



Face Recognition with Python, in Under 25 Lines of Code

by [Shantnu Tiwari](#) 115 Comments [data-science](#) [machine-learning](#)

[Tweet](#) [Share](#) [Email](#)

Table of Contents

- [OpenCV](#)
 - [Cascades in Practice](#)
- [Installing OpenCV](#)
- [Understanding the Code](#)
- [Checking the Results](#)
 - [What Happened?](#)
- [Extending to a Webcam](#)
- [Want to Know More?](#)



In this article, we'll look at a surprisingly simple way to get started with face recognition using Python and the open source library [OpenCV](#).

Before you ask any questions in the comments section:

1. Do not skip the article and just try to run the code. You must understand what the code does, not only to run it properly but also to troubleshoot it.
2. Make sure to use OpenCV v2.
3. Have a working webcam so this script can work properly.
4. Review the other comments and questions, since your questions have probably already been addressed.

Thank you.

Improve Your Python



Free Bonus: [Click here to get the Python Face Detection & OpenCV Examples Mini-Guide](#) that shows you practical code examples of real-world Python computer vision techniques.

Note: Also check out our [updated tutorial on face detection using Python](#).

AD

```
1# How to merge two dicts
2# in Python 3.5+
3
4>>> x = {'a': 1, 'b': 2}
5>>> y = {'b': 3, 'c': 4}
6
7>>> z = {**x, **y}
8
9>>> z
10{'c': 4, 'a': 1, 'b': 3}
```

Improve Your Python ×

...with a fresh  **Python Trick** 

code snippet every couple of days:

Email Address

Send Python Tricks »

OpenCV

OpenCV is the most popular library for computer vision. Originally written in C/C++, it now provides bindings for Python.

OpenCV uses machine learning algorithms to search for faces within a picture. Because faces are so complicated, there isn’t one simple test that will tell you if it found a face or not. Instead, there are thousands of small patterns and features that must be matched. The algorithms break the task of identifying the face into thousands of smaller, bite-sized tasks, each of which is easy to solve. These tasks are also called [classifiers](#).

For something like a face, you might have 6,000 or more classifiers, all of which must match for a face to be detected (within error limits, of course). But therein lies the problem: for face detection, the algorithm starts at the top left of a picture and moves down across small blocks of data, looking at each block, constantly asking, “Is this a face? ... Is this a face? ... Is this a face?” Since there are 6,000 or more tests per block, you might have millions of calculations to do, which will grind your computer to a halt.

To get around this, OpenCV uses [cascades](#). What’s a cascade? The best answer can be found in the [dictionary](#): “a waterfall or series of waterfalls.”

Like a series of waterfalls, the OpenCV cascade breaks the problem of detecting faces into multiple stages. For each block, it does a very rough and quick test. If that passes, it does a slightly more detailed test, and so on. The algorithm may have 30 to 50 of these stages or cascades, and it will only detect a face if all stages pass.

The advantage is that the majority of the picture will return a negative during the first few stages, which means the algorithm won’t waste time testing all 6,000 features on it. Instead of taking hours, face detection can now be done in real time.

Cascades in Practice

Though the theory may sound complicated, in practice it is quite easy. The cascades themselves are just a bunch of XML files that contain OpenCV data used to detect objects. You initialize your code with the cascade you want, and then it does the work for you.

Since face detection is such a common case, OpenCV comes with a number of built-in cascades for detecting everything from faces to eyes to hands to legs. There are even cascades for non-human things. For example, if you run a banana shop and want to track people stealing bananas, [this guy](#) has built one for that!

Installing OpenCV

First, you need to find the correct setup file for [your operatir](#)

Improve Your Python

I found that installing OpenCV was the hardest part of the task. If you get strange unexplainable errors, it could be due to library clashes, 32/64 bit differences, and so on. I found it easiest to just use a Linux virtual machine and install OpenCV from scratch.

Once you have completed the installation, you can test whether or not it works by firing up a Python session and typing:

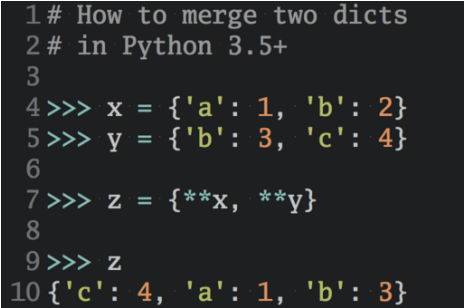
Python>>>

```
>>> import cv2
>>>
```



If you don't get any errors, you can move

Understanding the Code

Let's break down the actual code, which y
abba.png pic, and the **haarcascade_fron**



Improve Your Python×

...with a fresh  **Python Trick** 

code snippet every couple of days:

Email Address

Python

```
# Get user supplied values
imagePath = sys.argv[1]
cascPath = sys.argv[2]
```

You first pass in the image and cascade names as command-line arguments. We'll use the ABBA image as well as the default cascade for detecting faces provided by OpenCV.

Python

```
# Create the haar cascade
faceCascade = cv2.CascadeClassifier(cascPath)
```

Now we create the cascade and initialize it with our face cascade. This loads the face cascade into memory so it's ready for use. Remember, the cascade is just an XML file that contains the data to detect faces.

Python

```
# Read the image
image = cv2.imread(imagePath)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Here we read the image and convert it to grayscale. Many operations in OpenCV are done in grayscale.

Python

```
# Detect faces in the image
faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor=1.1,
    minNeighbors=5,
    minSize=(30, 30),
    flags = cv2.cv.CV_HAAR_SCALE_IMAGE
)
```

This function detects the actual face and is the key part of our code, so let's go over the options:

1. The [detectMultiScale function](#) is a general function that detects objects. Since we are calling it on the face cascade, that's what it detects.
2. The first option is the grayscale image.
3. The second is the `scaleFactor`. Since some faces may be closer to the camera, they would appear bigger than the faces in the back. The scale factor compensates for this.

4. The detection algorithm uses a moving window to detect objects. `minNeighbors` defines how many objects are detected near the current one before it declares the face found. `minSize`, meanwhile, gives the size of each window.

Note: I took commonly used values for these fields. In real life, you would experiment with different values for the window size, scale factor, and so on until you found one that works best for you.

The function returns a list of rectangles in which it believes it found a face. Next, we will loop over where it thinks it found something.

Python

```
print "Found {} faces!".format(len(faces))

# Draw a rectangle around the faces
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (255, 0, 0), 2)
```

```
1# How to merge two dicts
2# in Python 3.5+
3
4>>> x = {'a': 1, 'b': 2}
5>>> y = {'b': 3, 'c': 4}
6
7>>> z = {**x, **y}
8
9>>> z
10{'c': 4, 'a': 1, 'b': 3}
```

Improve Your Python

...with a fresh  **Python Trick** 
code snippet every couple of days:

[Send Python Tricks »](#)

This function returns 4 values: the x and y coordinates of the top-left corner of the rectangle, and the width and height of the rectangle.

We use these values to draw a rectangle around each face found.

Python

```
cv2.imshow("Faces found", image)
cv2.waitKey(0)
```

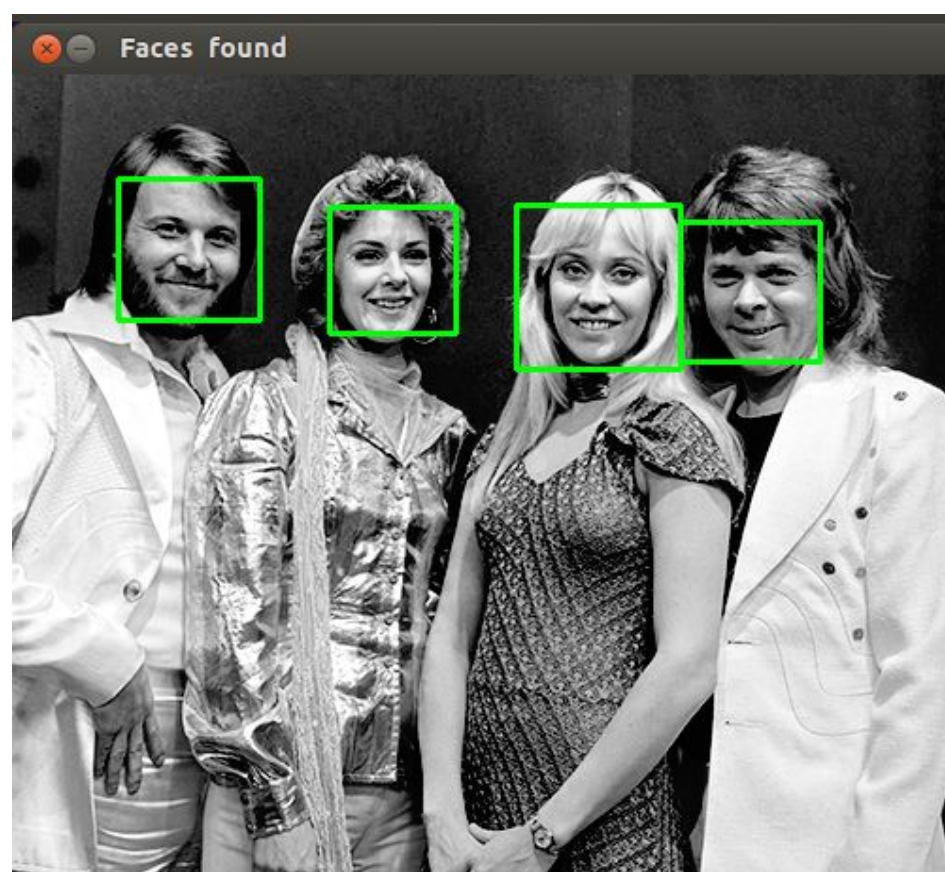
In the end, we display the image and wait for the user to press a key.

Checking the Results

Let's test against the ABBA photo:

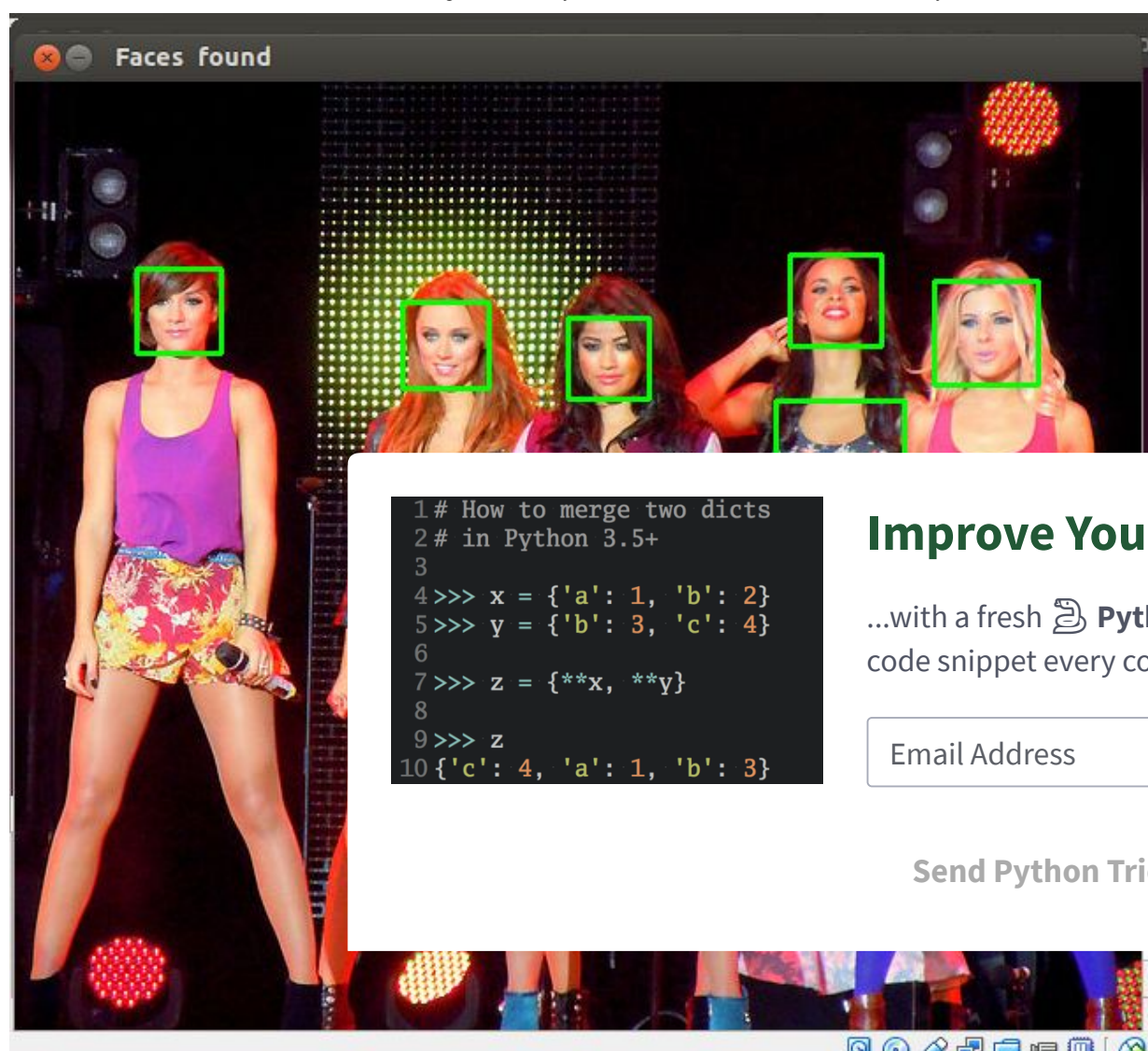
Shell

```
$ python face_detect.py abba.png haarcascade_frontalface_default.xml
```

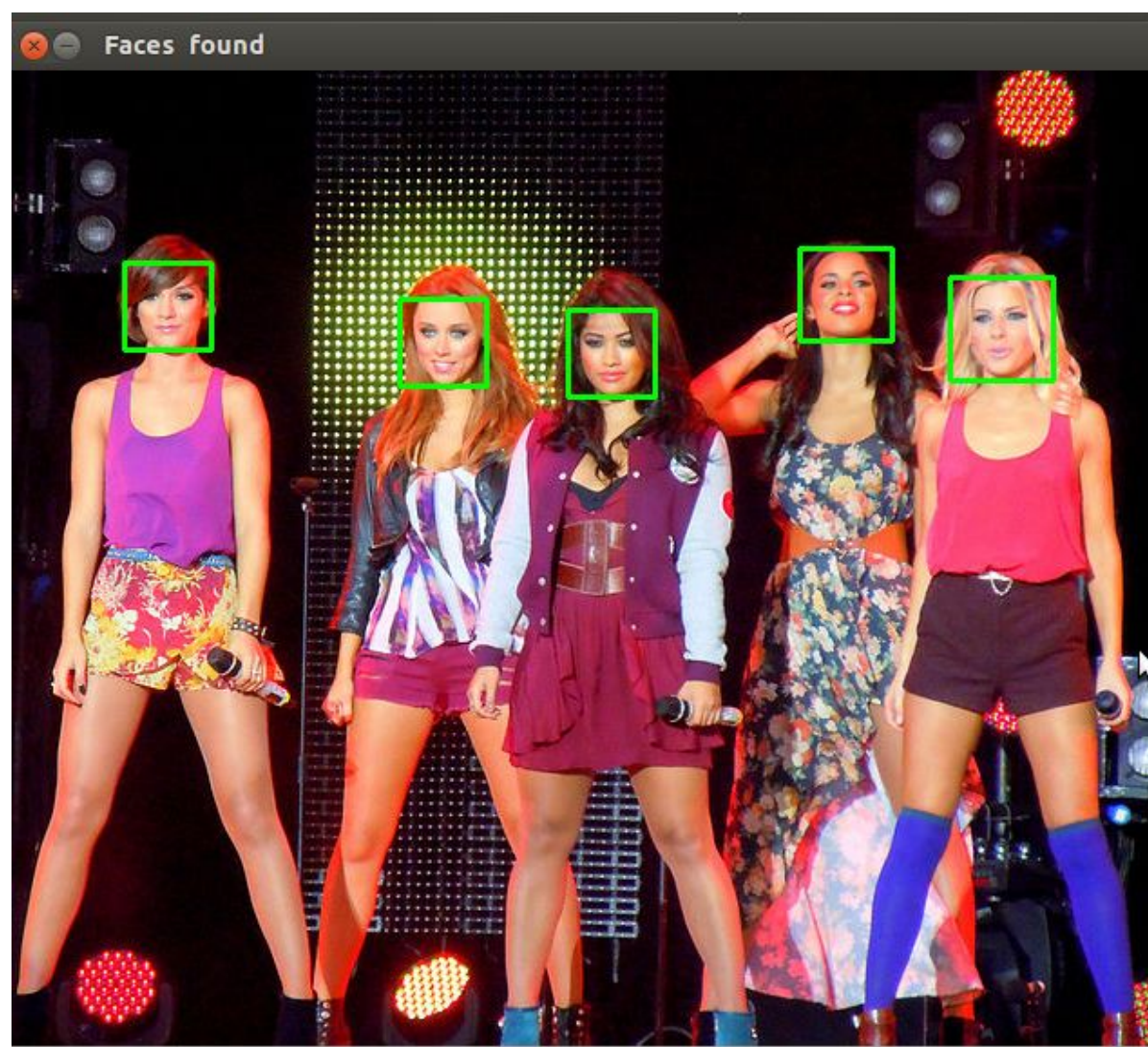


That worked. How about another photo:

[Improve Your Python](#)



That ... is not a face. Let's try again. I changed the parameters and found that setting the `scaleFactor` to 1.2 got rid of the wrong face.



What Happened?

Well, the first photo was taken fairly close up with a high quality camera. The second one seems to have been taken from afar and possibly with a mobile phone. This is why the `scaleFactor` had to be modified. As I said, you'll have to set up the algorithm on a case-by-case basis to avoid false positives.

Be warned though that since this is based on machine learning, the results will never be 100% accurate. You will get good enough results in most cases, but occasionally the algorithm will identify incorrect objects as faces.

The final code can be found [here](https://realpython.com/face-recognition-with-python/).

Extending to a Webcam

What if you want to use a webcam? OpenCV grabs each frame from the webcam, and you can then detect faces by processing each frame. You will need a powerful computer, but my five-year-old laptop seems to cope fine, as long as I don't dance around too much.

Update: The next article is live. Check out [Face Detection in Python Using a Webcam!](#)

Want to Know More?



Free Bonus: [Click here to get the Python practical code examples of real-world](#)

I will be covering this and more in my upcoming [Kickstarter](#). I will also cover machine learning

Thanks!

```
1 # How to merge two dicts
2 # in Python 3.5+
3
4 >>> x = {'a': 1, 'b': 2}
5 >>> y = {'b': 3, 'c': 4}
6
7 >>> z = {**x, **y}
8
9 >>> z
10 {'c': 4, 'a': 1, 'b': 3}
```

Improve Your Python

...with a fresh  **Python Trick**  code snippet every couple of days:

Email Address

Send Python Tricks »



Get a short & sweet **Python Trick** delivered to your inbox every couple of days. No spam ever. Unsubscribe any time. Curated by the Real Python team.

Email Address

Send Me Python Tricks »

```
1 # How to merge two dicts
2 # in Python 3.5+
3
4 >>> x = {'a': 1, 'b': 2}
5 >>> y = {'b': 3, 'c': 4}
6
7 >>> z = {**x, **y}
8
9 >>> z
10 {'c': 4, 'a': 1, 'b': 3}
```

About Shantnu Tiwari

Shantnu has worked in the low level/embedded domain for ten years. Shantnu suffered at the hands of C/C++ for several years before he discovered Python, and it felt like a breath of fresh air.

[» More about Shantnu](#)

Each tutorial at Real Python is created by a team of developers so that it meets our high quality standards. The team members who worked on this tutorial are:

Improve Your Python

[Aldren](#)

[Joanna](#)

Master Real-World Python Skills With Unlimited Access to Real Python



```
1# How to merge two dicts
2# in Python 3.5+
3
4>>> x = {'a': 1, 'b': 2}
5>>> y = {'b': 3, 'c': 4}
6
7>>> z = {**x, **y}
8
9>>> z
10{'c': 4, 'a': 1, 'b': 3}
```

Join us an
on vici

Pythonistas:

Level Up Your Python Skills »

Improve Your Python



...with a fresh 🐍 **Python Trick** 📧
code snippet every couple of days:

Email Address

Send Python Tricks »

What Do You Think?

Tweet Share Email

Real Python Comment Policy: The most useful comments are those written with the goal of learning from or helping out other readers—after reading the whole article and all the earlier comments. Complaints and insults generally won’t make the cut here.

What’s your #1 takeaway or favorite thing you learned? How are you going to put your newfound skills to use? Leave a comment below and let us know.

Keep Learning

Related Tutorial Categories: [data-science](#) [machine-learning](#)

— FREE Email Series —

Python Tricks

```
1# How to merge two dicts
2# in Python 3.5+
3
4>>> x = {'a': 1, 'b': 2}
5>>> y = {'b': 3, 'c': 4}
6
7>>> z = {**x, **y}
8
9>>> z
```

Improve Your Python


```
9 >>> z
10 {'c': 4, 'a': 1, 'b': 3}
```

Email...

Get Python Tricks »

 No spam. Unsubscribe any time.

All Tutorial Topics

- advanced

devops

projects
- api

django

python
- b

```
1 # How to merge two dicts
2 # in Python 3.5+
3
4 >>> x = {'a': 1, 'b': 2}
5 >>> y = {'b': 3, 'c': 4}
6
7 >>> z = {**x, **y}
8
9 >>> z
10 {'c': 4, 'a': 1, 'b': 3}
```

Improve Your Python



...with a fresh  **Python Trick** 
code snippet every couple of days:

Email Address

Send Python Tricks »



Table of Contents

- [OpenCV](#)
- [Installing OpenCV](#)
- [Understanding the Code](#)
- [Checking the Results](#)
- [Extending to a Webcam](#)
- [Want to Know More?](#)

-  Tweet
-  Share
-  Email




```
1# How to merge two dicts
2# in Python 3.5+
3
4>>> x = {'a': 1, 'b': 2}
5>>> y = {'b': 3, 'c': 4}
6
7>>> z = {**x, **y}
8
9>>> z
10{'c': 4, 'a': 1, 'b': 3}
```

Improve Your Python ×

...with a fresh 🐍 **Python Trick** 📧
code snippet every couple of days:

[Send Python Tricks »](#)