Django

Getting Ready for Django

- Before we start coding, it's really important that we set your development environment up correctly.

1. The terminal1 (on macOS or UNIX/Linux systems), or the Command Prompt2 (on Windows);
2. Python 3, including how to code and run Python scripts;
3. The Python Package Manager pip (It comes with python 3);
4. Virtual Environments;
5. your Integrated Development Environment (IDE) I strongly recommend VS Code, if you choose to use one; and a Version Control System (VCS) called Git.

Installations **Python 3**

To work with Django, it is expected of you to have installed on your computer a copy of the Python 3 programming language. Specifically, you need to ensure you are using a version of Python 3.8+ (as stated on the official Django 4 documentation3).

Installations **Virtual Environment**

With a working installation of Python 3, we can now setup our environment for the Django project, we'll be creating in this class. One super useful tool we strongly encourage you to use is a **virtual environment**.

A virtual environment allows for multiple installations of Python packages to exist in harmony, within unique Python environments. Why is this useful? Say you have a project, project A that you want to run in Django 2.1, and a further project, project B written for Django 4.0.

Installations The Python Package Manager (PIP)

Going hand in hand with virtual environments, we'll also be making use of the Python package manager, pip, to install several different Python software packages to our development environment.

Installations **Virtual Environment (cont.)**

This presents a problem as you would normally only be able to install one version of the required software at a time. By creating virtual environments for each project, you can install the respective versions of Django within each unique environment. This ensures that the software installed in one environment does not tamper with the software installed on another.

Installations **Django**

To ensure smooth sailing during your learning process, make sure you're using at least Django 4.0.0. Versioning of Django is similar to that of Python;

Installations **Pillow**

Pillow is a Python package providing support for handling image files (e.g. .jpg and .png files), something we'll be doing later in this class.

Windows Installations

	Command/link
Download and install python	https://www.python.org/downloads/
Install virtual environment	py -3 -m venv venv
Activate virtual environment	venv\scripts\activate
Install python and pillow	pip install -r requirements.txt

MAC Installations

	Command/link
Download and install python	https://www.python.org/downloads/
Install virtual environment	**python3 -m venv venv**
Activate virtual environment	source folder/bin/activate
Install Django and pillow	pip install -r requirements.txt

Installations **Django**

Django can be installed easily using pip in different ways.	
1st Method	In the command prompt, execute the following command: **pip install django**. This will download and install Django.
2nd Method	Create a .txt file name **requirements** and in the file type **django** hit enter and also type **pillow.** Save the file and on the command prompt execute the following command: **pip install -r requirements.txt.** This will download and install Django.
After the installation has completed, you can verify your Django installation by executing **django-admin --version** in the command prompt.	

Django Project

- A project in Django is a python package that represents the whole web application.
- A project in Django basically contains the configuration and setting related to the entire website.
- A single project can also have multiple apps in it that can be used to implement some functionality.

Starting Django Project

	Command
StartProject:	django-admin startproject crm_project .
The command above will invoke the django-admin script, which will set up a new Django project called crm_project for you	

Created Project files

Name	Description
__init__.py	a blank Python script whose presence indicates to the Python interpreter that the directory is a Python package
settings.py	the place to store all of your Django project's settings
urls.py	a Python script to store URL patterns for your project
asgi.py	a Python script used to let your Django project talk to a web server supporting the Asynchronous Server Gateway Interface
wsgi.py	a script similar to above, but for web servers supporting the older Web Server Gateway Interface
manage.py	allows you to run the built-in Django development server, test your application, and run various database commands.

Launch Your Application

- On your terminal type **python manage.py runserver**

Creating a Django App

- A Django project is a collection of configurations and apps that together make up a given web application or website.
- An app in Django is a sub-module of a project, and it is used to implement some functionality.
- Now, you can refer to an app as a standalone python module that is used to provide some functionality to your project.

Starting Django App

	Command
Start App:	django-admin startapp crm_app
The startapp command creates a new directory within your project's root. Unsurprisingly, this directory is called crm_app	

Created App files

Name	Description
__init__.py	serving the same purpose as discussed previously
admin.py	where you can register your models so that you can benefit from some Django machinery which creates a slick administrative interface for you (more on that later)
apps.py	that provides a place for any app-specific configuration
models.py	a place to store your app's data models – where you specify the entities and relationships between data
tests.py	where you can store a series of functions to test your implementation
views.py	where you can store a series of functions that handle requests and return responses
migrations directory	which stores database-specific information related to your models

Linking Your APP to your PROJECT

- You must first tell your Django project about the new app's existence. To do this, you need to modify the settings.py file, contained within your project's configuration directory. Open the file and find the INSTALLED_APPS list. Add the **psychology_app** to the end of the list, which should then look like the following example.

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

Creating a View

- With our app created, we can now create a simple view. Views handle a request that comes from the client, executes some code, and provides a response to the client. To fulfil the request, Django may contact other services, or query for data from other sources (such as a database, for example). The job of a view is to collate and package up the data required to handle the request, as we outlined above.

Creating a View

Views.py

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



def index(request):
        return HttpResponse("My First Django Project!")
```

urls.py

```
from crm_app import views

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