

```
1 # Find faces in picture
2 # https://github.com/ageitgey/face_recognition/blob/master/examples/find_faces_in_picture.py
3
4 from PIL import Image
5 import face_recognition
6
7 # Load the jpg file into a numpy array
8 image = face_recognition.load_image_file("office.jpg")
9
10 # Find all the faces in the image using the default HOG-based model.
11 # This method is fairly accurate, but not as accurate as the CNN model and not GPU accelerated.
12 # See also: find_faces_in_picture_cnn.py
13 face_locations = face_recognition.face_locations(image)
14
15 for face_location in face_locations:
16
17     # Print the location of each face in this image
18     top, right, bottom, left = face_location
19
20     # You can access the actual face itself like this:
21     face_image = image[top:bottom, left:right]
22     pil_image = Image.fromarray(face_image)
23     pil_image.show()
```

```
1 # Identify and draw box on David
2 # https://github.com/ageitgey/face_recognition/blob/master/examples/identify_and_draw_boxes_on_faces.py
3
4 import face_recognition
5 import numpy as np
6 from PIL import Image, ImageDraw
7
8 # Load a sample picture and learn how to recognize it.
9 known_image = face_recognition.load_image_file("toby.jpg")
10 encoding = face_recognition.face_encodings(known_image)[0]
11
12 # Load an image with unknown faces
13 unknown_image = face_recognition.load_image_file("office.jpg")
14
15 # Find all the faces and face encodings in the unknown image
16 face_locations = face_recognition.face_locations(unknown_image)
17 face_encodings = face_recognition.face_encodings(unknown_image, face_locations)
18
19 # Convert the image to a PIL-format image so that we can draw on top of it with the Pillow library
20 # See http://pillow.readthedocs.io/ for more about PIL/Pillow
21 pil_image = Image.fromarray(unknown_image)
22
23 # Create a Pillow ImageDraw Draw instance to draw with
24 draw = ImageDraw.Draw(pil_image)
25
26 # Loop through each face found in the unknown image
27 for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
28
29     # See if the face is a match for the known face(s)
30     matches = face_recognition.compare_faces([encoding], face_encoding)
31
32     # Use the known face with the smallest distance to the new face
33     face_distances = face_recognition.face_distance([encoding], face_encoding)
34     best_match_index = np.argmin(face_distances)
35     if matches[best_match_index]:
36
37         # Draw a box around the face using the Pillow module
38         draw.rectangle(((left - 20, top - 20), (right + 20, bottom + 20)), outline=(0, 255, 0), width=20)
39
40 # Remove the drawing library from memory as per the Pillow docs
41 del draw
42
```

```
43 # Display the resulting image
44 pil_image.show()
```

```
1 # Words in dictionary
2 words = set()
3
4
5 def check(word):
6     """Return true if word is in dictionary else false"""
7     if word.lower() in words:
8         return True
9     else:
10        return False
11
12
13 def load(dictionary):
14     """Load dictionary into memory, returning true if successful else false"""
15     file = open(dictionary, "r")
16     for line in file:
17         word = line.rstrip()
18         words.add(word)
19     file.close()
20     return True
21
22
23 def size():
24     """Returns number of words in dictionary if loaded else 0 if not yet loaded"""
25     return len(words)
26
27
28 def unload():
29     """Unloads dictionary from memory, returning true if successful else false"""
30     return True
```

```
1 // A program that says hello to the world
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     printf("hello, world\n");
8 }
```

```
1 // get_string and printf with %s
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     string answer = get_string("What's your name? ");
9     printf("hello, %s\n", answer);
10 }
```

```
1 # A program that says hello to the world
2
3 print("hello, world")
```

```
1 # get_string and print, with concatenation
2
3 from cs50 import get_string
4
5 answer = get_string("What's your name? ")
6 print("hello, " + answer)
```

```
1 # get_string and print, with format strings
2
3 from cs50 import get_string
4
5 answer = get_string("What's your name? ")
6 print(f"hello, {answer}")
```

```
1 # input and print, with format strings
2
3 answer = input("What's your name? ")
4 print(f"hello, {answer}")
```

```
1 // Addition with int
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Prompt user for x
9     int x = get_int("x: ");
10
11     // Prompt user for y
12     int y = get_int("y: ");
13
14     // Perform addition
15     printf("%i\n", x + y);
16 }
```

```
1 # Addition with int [using get_int]
2
3 from cs50 import get_int
4
5 # Prompt user for x
6 x = get_int("x: ")
7
8 # Prompt user for y
9 y = get_int("y: ")
10
11 # Perform addition
12 print(x + y)
```

```
1 # Addition with int [using input]
2
3 # Prompt user for x
4 x = int(input("x: "))
5
6 # Prompt user for y
7 y = int(input("y: "))
8
9 # Perform addition
10 print(x + y)
```

```
1 # Division with integers, demonstration lack of truncation
2
3 # Prompt user for x
4 x = int(input("x: "))
5
6 # Prompt user for y
7 y = int(input("y: "))
8
9 # Divide x by y
10 z = x / y
11 print(z)
```

```
1 # Floating-point imprecision
2
3 # Prompt user for x
4 x = int(input("x: "))
5
6 # Prompt user for y
7 y = int(input("y: "))
8
9 # Divide x by y
10 z = x / y
11 print(f"{z:.50f}")
```

```
1 // Conditionals, Boolean expressions, relational operators
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Prompt user for integers
9     int x = get_int("What's x? ");
10    int y = get_int("What's y? ");
11
12    // Compare integers
13    if (x < y)
14    {
15        printf("x is less than y\n");
16    }
17    else if (x > y)
18    {
19        printf("x is greater than y\n");
20    }
21    else
22    {
23        printf("x is equal to y\n");
24    }
25 }
```



```
1 # Conditionals, Boolean expressions, relational operators
2
3 from cs50 import get_int
4
5 # Prompt user for integers
6 x = get_int("What's x? ")
7 y = get_int("What's y? ")
8
9 # Compare integers
10 if x < y:
11     print("x is less than y")
12 elif x > y:
13     print("x is greater than y")
14 else:
15     print("x is equal to y")
```

```
1 // Logical operators
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Prompt user to agree
9     char c = get_char("Do you agree? ");
10
11     // Check whether agreed
12     if (c == 'Y' || c == 'y')
13     {
14         printf("Agreed.\n");
15     }
16     else if (c == 'N' || c == 'n')
17     {
18         printf("Not agreed.\n");
19     }
20 }
```

```
1 # Logical operators
2
3 from cs50 import get_string
4
5 # Prompt user to agree
6 s = get_string("Do you agree? ")
7
8 # Check whether agreed
9 if s == "Y" or s == "y":
10     print("Agreed.")
11 elif s == "N" or s == "n":
12     print("Not agreed.")
```

```
1 # Logical operators, using lists
2
3 from cs50 import get_string
4
5 # Prompt user to agree
6 s = get_string("Do you agree? ")
7
8 # Check whether agreed
9 if s.lower() in ["y", "yes"]:
10     print("Agreed.")
11 elif s.lower() in ["n", "no"]:
12     print("Not agreed.")
```

```
1 // Opportunity for better design
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     printf("meow\n");
8     printf("meow\n");
9     printf("meow\n");
10 }
```

```
1 # Opportunity for better design
2
3 print("meow")
4 print("meow")
5 print("meow")
```

```
1 // Demonstrates while loop
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     int i = 0;
8     while (i < 3)
9     {
10        printf("meow\n");
11        i++;
12    }
13 }
```

```
1 # Demonstrates while loop
2
3 i = 0
4 while i < 3:
5     print("meow")
6     i += 1
```

```
1 // Demonstrates for loop
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     for (int i = 0; i < 3; i++)
8     {
9         printf("meow\n");
10    }
11 }
```

```
1 # Opportunity for better design
2
3 for i in [0, 1, 2]:
4     print("meow")
```

```
1 # Better design
2
3 for i in range(3):
4     print("meow")
```

```
1 // Abstraction
2
3 #include <stdio.h>
4
5 void meow(void);
6
7 int main(void)
8 {
9     for (int i = 0; i < 3; i++)
10    {
11        meow();
12    }
13 }
14
15 // Meow once
16 void meow(void)
17 {
18     printf("meow\n");
19 }
```

```
1 # Abstraction
2
3 def main():
4     for i in range(3):
5         meow()
6
7 # Meow once
8 def meow():
9     print("meow")
10
11
12 main()
```

```
1 // Abstraction with parameterization
2
3 #include <stdio.h>
4
5 void meow(int n);
6
7 int main(void)
8 {
9     meow(3);
10 }
11
12 // Meow some number of times
13 void meow(int n)
14 {
15     for (int i = 0; i < n; i++)
16     {
17         printf("meow\n");
18     }
19 }
```

```
1 # Abstraction with parameterization
2
3 def main():
4     meow(3)
5
6
7 # Meow some number of times
8 def meow(n):
9     for i in range(n):
10        print("meow")
11
12
13 main()
```

```
1 # Prints a column of 3 bricks with a loop
2
3 for i in range(3):
4     print("#")
```



```
1 // Prints a column of height n
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Get height of column
9     int n;
10    do
11    {
12        n = get_int("Height: ");
13    }
14    while (n < 1);
15
16    // Print column of bricks
17    for (int i = 0; i < n; i++)
18    {
19        printf("#\n");
20    }
21 }
```

```
1 # Prints a column of n bricks with a loop
2
3 from cs50 import get_int
4
5 while True:
6     n = get_int("Height: ")
7     if n > 0:
8         break
9
10 for i in range(n):
11     print("#")
```

```
1 # Prints a column of bricks, using a helper function to get input
2
3 from cs50 import get_int
4
5
6 def main():
7     height = get_height()
8     for i in range(height):
9         print("#")
10
11
12 def get_height():
13     while True:
14         n = get_int("Height: ")
15         if n > 0:
16             return n
17
18
19 main()
```

```
1 # Prints a column of bricks, catching exceptions
2
3
4 def main():
5     height = get_height()
6     for i in range(height):
7         print("#")
8
9
10 def get_height():
11     while True:
12         try:
13             n = int(input("Height: "))
14             if n > 0:
15                 return n
16         except ValueError:
17             print("Not an integer")
18
19
20 main()
```

```
1 # Prints a row of 4 question marks with a loop
2
3 for i in range(4):
4     print("?", end="")
5 print()
```

```
1 # Prints a row of 4 question marks without a loop
2
3 print("?" * 4)
```

```
1 # Prints a 3-by-3 grid of bricks with loops
2
3 for i in range(3):
4     for j in range(3):
5         print("#", end="")
6     print()
```

```
1 # Prints a 3-by-3 grid of bricks with loop and * operator
2
3 for i in range(3):
4     print("#" * 3)
```

```
1 # Averages three numbers using a list
2
3 # Scores
4 scores = [72, 73, 33]
5
6 # Print average
7 average = sum(scores) / len(scores)
8 print(f"Average: {average}")
```

```
1 # Averages three numbers using a list and a loop
2
3 from cs50 import get_int
4
5 # Get scores
6 scores = []
7 for i in range(3):
8     score = get_int("Score: ")
9     scores.append(score)
10
11 # Print average
12 average = sum(scores) / len(scores)
13 print(f"Average: {average}")
```

```
1 # Averages three numbers using a list and a loop with + operator
2
3 from cs50 import get_int
4
5 # Get scores
6 scores = []
7 for i in range(3):
8     score = get_int("Score: ")
9     scores += [score]
10
11 # Print average
12 average = sum(scores) / len(scores)
13 print(f"Average: {average}")
```

```
1 # Uppercases string one character at a time
2
3 before = input("Before: ")
4 print("After: ", end="")
5 for c in before:
6     print(c.upper(), end="")
7 print()
```

```
1 # Uppercases string all at once
2
3 before = input("Before: ")
4 after = before.upper()
5 print(f"After: {after}")
```

```
1 # Prints a command-line argument
2
3 from sys import argv
4
5 if len(argv) == 2:
6     print(f"hello, {argv[1]}")
7 else:
8     print("hello, world")
```

```
1 # Printing command-line arguments, indexing into argv
2
3 from sys import argv
4
5 for i in range(len(argv)):
6     print(argv[i])
```

```
1 # Printing command-line arguments
2
3 from sys import argv
4
5 for arg in argv:
6     print(arg)
```

```
1 # Exits with explicit value, importing sys
2
3 import sys
4
5 if len(sys.argv) != 2:
6     print("Missing command-line argument")
7     sys.exit(1)
8
9 print(f"hello, {sys.argv[1]}")
10 sys.exit(0)
```

```
1 # Implements linear search for names using loop
2
3 import sys
4
5 # A list of names
6 names = ["Bill", "Charlie", "Fred", "George", "Ginny", "Percy", "Ron"]
7
8 # Ask for name
9 name = input("Name: ")
10
11 # Search for name
12 for n in names:
13     if name == n:
14         print("Found")
15         sys.exit(0)
16
17 print("Not found")
18 sys.exit(1)
```

```
1 # Implements linear search for names using `in`
2
3 import sys
4
5 # A list of names
6 names = ["Bill", "Charlie", "Fred", "George", "Ginny", "Percy", "Ron"]
7
8 # Ask for name
9 name = input("Name: ")
10
11 # Search for name
12 if name in names:
13     print("Found")
14     sys.exit(0)
15
16 print("Not found")
17 sys.exit(1)
```

```
1 # Implements linear search for names
2
3 import sys
4
5 # A list of names
6 names = ["Bill", "Charlie", "Fred", "George", "Ginny", "Percy", "Ron"]
7
8 # Ask for name
9 name = input("Name: ")
10
11 # Search for name
12 if name in names:
13     print("Found")
14     sys.exit(0)
15
16 print("Not found")
17 sys.exit(1)
```

```
1 # Implements a phone book
2
3 from cs50 import get_string
4
5 people = {
6     "Carter": "+1-617-495-1000",
7     "David": "+1-949-468-2750"
8 }
9
10 # Search for name
11 name = get_string("Name: ")
12 if name in people:
13     print(f"Number: {people[name]}")
```

```
1 # Compares two strings
2
3 # Get two strings
4 s = input("s: ")
5 t = input("t: ")
6
7 # Compare strings
8 if s == t:
9     print("Same")
10 else:
11     print("Different")
```

```
1 # Capitalizes a copy of a string
2
3 # Get a string
4 s = input("s: ")
5
6 # Capitalize copy of string
7 t = s.capitalize()
8
9 # Print strings
10 print(f"s: {s}")
11 print(f"t: {t}")
```

```
1 # Swaps two integers
2
3 x = 1
4 y = 2
5
6 print(f"x is {x}, y is {y}")
7 x, y = y, x
8 print(f"x is {x}, y is {y}")
```



```
1 # Saves names and numbers to a CSV file
2
3 import csv
4
5 # Open CSV file
6 file = open("phonebook.csv", "a")
7
8 # Get name and number
9 name = input("Name: ")
10 number = input("Number: ")
11
12 # Print to file
13 writer = csv.writer(file)
14 writer.writerow([name, number])
15
16 # Close file
17 file.close()
```

```
1 # Saves names and numbers to a CSV file
2
3 import csv
4
5 # Get name and number
6 name = input("Name: ")
7 number = input("Number: ")
8
9 # Open CSV file
10 with open("phonebook.csv", "a") as file:
11
12     # Print to file
13     writer = csv.writer(file)
14     writer.writerow([name, number])
```

```
1 # Saves names and numbers to a CSV file using a DictWriter
2
3 import csv
4
5 # Get name and number
6 name = input("Name: ")
7 number = input("Number: ")
8
9 # Open CSV file
10 with open("phonebook.csv", "a") as file:
11
12     # Print to file
13     writer = csv.DictWriter(file, fieldnames=["name", "number"])
14     writer.writerow({"name": name, "number": number})
```

```
1 # Logical operators, using regular expressions
2
3 import re
4
5 from cs50 import get_string
6
7 # Prompt user to agree
8 s = get_string("Do you agree? ")
9
10 # Check whether agreed
11 if re.search("^y(es)?$", s, re.IGNORECASE):
12     print("Agreed.")
13 elif re.search("^no?$", s, re.IGNORECASE):
14     print("Not agreed.")
```

```
1 # Says hello
2
3 import pyttsx3
4
5 engine = pyttsx3.init()
6 engine.say("hello, world")
7 engine.runAndWait()
```

```
1 # Says hello to someone
2
3 import pyttsx3
4
5 engine = pyttsx3.init()
6 name = input("What's your name? ")
7 engine.say(f"hello, {name}")
8 engine.runAndWait()
```

```
1 # Generates a QR code
2 # https://github.com/lincolnloop/python-qrcode
3
4 import os
5 import qrcode
6
7 # Generate QR code
8 img = qrcode.make("https://youtu.be/xvFZjo5PgG0")
9
10 # Save as file
11 img.save("qr.png", "PNG")
12
13 # Open file
14 os.system("open qr.png")
```

```
1 # Says "This was CS50"
2
3 import pyttsx3
4
5 engine = pyttsx3.init()
6 engine.say("This was CS50")
7 engine.runAndWait()
```