Step 1:

Each and every one of the 8 billion people on this earth have different facial features, and even though we do not interact with many people in our daily lives, we can somehow distinguish between every person based on their facial features with relative ease. How does our mind so easily map these complex features to a name or identity? This task which is so easy for our human brains is a complex feat in computer science. How can we create a computer programme that identifies the features in a multiple face image and then maps each of them to a name or identity? We can firstly try to find a relation between the relative positioning of the eyes, nose and mouth and the structure of our face. Skin tone can also be used as an additional feature though inconsistent lighting can affect this feature.

In this current day and age, where public security is being monitored using security cameras and drones, identification of a culprit in a vast sea of people can easily be done using facial recognition algorithm to help the authorities find the culprit. Also, as these days online classes are being taken, taking attendance using the old method of calling a child's name is tedious and requires a lot of work. Instead facial recognition algorithm's can be used to identify each child and make the work of the teacher easier and less time consuming. Also using facial features as a mean of authentication for personal and sensitive information is a great example of why facial recognition should be developed

So, we are looking forward to develop a programme that identifies faces in a still image using a dataset of known faces.

Step 2:

Over the past decade face detection and recognition has transcended from esoteric to popular areas of research in computer vision and one of the better and successful applications of image analysis and algorithm-based understanding.

A general statement of the face recognition problem (in computer vision) can be formulated as follows: given an image identify or verify one or more persons in the scene using a stored database of faces and it generally involves two stages.

1. Face Detection: Where a photo is searched to find a face and then the photo is processed to extract the persons face for easier recognition.
2. Face Recognition: The detected face is compared to existing database of known faces to see who the person is

Since long, face detection can be performed easily using Intel's open source network called OpenCV. OpenCV is a strong framework that's works in 90-95% of cases. Only the lack of brightness, shadows and blurriness can cause the network to not work properly.

However, face recognition is much less reliable than face detection with an accuracy of 30-70% in general and this has been a strong field of research. The Eigenface technique is considered the simplest method of accurate face recognition.

Step 3:

To determine the basic ingredients, we must first determine the necessary requirements for our model. For proper face recognition and detection, we need a well light area and a camera that is well equipped to click a photo that is not blurry and does not add excessive contrast as well. Also, we need to create a dataset so that we can train a model to identify faces and a computer that can handle such computations. The initial images we train our model on must be of good quality so that once we train our model to detect faces, we can use the same weights to make the model learn the features of a new face.

Also, as many people wear specs, apply makeup etc. it can also affect the output of the data, so it also important to consider it as part of training.

Now the important part is to find a library that can help us create an accurate model:

OpenCV: Its functionality will be used in facial recognition which is contained in many modules

1. CXCORE: It contains basic data type definitions, linear algebra and statistics methods, the persistence functions and the error handlers.
2. CV: It contains image processing and camera calibration methods.
3. CVAUX: The simplest interfaces for face recognition are in this module. The code behind them is specialized for face recognition, and they're widely used for that purpose.
4. ML: Contains machine learning interface
5. HighGUI: Contains the basic I/O interfaces and multi-platform windowing capabilities.

Step 4:

Very large datasets may not always be good. Proper training size depends on different condition including correlated variables. All of this can have a negative impact on the accuracy.

To be able to handle these risks, you must have a basic understanding of

1. Dimension reduction
2. Eigenvectors and Eigenvalues
3. PCA

PCA: It is a statistical method to make data visualisation and navigation a simple process

PCA for images: if you have an image that needs recognition, machine learning algorithms check the differences between the target image and each of the principal components.

Eigenvectors and Eigenvalues: An eigenvector is expressed by a real nonzero eigenvalue, that points in a direction that is stretched by a transformation. On the other side, the eigenvalue is the factor by which it is stretched. A typical application is image compression.

Dimension reduction: It is an algorithm to reduce the number of features by choosing the most important ones

Step 5:

When image quality is taken into consideration, there is a plethora of factors that influence the system’s accuracy. So, it is very important to apply pre processing techniques to standardise the images as lighting, makeup, emotions and specs can cause issues with face recognition.
So, for simplicity we use Eigenfaces using greyscale images for face recognition as it is really easy to apply Histogram Equalisation for standardising the brightness and image contrast. Also, we can apply edge detection, contour detection and resize the image.
Also, OpenCV uses a Haar Cascade Classifier as a type of face detector. Given an image, which can come from a file or from live video, the face detector examines each image location and classifies it as "Face" or “Not Face”. The classifier uses data stored in an XML file to decide how to classify each image location.

Step 6:

1. Selecting Size of dataset
2. Normalising the images (To get uniform size and lighting)
3. Convert to grayscale (To remove color as it is an unnecessary variable)
4. Train model for face detection
    a. Haar Classifier:
        i. Feature extraction
        ii. Integral Image concept
        iii. Adaboost to improve accuracy
        iv. Cascade of classifiers
    b. Face Detection:
        i. Loading necessary libraries
        ii. Loading classifiers for frontal face
        iii. Using:
            1. Scale factors: Compensates for distance of faces
            2. Min neighbours: No. of neighbours a rectangle to have to be called a face
5. Testing for new image

https://github.com/parulnith/Face-Detection-in-Python-using-OpenCV/blob/master/Face%20Detection%20with%20OpenCV-Python.ipynb

Step 9:

Once we have finished our model the most important aspect is to test it on different faces, lighting condition and different facial expressions

First, we should test our model on a single photo and then if it works well try to see if it works the same on a group photo.
Only after it has passed these tests should we consider our model to be well trained to be working for real life condition.