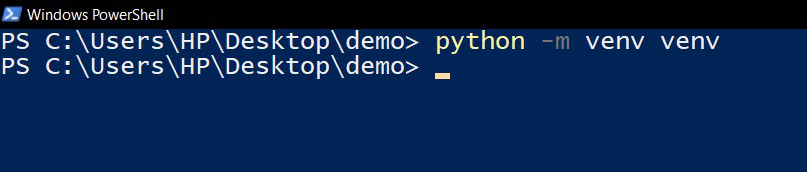
**Step 1: Create a virtual environment**

Open the directory where you want to create your project. open cmd/powershell and navigate to the same directory and run the following commands to create a virtual environment.

python -m venv venv



**Step 2: Activate the virtual environment**

Now as we have our virtual environment let’s activate it.

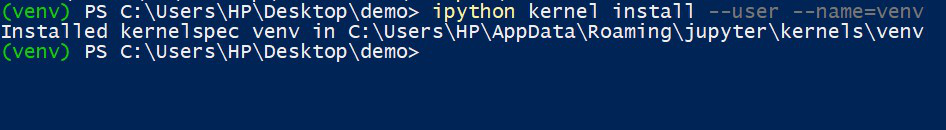
venv\Scripts\activate



**Step 3: Install jupyter kernel for the virtual environment using the following command:**

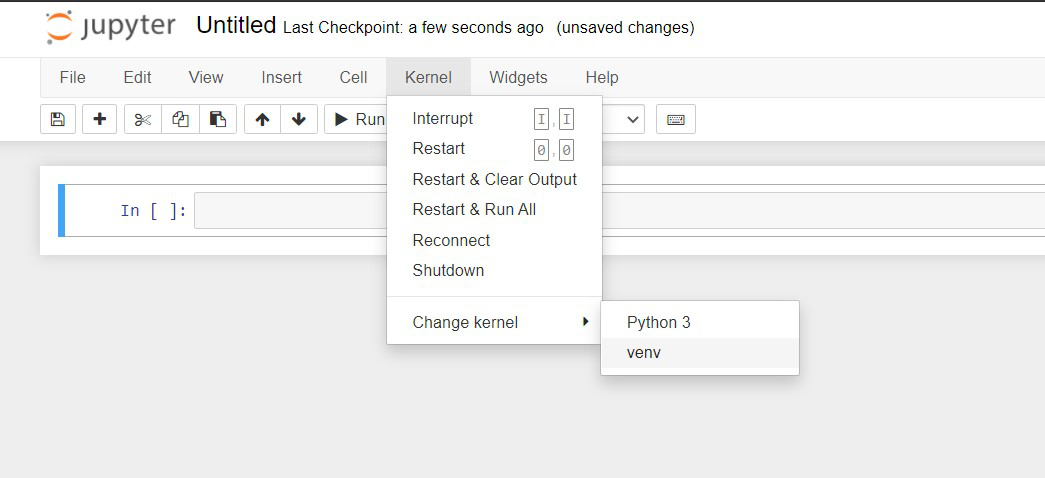
Running the following command will create a kernel that can be used to run jupyter notebook commands inside the virtual environment.

ipython kernel install --user --name=venv



**Step 4: Select the installed kernel when you want to use jupyter notebook in this virtual environment**

Let’s now check if our kernel is created. Just run “jupyter notebook” command in the command prompt or Powershell and the jupyter environment will open up. Click on the kernel and click change kernel you will be able to see the kernel you just created.



**FaceNet for Face Recognition:**

**Introduction:**

FaceNet is a deep learning model designed for face recognition tasks. Developed by Google researchers, it employs a deep convolutional neural network (CNN) to map facial features into a high-dimensional space, ensuring that similar faces are close to each other in this space. Notably, FaceNet uses a triplet loss function during training to minimize the distance between positive face pairs while maximizing the distance between negative face pairs.

**Key Features:**

Triplet Loss: FaceNet uses a triplet loss function to train the model, which aids in learning a robust face representation.

Embedding Space: Faces are represented as points in a high-dimensional space, ensuring that similar faces are close together.

**Performance:**

Accuracy: FaceNet has demonstrated state-of-the-art accuracy in face recognition tasks, achieving high levels of precision and recall.

Robustness: The model is known for its robustness to variations in lighting conditions, facial expressions, and poses.

**Challenges:**

Computational Intensity: Training and running FaceNet can be computationally intensive, requiring substantial resources.

Data Privacy Concerns: Face recognition models, including FaceNet, raise concerns about privacy and potential misuse.

**For graph and official docs reading kindly go through this link:**

<https://www.geeksforgeeks.org/facenet-using-facial-recognition-system/>

**File Organization 🗄️**

├── Real-time-face-recognition-Using-Facenet (Current Directory)

├── encodings

├── architecture.py

├── detect.py

├── facenet\_keras\_weights.h5

├── train\_v2.py

├── requirements.txt

├── Faces

├── Azam

└── winnie

└── JackieChan

# Dependencies 💾

Python 3.11.5 This code was working properly on tensroflow 2.3.0.

* Tensorflow 2.X
* numpy
* opencv-python
* mtcnn
* scikit-learn
* scipy

## Code Requirements 🦄

pip install requirements.txt

## SetUp 🖥️

1. Make a directory of your name inside the **Faces** folder and upload your 2-3 pictures of you.
2. Train Your System/ Which is already integrated into Notebook

python train\_v2.py

1. To test your image, rename the image which will be store into test folder and the out will be saved.
2. It is necessary that each time you want to run for new image, the input and output image name should be update/changed.

**YOLOv8 for Object Detection:**

**Introduction:**

You Only Look Once (YOLO) is a popular object detection algorithm, and YOLOv8 is one of its iterations. YOLOv8 divides an image into a grid and predicts bounding boxes and class probabilities for each grid cell. This model is known for its real-time processing capabilities and high accuracy.

**Key Features:**

Single Forward Pass: YOLOv8 processes the entire image in a single forward pass, making it efficient for real-time applications.

Anchor Boxes: The model uses anchor boxes to improve bounding box predictions, enhancing localization accuracy.

**Performance:**

Real-time Processing: YOLOv8 is optimized for real-time object detection, making it suitable for applications such as video surveillance and autonomous vehicles.

Accuracy: YOLOv8 achieves competitive accuracy on object detection benchmarks, balancing speed and precision.

**Challenges:**

Fine-grained Detection: YOLO may face challenges in detecting small or closely spaced objects due to its coarse grid-based approach.

Training Data Requirements: YOLOv8 requires a diverse and well-annotated dataset for effective training.

**For graph and official docs reading kindly go through this link:**

<https://docs.ultralytics.com/#where-to-start>

**Instructions to run:**

1. First install this library once

Pip install ultralytics

1. To Test images Run bellow command Source name should be change always for new image to test.

!yolo detect predict model=yolov8n.pt source="gettyimages-530330473.jpg"

The result will be stored into run/detect/pred folder each time you run it will create new folder to see the result change the number for each image in the path for example run/detect/pred1 or pred2 or pred\*