Django 101 : Basic Training

EDS Technical Data TDE Initiative

Django 101

01 The Overture
02 The Setup
03 The Basics
04 Docs & References
05 The Assignment

EFTECH

Django 101

The Overture

Prerequisites

A checklist of things you need to join this session:

- A laptop (preferably procured by EDS)
- Basic **Python** knowledge
- Basic **HTML & CSS** experience
- **Python installed** on your system
- Basic **CLI** experience (cmd, Bash, Terminal, PowerShell, etc)
- Acceptable movie knowledge of **Django: Unchained**


Django.
The **D** is **silent**.
Shall we start?

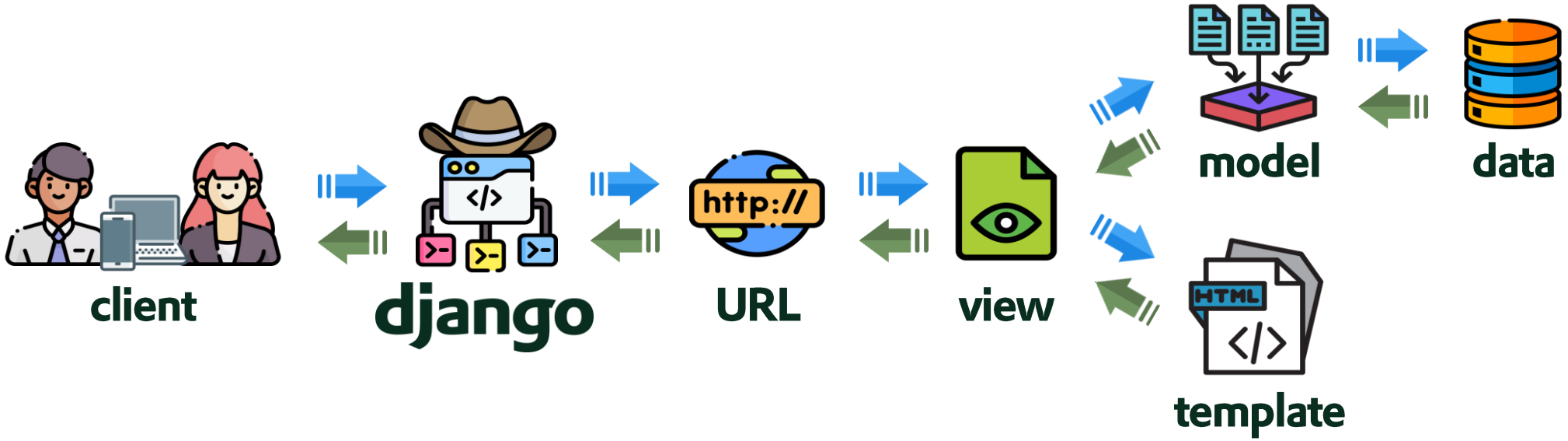
Django Web Framework

- **What is Django?** - A high-level **Python web framework** that encourages rapid development & clean, practical design.
- A proponent of **DRY** (Don't Repeat Yourself)
- It has **built-in functionalities** so that developers can **focus on writing apps** without reinventing the wheel.
- Django is **FREE** & **open source**!

Among the global brands that leverage on Django to build powerful apps.

Django MVT

- The **Model-View-Template (MVT)** is a software design pattern used by Django.
- Although Django follows the **Model-View-Controller (MVC)** pattern, it maintains its own conventions where control is handled by the **framework** itself.
- 3 important components:
  - **Model:** manages the dynamic data structure, query & logic.
  - **Template:** presentation layer which renders the response.
  - **View:** executes the business logic & interact with a **Model** to carry data & render a **Template**.



Django MVT Control Flow

1. The **Client** sends a **URL request** for a resource to **Django**.
2. Then, **Django** searches for the **URL** resource & if the **URL** path matches a **View**, then that specific **View** is called.
3. Next, the **View** will then interact with the **Model** & retrieve the relevant **Data** from the DB.
4. Finally, the **View** then renders back a corresponding **Template** carrying the retrieved data to the **Client**.

EFTECH

Django 101

The Setup

What Do You Need?

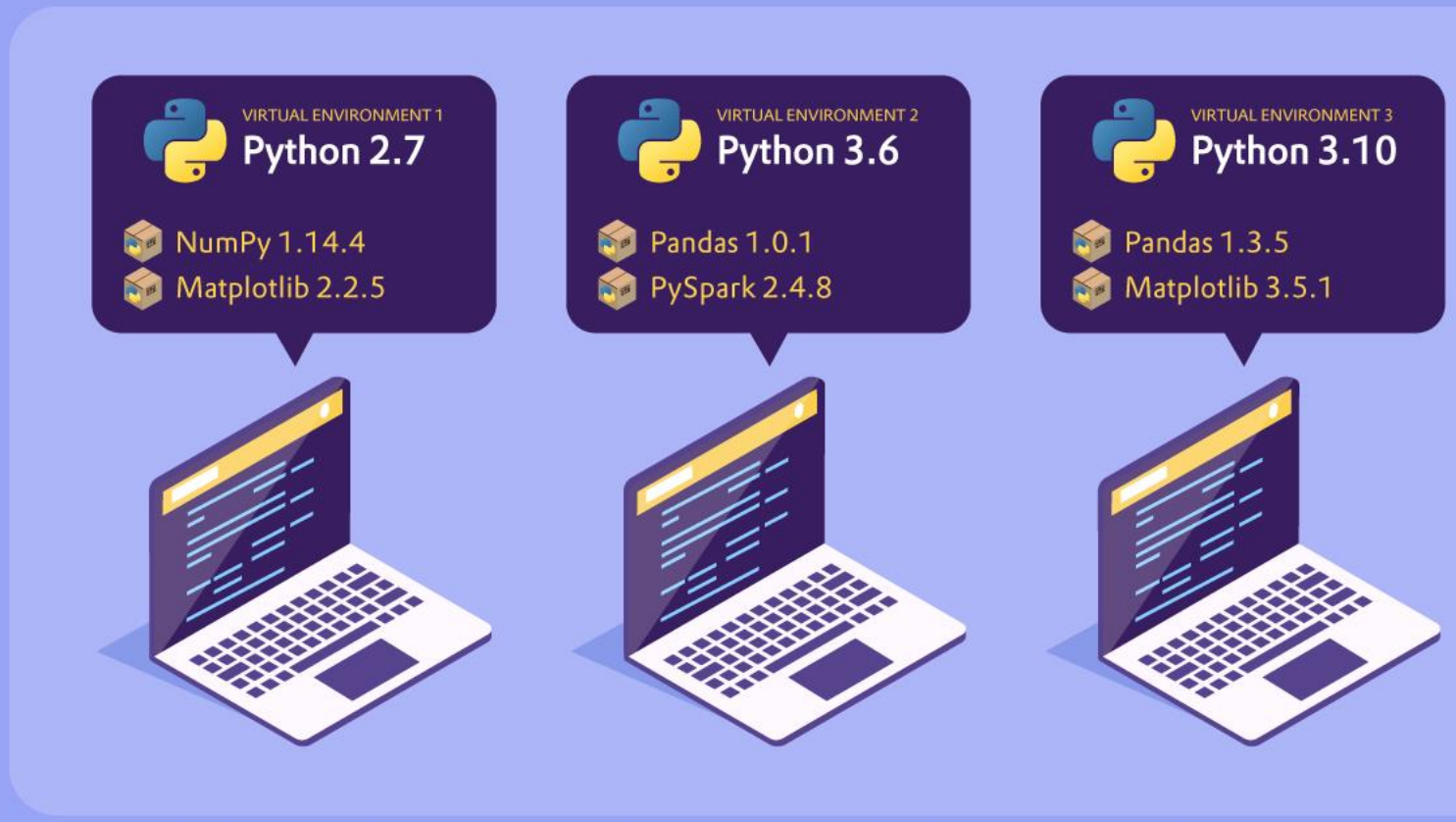
- **Python** & **pip**
- Virtual Environment Setup Tools:
  virtualenv (crucial) & **virtualenvwrapper** (optional)
- **Django**
- **CLI** - cmd, PowerShell, Bash, GitBash, fish, etc
- **Code Editor** or **IDE**
- A **Browser**: Chrome, Firefox, Opera, Safari, etc

Install Python

- Go to : https://www.python.org/downloads/
- Read up :
  - Python : https://www.w3schools.com/python/default.asp
  - Package Installer (PIP) : https://www.w3schools.com/python/python_pip.asp
- Check installation using CMD :

  python --version
- Upgrade pip :

  python –m pip install --upgrade pip
- Now we can proceed to install **virtualenv** to setup our projects.



Python
Virtual Environments

Create Virtual Environment

- For each Django project, we will **create a virtual environment** to install Django itself & other required libraries.
- This is a good practice as the **virtual environment** can be a **sandbox** that you can **play around** with to install/remove packages without breaking anything.
- We will be using the **virtualenv** package for this purpose:

  pip install virtualenv
- To create a new **virtual environment**, cd to your chosen path & type:

  virtualenv <virtual-environment-name>

  virtualenv <virtual-environment-name> -p <python-version>

  e.g. **virtualenv django_env**

  e.g. **virtualenv django_env –p python3.9**

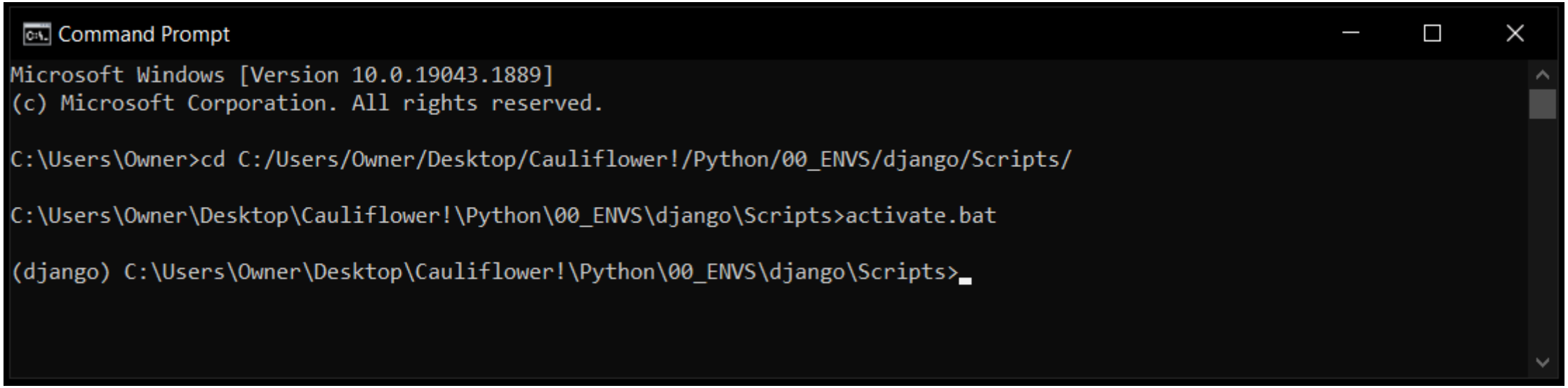
Activate Virtual Environment

- To activate the **venv**, run following command (system-based):

Platform	Shell	Command to activate virtual environment
POSIX	bash/zsh	$ source <venv>/bin/activate
	fish	$ source <venv>/bin /activate.fish
	csh/tcsh	$ source <venv>/bin /activate.csh
	PowerShell Core	$ <venv>/bin/Activate.ps1
Windows	cmd.exe	C:\> <venv>\Scripts \activate.bat
	PowerShell	PS C:\> <venv>\Scripts \Activate.ps1

- For this training, because our developers are supplied with **Windows** machines, we assume that the venv activation command will follow the convention for the **Windows** platform.
- Read up: **virtualenvwrapper** & **virtualenvwrapper-win**

Navigate to your folder in Windows Explorer, enter **cmd** in the location bar & press **ENTER** **OR** from **CMD**, change directory (cd) to your folder as follows:



As illustrated here, once a virtual environment is **activated**, the environment name (**django** in this example) will appear in the command line as a **prefix** to the current directory.

Install Django

- Individual Installation
  - Activate venv and type:

    pip install django
- Auto:

  (This option copies all dependencies from an existing venv to a text file, which can be used in another environment)
  - **Freeze installed packages for easier future setup**, cd to path:

    pip freeze > requirements.txt
  - Activate the your **venv** and install **packages via** requirements:

    pip install -r requirements.txt

Visual Studio Code

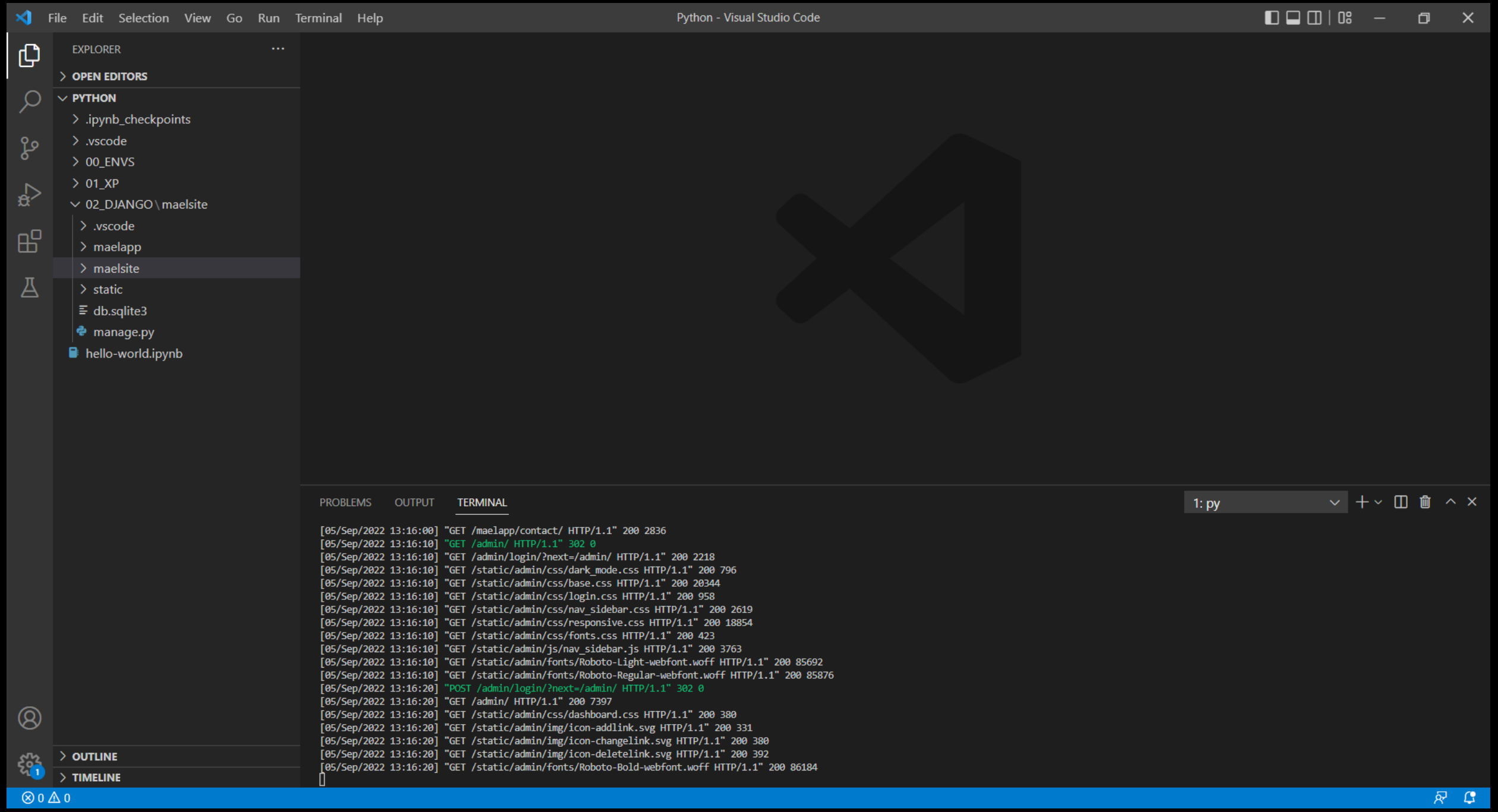
- **Visual Studio code** (VS Code/VSC) is a free code editor with a built-in extension manager.
- Integrated tabbed **terminals** & cohesive **Python extension**.
- Built-in support & **IntelliSense** for **HTML, JavaScript** & **CSS**.
- To select a **Python Interpreter** & activate a **venv** on VSC:

  Command Palette (Ctrl+Shift+P):

  > Python: Select Interpreter

  > Browse to & select: ./your_venv/Scripts/python.exe

- To switch terminal tabs **Ctrl + (PageUp/PageDown)**


File Edit Selection View Go Run Terminal Help
Python - Visual Studio Code
EXPLORER
OPEN EDITORS
PYTHON
.ipynb_checkpoints
.vscode
00_ENVS
01_XP
02_DJANGO \ maelsite
.vscode
maelapp
maelsite
static
db.sqlite3
manage.py
hello-world.ipynb
OUTLINE
TIMELINE
PROBLEMS
OUTPUT
TERMINAL
1: py
[05/Sep/2022 13:16:00] "GET /maelapp/contact/ HTTP/1.1" 200 2836
[05/Sep/2022 13:16:10] "GET /admin/ HTTP/1.1" 302 0
[05/Sep/2022 13:16:10] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 2218
[05/Sep/2022 13:16:10] "GET /static/admin/css/dark_mode.css HTTP/1.1" 200 796
[05/Sep/2022 13:16:10] "GET /static/admin/css/base.css HTTP/1.1" 200 20344
[05/Sep/2022 13:16:10] "GET /static/admin/css/login.css HTTP/1.1" 200 958
[05/Sep/2022 13:16:10] "GET /static/admin/css/nav_sidebar.css HTTP/1.1" 200 2619
[05/Sep/2022 13:16:10] "GET /static/admin/css/responsive.css HTTP/1.1" 200 18854
[05/Sep/2022 13:16:10] "GET /static/admin/css/fonts.css HTTP/1.1" 200 423
[05/Sep/2022 13:16:10] "GET /static/admin/js/nav_sidebar.js HTTP/1.1" 200 3763
[05/Sep/2022 13:16:10] "GET /static/admin/fonts/Roboto-Light-webfont.woff HTTP/1.1" 200 85692
[05/Sep/2022 13:16:10] "GET /static/admin/fonts/Roboto-Regular-webfont.woff HTTP/1.1" 200 85876
[05/Sep/2022 13:16:20] "POST /admin/login/?next=/admin/ HTTP/1.1" 302 0
[05/Sep/2022 13:16:20] "GET /admin/ HTTP/1.1" 200 7397
[05/Sep/2022 13:16:20] "GET /static/admin/css/dashboard.css HTTP/1.1" 200 380
[05/Sep/2022 13:16:20] "GET /static/admin/img/icon-addlink.svg HTTP/1.1" 200 331
[05/Sep/2022 13:16:20] "GET /static/admin/img/icon-changelink.svg HTTP/1.1" 200 380
[05/Sep/2022 13:16:20] "GET /static/admin/img/icon-deletelink.svg HTTP/1.1" 200 392
[05/Sep/2022 13:16:20] "GET /static/admin/fonts/Roboto-Bold-webfont.woff HTTP/1.1" 200 86184
0 0

Django 101

The Basics

Create Project

- Activate your venv & cd to your intended project path.
- Create your project:
  django-admin startproject <projectname>
  e.g. **django-admin startproject myproject**
- This creates a new directory with Django project folders & files.
- cd to your project path and test your project:
  python manage.py runserver <localhost:port>
  e.g. **python manage.py runserver localhost:8001**
  *** by default Django will run on localhost:8000**
- Open **http://localhost:8001/** in a new browser window
  (Ctrl+Click in VSC Console)

*<localhost:port> is optional as it is useful when running multiple projects simultaneously in 1 venv

The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

Django Documentation
Topics, references, & how-to's

Tutorial: A Polling App
Get started with Django

Django Community
Connect, get help, or contribute

```
myproject/
    manage.py
    myproject/
        __init__.py
        settings.py
        urls.py
        asgi.py
        wsgi.py
```

- The outer **myproject/** root directory is a container for your project.
- **manage.py**: A command-line utility that lets you interact with this Django project.
- The inner **myproject /** directory is the actual Python package for your project:
  - **__init__.py**: An empty file that tells Python that this directory should be considered a Python package.
  - **settings.py**: Settings/configuration for this Django project.
  - **urls.py**: The URL declarations for this Django project
  - **asgi.py** & **wsgi.py**: Entry-points for ASGI & WSGI compatible web servers for deployment.

Create App

- Activate your venv & cd to your project path.
- **By default, when first creating project, an app was automatically created with the same name as project name.**
- Create your app :

  python manage.py startapp <appname>

  e.g. **python manage.py startapp myapp**
- This creates a new directory with Django app folders & files.
- Next, we will add **myapp** to the constant **INSTALLED_APPS** found in **settings.py**.

Register The App

- Open **myproject/myproject/settings.py** & add as follows:

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

- This tells Django that **a new app has been created** and it will allow us to perform more complicated tasks.

Create View

- **Django views** are Python functions that **takes http requests** and **returns http response** e.g. HTML documents.
- A website that uses Django is full of **views with different tasks**.
- Views can be created in **views.py** located in your app folder.
- In **myapp/views.py**, recreate the following lines:

```
from django.shortcuts import render
from django.http import HttpResponse

def index(request):
    return HttpResponse('Hello World!')
```

Create View: URL

- Next, create a file named **urls.py** in the same folder as the **views.py** file, & type in the following lines:

```
from django.urls import path
from myapp.views import *

urlpatterns = [
    path('', index, name='index'),

]
```

- The **urls.py** file you just created is specific for the **myapp** application.
- We have to do some routing in the root directory **myproject** as well.

Create View: Root Routing

- There is a file called **myproject/myproject/urls.py**
- Open that file and add the **include** & **path** modules in the import statement
- Add a **path()** function in the **urlpatterns = []** list, with arguments that will route users that comes in via **http://localhost:8001/myapp/**

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('myapp/', include('myapp.urls')),
    path('admin/', admin.site.urls),
]
```

Test Your App

- cd to your project path and test your app :
  python manage.py runserver <localhost:port>
  e.g. **python manage.py runserver localhost:8001**
- Open **http://localhost:8001/myapp/** in a new browser window

*<localhost:port> is optional as it is useful when running multiple projects simultaneously in 1 venv

Create a HTML Template

- **Templates** can help us better manage & style our response renditions.
- Create a **templates** folder inside the **myapp** folder, and create a HTML file named **myfirst.html**.
- The file structure should be something like this:

```
myproject
    manage.py
    myproject/
    myapp/
        templates/
            myfirst.html
```

Editing the HTML Template

- Open & edit **myapp/templates/myfirst.html** as follows:

```
<!DOCTYPE html>
<html>

<body>

<h1>Hello World!</h1>
<p>Welcome to my first Django Project!</p>

</body>
</html>
```

- We can now call this template in our **View** by editing **views.py**

Add to View

- Open **myapp/views.py** & replace the lines with the following:

```
from django.shortcuts import render


def index(request):
    return render(request, "myfirst.html");
```

- The new lines will call the file **myfirst.html** using it as the **template** to render the index **view** for **myapp**.

Test Your Template

- cd to your project path and test your app:
  python manage.py runserver <localhost:port>
  e.g. **python manage.py runserver localhost:8001**
- Open **http://localhost:8001/myapp/** in a new browser window

*<localhost:port> is optional as it is useful when running multiple projects simultaneously in 1 venv

Create Model

- A Django **model** is a **table** in the Django project **database**.
- When we created the Django project **myproject**, we automatically created an empty **SQLite database** in the **myproject** root folder.
- To create a new **table**, we must make changes to **models.py**.
- Open **myapp/models.py** and observe that it is almost empty with only an import statement & a comment:

```
from django.db import models

# Create your models here.
```

Add a Table

- To add a **Members** table in the **database**, start by creating a Members **class**, & describe the fields in it:

```
from django.db import models

class Members(models.Model):
    firstname = models.CharField(max_length=255)
    lastname = models.CharField(max_length=255)
```

- We've created two fields **firstname** & **lastname** each with a max length.
- Next, we need to migrate these changes to the **model** to create the table.
- **Migrations** are Django's way of propagating changes you make to your models (e.g. adding a field, deleting a model) into your DB schema.

Make Migrations

- cd to the root myproject folder and run:

  py manage.py makemigrations

- You will see the following output:

```
Migrations for 'myapp':
  myapp\migrations\0001_initial.py
    - Create model Members
```

- Django creates a file with new changes (table creation, field name change, etc) & stores it in the **migrations** folder.
- Django will create and execute an SQL statement, based on the content of the new file in the migrations folder, when you run the migrate command:

  py manage.py migrate

Complete Migrations

- Once you run the **migrate** command, you will see the following:

```
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, myapp, sessions
Running migrations:
  Applying myapp.0001_initial... OK
```

- This means that Django has created the **Members** table in your SQLite DB.
- Next, you can perform CRUD operations to your tables using the Django **admin** by registering your models to **admin.py**.

Django Admin

- Django comes with a powerful administrative tool called **admin**.
- You can use it out of the box to perform CRUD operations on the database from a web interface.
- Go to **http://localhost:8001/admin/** to see the login page.
- You need to create a super user in order to login to **admin**.
- In the terminal, run the following command:

  py manage.py createsuperuser

- Django will prompt for you enter the following credentials: **username**, **email**, **password** & **password confirmation**.
- Once the **superuser** is created, you can login to **admin** with the credentials.

Add admin to urls.py

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('myapp/', include('myapp.urls')),
    path('admin/', admin.site.urls),
]
```

To use the Django **admin** dashboard, make sure **admin.site.urls** is added into **myproject/myproject/urls.py**

Register Model to Admin

- To perform CRUD to your tables via Django **admin**, you must first register the models in **myapp/admin.py**.
- Add a reference to the models by entering the following lines in **admin.py**:

```
from django.contrib import admin
from myapp.models import Members

admin.site.register(Members)
```

- Navigate to **admin**: http://localhost:8001/admin/
- You can now see **Members** appear under **myapp** & perform your CRUD operations using the **admin** UI.

Django 101

Docs & References

Django Documentation

- **Direct Link:** https://docs.djangoproject.com/en/4.1/
- **Tutorials** take you by the hand through a series of steps to create a web application. Start here if you're new to Django or web application development. Also look at the "**First steps**".
- **Topic guides** discuss key topics and concepts at a fairly high level and provide useful background information and explanation.
- **Reference guides** contain technical reference for APIs and other aspects of Django's machinery. They describe how it works and how to use it but assume that you have a basic understanding of key concepts.
- **How-to guides** are recipes. They guide you through the steps involved in addressing key problems and use-cases. They are more advanced than tutorials and assume some knowledge of how Django works.

Other Good References

- Django :
  - W3Schools Tutorials : https://www.w3schools.com/django/index.php
  - JavaTPoint : https://www.javatpoint.com/django-tutorial
  - A Complete Beginner's Guide to Django : https://simpleisbetterthancomplex.com/series/beginners-guide/1.11/
  - OpenTechSchool : http://opentechschool.github.io/django-101/en/index.html
- For Template & UI :
  - HTML : https://www.w3schools.com/html/default.asp
  - CSS : https://www.w3schools.com/css/default.asp

Django 101

The Assignment

Learn By Doing!

- **The Assignment** has 2 parts :
  - Part 1 : **Read Up!**
  - Part 2 : **Personal App**
- You will be required to **extend your Django knowledge** from this basic training.
- You have **1 week** to complete this assignment.
- You will present your project during the basic training follow up session.

Part 1 : Read Up!

- Look through the references & focus on the following:
  1) The **URL Dispatcher**: Path Converters & Using Regex
  2) Using **Static Files** in Your Django Project
  3) Django **Models**: Field & Meta Options
  4) Displaying **Models** in your Django **Template**
  5) Django **Forms** & Creating Forms from **Models**
  6) Django **Templates** & Template Inheritance

- **Remember:** These topics will help you accomplish **Part 2** of The Assignment.

Part 2 : Personal App

- Create a new virtual environment & install Django
- Create a new Django project - **yournamesite**
- Create a new Django app - **yournameapp**
- Create templates, views & models to present the following :
  - **Homepage - About Me**

    HTML with External CSS Styling (Create Django Template & Static files)
  - **Portfolio**

    Listing Past Projects (Create Django Model & Register to Admin)
  - **Contact**

    Feedback Form (Create Form with Django Model Forms)


fin.