# Find faces in pictures

We start by loading an example picture using Python imaging library. Go ahead and **run the cell bellow** to see the image.

from PIL import Image, ImageDraw  
from IPython.display import display  
  
# The program we will be finding faces on the example below  
pil\_im = Image.open('two\_people.jpg')  
display(pil\_im)

# Learning from example

Now we show the library two different faces ([Joe Biden](/biden.jpg), [Barack Obama](/obama.jpg)) and generate the encodings for them. Encoding is simply a low dimensional representation of a face that can be easily compared with other faces the library will recognize in the future.

import face\_recognition  
import numpy as np  
from PIL import Image, ImageDraw  
from IPython.display import display  
  
# This is an example of running face recognition on a single image  
# and drawing a box around each person that was identified.  
  
# Load a sample picture and learn how to recognize it.  
obama\_image = face\_recognition.load\_image\_file("obama.jpg")  
obama\_face\_encoding = face\_recognition.face\_encodings(obama\_image)[0]  
  
# Load a second sample picture and learn how to recognize it.  
biden\_image = face\_recognition.load\_image\_file("biden.jpg")  
biden\_face\_encoding = face\_recognition.face\_encodings(biden\_image)[0]  
  
# Create arrays of known face encodings and their names  
known\_face\_encodings = [  
 obama\_face\_encoding,  
 biden\_face\_encoding  
]  
known\_face\_names = [  
 "Barack Obama",  
 "Joe Biden"  
]  
print('Learned encoding for', len(known\_face\_encodings), 'images.')

# Showtime

Finally, we load the image we looked at in the first cell, find the faces in the image and compare them with the encodings the library generated in the previous step. We can see that library now correctly recognizes Barack and Joe in the input.

# Load an image with an unknown face  
unknown\_image = face\_recognition.load\_image\_file("two\_people.jpg")  
  
# Find all the faces and face encodings in the unknown image  
face\_locations = face\_recognition.face\_locations(unknown\_image)  
face\_encodings = face\_recognition.face\_encodings(unknown\_image, face\_locations)  
  
# Convert the image to a PIL-format image so that we can draw on top of it with the Pillow library  
# See http://pillow.readthedocs.io/ for more about PIL/Pillow  
pil\_image = Image.fromarray(unknown\_image)  
# Create a Pillow ImageDraw Draw instance to draw with  
draw = ImageDraw.Draw(pil\_image)  
  
# Loop through each face found in the unknown image  
for (top, right, bottom, left), face\_encoding in zip(face\_locations, face\_encodings):  
 # See if the face is a match for the known face(s)  
 matches = face\_recognition.compare\_faces(known\_face\_encodings, face\_encoding)  
  
 name = "Unknown"  
  
 # Or instead, use the known face with the smallest distance to the new face  
 face\_distances = face\_recognition.face\_distance(known\_face\_encodings, face\_encoding)  
 best\_match\_index = np.argmin(face\_distances)  
 if matches[best\_match\_index]:  
 name = known\_face\_names[best\_match\_index]  
  
 # Draw a box around the face using the Pillow module  
 draw.rectangle(((left, top), (right, bottom)), outline=(0, 0, 255))  
  
 # Draw a label with a name below the face  
 text\_width, text\_height = draw.textsize(name)  
 draw.rectangle(((left, bottom - text\_height - 10), (right, bottom)), fill=(0, 0, 255), outline=(0, 0, 255))  
 draw.text((left + 6, bottom - text\_height - 5), name, fill=(255, 255, 255, 255))  
  
  
# Remove the drawing library from memory as per the Pillow docs  
del draw  
  
# Display the resulting image  
display(pil\_image)