

---

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

---

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

---

```
1  # Logical operators
2
3  # Prompt user for answer
4  c = input("Answer: ")
5
6  # Check answer
7  if c == "Y" or c == "y":
8      print("yes")
9  elif c == "N" or c == "n":
10     print("no")
```

---

```
1  # Conditions and relational operators
2
3  # Prompt user for x
4  x = int(input("x: "))
5
6  # Prompt user for y
7  y = int(input("y: "))
8
9  # Compare x and y
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  # Opportunity for better design
2
3  print("cough")
4  print("cough")
5  print("cough")
```

---

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

---

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

---

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



```
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("yale.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("malan.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("harvard.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  # Says hello to the world
2
3  print("hello, world")
```

---

```
1  # Says hello to someone
2
3  name = input("Name: ")
4  print("hello,", name)
```

---

```
1 # Floating-point imprecision
2
3 print(f"{1/10:.50f}")
```

---

```
1  # Abstraction and scope
2
3
4  def main():
5      i = get_positive_int("Positive integer: ")
6      print(i)
7
8
9  def get_positive_int(prompt):
10     while True:
11         n = int(input(prompt))
12         if n > 0:
13             break
14     return n
15
16
17 main()
```

---

```
1  # Generates a QR code
2  # https://github.com/lincolnloop/python-qrcode
3
4  import qrcode
5
6  # Generate QR code
7  img = qrcode.make("https://youtu.be/oHg5SJYRHA0")
8
9  # Save as file
10 img.save("qr.png", "PNG")
```

```
1  # Generates a bar chart of three scores
2
3  # Get scores from user
4  score1 = int(input("Score 1: "))
5  score2 = int(input("Score 2: "))
6  score3 = int(input("Score 3: "))
7
8  # Generate first bar
9  print("Score 1: ", end="");
10 for i in range(score1):
11     print("#", end="")
12 print()
13
14 # Generate second bar
15 print("Score 2: ", end="");
16 for i in range(score2):
17     print("#", end="")
18 print()
19
20 # Generate third bar
21 print("Score 3: ", end="");
22 for i in range(score3):
23     print("#", end="")
24 print()
```