PYTHON COURSEWORK

Module No.: CCE1000

Lecturer: Dr. Priscilla Ramsamy

Date: 29/03/2019

Beeharry Farhaan (M00681483)
Fakeermahamood Zaffar (M00685151)

Table of Contents

Introduction

Python is an interpreted high-level object computer programming language, widely known for its simplicity in coding as well as its powerful features enabling both beginners and experts to collaborate.

Python language, being one of the most dominant language, is widely used in real world sectors such as in the making GUI Application, Scientific and Numeric Computation, Data Analysis etc…

The aim of this documentation is to demonstrate light animations on an array of LED's using Python language together with the help of Arduino microcontroller and sensors (Potentiometer and Sound).

Why use Python?

- Simply to code
- Super easy to learn due to its structure
- Portable and Interpreted
- Scalable and Embeddable
- Easy to maintain due to Free and Open Source
- Huge amount of usable libraries

Sensor

❖ Potentiometer

An electrical sensor that works on the principle change of resistance of the wire with its length. The resistance of this electric sensor is manually changed by the sliding contacts.

❖ Sound sensor

A small board combining microphone and some processing circuitry, which not only provides an audio output, but also a binary indication of the presence of sound, and an analog representation of the amplitude.

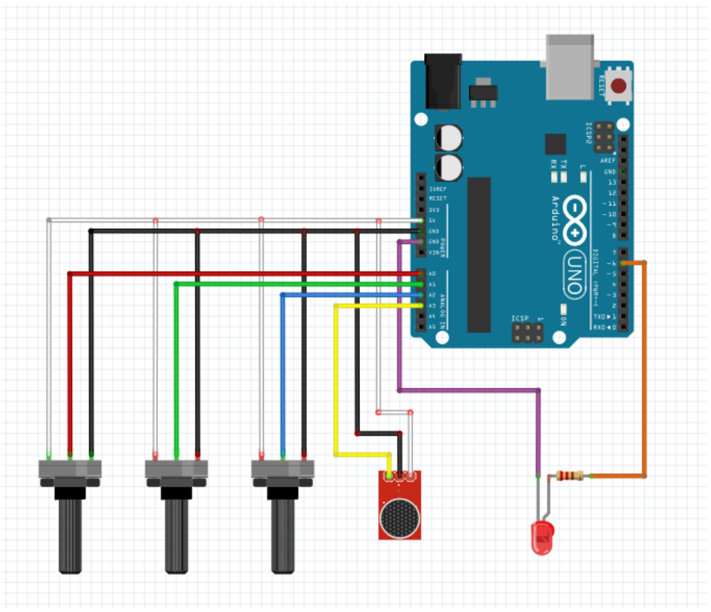
Microcontroller

A microcontroller is a small, self-contained computer integrated on a chip that can be used as an embedded system containing memory, a processor and programmable input/output peripherals.

Throughout this whole documentation, Arduino Uno R3 is the only microcontroller being used.

Arduino Uno R3: A microcontroller board containing 14 digital input/output pins, 6 analog inputs, a reset button and a USB connection port. The Arduino microcontroller is easily programmable using the Arduino IDE which contains a boot loader that executes on whenever the Arduino is switched on.

Connection:



The installation has 6 led strips on the Simulator where each strip has 60 RGB LED's and 360 RGB LED's in total. The Led strip are setup in an array of size 360(0-359).

All input pins used for implementing the connection are analog ones (A0 – A3).

All ground outputs from the 3 potentiometers and the sound sensor are connected to the ground(gnd) pin of the Arduino chip using black wires and all Vcc is connected to the 5V pin using white wires.

The output from the red potentiometer, green potentiometer, blue potentiometer and sound sensor are connected to A0, A1, A2 and A3 using red, green, blue and yellow wires respectively.

A red LED has also been implemented into the system indicating whether the device is running or not. Its positive pole is connected to the digital output pin 6 via a 220Ω resistor and the negative pole to the gnd pin of the Arduino.

Communication:

Serial communication has been used to send data from Arduino to the computer through USB serial COM ports.

Programming Platforms

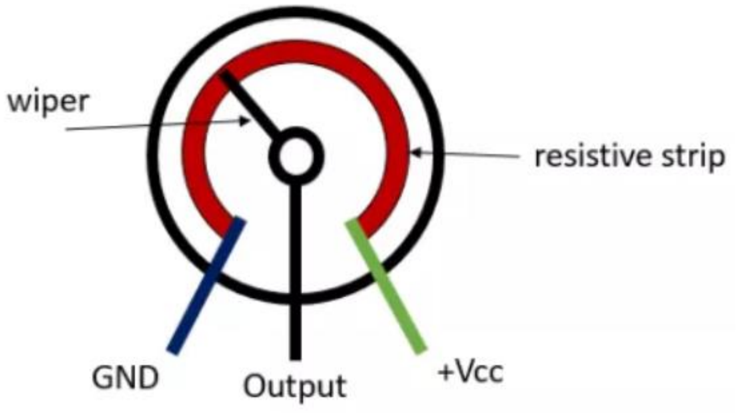
Throughout this documentation, Python 3.7 is being used together with the help of its independent libraries:

Library	Description
OPC	To allow communication between the Python IDE and the simulator
Time	To handle time related tasks like sleep function
Random	Python offers random module that can generate random numbers. These pseudo-random number as the sequence of number generated depends on the seed
Sys	Have a function which allows the user to exit the application
Pyfirmata	A python interface for the Firmata protocol allowing communication between microcontrollers from software on a computer the Firmata board node can talk to the Arduino that is loaded with the Firmata library

Implementation of sensors on Arduino:

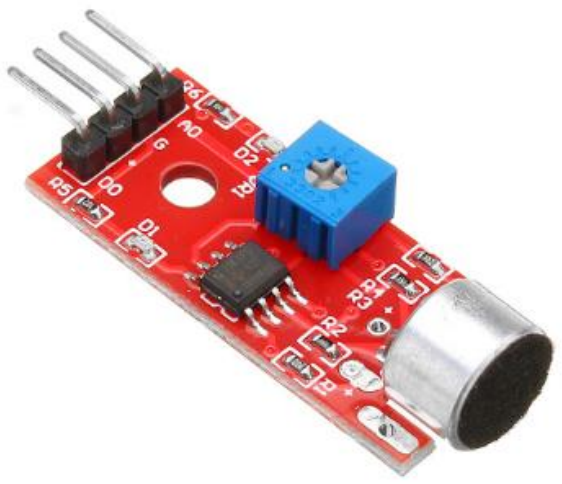
- Potentiometer

The potentiometer here is used as a voltage divider where all three pins are connected to the Arduino. One of the outer pins is connected to the GND, one to Vcc (5V) and the middle one as an analog input to the microcontroller.



- Sound

The sound sensor detects vibration from the surroundings using the microphone integrated in the chip which converts the reading into analog. Then the Arduino can process the analog data. The 'G' pin is connected to the GND, the other to Vcc and the last to A3 analog pin as input.



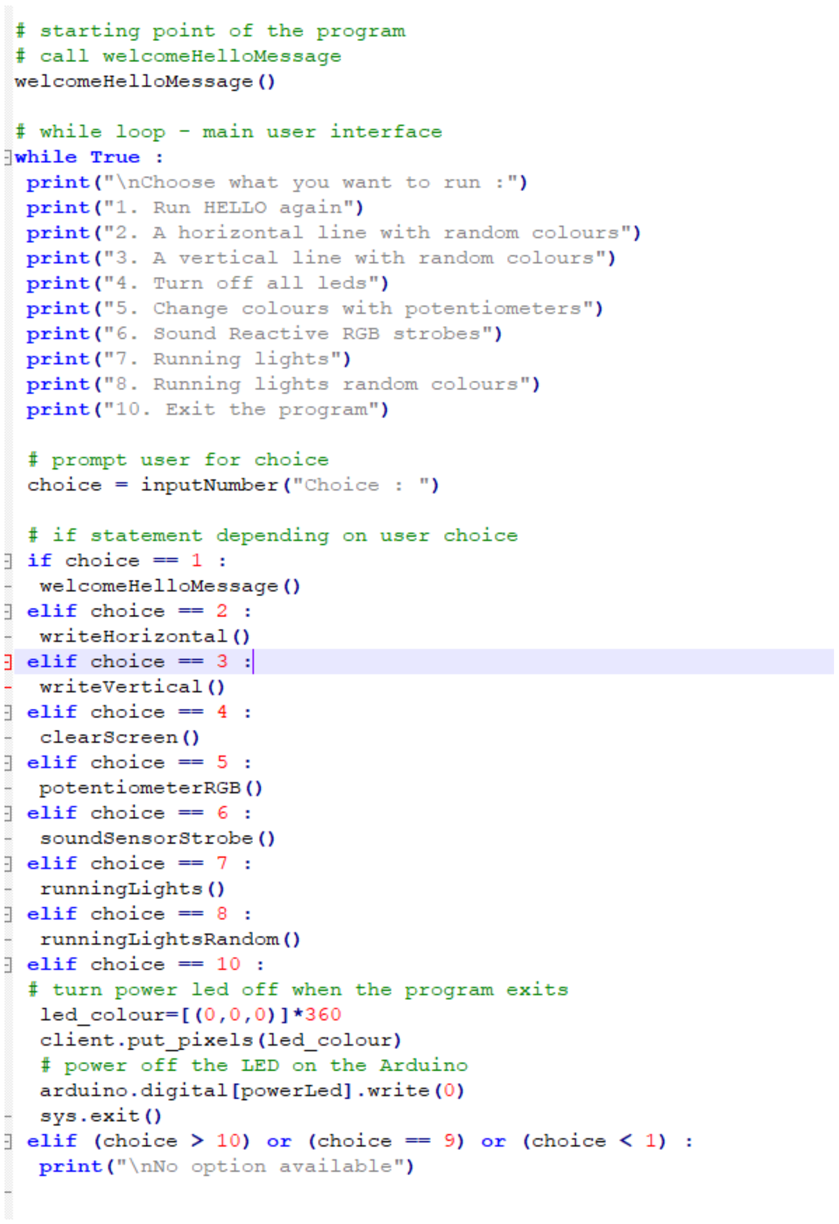
Explanation of code:

The below code is fully written in Python language where it allows the user to make choices for any desired animations from the given list.

All light animations will be on the Simulator provided and readings will either be from user for choices and Arduino for sensors.

The main user interface is displayed every time the program is run. It prompts user for a list of choices and will display the animation accordingly.

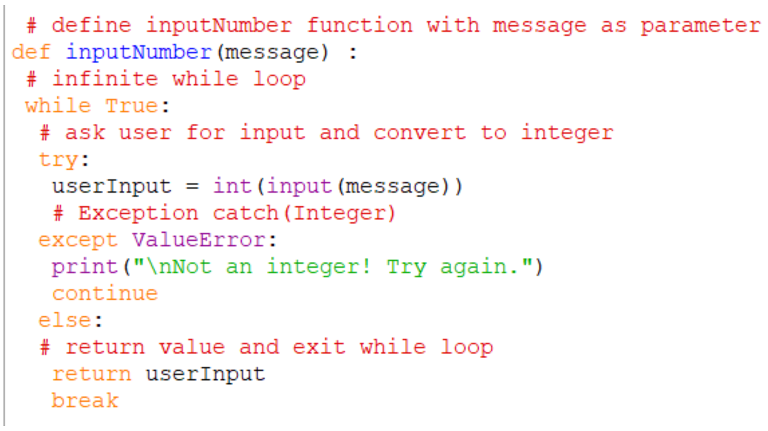
By default, a welcome message is displayed on the simulator and any invalid choices from the user will be rejected.



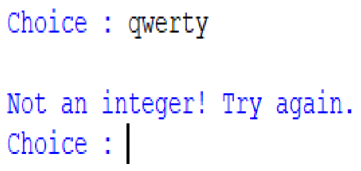
```python
# starting point of the program
# call welcomeHelloMessage
welcomeHelloMessage()

# while loop - main user interface
while True :
 print("\nChoose what you want to run :")
 print("1. Run HELLO again")
 print("2. A horizontal line with random colours")
 print("3. A vertical line with random colours")
 print("4. Turn off all leds")
 print("5. Change colours with potentiometers")
 print("6. Sound Reactive RGB strobes")
 print("7. Running lights")
 print("8. Running lights random colours")
 print("10. Exit the program")

 # prompt user for choice
 choice = inputNumber("Choice : ")

 # if statement depending on user choice
 if choice == 1 :
  welcomeHelloMessage()
 elif choice == 2 :
  writeHorizontal()
 elif choice == 3 :
  writeVertical()
 elif choice == 4 :
  clearScreen()
 elif choice == 5 :
  potentiometerRGB()
 elif choice == 6 :
  soundSensorStrobe()
 elif choice == 7 :
  runningLights()
 elif choice == 8 :
  runningLightsRandom()
 elif choice == 10 :
 # turn power led off when the program exits
  led_colour=[(0,0,0)]*360
  client.put_pixels(led_colour)
  # power off the LED on the Arduino
  arduino.digital[powerLed].write(0)
  sys.exit()
 elif (choice > 10) or (choice == 9) or (choice < 1) :
  print("\nNo option available")
```

Input Number Function

The below code is used as a validation check every time the user is prompted. Any non- integer value is rejected and will keep prompted until an integer value has been entered.



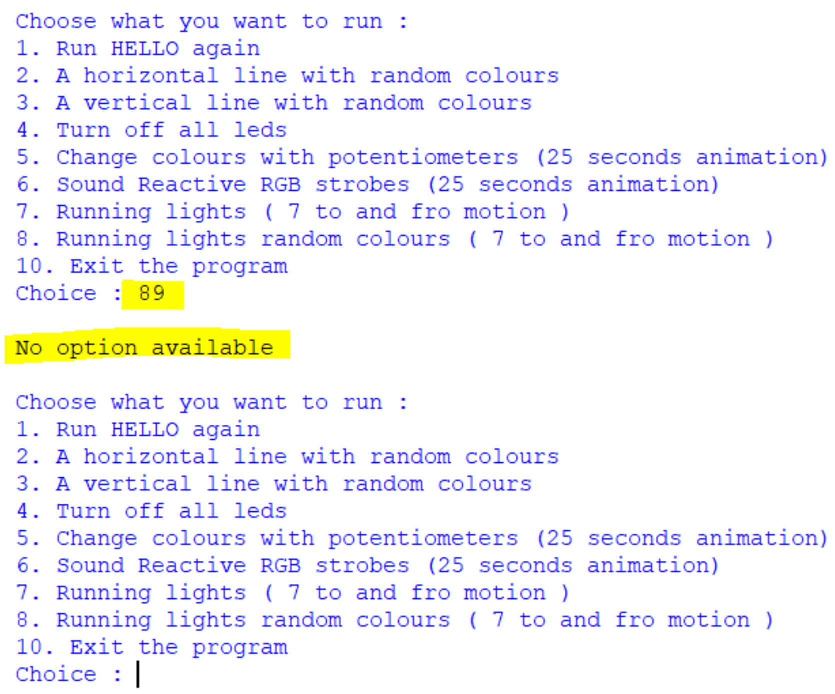
```
 # define inputNumber function with message as parameter
def inputNumber(message) :
 # infinite while loop
 while True:
  # ask user for input and convert to integer
  try:
   userInput = int(input(message))
   # Exception catch(Integer)
  except ValueError:
   print("\nNot an integer! Try again.")
   continue
  else:
  # return value and exit while loop
   return userInput
   break
```



```
Choice : qwerty

Not an integer! Try again.
Choice :
```

Invalid choice made by user

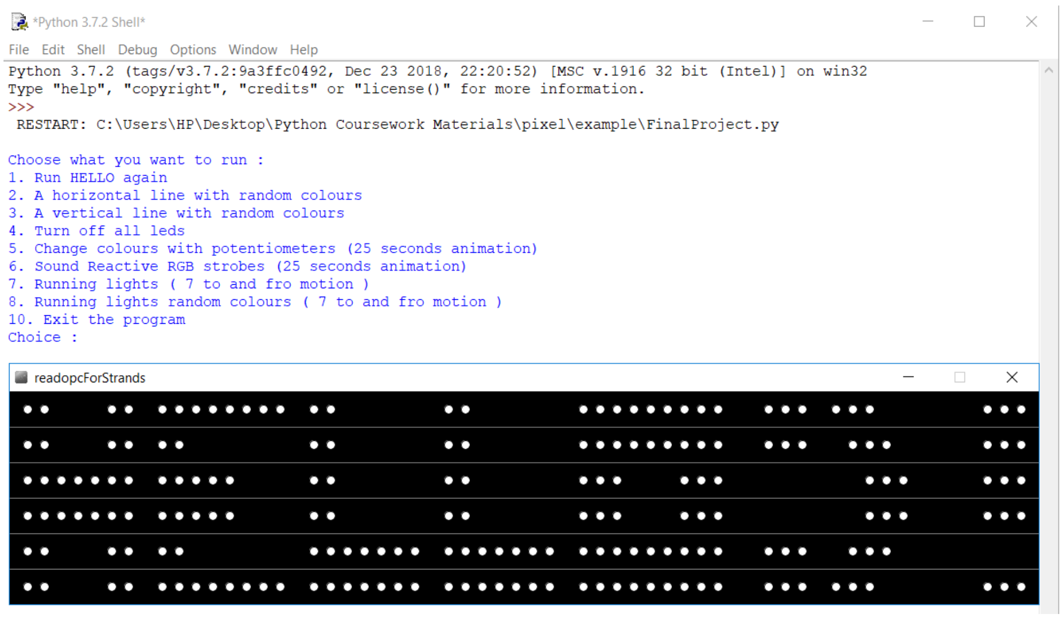
Any invalid integer choice is rejected by the program and will continue to loop until a valid choice has been made or the user decides to terminate the program manually.



```
Choose what you want to run :
1. Run HELLO again
2. A horizontal line with random colours
3. A vertical line with random colours
4. Turn off all leds
5. Change colours with potentiometers (25 seconds animation)
6. Sound Reactive RGB strobes (25 seconds animation)
7. Running lights ( 7 to and fro motion )
8. Running lights random colours ( 7 to and fro motion )
10. Exit the program
Choice : 89

No option available

Choose what you want to run :
1. Run HELLO again
2. A horizontal line with random colours
3. A vertical line with random colours
4. Turn off all leds
5. Change colours with potentiometers (25 seconds animation)
6. Sound Reactive RGB strobes (25 seconds animation)
7. Running lights ( 7 to and fro motion )
8. Running lights random colours ( 7 to and fro motion )
10. Exit the program
Choice :
```

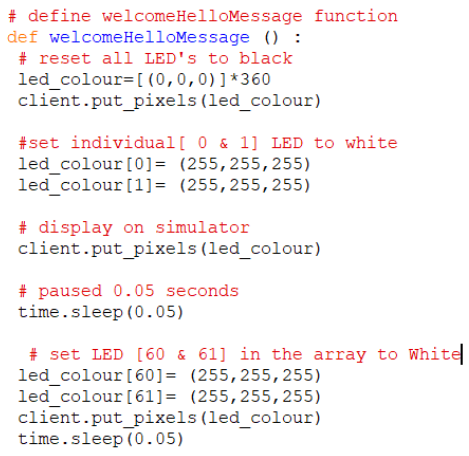
Animations:

HELLO animation

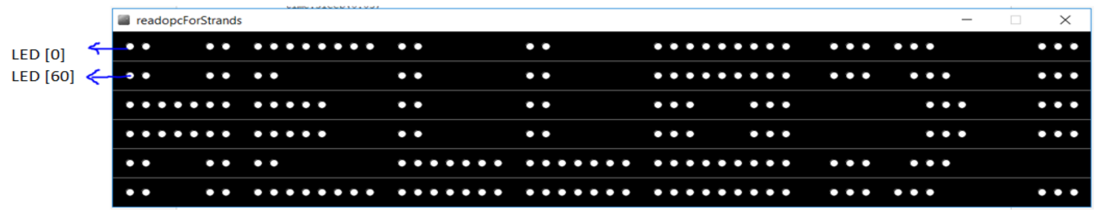


```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\HP\Desktop\Python Coursework Materials\pixel\example\FinalProject.py

Choose what you want to run :
1. Run HELLO again
2. A horizontal line with random colours
3. A vertical line with random colours
4. Turn off all leds
5. Change colours with potentiometers (25 seconds animation)
6. Sound Reactive RGB strobes (25 seconds animation)
7. Running lights ( 7 to and fro motion )
8. Running lights random colours ( 7 to and fro motion )
10. Exit the program
Choice :
```

In order to display, the “HELLO” message, each LED has been assigned RGB (0,0,0), i.e. Black colour and for the White colour, the location of the LED has been precisely calculated and set to RGB (255,255,255).



```
# define welcomeHelloMessage function
def welcomeHelloMessage () :
 # reset all LED's to black
 led_colour=[(0,0,0)]*360
 client.put_pixels(led_colour)

 #set individual[ 0 & 1] LED to white
 led_colour[0]= (255,255,255)
 led_colour[1]= (255,255,255)

 # display on simulator
 client.put_pixels(led_colour)

 # paused 0.05 seconds
 time.sleep(0.05)

  # set LED [60 & 61] in the array to White
 led_colour[60]= (255,255,255)
 led_colour[61]= (255,255,255)
 client.put_pixels(led_colour)
 time.sleep(0.05)
```

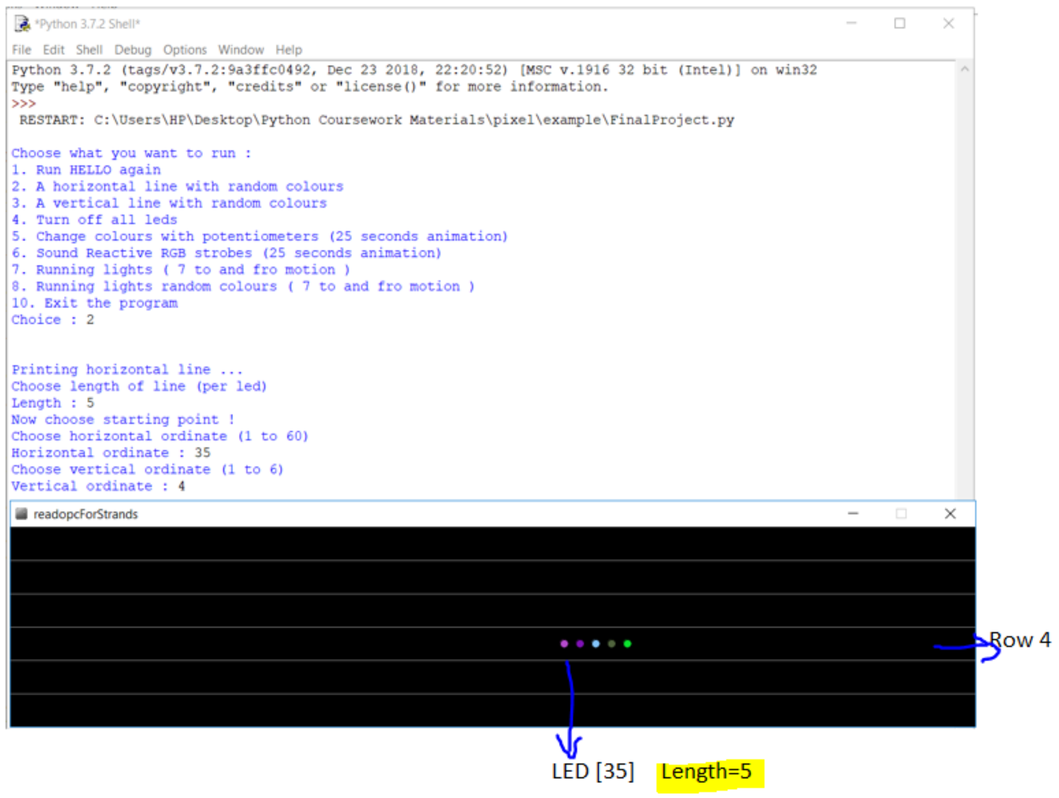


WriteHorizontal animation

This function is used to display a horizontal line on the simulator where each LED is given a random colour.

The user selects the starting location, the length of the line as well as the row on which the line is to be displayed.

Again, any invalid data is rejected by the 'if' statement.

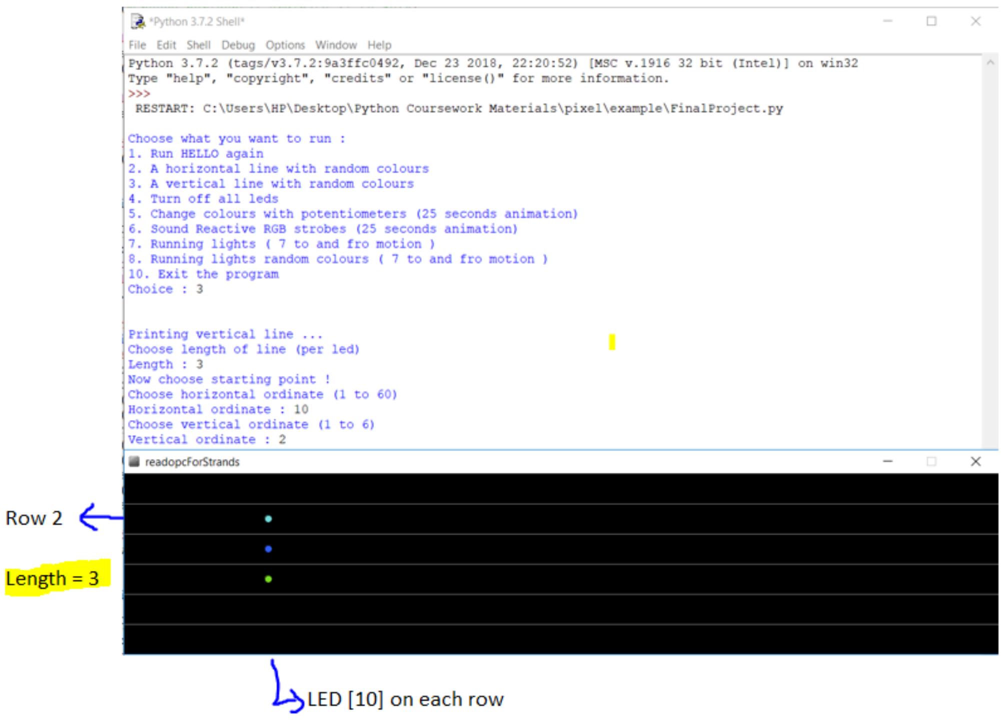


```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\HP\Desktop\Python Coursework Materials\pixel\example\FinalProject.py

Choose what you want to run :
1. Run HELLO again
2. A horizontal line with random colours
3. A vertical line with random colours
4. Turn off all leds
5. Change colours with potentiometers (25 seconds animation)
6. Sound Reactive RGB strobes (25 seconds animation)
7. Running lights ( 7 to and fro motion )
8. Running lights random colours ( 7 to and fro motion )
10. Exit the program
Choice : 2


Printing horizontal line ...
Choose length of line (per led)
Length : 5
Now choose starting point !
Choose horizontal ordinate (1 to 60)
Horizontal ordinate : 35
Choose vertical ordinate (1 to 6)
Vertical ordinate : 4
```

WriteVertical animation

This animation is similar to writeHorizontal one whereby instead of displaying a horizontal line, the program will display a vertical line.

```
# define writeVertical function
def writeVertical():
 # reset all LED's to black
 led_colour=[(0,0,0)]*360
 client.put_pixels(led_colour)
 print("\n\nPrinting vertical line ...")
 print("Choose length of line (per led)")
 length = inputNumber("Length : ")
 print("Now choose starting point !")
 print("Choose horizontal ordinate (1 to 60)")
 hOffset = inputNumber("Horizontal ordinate : ")
 print("Choose vertical ordinate (1 to 6)")
 vOffset = inputNumber("Vertical ordinate : ")

 # condition in case of invalid values for length, vOffset, hOffset and array bounds
 if ((vOffset + length) < 8 ) and (hOffset < 61) and (hOffset >0) and (vOffset>0):

     # for loop to add random colours to the LED's which have to be switched on
  for i in range(vOffset - 1, length + (vOffset - 1)):
   led_colour[hOffset - 1 + (i*60)] = (randint(0,256),randint(0,256),randint(0,256))
  client.put_pixels(led_colour)
 else :
  print("\nLed amount out of bounds... returning to main menu !")
  time.sleep(1)
```



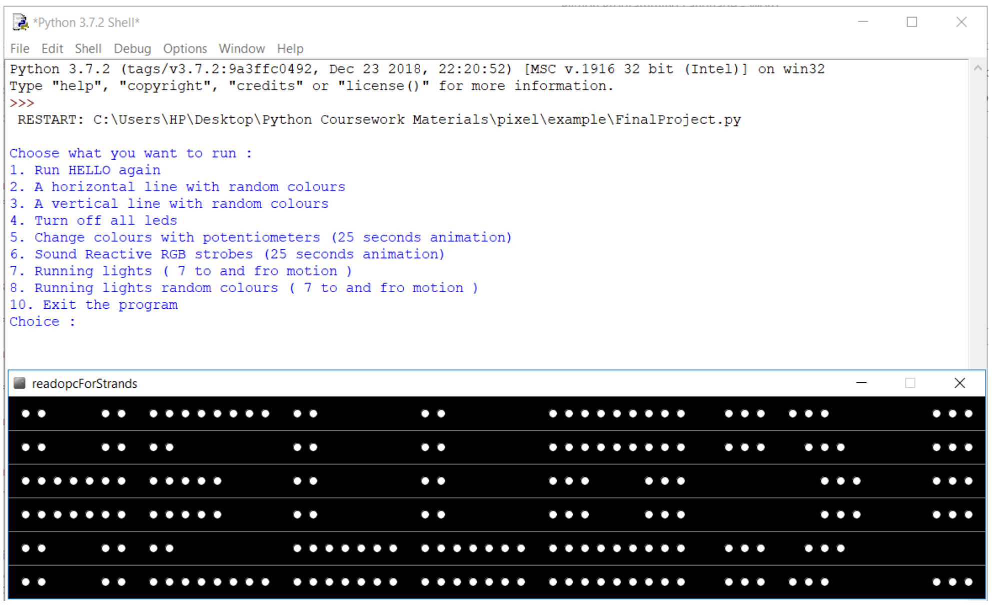
```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\HP\Desktop\Python Coursework Materials\pixel\example\FinalProject.py

Choose what you want to run :
1. Run HELLO again
2. A horizontal line with random colours
3. A vertical line with random colours
4. Turn off all leds
5. Change colours with potentiometers (25 seconds animation)
6. Sound Reactive RGB strobes (25 seconds animation)
7. Running lights ( 7 to and fro motion )
8. Running lights random colours ( 7 to and fro motion )
10. Exit the program
Choice : 3


Printing vertical line ...
Choose length of line (per led)
Length : 3
Now choose starting point !
Choose horizontal ordinate (1 to 60)
Horizontal ordinate : 10
Choose vertical ordinate (1 to 6)
Vertical ordinate : 2
```

Turn off all LED animation

This animation simply reset all LEDs to black {RGB (0,0,0)}.

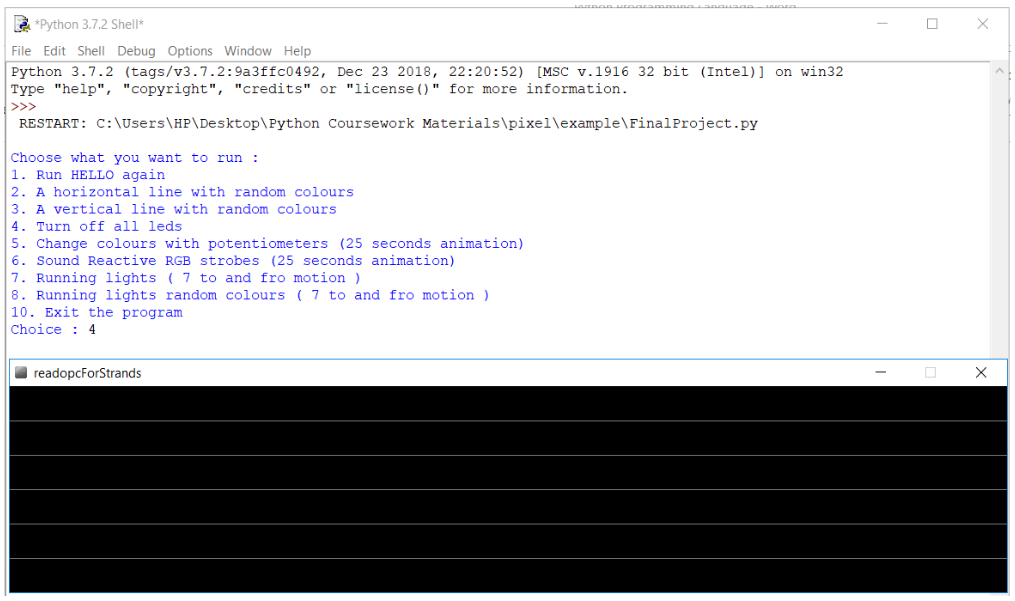
```
# define clearScreen function
def clearScreen() :
  # reset all LED's to black
  led_colour=[(0,0,0)]*360
  client.put_pixels(led_colour)
```

- Before



```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\HP\Desktop\Python Coursework Materials\pixel\example\FinalProject.py

Choose what you want to run :
1. Run HELLO again
2. A horizontal line with random colours
3. A vertical line with random colours
4. Turn off all leds
5. Change colours with potentiometers (25 seconds animation)
6. Sound Reactive RGB strobes (25 seconds animation)
7. Running lights ( 7 to and fro motion )
8. Running lights random colours ( 7 to and fro motion )
10. Exit the program
Choice :
```

- After



```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\HP\Desktop\Python Coursework Materials\pixel\example\FinalProject.py

Choose what you want to run :
1. Run HELLO again
2. A horizontal line with random colours
3. A vertical line with random colours
4. Turn off all leds
5. Change colours with potentiometers (25 seconds animation)
6. Sound Reactive RGB strobes (25 seconds animation)
7. Running lights ( 7 to and fro motion )
8. Running lights random colours ( 7 to and fro motion )
10. Exit the program
Choice : 4
```

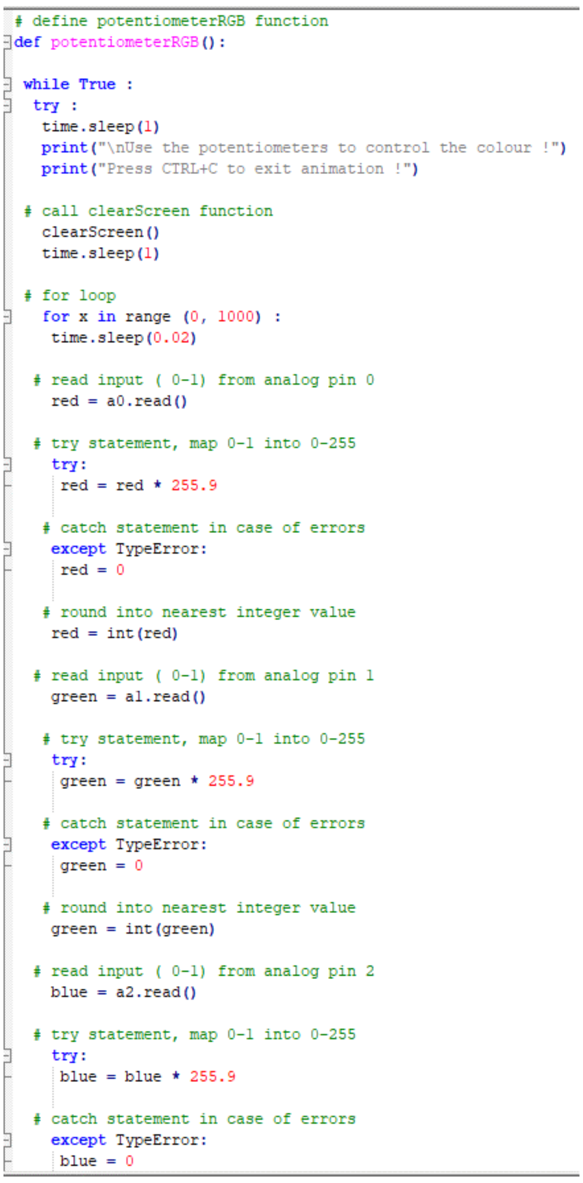
PotentiometerRGB animation

This function displays RGB colours to all 360 LEDs by making the use of 3 potentiometer.

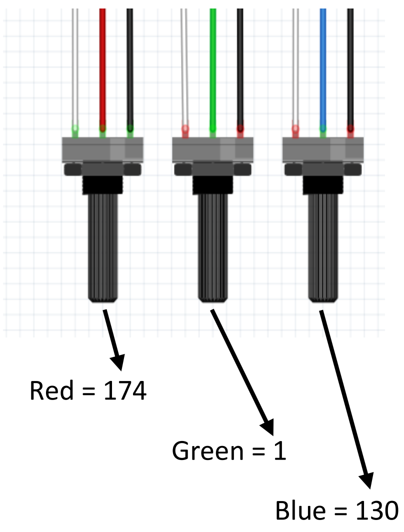
The user rotates the wiper which returns the analog reading from the Arduino microcontroller to Python.

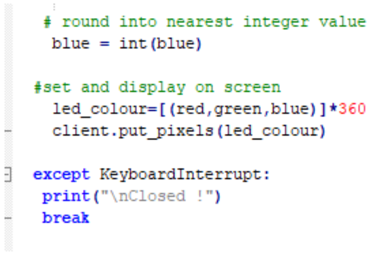
Since the analog reading is between 0 and 1, a simple mathematical formula is being applied in order to map the value from 0 to 255.
In case of decimal value, the reading is being rounded to the nearest whole number.

Thus, the user can set the RGB colours of the LED strip and display them.

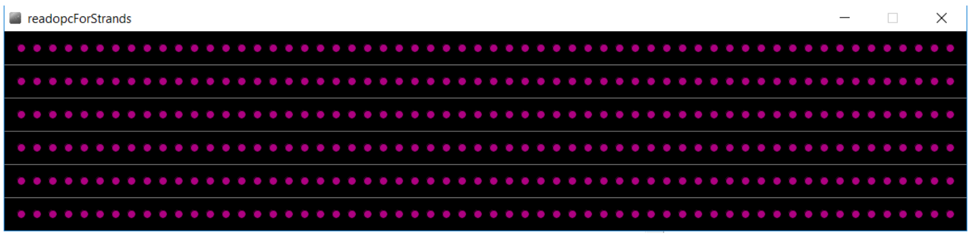


```
# define potentiometerRGB function
def potentiometerRGB():

 while True :
  try :
   time.sleep(1)
   print("\nUse the potentiometers to control the colour !")
   print("Press CTRL+C to exit animation !")

 # call clearScreen function
   clearScreen()
   time.sleep(1)

 # for loop
   for x in range (0, 1000) :
    time.sleep(0.02)

  # read input ( 0-1) from analog pin 0
    red = a0.read()

  # try statement, map 0-1 into 0-255
    try:
     red = red * 255.9

   # catch statement in case of errors
    except TypeError:
     red = 0

   # round into nearest integer value
    red = int(red)

  # read input ( 0-1) from analog pin 1
    green = a1.read()

   # try statement, map 0-1 into 0-255
    try:
     green = green * 255.9

   # catch statement in case of errors
    except TypeError:
     green = 0

   # round into nearest integer value
    green = int(green)

  # read input ( 0-1) from analog pin 2
    blue = a2.read()

  # try statement, map 0-1 into 0-255
    try:
     blue = blue * 255.9

  # catch statement in case of errors
    except TypeError:
     blue = 0
```





```
   # round into nearest integer value
    blue = int(blue)

  #set and display on screen
    led_colour=[(red,green,blue)]*360
    client.put_pixels(led_colour)

  except KeyboardInterrupt:
   print("\nClosed !")
   break
```

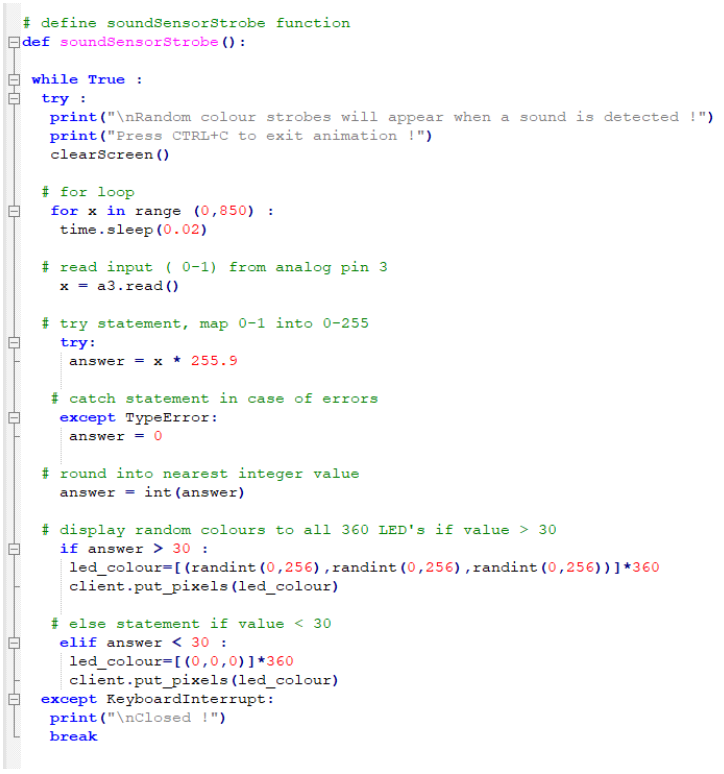


SoundSensorStrobe animation

This animation makes the use of a sound sensor which detects any sound waves by the microphone. The intensity of the sound wave is then fed into Arduino by its analog pin and returns it to the Python IDE.

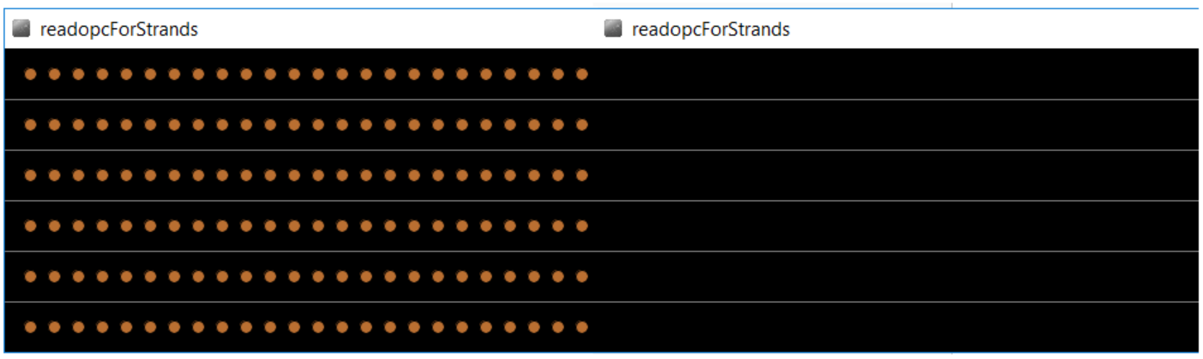
From there, a proper mapping of values occurs and then display random colours to all 360 LEDs if the intensity of the sound wave met the criteria of the code.

Animation will continue until user presses ‘CTRL + C’.



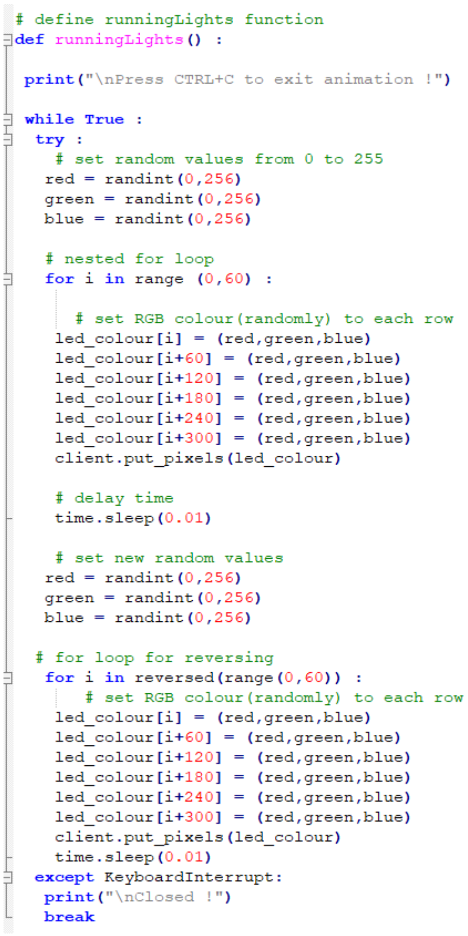
```
# define soundSensorStrobe function
def soundSensorStrobe():

 while True :
  try :
   print("\nRandom colour strobes will appear when a sound is detected !")
   print("Press CTRL+C to exit animation !")
   clearScreen()

  # for loop
   for x in range (0,850) :
    time.sleep(0.02)

  # read input ( 0-1) from analog pin 3
    x = a3.read()

  # try statement, map 0-1 into 0-255
    try:
     answer = x * 255.9

   # catch statement in case of errors
    except TypeError:
     answer = 0

  # round into nearest integer value
    answer = int(answer)

  # display random colours to all 360 LED's if value > 30
    if answer > 30 :
     led_colour=[(randint(0,256),randint(0,256),randint(0,256))]*360
     client.put_pixels(led_colour)

   # else statement if value < 30
    elif answer < 30 :
     led_colour=[(0,0,0)]*360
     client.put_pixels(led_colour)
  except KeyboardInterrupt:
   print("\nClosed !")
   break
```

WITH SOUND VS NO SOUND



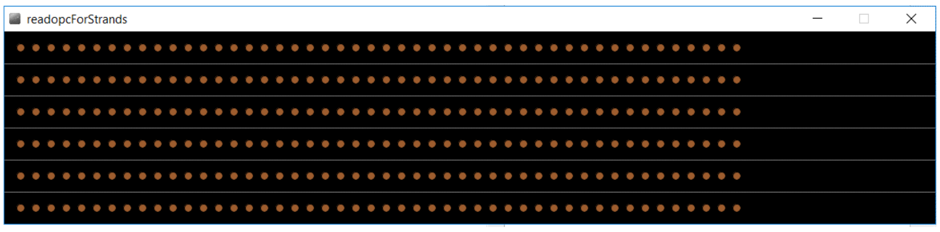
RUNNING LIGHT

This animation picks a random colour and then display the same or different colour for each LED on each row. When it reaches the 60th LED on each row, another random is selected and the animation happens in the reverse order until user presses ‘CTRL +C’

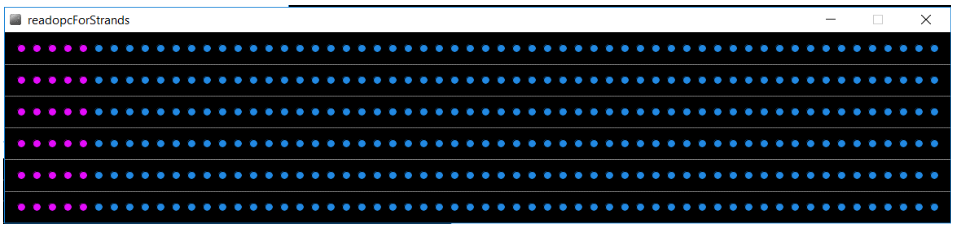


```
# define runningLights function
def runningLights() :

 print("\nPress CTRL+C to exit animation !")

 while True :
  try :
    # set random values from 0 to 255
   red = randint(0,256)
   green = randint(0,256)
   blue = randint(0,256)

   # nested for loop
   for i in range (0,60) :

      # set RGB colour(randomly) to each row
    led_colour[i] = (red,green,blue)
    led_colour[i+60] = (red,green,blue)
    led_colour[i+120] = (red,green,blue)
    led_colour[i+180] = (red,green,blue)
    led_colour[i+240] = (red,green,blue)
    led_colour[i+300] = (red,green,blue)
    client.put_pixels(led_colour)

    # delay time
    time.sleep(0.01)

    # set new random values
   red = randint(0,256)
   green = randint(0,256)
   blue = randint(0,256)

  # for loop for reversing
   for i in reversed(range(0,60)) :
       # set RGB colour(randomly) to each row
    led_colour[i] = (red,green,blue)
    led_colour[i+60] = (red,green,blue)
    led_colour[i+120] = (red,green,blue)
    led_colour[i+180] = (red,green,blue)
    led_colour[i+240] = (red,green,blue)
    led_colour[i+300] = (red,green,blue)
    client.put_pixels(led_colour)
    time.sleep(0.01)
  except KeyboardInterrupt:
   print("\nClosed !")
   break
```

- Normal

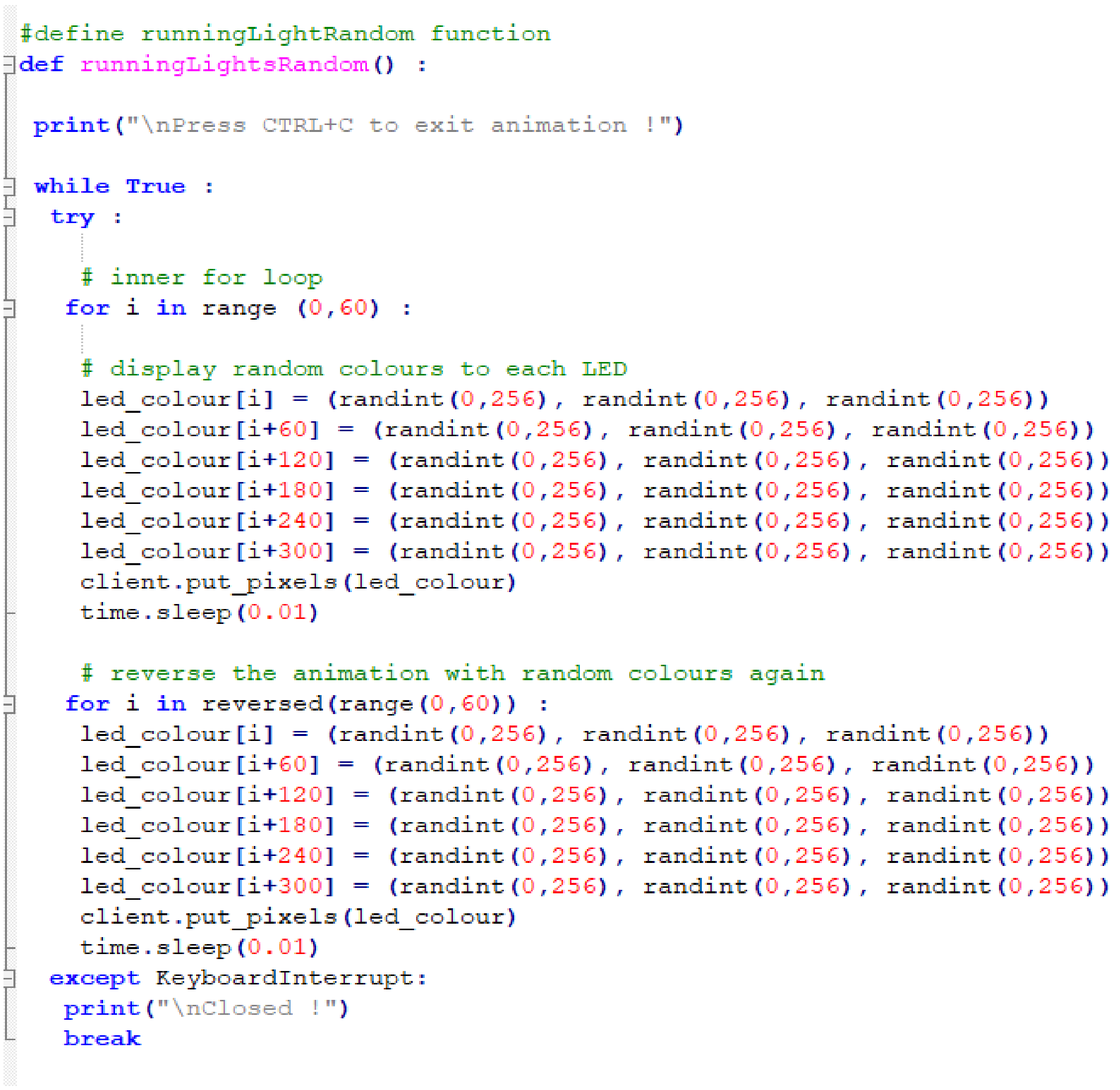


- Reverse order



RUNNING LIGHT Random colours

This animation is similar to the previous one (Running light) but instead of displaying the same colour for all the LEDs, it displays a random colour for each individual LED until user terminates it by pressing ‘CTRL + C’.



```
#define runningLightRandom function
def runningLightsRandom() :

 print("\nPress CTRL+C to exit animation !")

 while True :
  try :

    # inner for loop
   for i in range (0,60) :

    # display random colours to each LED
    led_colour[i] = (randint(0,256), randint(0,256), randint(0,256))
    led_colour[i+60] = (randint(0,256), randint(0,256), randint(0,256))
    led_colour[i+120] = (randint(0,256), randint(0,256), randint(0,256))
    led_colour[i+180] = (randint(0,256), randint(0,256), randint(0,256))
    led_colour[i+240] = (randint(0,256), randint(0,256), randint(0,256))
    led_colour[i+300] = (randint(0,256), randint(0,256), randint(0,256))
    client.put_pixels(led_colour)
    time.sleep(0.01)

    # reverse the animation with random colours again
   for i in reversed(range(0,60)) :
    led_colour[i] = (randint(0,256), randint(0,256), randint(0,256))
    led_colour[i+60] = (randint(0,256), randint(0,256), randint(0,256))
    led_colour[i+120] = (randint(0,256), randint(0,256), randint(0,256))
    led_colour[i+180] = (randint(0,256), randint(0,256), randint(0,256))
    led_colour[i+240] = (randint(0,256), randint(0,256), randint(0,256))
    led_colour[i+300] = (randint(0,256), randint(0,256), randint(0,256))
    client.put_pixels(led_colour)
    time.sleep(0.01)
  except KeyboardInterrupt:
   print("\nClosed !")
   break
```