

**YouTube**

**Workshop Kit v2.0**

**Tutorial 6/9: Morse Code**

## Contents

Things you will need .....	3
Prerequisites .....	3
Wiring Diagram .....	4
Introduction .....	5
Getting Started.....	5
Save the Program.....	6
Writing the Code .....	7
Display Variables .....	7
Setting up the GPIO.....	7
Update Display Method .....	7
Set Outputs Method .....	8
Morse Code Method.....	8
The Action Code – User Input .....	8
Running the Program .....	9
Results.....	9
Code on GitHub.....	9
Thanks .....	9

## Things you will need

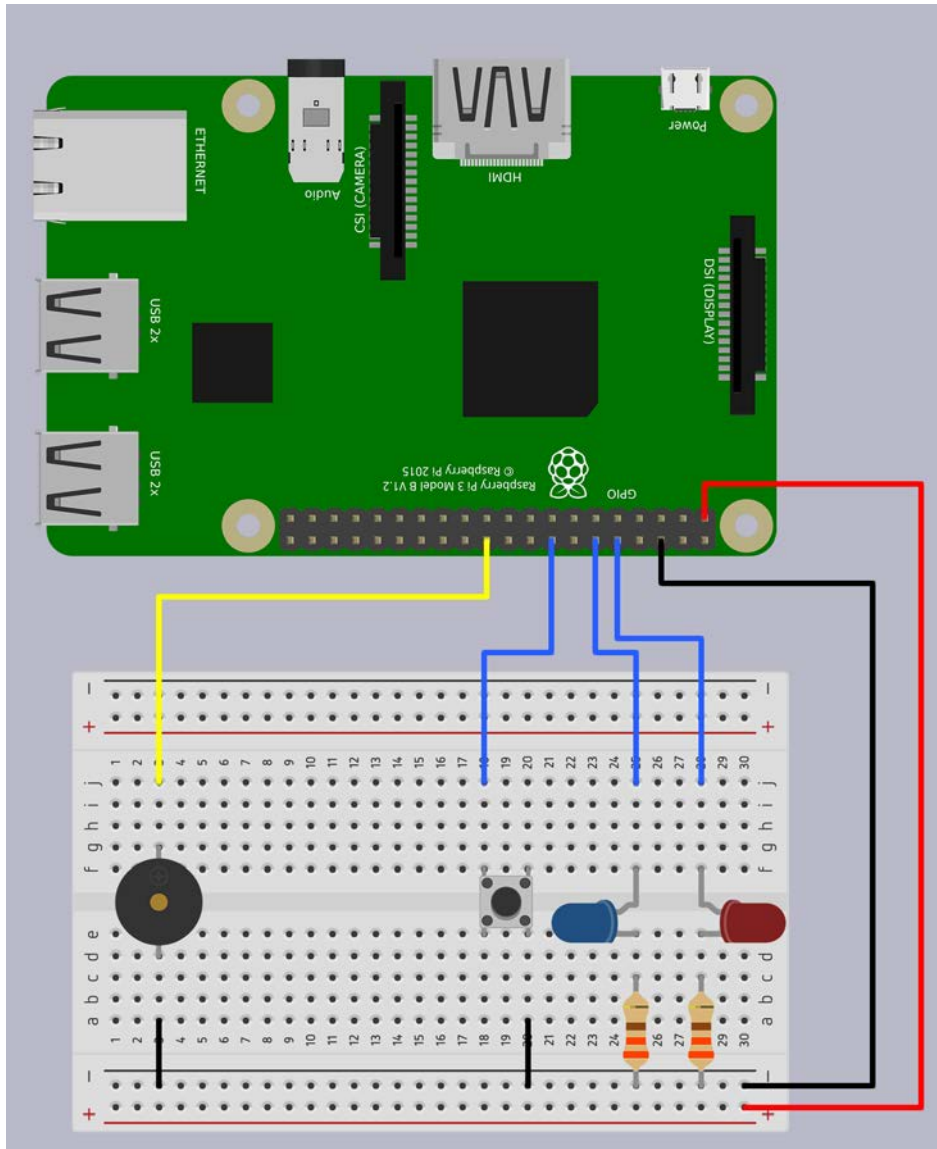
Raspberry Pi 3 Model B  
Class 10 Micro SD Card  
Keyboard + Mouse  
Monitor + HDMI Cable  
Power Supply (Recommended: 5V 2.5A)  
Breadboard  
1x Red LED  
1x Blue LED  
2x 330Ω Resistor  
2x M/M Jumper Wire  
5x M/F Jumper Wire  
1x Button  
1x Buzzer

*If you are connecting to your Raspberry Pi remotely using VNC or other means then Keyboard, mouse Monitor and HDMI cable are optional.*

## Prerequisites

You will need to install the latest version of Raspbian on to your Micro SD Card. Initial setup will require a keyboard, mouse, HDMI cable and Monitor/TV.

## Wiring Diagram



## Introduction

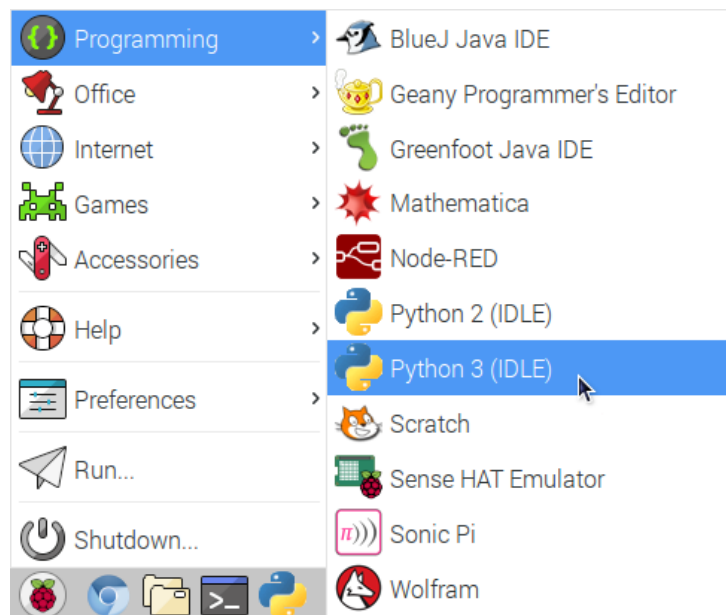
In this tutorial we will be making a buzzer and/or LED sound/flash Morse Code.

These tutorials are written on a Raspberry Pi 3 Model B with a clean install of Rasbian 4.4.

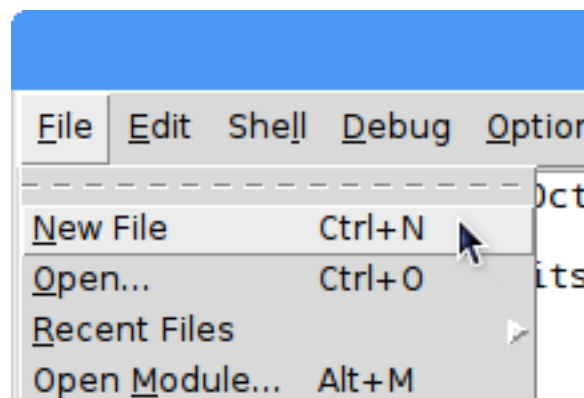
Some parts of the tutorials may look familiar as the code examples are written in a way that there is very little work needed and minor modifications to the previous code to get you up and running faster.

## Getting Started

To get started, first you need to open Python 3 (IDLE). To do this click on the Raspberry Pi icon on the task bar, highlight “Programming” then click on “Python 3 (IDLE)”

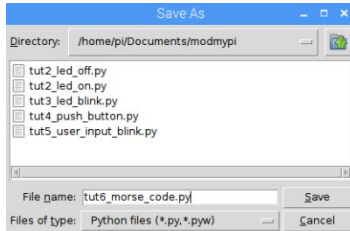


When IDLE has loaded, you will want to start working on a new file. You can do this by clicking on File and select “New File” or by pressing Ctrl+N



## Save the Program

Now that we have IDLE running, first save a new file. First open up the File Manager by clicking on this icon on the taskbar and open the Documents folder.



Go back to the IDLE and click on File and select “Save As”. Navigate to /home/pi/Documents/modmypi and enter tut6\_morse\_code.py for the filename then click Save.

## Writing the Code

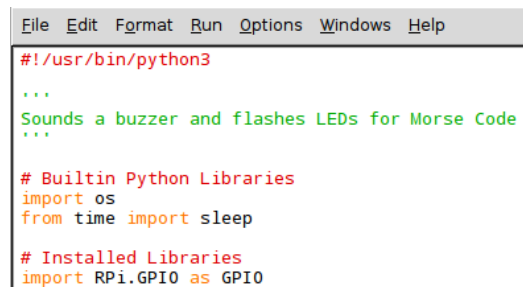
The first thing you should type is a shebang line, docstring and import your modules

```
#!/usr/bin/python3

'''
Sounds a buzzer and flashes LEDs for Morse Code
'''

# Builtin Python Libraries
import os
from time import sleep

# Installed Libraries
import RPi.GPIO as GPIO
```



```
File Edit Format Run Options Windows Help

#!/usr/bin/python3

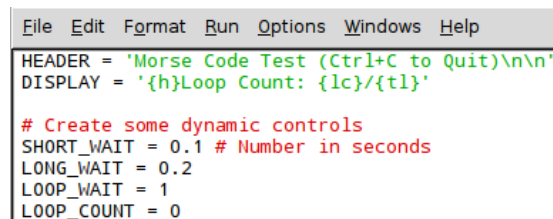
'''
Sounds a buzzer and flashes LEDs for Morse Code
'''

# Builtin Python Libraries
import os
from time import sleep

# Installed Libraries
import RPi.GPIO as GPIO
```

## Display Variables

Now you need to create some variables, 2 string variables and 1 integer variables. The string variables will be used to display text in the terminal to let us know what the program is doing. The integer variables will be used for a sleep timer.



```
File Edit Format Run Options Windows Help

HEADER = 'Morse Code Test (Ctrl+C to Quit)\n\n'
DISPLAY = '{h}Loop Count: {lc}/{t1}'

# Create some dynamic controls
SHORT_WAIT = 0.1 # Number in seconds
LONG_WAIT = 0.2
LOOP_WAIT = 1
LOOP_COUNT = 0
```

```
# Display Variables
HEADER = Morse Code Test (Ctrl+C to Quit)\n\n'
DISPLAY = '{h}Loop Count: {lc}/{t1}'

# Create some dynamic controls
SHORT_WAIT = 0.1 # Number in seconds
LONG_WAIT = 0.2 # Number in seconds
LOOP_WAIT = 1 # Number in seconds
LOOP_COUNT = 0
```

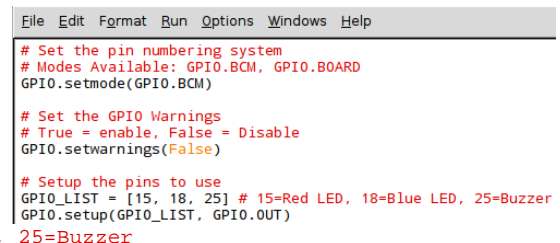
## Setting up the GPIO

Time to setup the GPIO. We will set mode and warnings and also which pins that you will be using.

```
# Set the pin numbering system
# Modes Available: GPIO.BCM, GPIO.BOARD
GPIO.setmode(GPIO.BCM)

# Set the GPIO Warnings
# True = enable, False = Disable
GPIO.setwarnings(False)

# Setup the pins to use
GPIO_LIST = [15, 18, 25] # 15=Red LED, 18=Blue LED, 25=Buzzer
GPIO.setup(GPIO_LIST, GPIO.OUT)
```



```
File Edit Format Run Options Windows Help

# Set the pin numbering system
# Modes Available: GPIO.BCM, GPIO.BOARD
GPIO.setmode(GPIO.BCM)

# Set the GPIO Warnings
# True = enable, False = Disable
GPIO.setwarnings(False)

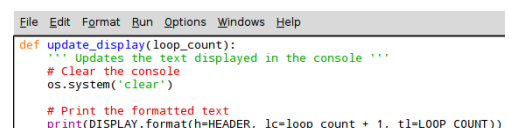
# Setup the pins to use
GPIO_LIST = [15, 18, 25] # 15=Red LED, 18=Blue LED, 25=Buzzer
GPIO.setup(GPIO_LIST, GPIO.OUT)
```

## Update Display Method

Next, it is time to create a Method that will update the text that will be displayed in the console. This will be called when you want to update the information in the console.

```
def update_display(status):
    ''' Updates the text displayed in the console '''

    # Clear the console
    os.system('clear')
    # Print the formatted text to the console
    print(DISPLAY.format(h=HEADER, lc=loop_count + 1, t1=LOOP_COUNT))
```



```
File Edit Format Run Options Windows Help

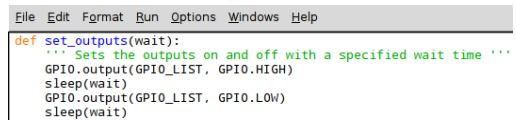
def update_display(loop_count):
    ''' Updates the text displayed in the console '''
    # Clear the console
    os.system('clear')

    # Print the formatted text
    print(DISPLAY.format(h=HEADER, lc=loop_count + 1, t1=LOOP_COUNT))
```

## Set Outputs Method

This method is used to sets the outputs on and off.

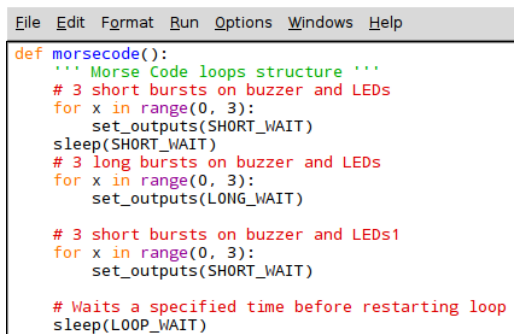
```
def set_outputs(wait):  
    ''' Sets the outputs on and off with a specified wait time '''  
    GPIO.output(GPIO_LIST, GPIO.HIGH)  
    sleep(wait)  
    GPIO.output(GPIO_LIST, GPIO.LOW)  
    sleep(wait)
```



```
File Edit Format Run Options Windows Help  
def set_outputs(wait):  
    ''' Sets the outputs on and off with a specified wait time '''  
    GPIO.output(GPIO_LIST, GPIO.HIGH)  
    sleep(wait)  
    GPIO.output(GPIO_LIST, GPIO.LOW)  
    sleep(wait)
```

## Morse Code Method

```
def morsecode():  
    ''' Morse Code loops structure '''  
    # 3 short bursts on buzzer and LEDs  
    for x in range(0, 3):  
        set_outputs(SHORT_WAIT)  
        sleep(SHORT_WAIT)  
    # 3 long bursts on buzzer and LEDs  
    for x in range(0, 3):  
        set_outputs(LONG_WAIT)  
  
    # 3 short bursts on buzzer and LEDs  
    for x in range(0, 3):  
        set_outputs(SHORT_WAIT)  
  
    # Waits a specified time before restarting loop  
    sleep(LOOP_WAIT)
```

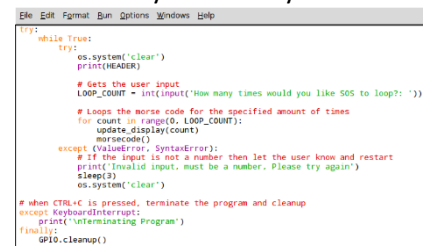


```
File Edit Format Run Options Windows Help  
def morsecode():  
    ''' Morse Code loops structure '''  
    # 3 short bursts on buzzer and LEDs  
    for x in range(0, 3):  
        set_outputs(SHORT_WAIT)  
        sleep(SHORT_WAIT)  
    # 3 long bursts on buzzer and LEDs  
    for x in range(0, 3):  
        set_outputs(LONG_WAIT)  
  
    # 3 short bursts on buzzer and LEDs1  
    for x in range(0, 3):  
        set_outputs(SHORT_WAIT)  
  
    # Waits a specified time before restarting loop  
    sleep(LOOP_WAIT)
```

## The Action Code – User Input

You will now need to add code to get the user to enter a number for how many times they would like the Morse Code to loop.

```
try:  
    while True:  
        try:  
            os.system('clear')  
            print(HEADER)  
  
            # Gets the user input  
            LOOP_COUNT = int(input('How many times would you like SOS to loop?: '))  
  
            # Loops the morse code for the specified amount of times  
            for count in range(0, LOOP_COUNT):  
                update_display(count)  
                morsecode()  
        except (ValueError, SyntaxError):  
            # If the input is not a number then let the user know and restart  
            print('Invalid input, must be a number, Please try again')  
            sleep(3)  
            os.system('clear')  
  
# when CTRL+C is pressed, terminate the program and cleanup  
except KeyboardInterrupt:  
    print('\nTerminating Program')  
finally:  
    GPIO.cleanup()
```



```
File Edit Format Run Options Windows Help  
try:  
    while True:  
        try:  
            os.system('clear')  
            print(HEADER)  
  
            # Gets the user input  
            LOOP_COUNT = int(input('How many times would you like SOS to loop?: '))  
  
            # Loops the morse code for the specified amount of times  
            for count in range(0, LOOP_COUNT):  
                update_display(count)  
                morsecode()  
        except (ValueError, SyntaxError):  
            # If the input is not a number then let the user know and restart  
            print('Invalid input, must be a number, Please try again')  
            sleep(3)  
            os.system('clear')  
  
# when CTRL+C is pressed, terminate the program and cleanup  
except KeyboardInterrupt:  
    print('Terminating Program')  
finally:  
    GPIO.cleanup()
```

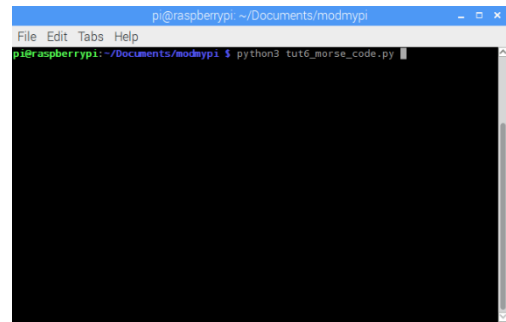
Make sure to save your work by clicking File and select Save, or press Ctrl+S



## Running the Program

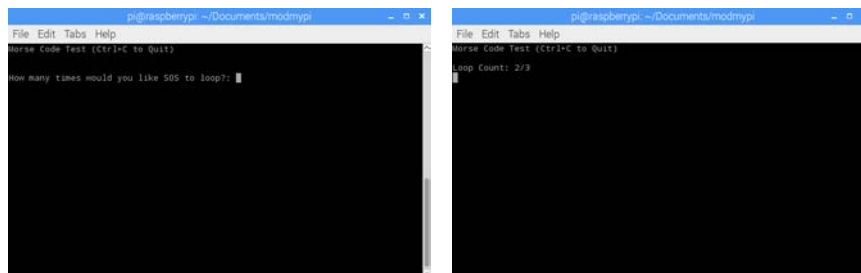
Save your work and it is time to run it so that you can make sure that it works as it should. Go back to the File Manager and open the modmypi folder you created. Next click on tools and select “Open Current Folder in Terminal” or press F4.

In the terminal, type  
`python3 tut6_morse_code.py`  
and press enter



## Results

If everything is working correctly you should see in the console that it is asking you enter how many times you would like the SOS to loop. In the console you should see something similar to this when running:



## Code on GitHub

If you would like to download a copy of the code, you can download it from along with all the other tutorials, code and wiring diagrams from [GitHub here](#)

## Thanks

Thank you for taking the time to follow this tutorial and hope that you have found this useful. Please feel free to follow the other tutorials that have been created for the ModMyPi YouTube Workshop Kit.