, 'index.html') ; #render is used to return the template
```

Sample template file

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Template is working</title>
</head>
<body>
<h1>This is a sample django template.</h1>
</body>
</html>
```

Migrations in Django

Migrations are Django's way of updating the database schema according to the changes that you make to your models.

Creating a migration

The below command is used to make migration but no changes are made to the actual database.

python manage.py makemigrations

Applying the migration

The below command is used to apply the changes to the actual database.

python manage.py migrate

Admin interface in Django

Django comes with a ready-to-use admin interface.

Creating the admin user

python manage.py createsuperuser

Page Redirection

Redirection is used to redirect the user to a specific page of the application on the occurrence of an event.

Redirect method

```
from django.shortcuts import render, redirect

def redirecting(request):
return redirect("https://www.google.com")
```