```python
# Logical operators

# Prompt user to agree
s = input("Do you agree? ")

# Check whether agreed
if s == "Y" or s == "y":
    print("Agreed.")
elif s == "N" or s == "n":
    print("Not agreed.")
```

```python
# Logical operators, using lists

# Prompt user to agree
s = input("Do you agree? ")

# Check whether agreed
if s.lower() in ["y", "yes"]:
    print("Agreed.")
elif s.lower() in ["n", "no"]:
    print("Not agreed.")
```

```python
# Logical operators

# Prompt user for answer
c = input("Answer: ")

# Check answer
if c == "Y" or c == "y":
    print("yes")
elif c == "N" or c == "n":
    print("no")
```

```python
# Conditions and relational operators

# Prompt user for x
x = int(input("x: "))

# Prompt user for y
y = int(input("y: "))

# Compare x and y
if x < y:
    print("x is less than y")
elif x > y:
    print("x is greater than y")
else:
    print("x is equal to y")
```

```python
# Opportunity for better design

print("cough")
print("cough")
print("cough")
```

```python
# Better design

for i in range(3):
    print("cough")
```

```python
# Abstraction


def main():
    for i in range(3):
        cough()


def cough():
    print("cough")


main()
```

```python
# Abstraction with parameterization


def main():
    cough(3)


def cough(n):
    for i in range(n):
        print("cough")


main()
```

```python
# Find faces in picture
# https://github.com/ageitgey/face_recognition/blob/master/examples/find_faces_in_picture.py

from PIL import Image
import face_recognition

# Load the jpg file into a numpy array
image = face_recognition.load_image_file("yale.jpg")

# Find all the faces in the image using the default HOG-based model.
# This method is fairly accurate, but not as accurate as the CNN model and not GPU accelerated.
# See also: find_faces_in_picture_cnn.py
face_locations = face_recognition.face_locations(image)

for face_location in face_locations:

    # Print the location of each face in this image
    top, right, bottom, left = face_location

    # You can access the actual face itself like this:
    face_image = image[top:bottom, left:right]
    pil_image = Image.fromarray(face_image)
    pil_image.show()
```

```python
# Identify and draw box on David
# https://github.com/ageitgey/face_recognition/blob/master/examples/identify_and_draw_boxes_on_faces.py

import face_recognition
import numpy as np
from PIL import Image, ImageDraw

# Load a sample picture and learn how to recognize it.
known_image = face_recognition.load_image_file("malan.jpg")
encoding = face_recognition.face_encodings(known_image)[0]

# Load an image with unknown faces
unknown_image = face_recognition.load_image_file("harvard.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([encoding], face_encoding)

    # Use the known face with the smallest distance to the new face
    face_distances = face_recognition.face_distance([encoding], face_encoding)
    best_match_index = np.argmin(face_distances)
    if matches[best_match_index]:

        # Draw a box around the face using the Pillow module
        draw.rectangle(((left - 20, top - 20), (right + 20, bottom + 20)), outline=(0, 255, 0), width=20)

# Remove the drawing library from memory as per the Pillow docs
del draw

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

```python
# Says hello to the world

print("hello, world")
```

```python
# Says hello to someone

name = input("Name: ")
print("hello,", name)
```

```python
# Floating-point imprecision

print(f"{1/10:.50f}")
```

```python
# Prints a row of 4 question marks with a loop

for i in range(4):
    print("?", end="")
print()
```

```python
# Prints a row of 4 question marks without a loop

print("?" * 4)
```

```python
# Prints a column of 3 bricks with a loop

for i in range(3):
    print("#")
```

```python
# Prints a column of 3 bricks without a loop

print("#\n" * 3, end="")
```

```python
# Prints a 3-by-3 grid of bricks with loops

for i in range(3):
    for j in range(3):
        print("#", end="")
    print()
```

```python
# Abstraction and scope


def main():
    i = get_positive_int("Positive integer: ")
    print(i)


def get_positive_int(prompt):
    while True:
        n = int(input(prompt))
        if n > 0:
            break
    return n


main()
```

```python
# Generates a QR code
# https://github.com/lincolnloop/python-qrcode

import qrcode

# Generate QR code
img = qrcode.make("https://youtu.be/oHg5SJYRHA0")

# Save as file
img.save("qr.png", "PNG")
```

```python
# Generates a bar chart of three scores

# Get scores from user
score1 = int(input("Score 1: "))
score2 = int(input("Score 2: "))
score3 = int(input("Score 3: "))

# Generate first bar
print("Score 1: ", end="");
for i in range(score1):
    print("#", end="")
print()

# Generate second bar
print("Score 2: ", end="");
for i in range(score2):
    print("#", end="")
print()

# Generate third bar
print("Score 3: ", end="");
for i in range(score3):
    print("#", end="")
print()
```