PYTHON CODE TO BLINK THE LED USING RASBERRY PI:

Before we start writing the software we first need to install the Raspberry Pi GPIO Python module. This is a library that allows us to access the GPIO port directly from Python.

To install the Python library open a terminal and execute the following:

```
$ sudo apt-get install python-rpi.gpio python3-rpi.gpio
```

With the library installed now open your favourite Python IDE (I recommend Thonny Python IDE more information about using it here).

Our script needs to do the following:

- Initialize the GPIO ports
- Turn the LED on and off in 1 second intervals

To initialize the GPIO ports on the Raspberry Pi we need to first import the Python library, the initialize the library and setup pin 8 as an output pin.

```
import RPi.GPIO as GPIO   # Import Raspberry Pi GPIO library

from time import sleep    # Import the sleep function from the time module

GPIO.setwarnings(False)   # Ignore warning for now

GPIO.setmode(GPIO.BOARD)  # Use physical pin numbering

GPIO.setup(8, GPIO.OUT, initial=GPIO.LOW)  # Set pin 8 to be an output pin and set initial value to
 low (off)
```

Next we need to turn the LED on and off in 1 second intervals by setting the output pin to either high (on) or low (off). We do this inside a infinite loop so our program keep executing until we manually stop it.

```
while True: # Run forever

  GPIO.output(8, GPIO.HIGH) # Turn on

  sleep(1)              # Sleep for 1 second

  GPIO.output(8, GPIO.LOW)  # Turn off
```

```
sleep(1)          # Sleep for 1 second
```

Combining the initialization and the blink code should give you the following full Python program:

```
import RPi.GPIO as GPIO # Import Raspberry Pi GPIO library

from time import sleep # Import the sleep function from the time module

GPIO.setwarnings(False) # Ignore warning for now

GPIO.setmode(GPIO.BOARD) # Use physical pin numbering

GPIO.setup(8, GPIO.OUT, initial=GPIO.LOW) # Set pin 8 to be an output pin and set initial value to low (off)

while True: # Run forever

 GPIO.output(8, GPIO.HIGH) # Turn on

 sleep(1) # Sleep for 1 second

 GPIO.output(8, GPIO.LOW) # Turn off

 sleep(1) # Sleep for 1 second
```

With our program finished, save it as blinking_led.py and run it either inside your IDE or in the console with:

```
$ python blinking_led.py
```

PYTHON CODE FOR TRAFFIC LIGHT USING RASBERRY PI:

GPIO Zero is a new Python library which provides a simple interface to everyday GPIO component. It comes installed by default in Rasbian.

Open IDLE(*I*ntegrated Development Environment), which you can use to write and run code.
Raspbian Menu Icon >> Programming >> Python 3 (IDLE).

- To create a new file in IDLE, You can click on File and then New File in IDLE's menu bar.
- Create a new file by clicking File >> New File
- Save the new file by clicking File >> Save. Save the file as *trffic.py*
- You'll need the LED Class, and to tell it that the LED is on pin 17. Write the following code in your new file.

```
1.  from gpiozero import LED
2.  led = LED(17)
```

- To make the LED switch on, type the following and press Enter

```
1.  led.on()
```

- To make it switch off you can type

```
1.  led.off()
```

Your LED should switch on and then off again. But that's not all you can do. Similarly checks the Buzzer and Button. Just import a Buzzer and Button for the header file.

Making Traffic Light

We need a breadboard, three LEDs, a button, a buzzer, and the necessary jumper cables and registers.

Wiring

First, you need to understand how each component is connected.
- A push-button requires 1 ground pin and 1 GPIO pin
- An LED requires 1 ground pin and 1 GPIO pin, with a current limiting register
- A buzzer requires 1 ground pin and 1 GPIO pin

Place the components on the breadboard and connect them to the Raspberry Pi GPIO pins, according to the following diagram.

Component GPIO pin

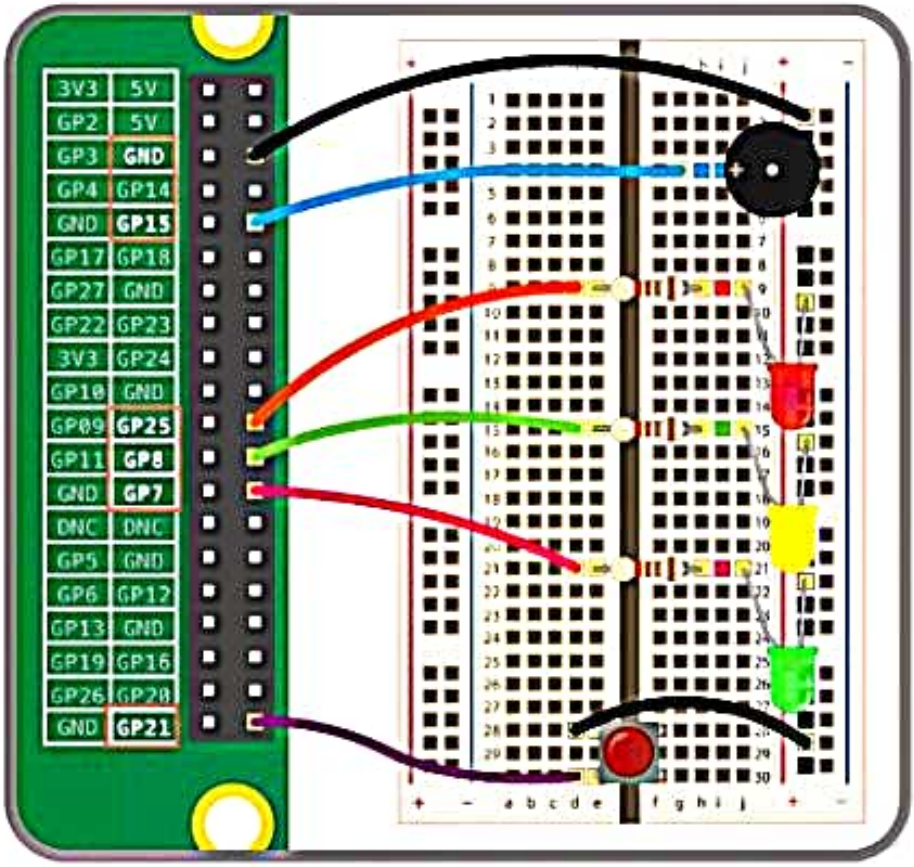
Button	21
Red LED	25
Yellow LED	8
Green LED	7
Buzzer	15

This is the same as the Switching and LED on and off step
- Open Python 3 from the main menu
- Create a new file just save with the project name.py

Add Traffic Light, Button and Buzzer Code

```
1.  from gpiozero import Button, TrafficLights, Buzzer
2.  from time import sleep
3.
4.  buzzer = Buzzer(15)
5.  button = Button(21)
6.  lights = TrafficLights(25, 8, 7)
7.
8.  while True:
9.      button.wait_for_press()
10.     buzzer.on()
11.     light.green.on()
12.     sleep(1)
13.     lights.amber.on()
14.     sleep(1)
15.     lights.red.on()
16.     sleep(1)
17.     lights.off()
18.     buzzer.off()
```

Finally, we have successfully created a smart traffic system using a Raspberry Pi.



PYTHON CODE TO BLINK THE LED USING RASBERRY PI:

Before we start writing the software we first need to install the Raspberry Pi GPIO Python module. This is a library that allows us to access the GPIO port directly from Python.

To install the Python library open a terminal and execute the following:

```
$ sudo apt-get install python-rpi.gpio python3-rpi.gpio
```

With the library installed now open your favourite Python IDE (I recommend Thonny Python IDE more information about using it here).

Our script needs to do the following:

- Initialize the GPIO ports
- Turn the LED on and off in 1 second intervals

To initialize the GPIO ports on the Raspberry Pi we need to first import the Python library, the initialize the library and setup pin 8 as an output pin.

```
import RPi.GPIO as GPIO   # Import Raspberry Pi GPIO library

from time import sleep    # Import the sleep function from the time module

GPIO.setwarnings(False)   # Ignore warning for now

GPIO.setmode(GPIO.BOARD)  # Use physical pin numbering

GPIO.setup(8, GPIO.OUT, initial=GPIO.LOW)  # Set pin 8 to be an output pin and set initial value to
low (off)
```

Next we need to turn the LED on and off in 1 second intervals by setting the output pin to either high (on) or low (off). We do this inside a infinite loop so our program keep executing until we manually stop it.

```
while True: # Run forever

  GPIO.output(8, GPIO.HIGH) # Turn on

  sleep(1)              # Sleep for 1 second

  GPIO.output(8, GPIO.LOW)  # Turn off
```