

FACE DETECTION APPLICATION

1. What Are We Building?

We are building a **Real-Time Face Detection Application** .

This application:

-  Opens your webcam
-  Detects human faces
-  Draws a rectangle around faces
-  Shows live video on screen

It works in real-time using your computer camera.

2. Technologies Used

Technology	What it is used for
Python	Main programming language
OpenCV	Used for camera access and face detection
Haar Cascade	Pre-trained face detection model
Webcam	Captures live video

3. Before Starting (Setup)

Step 3.1: Install Python

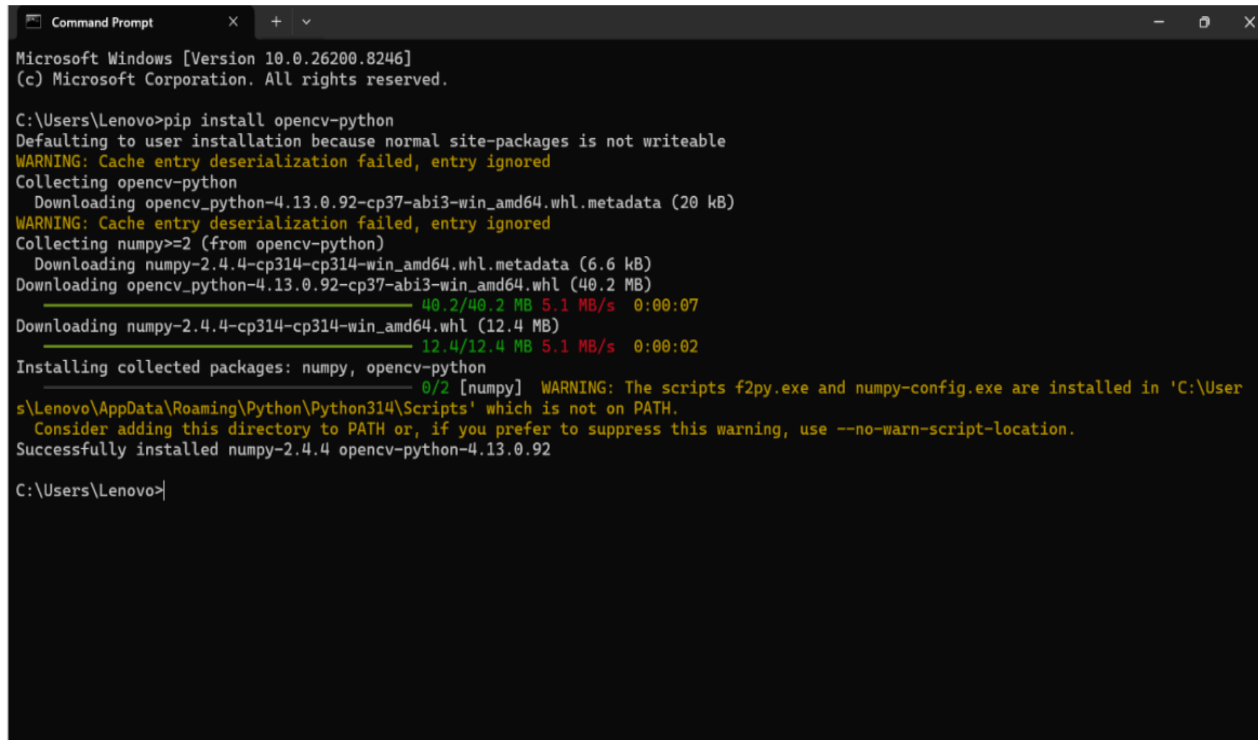
Download from: <https://www.python.org>

While installing → check “Add Python to PATH”

Step 3.2: Install OpenCV

Open terminal / command prompt and type:

```
pip install opencv-python
```



```
Microsoft Windows [Version 10.0.26200.8246]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo>pip install opencv-python
Defaulting to user installation because normal site-packages is not writeable
WARNING: Cache entry deserialization failed, entry ignored
Collecting opencv-python
  Downloading opencv_python-4.13.0.92-cp37-abi3-win_amd64.whl.metadata (20 kB)
WARNING: Cache entry deserialization failed, entry ignored
Collecting numpy>=2 (from opencv-python)
  Downloading numpy-2.4.4-cp314-cp314-win_amd64.whl.metadata (6.6 kB)
  Downloading opencv_python-4.13.0.92-cp37-abi3-win_amd64.whl (40.2 MB)
    40.2/40.2 MB 5.1 MB/s 0:00:07
  Downloading numpy-2.4.4-cp314-cp314-win_amd64.whl (12.4 MB)
    12.4/12.4 MB 5.1 MB/s 0:00:02
Installing collected packages: numpy, opencv-python
  0/2 [numpy] WARNING: The scripts f2py.exe and numpy-config.exe are installed in 'C:\User
s\Lenovo\AppData\Roaming\Python\Python314\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed numpy-2.4.4 opencv-python-4.13.0.92

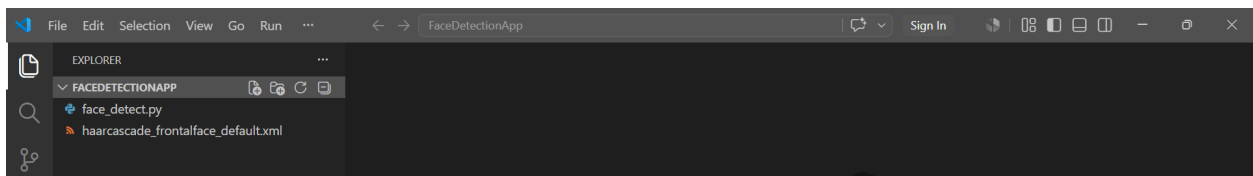
C:\Users\Lenovo>
```

This installs OpenCV library required for face detection.

4. Create Project Structure

Create a folder named FaceDetectionApp

Inside it, create:



5. Backend Development (`face_detect.py`)

This file contains the complete face detection logic.

It controls:

- Webcam
- Face detection
- Rectangle drawing
- Live display

Step 5.1: Import Required Modules

```
import cv2
```

`cv2` is the OpenCV library.

It helps to:

- Access webcam
- Process images
- Detect faces
- Show video frames

innovbit AI Lab

Step 5.2: Start Webcam

```
cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
```

Explanation:

- 0 means default webcam
- `CAP_DSHOW` improves camera compatibility on Windows

Step 5.3: Check Camera Availability

```
if not cap.isOpened():  
    print("Camera not detected")  
    exit()
```

Prevents errors if the webcam is not connected.

If camera is missing:

- Message is shown
- Program stops safely

Step 5.4: Load Face Detection Model

```
face_cascade =  
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

This loads the Haar Cascade model.

The XML file already contains trained face detection data.

Step 5.5: Start Infinite Loop

```
while True:
```

Keeps the webcam running continuously.

Without this loop:

- Camera would close instantly

Step 5.6: Load Face Detection Model

```
ret, frame = cap.read()
```

Explanation:

Technology	What it is used for
ret	Checks if frame captured successfully
frame	Current camera image

Step 5.7: Check Frame Capture

```
if not ret:
    print("Failed to grab frame")
    break
```

Stops program safely if frame capture fails.

Step 5.8: Convert Image to Grayscale

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

Face detection works better on grayscale images.

Why grayscale?

- Faster processing
- Less computation
- Better detection accuracy

Step 5.9: Detect Faces



```
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
```

This detects all faces in the frame

Parameters:

Parameters	Meaning
gray	Grayscale image
1.3	Scale factor
5	Minimum neighbor detections

Step 5.10: Draw Rectangle Around Faces

```
for (x, y, w, h) in faces:
```

Loops through all detected faces.

Rectangle code:

```
cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
```

Explanation:

Value	Meaning
(x,y)	Starting point
(x+w,y+h)	Ending Point
(255,0,0)	Blue rectangle color
2	Thickness

6. Display Output Window

innovbit AI Lab

Step 6.1: Show Video

```
cv2.imshow("Face Detection", frame)
```

Opens a live webcam window.

Detected faces appear with rectangles.

Step 6.2: Exit Application

```
if cv2.waitKey(1) == ord('q'):  
    break
```

Pressing q closes the application.

Step 6.3: Release Camera

```
cap.release()  
cv2.destroyAllWindows()
```

Properly closes:

- Webcam
- OpenCV windows

7. How Face Detection Works

Haar Cascade Algorithm

The application uses:

```
haarcascade_frontalface_default.xml
```

This is a pre-trained model created by OpenCV.

It detects:

- Eyes
- Nose
- Face structure

The logo for Innovbit AI Lab, featuring the word "innovbit" in blue, "AI" in red, and "Lab" in blue.

to identify human faces.

8. Run the Application

Open terminal inside project folder:

```
python face_detect.py
```

9. How Everything Works Together (Flow)

- Program starts
- Webcam opens
- Video frames are captured
- Frame converts to grayscale
- Face detection model scans image
- Faces are detected
- Rectangles are drawn

- Live video appears on screen
- User presses **q** to exit

10. Common Mistakes (Very Important)

- Forgetting to install OpenCV
- Missing XML file
- Wrong XML file path
- Webcam permission denied
- Camera already used by another app
- Running script outside project folder

11. Tips & Tricks

11.1. Improve Detection Accuracy

Use:

`detectMultiScale(gray, 1.1, 6)`

Gives more accurate results.

11.2. Add Eye Detection

You can use another Haar Cascade file:

`haarcascade_eye.xml`

11.3. Detect Multiple Faces

The current project already supports multiple faces automatically.

11.4. Save Captured Images

Use:

`cv2.imwrite()`

Saves images to the computer.

12. Final Summary

This project teaches:

- OpenCV basics
- Webcam handling
- Real-time video processing
- Face detection
- Image processing
- Haar Cascade classifiers

It is a great beginner project for learning:

- Computer Vision
- Artificial Intelligence basics
- Real-time image processing using Python

The logo for Innovbit AI Lab. It features the word "innovbit" in a light blue, lowercase sans-serif font. To its right is the text "AI Lab" in a larger, bold, light blue sans-serif font. The "AI" is in a darker blue, and the "Lab" is in a lighter blue. A small pink triangle is positioned above the "i" in "innovbit".