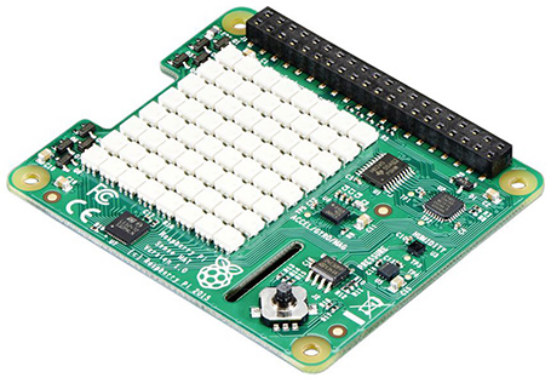
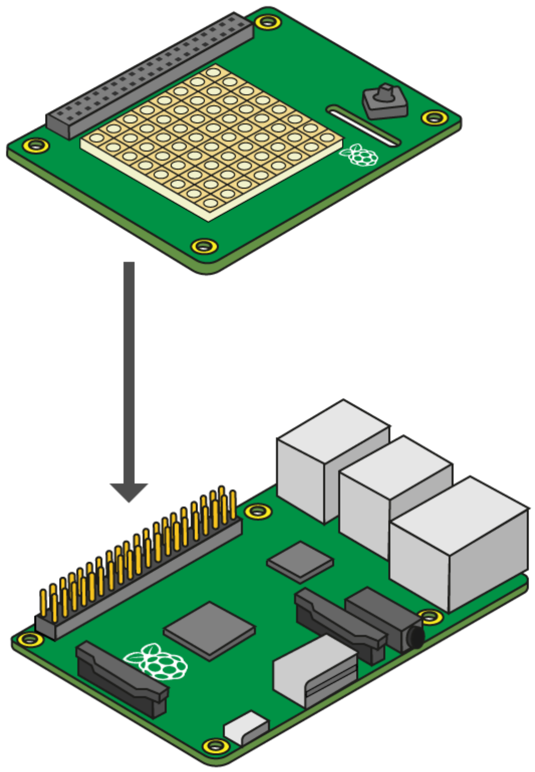
Raspberry Pi Sense HAT User Manual



Assembling the Sense HAT

1. If you have not yet installed the Sense HAT, now is the time to do so. Do this with the Raspberry Pi shut down, disconnected from the mains, and with all other cables disconnected.
2. The Sense HAT comes in a silver anti-static bag, along with the following fixtures and fittings:
   - 1 x GPIO pin extension header
   - 4 x Hexagon stand-offs (female-to-female)
   - 8 x M2.5 screws

   Check that these are all present before proceeding.
3. This diagram shows how it all fits together.

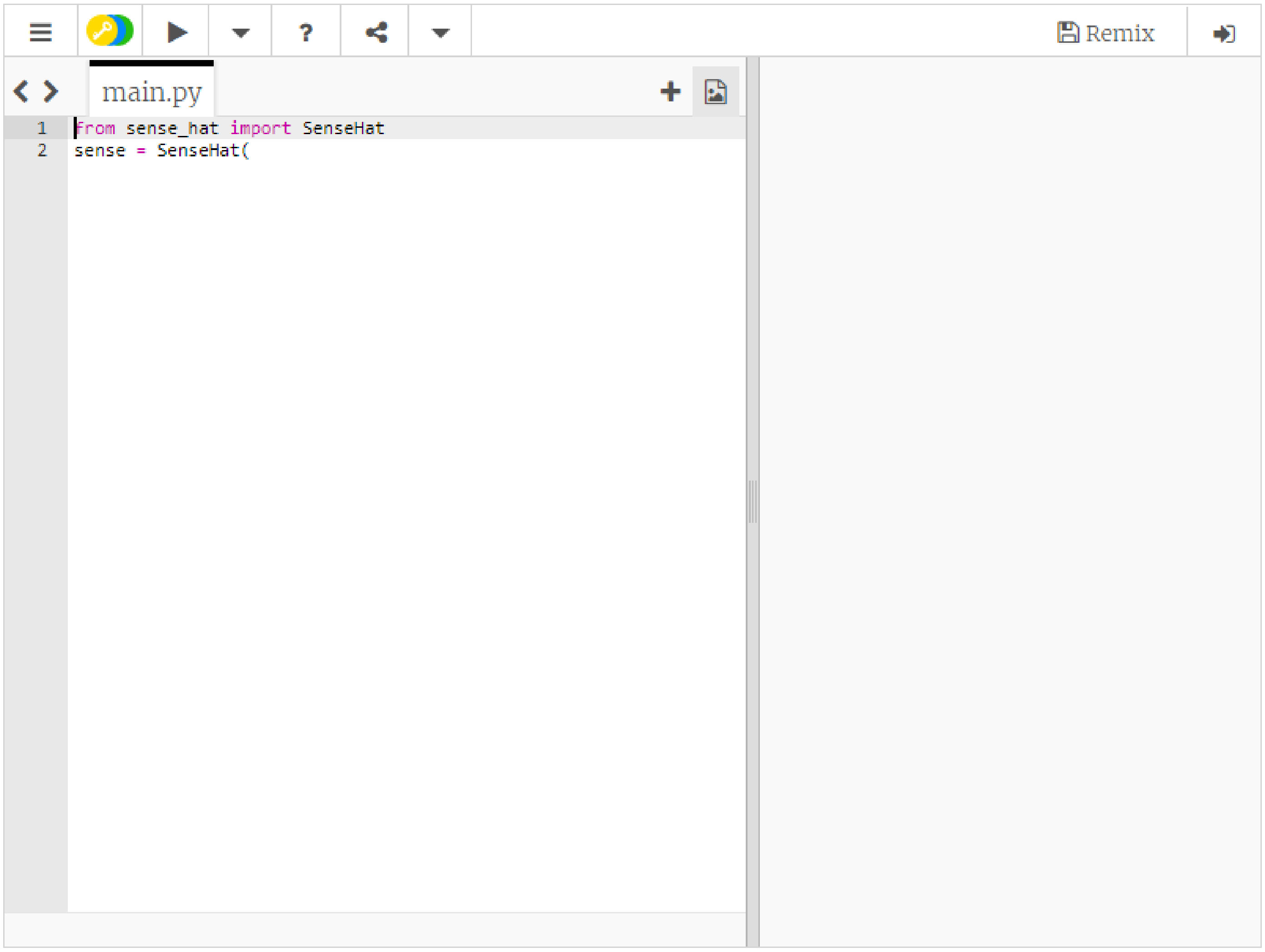


4. First, put the GPIO extension header block onto the Raspberry Pi GPIO pins.

5. Screw the hexagon stand-offs to the Raspberry Pi itself, by threading the screws through from the bottom and turning the hexagon stand-offs between finger and thumb.
6. Next, insert the Sense HAT into the GPIO pin extension header. The corner holes should line up with the hexagon stand-offs.
7. Lastly, put the remaining screws through from the top.
8. Use a small Phillips screwdriver to tighten each corner stand-off individually. They don't need to be especially tight, just enough to ensure that the HAT doesn't become loose.

Emulating the Sense HAT

The talented programmers over at trinket.io have created a Sense HAT emulator. This can be used to play with scripts when your Sense HAT is not available, or to create tutorials on using the Sense HAT that can be embedded on a web-page.



```
from sense_hat import SenseHat
sense = SenseHat(
```

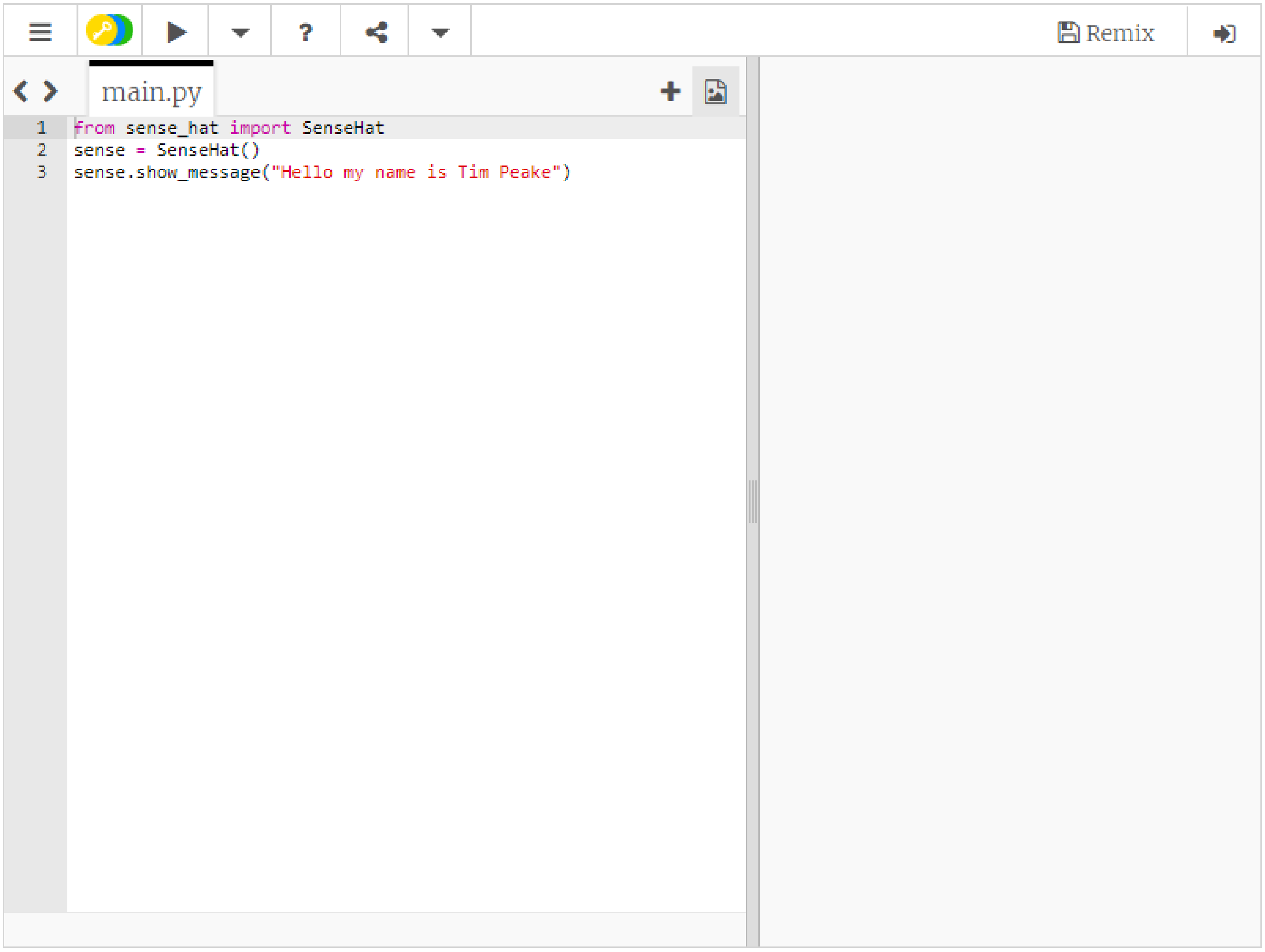
The SenseHAT emulator will be used throughout this guide to explain certain features of the Sense HAT. You can test any of the programs by clicking on the run button, and even edit the code in the embedded examples to play with different features.

Sense HAT: first program

Open Python 3 using `Menu > Programming > Python 3`. This will cause a Python Shell window to appear. Select `File > New File`, and type in the following code:

```
from sense_hat import SenseHat
sense = SenseHat()
sense.show_message("Hello my name is Tim Peake")
```

Select `File > Save` and choose a file name for your program, then select `Run > Run module`. Your message should then scroll across the LED matrix in white text.



```
from sense_hat import SenseHat
sense = SenseHat()
sense.show_message("Hello my name is Tim Peake")
```

Why not try changing the message between the double quotation marks and running your code again?

Sensors

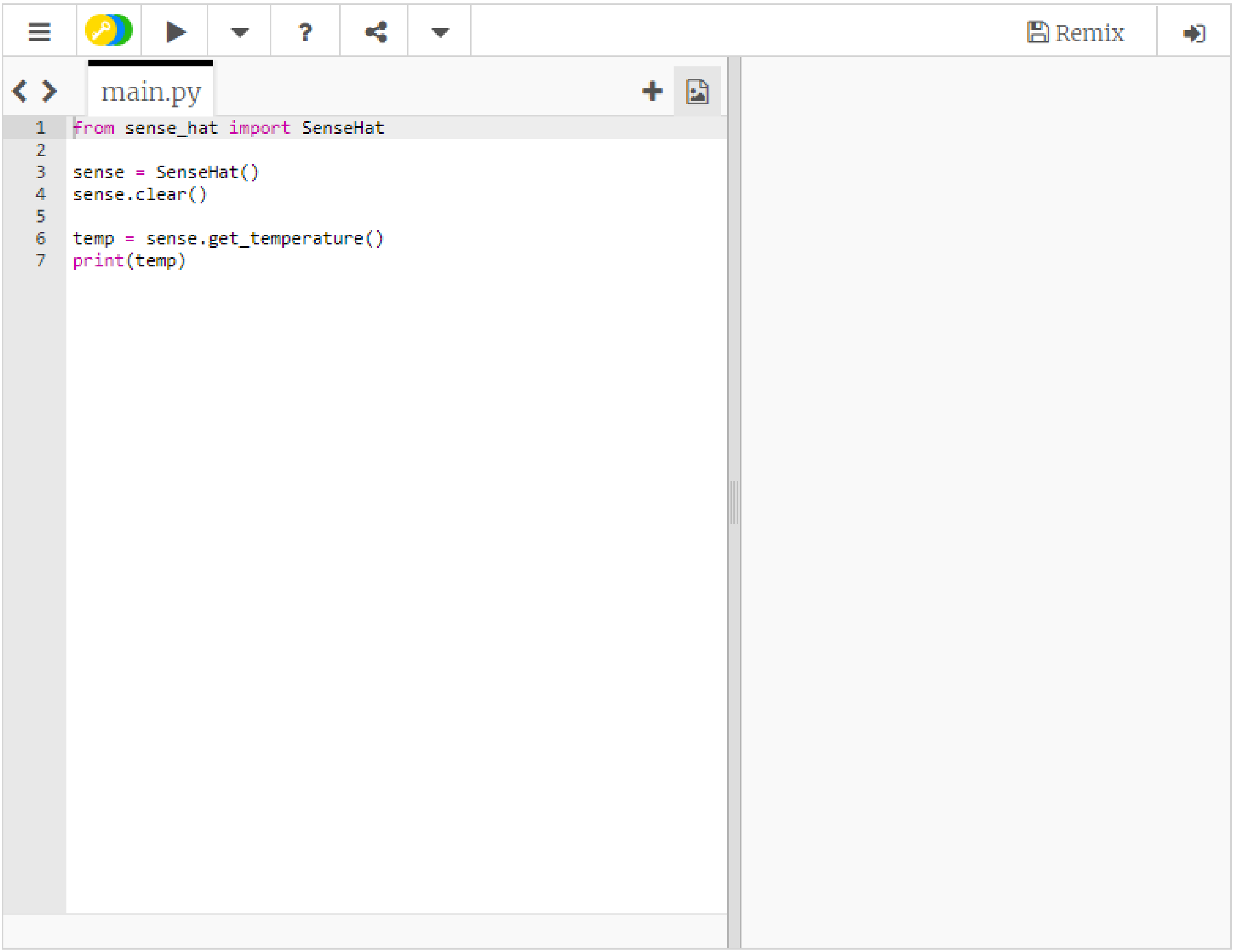
I. Temperature

What is the Temperature?

1. Click on `Menu` > `Programming` > `Python 3 (IDLE)` to open a new Python shell. Then click `File` > `New File` to open a new file.
2. Enter the following code into a new window:

```python
from sense_hat import SenseHat

sense = SenseHat()
sense.clear()

temp = sense.get_temperature()
print(temp)
```

3. Select `File > Save` and choose a file name for your program.
4. Select `Run > Run module`.



```python
from sense_hat import SenseHat

sense = SenseHat()
sense.clear()

temp = sense.get_temperature()
print(temp)
```

5. If you see the error `Humidity Init Failed, please run as root / use sudo` (look on the last line of the message in red), it means you haven't followed the instructions above. Close everything and go back to step 1.

6. You should see something like this:

```
Humidity sensor Init Succeeded
28.6293258667
```

7. Just before the `print(temp)` line add this line below:

```
temp = round(temp, 1)
```

8. You should now see something like this (without all the numbers after the decimal point):

```
Humidity sensor Init Succeeded
28.6
```

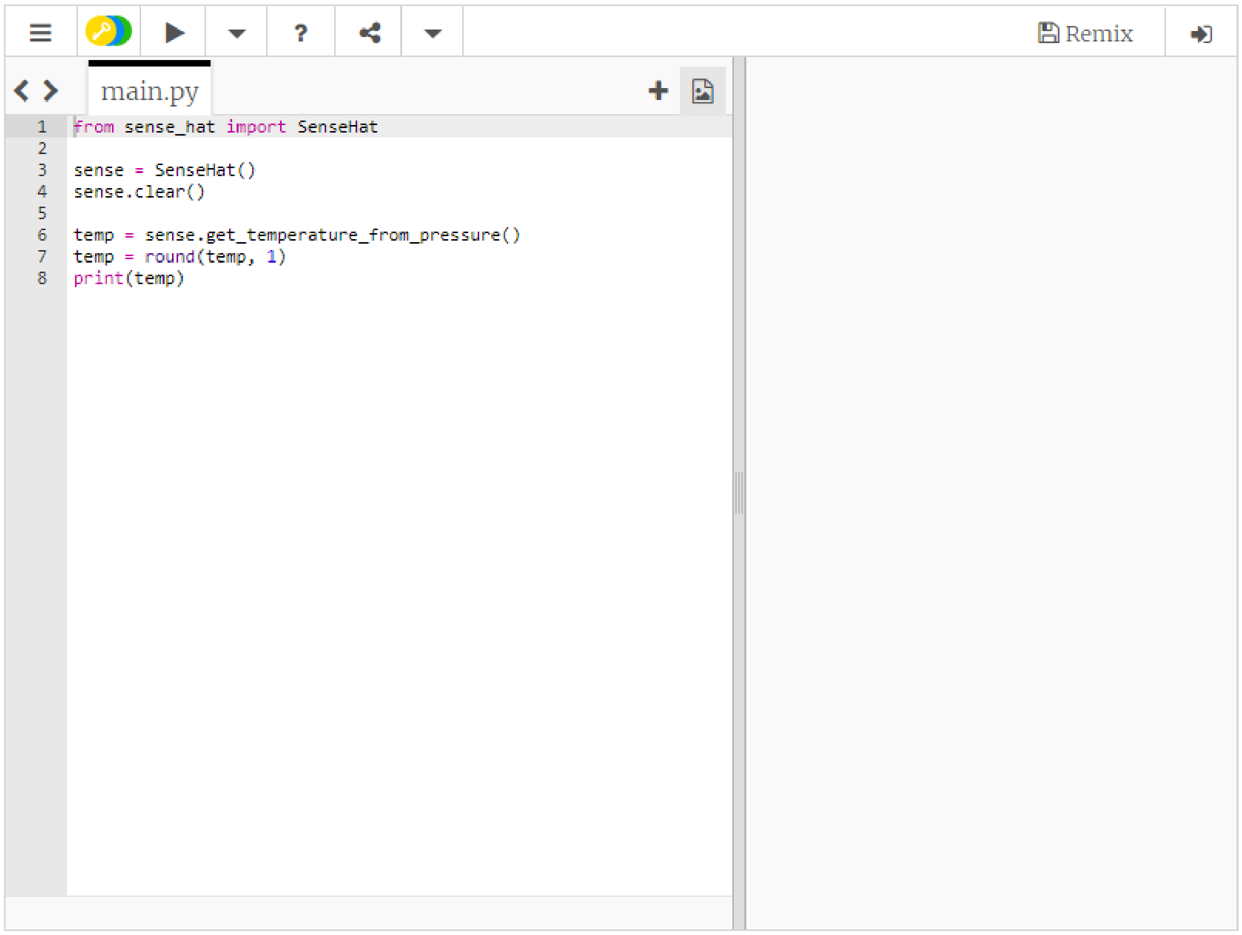
9. Try the following functions instead of `get_temperature`.

- `get_temperature_from_humidity` (uses the humidity sensor, `get_temperature` is a short version of this)
- `get_temperature_from_pressure` (uses the pressure sensor)

For example:

```
from sense_hat import SenseHat

sense = SenseHat()
sense.clear()

temp = sense.get_temperature_from_pressure()
temp = round(temp, 1)
print(temp)
```

Your code takes one measurement and then exits.



```
from sense_hat import SenseHat

sense = SenseHat()
sense.clear()

temp = sense.get_temperature_from_pressure()
temp = round(temp, 1)
print(temp)
```

Monitoring the Temperature

1. It would be good to monitor the temperature as it changes, so let's put our code into a while loop and run it again.

```
while True:
  temp = sense.get_temperature()
  temp = round(temp, 1)
  print(temp)
```

2. When you run the code the temperature values will scroll up the screen with the latest ones at the bottom.
3. Put your thumb over the sensor and hold it there. You should see the measurement start to rise.
4. Blow on it (or give the sensors a short blast from an air duster, if available). The measurement should fall.
5. Press `Ctrl + C` to stop the program.

Display the Temperature on the LED Matrix

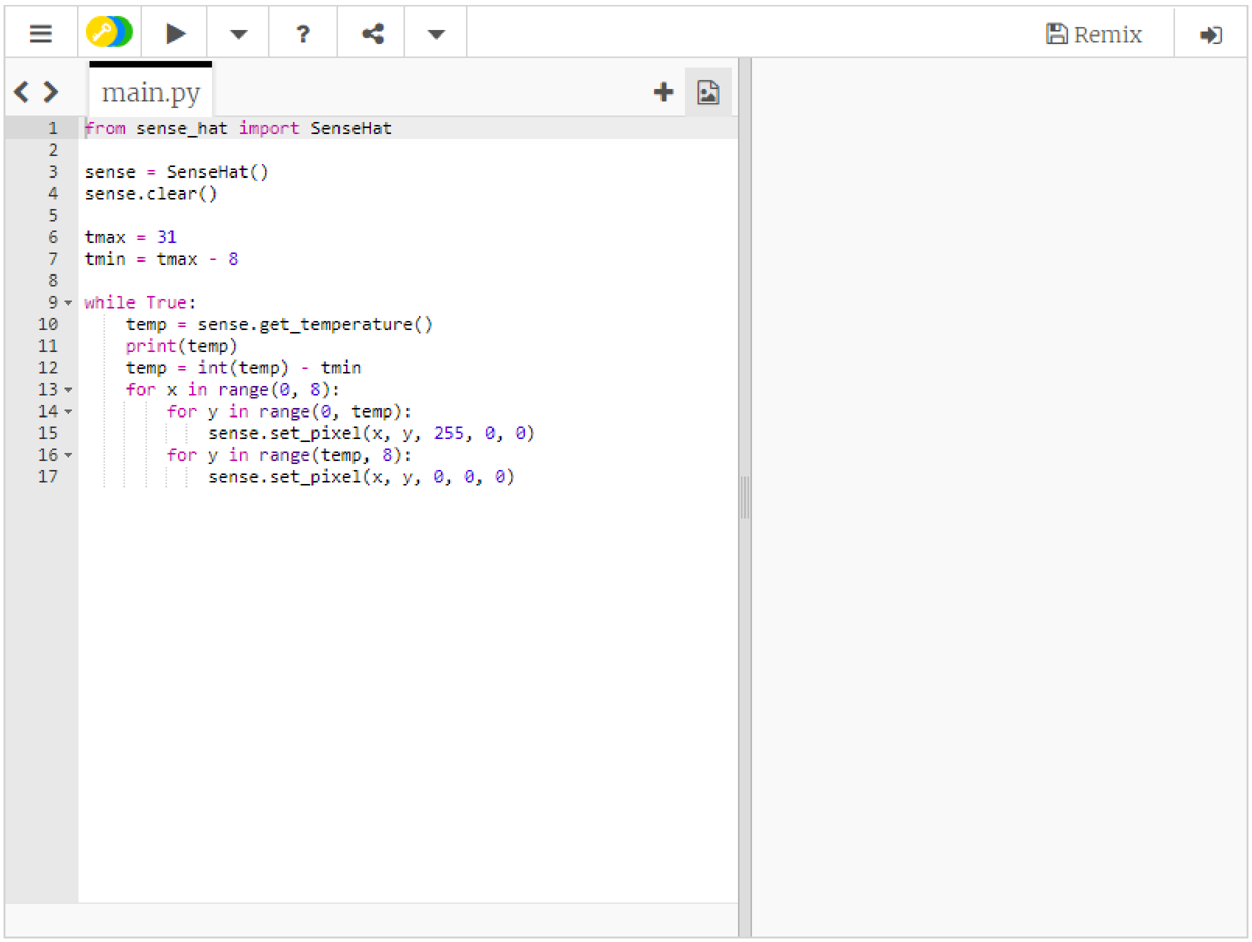
- Think about how you could show the temperature information on the LED matrix in some way (see the LED Matrix guide for more information). The obvious choice would be to use the `show_message` function, but, while this would work, there are probably better ways to do it. For example, you could:

- Use the `clear` function to display some predefined colours based on ranges that the temperature falls in. For example 0 to 5 degrees could be blue?
- Use the `clear` function to display a single colour but change the brightness of red (0 to 255) based on the measured temperature?
- Use the `set_pixel` function to display a bar that moves up and down similar to a thermometer.

Below is some starter code for the final suggestion above. This code will display a bar that has a range of 8 degrees Celsius (one degree per horizontal row of LEDs). The maximum it can display is `27` (hard coded; feel free to edit this) and so the minimum is `27 - 8` which is `19`. If the measured temperature goes outside of that range then errors can occur. You can add code to clamp the measured temperature to prevent these errors if you like.

```
from sense_hat import SenseHat

sense = SenseHat()
sense.clear()

tmax = 27
tmin = tmax - 8

while True:
    temp = sense.get_temperature()
    print(temp)
    temp = int(temp) - tmin
    for x in range(0, 8):
        for y in range(0, temp):
            sense.set_pixel(x, y, 255, 0, 0)
        for y in range(temp, 8):

            sense.set_pixel(x, y, 0, 0, 0)
```

It works by subtracting the minimum value from the measured value which should give a number between 0 and 8. We then use two nested `for` loops. The outer loop is for the x axis and the two inner loops are for the `y` axis. We use two loops here because we want to turn all the LEDs below the measurement red with `set_pixel` and those above it off. That way the bar will appear to move up and down the `y` axis following the measured temperature.



```
from sense_hat import SenseHat

sense = SenseHat()
sense.clear()

tmax = 31
tmin = tmax - 8

while True:
    temp = sense.get_temperature()
    print(temp)
    temp = int(temp) - tmin
    for x in range(0, 8):
        for y in range(0, temp):
            sense.set_pixel(x, y, 255, 0, 0)
        for y in range(temp, 8):
            sense.set_pixel(x, y, 0, 0, 0)
```

Remember that you can use `sense.set_rotation(n)` (where `n` is 0, 90, 180 or 270) at the start of the program just after `sense.clear()` if you want to change the orientation of the bar.

II. Humidity

What is the current Humidity?

1. Click on `Menu` > `Programming` > `Python 3 (IDLE)` to open a new Python shell.
2. Select `File > New Window` and enter the following code:

```
from sense_hat import SenseHat
```

sense = SenseHat() sense.clear()
humidity = sense.get_humidity() print(humidity)

```
1. Select `File > Save` and choose a file name for your program.
1. Select `Run > Run module`.
1. If you see the error `Humidity Init Failed, please run as root / use sudo` on
the last line in red, it means you haven't followed the instructions above. Close
everything and go back to step 1.
1. You should see something like this:
```

~~~
    ```bash
    Humidity sensor Init Succeeded
    34.6234588623
<iframe src="https://trinket.io/embed/python/cd35fdd905" width="100%" height="600"
frameborder="0" marginwidth="0" marginheight="0" allowfullscreen></iframe>
~~~

1. Just before the `print(humidity)` line add this line below:

```
humidity = round(humidity, 1)
```

2. You should now see something like this, without all the numbers after the decimal point:

```
Humidity sensor Init Succeeded
34.6
```

Monitoring humidity over time

1. It would be good to monitor the humidity as it changes, so let's put your code into a `while` loop and run it again:

```
while True:
    humidity = sense.get_humidity()
    humidity = round(humidity, 1)
    print(humidity)
```

2. Exhale slowly onto the sensors. The water vapour in your breath should cause the readings to jump up.
3. Keep watching and it should slowly fall back to the background humidity of the room.
4. Press `Ctrl + C` to stop the program.

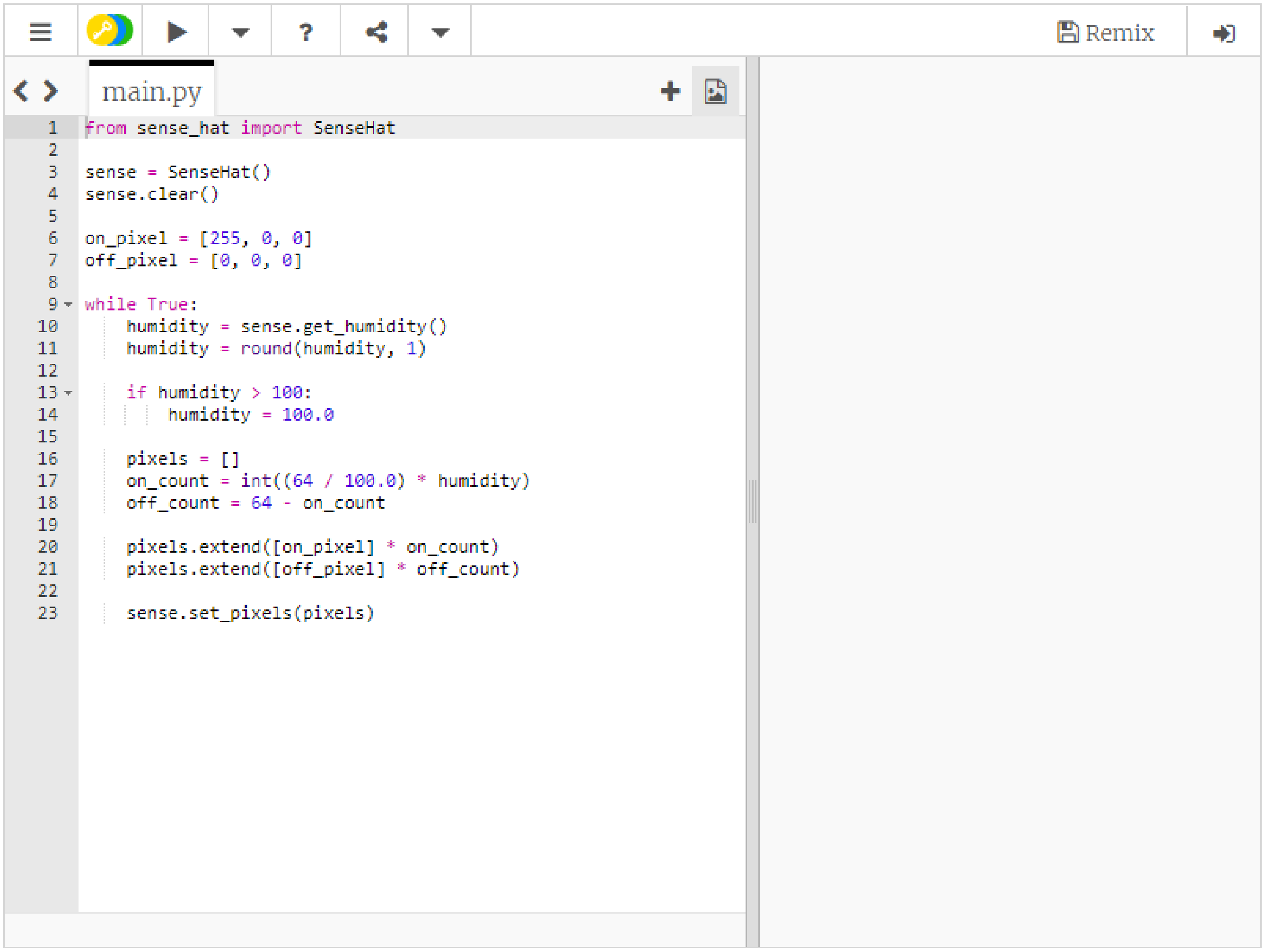
Display the Humidity on the LED Matrix

Think about how you might use the matrix to display the humidity. One way is to divide 64 (the number of LEDs in the matrix) by 100, then multiply that by the percentage of relative humidity, which gives you the number of LEDs you should turn on. 100%, for example, would be all 64 LEDs turned on. To do this, you need to build up a pixel list of the right number of light vs. dark pixels, and then call the `set_pixels` function.

Here is some example code (notice that you have to clamp the humidity measurement to 100):

```
from sense_hat import SenseHat
```

```python
sense = SenseHat()
sense.clear()

on_pixel = [255, 0, 0]
off_pixel = [0, 0, 0]

while True:
    humidity = sense.get_humidity()
    humidity = round(humidity, 1)

    if humidity > 100:
        humidity = 100.0

    pixels = []
    on_count = int((64 / 100.0) * humidity)
    off_count = 64 - on_count

    pixels.extend([on_pixel] * on_count)
    pixels.extend([off_pixel] * off_count)

    sense.set_pixels(pixels)
```



```
from sense_hat import SenseHat

sense = SenseHat()
sense.clear()

on_pixel = [255, 0, 0]
off_pixel = [0, 0, 0]

while True:
    humidity = sense.get_humidity()
    humidity = round(humidity, 1)

    if humidity > 100:
        humidity = 100.0

    pixels = []
    on_count = int((64 / 100.0) * humidity)
    off_count = 64 - on_count

    pixels.extend([on_pixel] * on_count)
    pixels.extend([off_pixel] * off_count)

    sense.set_pixels(pixels)
```

Please note that it is possible to get a value higher than 100 from the humidity sensor.

III. Pressure

What is the Pressure?

1. Click on `Menu` > `Programming` > `Python 3 (IDLE)` to open a new Python shell.
2. Select `File > New Window` and enter the following code:

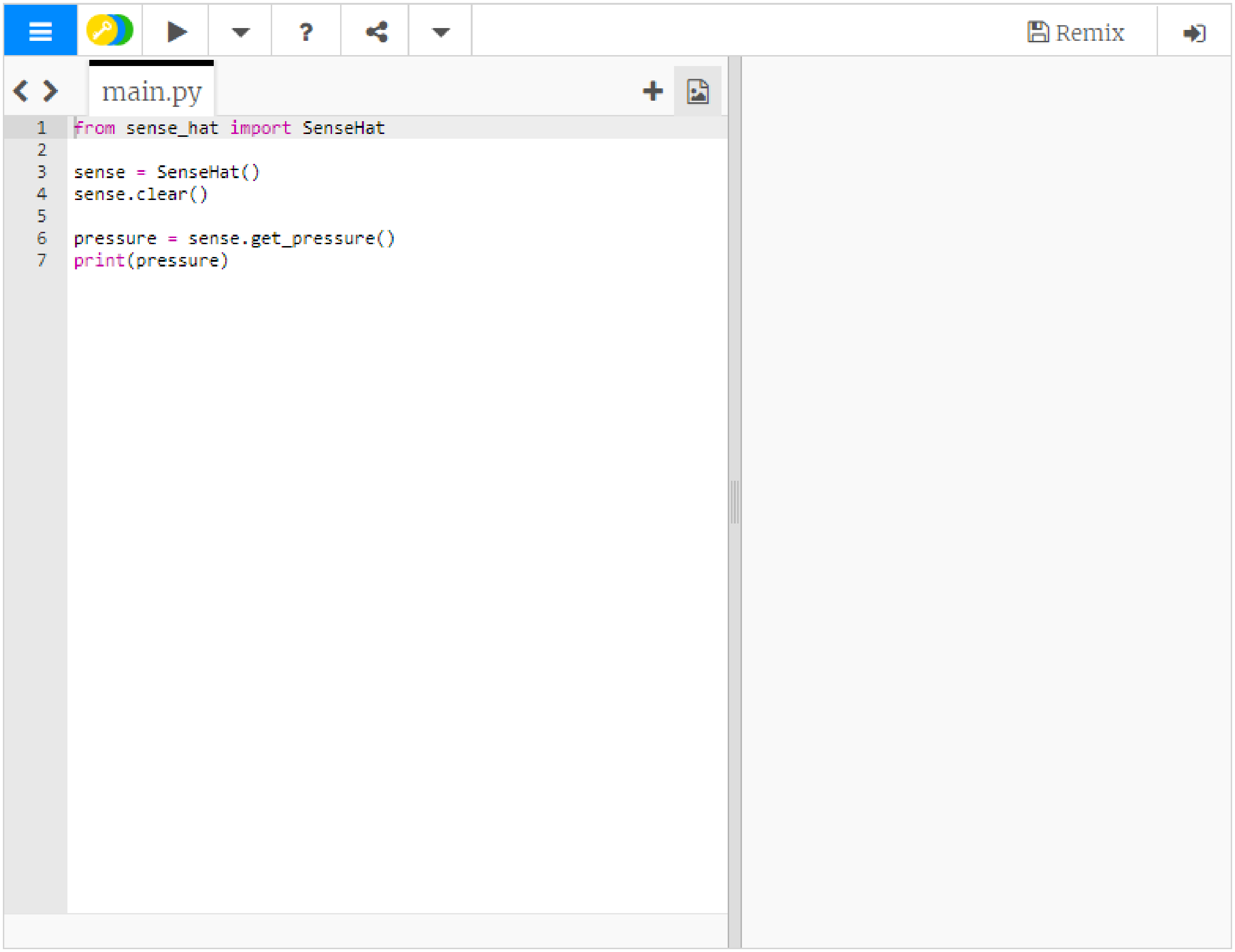
```
from sense_hat import SenseHat

sense = SenseHat()
sense.clear()

pressure = sense.get_pressure()
print(pressure)
```

3. Select `File > Save` and choose a file name for your program.
4. Select `Run > Run module`.

5. If you see the error `Pressure Init Failed, please run as root / use sudo` on the last line in red, it means you haven't followed the instructions above. Close everything and go back to step 1.
6. You should see something like this:

```
Pressure sensor Init Succeeded
1013.40380859
```

If you get `0` just run the code again. This sometimes happens when you use the pressure sensor for the first time.



```python
from sense_hat import SenseHat

sense = SenseHat()
sense.clear()

pressure = sense.get_pressure()
print(pressure)
```

Just before the `print(pressure)` line, add this line below:

```
pressure = round(pressure, 1)
```

Monitoring the Pressure

1. It would be good to monitor the pressure as it changes, so let's put your code into a `while` loop and run it again:

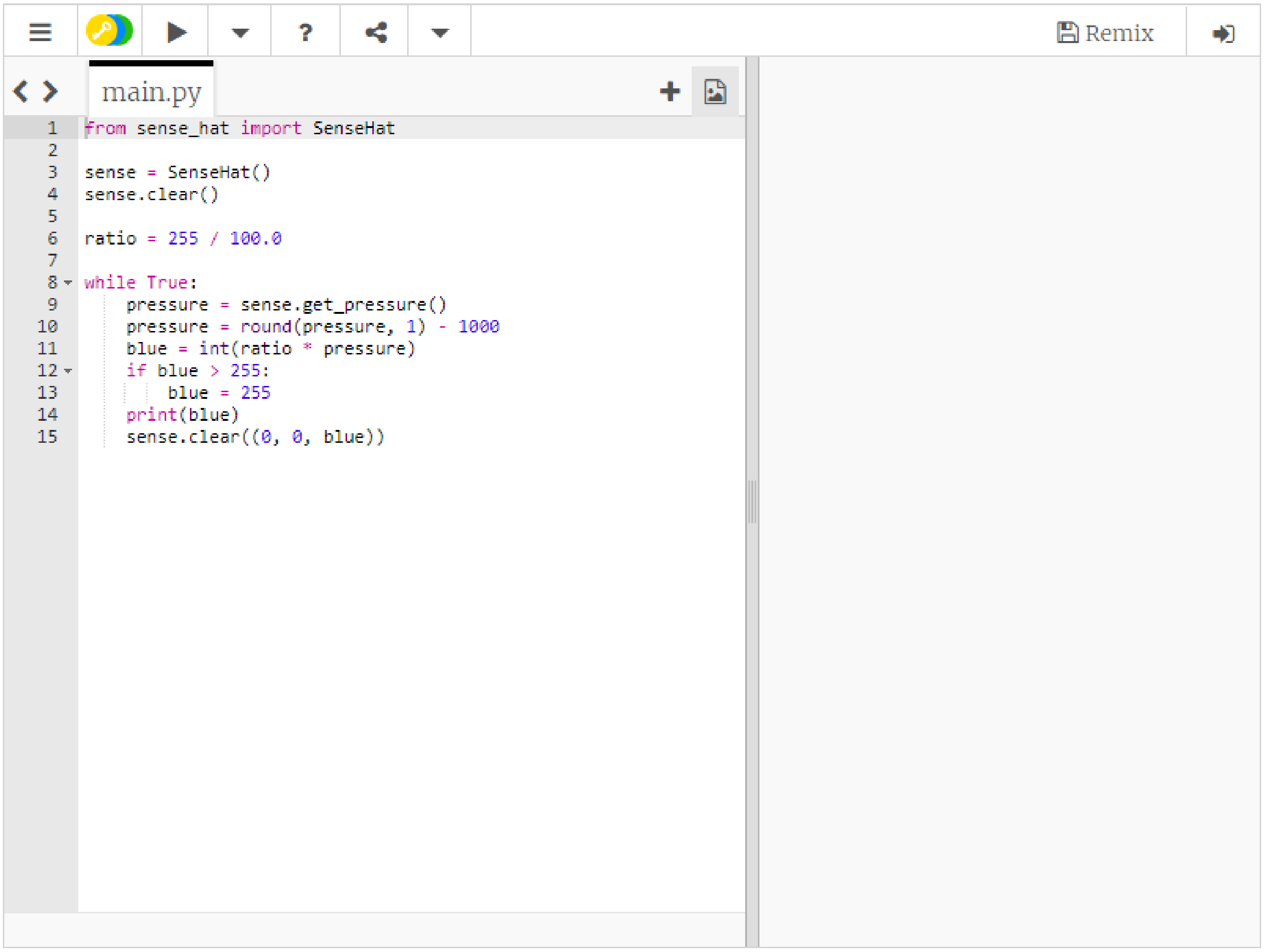
```
while True:
    pressure = sense.get_pressure()
    pressure = round(pressure, 1)
    print(pressure)
```

2. Unfortunately, it's not as easy to make it change as holding your thumb on the sensor or breathing on it, so use the plastic bottle experiment below to test your code.

Display the Pressure on the LED Matrix

Below is the code that was used in the video. It can cope with 1000 to 1100 millibars, so that's 100 millibars of range. We know that the LED matrix colours have a range of 0 to 255, so the first thing it does is create a ratio between the pressure range and the colour range. The plan is then to multiply the measured pressure by that ratio to get the colour. You have to subtract 1000 from the measured pressure to make this work, so you're multiplying a number between 0 and 100 by the ratio. It then clamps the colour to a maximum of 255, in case there is someone with very strong lungs who can drive the pressure higher than 1100 millibars.

```
from sense_hat import SenseHat

sense = SenseHat()
sense.clear()

ratio = 255 / 100.0

while True:
    pressure = sense.get_pressure()
    pressure = round(pressure, 1) - 1000
    blue = int(ratio * pressure)
    if blue > 255:
        blue = 255
    elif blue < 0:
        blue = 0

    sense.clear((0, 0, blue))
```



```python
from sense_hat import SenseHat

sense = SenseHat()
sense.clear()

ratio = 255 / 100.0

while True:
    pressure = sense.get_pressure()
    pressure = round(pressure, 1) - 1000
    blue = int(ratio * pressure)
    if blue > 255:
        blue = 255
    print(blue)
    sense.clear((0, 0, blue))
```

IV. Movement

The Sense HAT has a movement sensor called an IMU, which can measure the kind of movement it is experiencing.

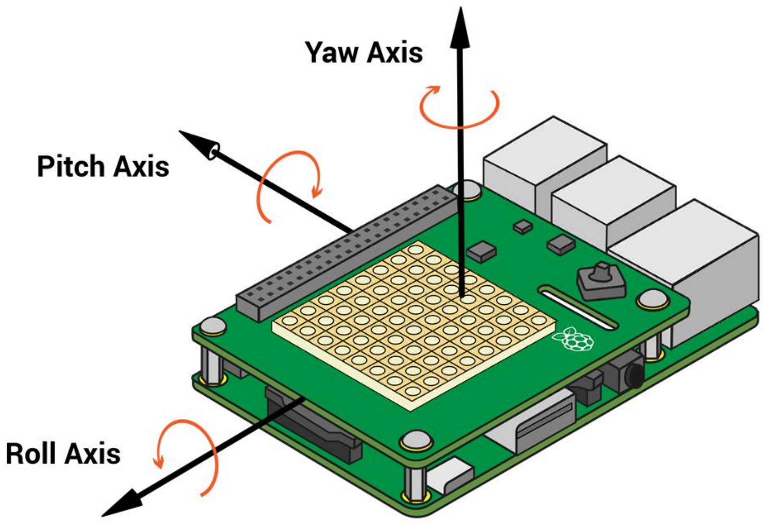
How is orientation presented?

We all know the Earth rotates around an axis that runs between the North and South Poles. All objects in space or otherwise have *three* axes around which they can rotate. If you know how much rotation has happened on each axis, then you know which way the object is pointing.

The three axes are:

- **Pitch** (like a plane taking off)
- **Roll** (the plane doing a victory roll)
- **Yaw** (imagine steering the plane like a car)

Watch this short video that shows where these axes are in relation to a plane. Try to imagine the plane pointing in any random direction. To get the plane into that position, you can rotate it by a known amount around each axis to get it to the orientation that you imagined.



The image above shows where these axes are in relation to the Sense HAT.

Let's download and run a 3D demo program to explore this.

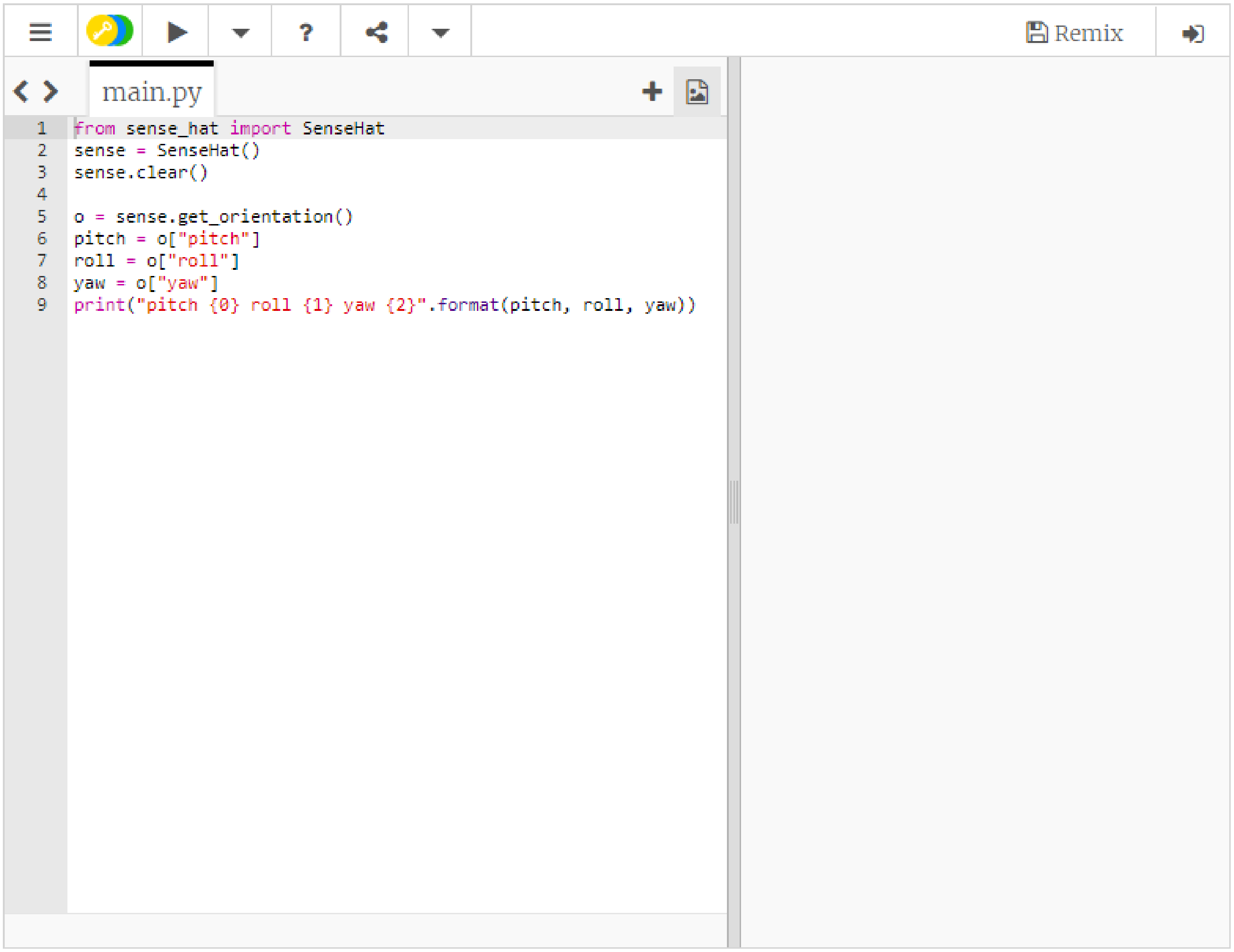
Which way am I pointing?

1. Click on `Menu` > `Programming` > `Python 3 (IDLE)` to open a new Python shell.
2. Select `File > New Window` and enter the following code:

```
from sense_hat import SenseHat
sense = SenseHat()
sense.clear()

o = sense.get_orientation()
pitch = o["pitch"]
roll = o["roll"]
yaw = o["yaw"]
print("pitch {0} roll {1} yaw {2}".format(pitch, roll, yaw))
```

3. Select `File > Save` and choose a file name for your program.
4. Select `Run > Run module`.
5. If you see the error `IMU Init Failed, please run as root / use sudo` on the last line in red, it means you haven't followed the instructions above. Close everything and go back to step 1.
6. You should now see something like this:

```
IMU Init Succeeded
pitch 356.35723002363454 roll 303.4986602798494 yaw 339.19880231669873
```



```python
from sense_hat import SenseHat
sense = SenseHat()
sense.clear()

o = sense.get_orientation()
pitch = o["pitch"]
roll = o["roll"]
yaw = o["yaw"]
print("pitch {0} roll {1} yaw {2}".format(pitch, roll, yaw))
```

7. We don't need all the numbers after the decimal point so let's round them off. Just before the `print("pitch %s roll %s yaw %s" % (pitch, roll, yaw))` line, add these lines below:

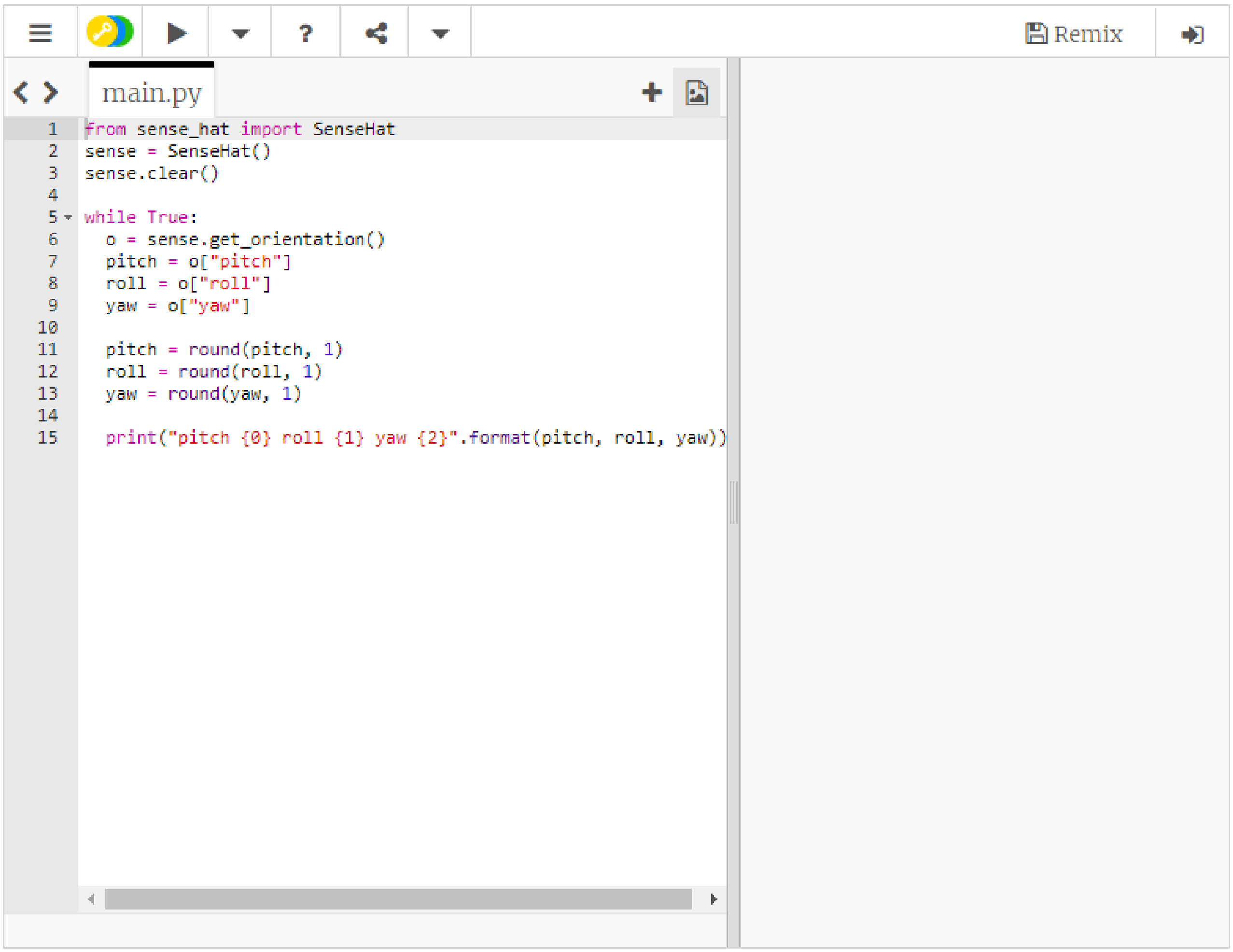
```python
pitch = round(pitch, 1)
roll = round(roll, 1)
yaw = round(yaw, 1)
```

Monitor movement over time

1. It would be good to monitor the axis values changing during movements, so let's put your code into a `while` loop and run it again:

```python
while True:
    o = sense.get_orientation()
    pitch = o["pitch"]
    roll = o["roll"]
    yaw = o["yaw"]

    pitch = round(pitch, 1)
    roll = round(roll, 1)
    yaw = round(yaw, 1)

    print("pitch {0} roll {1} yaw {2}".format(pitch, roll, yaw))
```

2. Move the Pi around in your hand and you should see the numbers changing. See if you can just make one axis change by moving only in the pitch direction for example. Do this for all three axes. Press `Ctrl - C` to stop the program.



```
from sense_hat import SenseHat
sense = SenseHat()
sense.clear()

while True:
  o = sense.get_orientation()
  pitch = o["pitch"]
  roll = o["roll"]
  yaw = o["yaw"]

  pitch = round(pitch, 1)
  roll = round(roll, 1)
  yaw = round(yaw, 1)

  print("pitch {0} roll {1} yaw {2}".format(pitch, roll, yaw))
```

Display orientation on the LED Matrix

1. Displaying something which is 3D in a 2D way is always a challenge, especially when your screen is only 8 x 8 pixels in size. One way which might work well is to have one LED for each axis and then make them move in different ways. For example:

- The pitch LED could go up and down
- The roll LED could go side to side
- The yaw LED could chase around the edge

2. Here is a clever trick you can do. The table below shows the sequential LED numbers laid out in horizontal rows.

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

For any of those numbers you can convert them into their X Y coordinate using the code below.

```
y = number // 8
x = number % 8
```

For the `y` value you floor divide `//` the number by 8. This is integer division and ignores the remainder. Then for the `x` value you do the modulus `%` of 8 which gives you **only** the remainder.

For example (using the number 60 which is on the bottom row):

- `60 // 8 = 7`
- `60 % 8 = 4`

3. Try this code:

```
number = 60

y = number // 8
x = number % 8

sense.set_pixel(x, y, 255, 255, 255)
```

4. The clever trick is to make a list containing the LED numbers for the path you want it to move back and forth through. So say you want to make an LED chase around the edge you would read the numbers accross the top of the table, down the right hand side, backwards along to bottom and up the left side. So it would be:

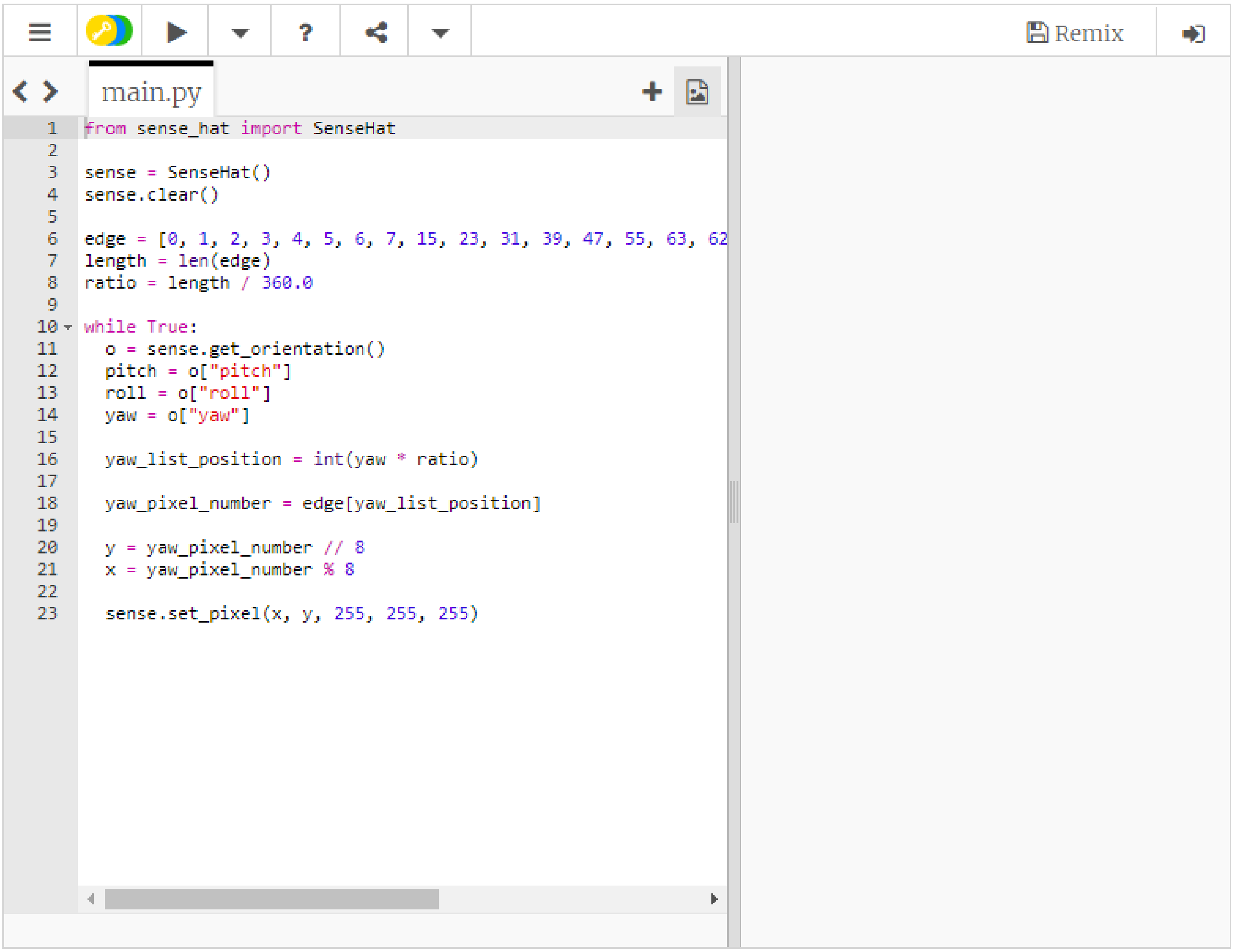
```
edge = [0, 1, 2, 3, 4, 5, 6, 7, 15, 23, 31, 39, 47, 55, 63, 62, 61, 60, 59, 58,
57, 56, 48, 40, 32, 24, 16, 8]
```

We can then find the length of the list using the `len` function:

```
length = len(edge)
```

So the length is 28. If we divide 28 by 360 we have a ratio between, say, the yaw measurement and the positions in our list (how far around the edge we are). We can then get the sequential pixel number out of the list at the calculated position, work out its coordinate and then switch the LED on! Like this:

```
from sense_hat import SenseHat

sense = SenseHat()
sense.clear()

edge = [0, 1, 2, 3, 4, 5, 6, 7, 15, 23, 31, 39, 47, 55, 63, 62, 61, 60, 59, 58,
57, 56, 48, 40, 32, 24, 16, 8]
length = len(edge)
ratio = length / 360.0

while True:
    o = sense.get_orientation()
    pitch = o["pitch"]
    roll = o["roll"]
    yaw = o["yaw"]

    yaw_list_position = int(yaw * ratio)

    yaw_pixel_number = edge[yaw_list_position]

    y = yaw_pixel_number // 8
    x = yaw_pixel_number % 8

    sense.set_pixel(x, y, 255, 255, 255)
```



```
from sense_hat import SenseHat

sense = SenseHat()
sense.clear()

edge = [0, 1, 2, 3, 4, 5, 6, 7, 15, 23, 31, 39, 47, 55, 63, 62
length = len(edge)
ratio = length / 360.0

while True:
  o = sense.get_orientation()
  pitch = o["pitch"]
  roll = o["roll"]
  yaw = o["yaw"]

  yaw_list_position = int(yaw * ratio)

  yaw_pixel_number = edge[yaw_list_position]

  y = yaw_pixel_number // 8
  x = yaw_pixel_number % 8

  sense.set_pixel(x, y, 255, 255, 255)
```

5. What you'll notice is that the above code only turns LEDs on, you'll need to figure out how to turn them off yourself. Try having a variable for the previous `x` and `y` from the last time around the loop and if this is different from the new `x` and `y` you use `set_pixel` to set the LED to black.

Inputs & Outputs

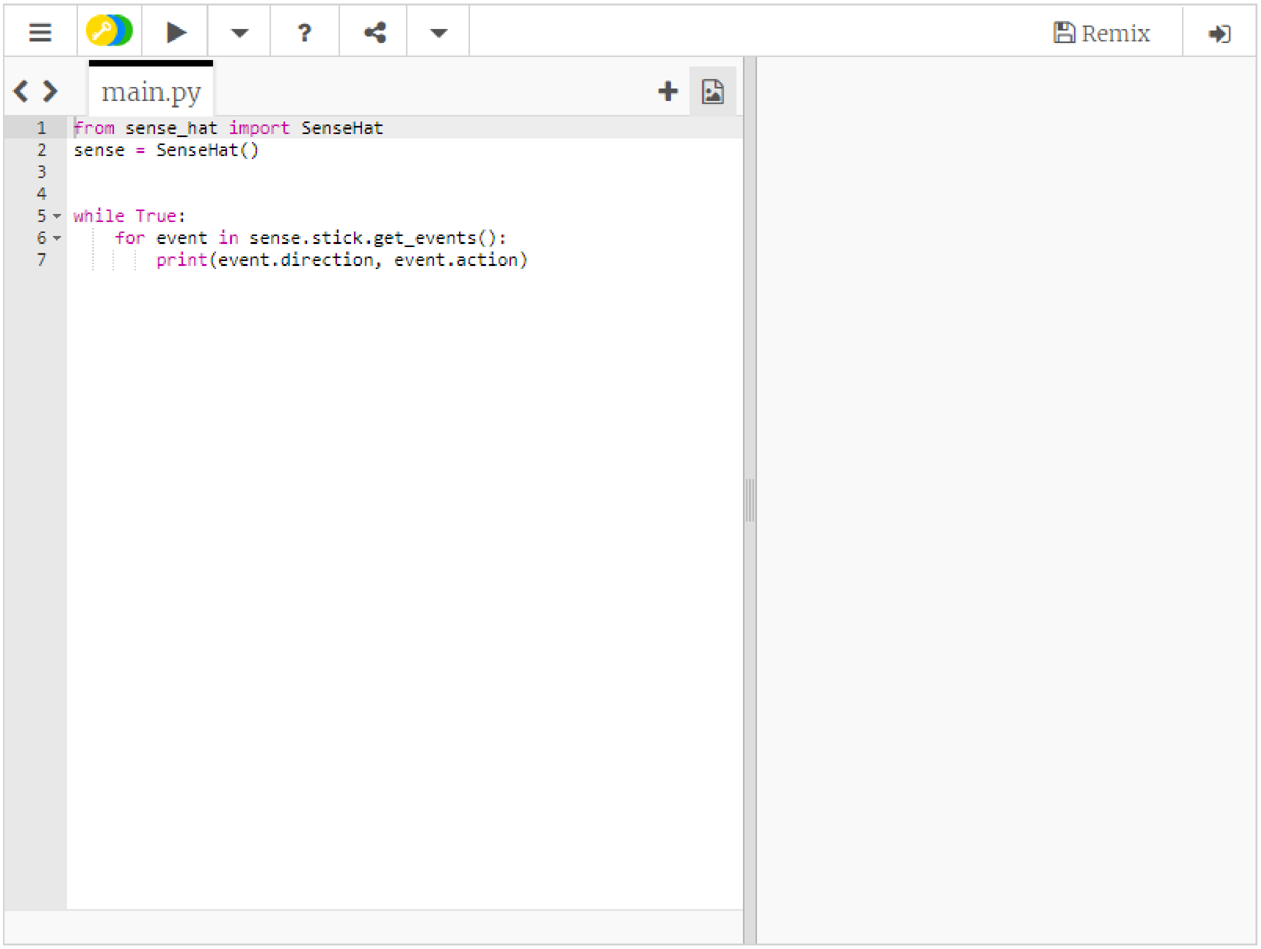
I. Joystick

Accessing the joystick events

To find out what each joystick event outputs, you can print the events to the shell. This bit of code will print out the direction that the joystick was pushed in, or the direction from which it was released.

```
from sense_hat import SenseHat
sense = SenseHat()

while True:
    for event in sense.stick.get_events():
        print(event.direction, event.action)
```

In the trinket emulator, you can pretend to be moving the joystick using your keyboard's cursor keys.



```
from sense_hat import SenseHat
sense = SenseHat()


while True:
    for event in sense.stick.get_events():
        print(event.direction, event.action)
```

You should see something like the following printed out, as you press the joystick in various directions.

```
('up', 'pressed')
('up', 'released')
('down', 'pressed')
('down', 'released')
('left', 'pressed')
('left', 'released')
('right', 'pressed')
('right', 'released')
('middle', 'pressed')
('middle', 'released')
```

Moving a pixel using the joystick

One example of how you could use this in a program would be to control a pixel on the LED matrix, moving it around and changing its colour, for instance.

You can start by setting the `x` and `y` coordinates for the pixel to be illuminated, providing a list of colours, and then providing a value for which colour in the list to use on the illuminated pixel.

```
from sense_hat import SenseHat
sense = SenseHat()

x, y = 0, 0
colours = [[255,0,0], [0,255,0], [0,0,255], [255,255,0], [255,0,255], [0,255,255]]
colour = 0
```

Next, you can use the same code as before to detect joystick movements, but set a pixel each time the joystick is moved, instead of printing.

```
while True:
    for event in sense.stick.get_events():
        sense.set_pixel(x, y, colours[colour])
```

The next stage is to detect when the joystick is pressed, and the direction it's pressed in, and change the `x` and `y` position of the pixel.

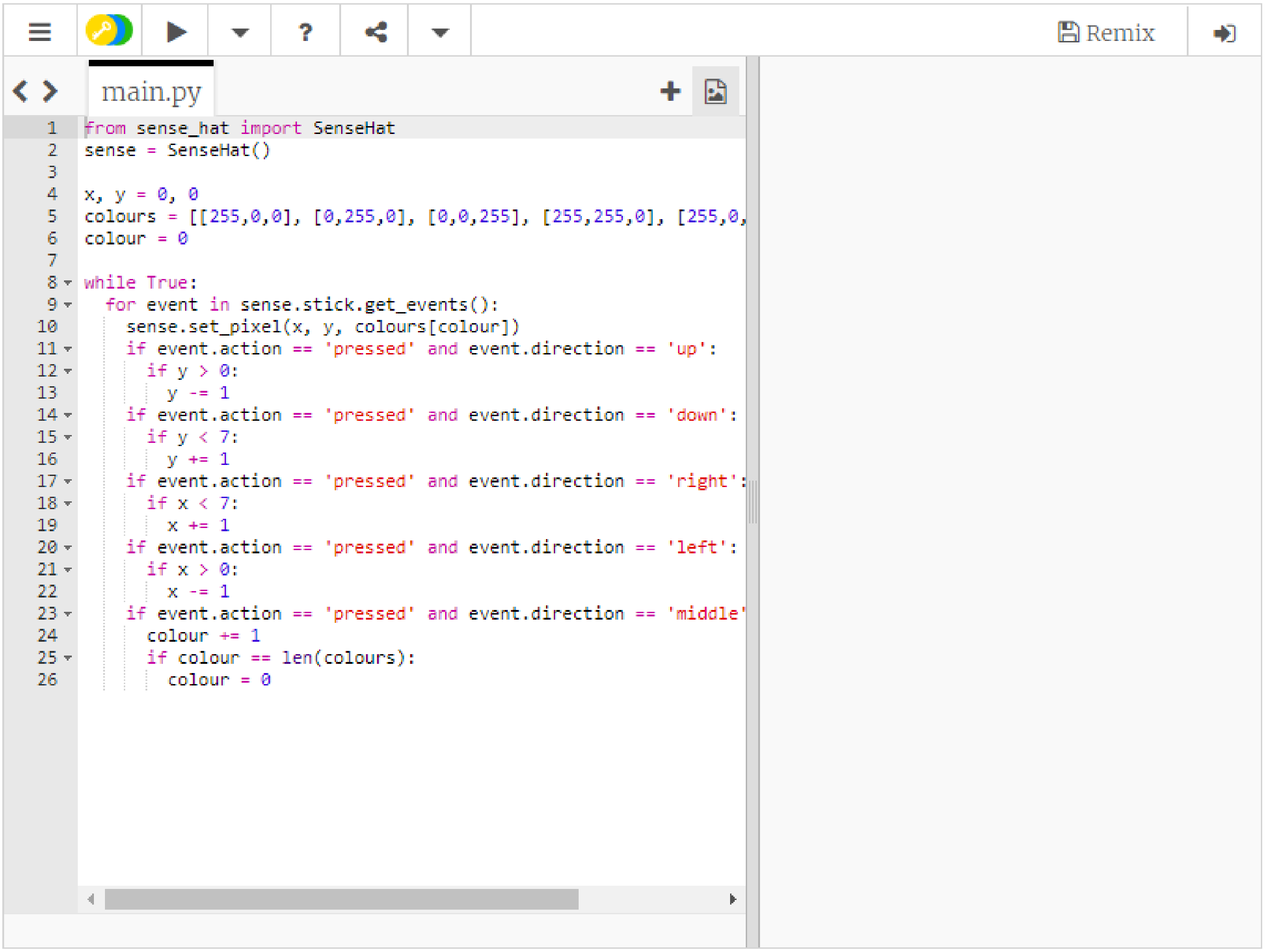
```
while True:
    for event in sense.stick.get_events():
        sense.set_pixel(x, y, colours[colour])
        if event.action == 'pressed' and event.direction == 'up':
            y -= 1
        if event.action == 'pressed' and event.direction == 'down':
            y += 1
        if event.action == 'pressed' and event.direction == 'right':
            x += 1
        if event.action == 'pressed' and event.direction == 'left':
             x -= 1
```

You can try this out if you like, but there's a problem. If the pixel falls off the edge of the matrix, the program will produce an error, so this needs handling in the code.

```
while True:
    for event in sense.stick.get_events():
        sense.set_pixel(x, y, colours[colour])
        if event.action == 'pressed' and event.direction == 'up':
            if y > 0:
                y -= 1
        if event.action == 'pressed' and event.direction == 'down':
            if y < 7:
                y += 1
        if event.action == 'pressed' and event.direction == 'right':
            if x < 7:
                x += 1
        if event.action == 'pressed' and event.direction == 'left':
            if x > 0:
                x -= 1
```

Lastly, if the joystick's direction is middle, you can change the `colour`. If the `colour` value is longer than the length of the `colours` list, then it needs to reset to `0`.

```
while True:
    for event in sense.stick.get_events():
        sense.set_pixel(x, y, colours[colour])
        if event.action == 'pressed' and event.direction == 'up':
            if y > 0:
                y -= 1
        if event.action == 'pressed' and event.direction == 'down':
            if y < 7:
                y += 1
        if event.action == 'pressed' and event.direction == 'right':
            if x < 7:
                x += 1
        if event.action == 'pressed' and event.direction == 'left':
            if x > 0:
                x -= 1
        if event.action == 'pressed' and event.direction == 'middle':
            colour += 1
            if colour == len(colours):
                colour = 0
```

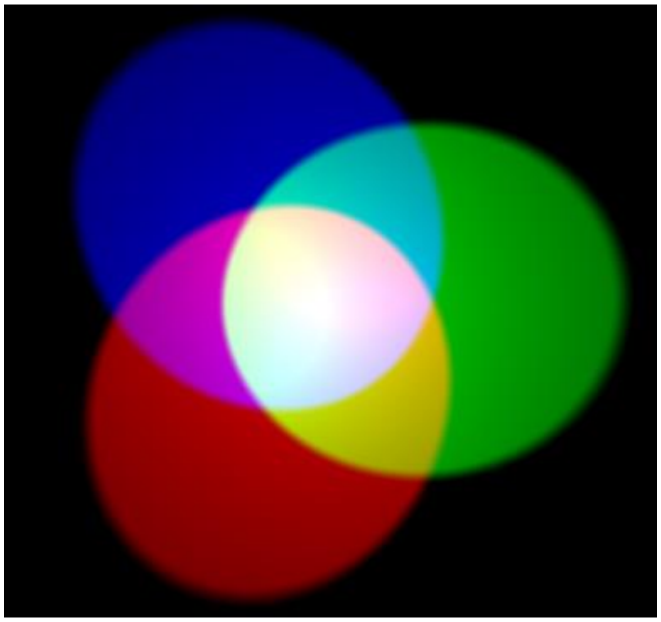


```python
from sense_hat import SenseHat
sense = SenseHat()

x, y = 0, 0
colours = [[255,0,0], [0,255,0], [0,0,255], [255,255,0], [255,0,
colour = 0

while True:
  for event in sense.stick.get_events():
    sense.set_pixel(x, y, colours[colour])
    if event.action == 'pressed' and event.direction == 'up':
      if y > 0:
        y -= 1
    if event.action == 'pressed' and event.direction == 'down':
      if y < 7:
        y += 1
    if event.action == 'pressed' and event.direction == 'right':
      if x < 7:
        x += 1
    if event.action == 'pressed' and event.direction == 'left':
      if x > 0:
        x -= 1
    if event.action == 'pressed' and event.direction == 'middle'
      colour += 1
      if colour == len(colours):
        colour = 0
```

II. LED Matrix

The Sense HAT LED matrix contains 64 multi-colour LEDs. Each of the 64 LEDs actually have three smaller LEDs inside them, just like a pixel on a TV, monitor or smartphone screen.

Simple color mixing

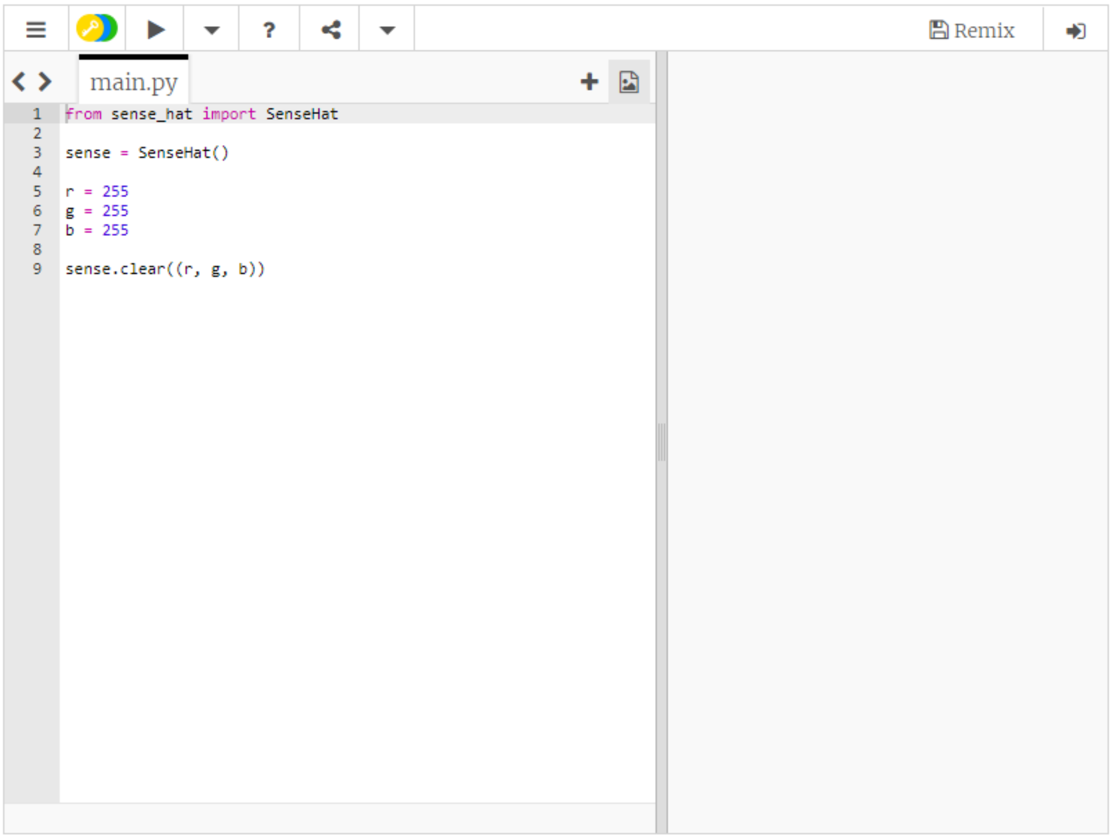


In additive colour mixing three colours: red, green, and blue are used to make other colours. In the image above, there are three spotlights of equal brightness, one for each colour. In the absence of any colour the result is black. If all three colours are mixed, the result is white. When red and green combine, the result is yellow. When red and blue combine, the result is magenta. When blue and green combine, the result is cyan. It's possible to make even more colours than this by varying the brightness of the three original colours used.

1. Click on `Menu` > `Programming` > `Python 3 (IDLE)` to open a new Python shell.
2. Select `File > New Window`.
3. Type in the following code:

```
from sense_hat import SenseHat

sense = SenseHat()

r = 255
g = 255
b = 255

sense.clear((r, g, b))
```

4. Select `File > Save` and choose a file name for your program.
5. Then select `Run > Run module`.
6. The LED matrix will then go bright white.



```
from sense_hat import SenseHat

sense = SenseHat()

r = 255
g = 255
b = 255

sense.clear((r, g, b))
```

1. The variables `r`, `g` and `b` represent the colours red, green, and blue. The numbers they contain specify how bright each colour should be; they can be between 0 and 255. In the above code the maximum value for each colour has been used, so the result is white.

2. Change the values to specify 255 red but 0 green and blue, then run the code again.

3. What other colours can you make?

Changing foreground and background colours

This colour mixing system is used throughout the Astro Pi programming module. You can use colour mixing to great effect by programming scrolling text. In this example, you can set the colour of the text that will appear on the matrix.

1. Type the following code into a new file:

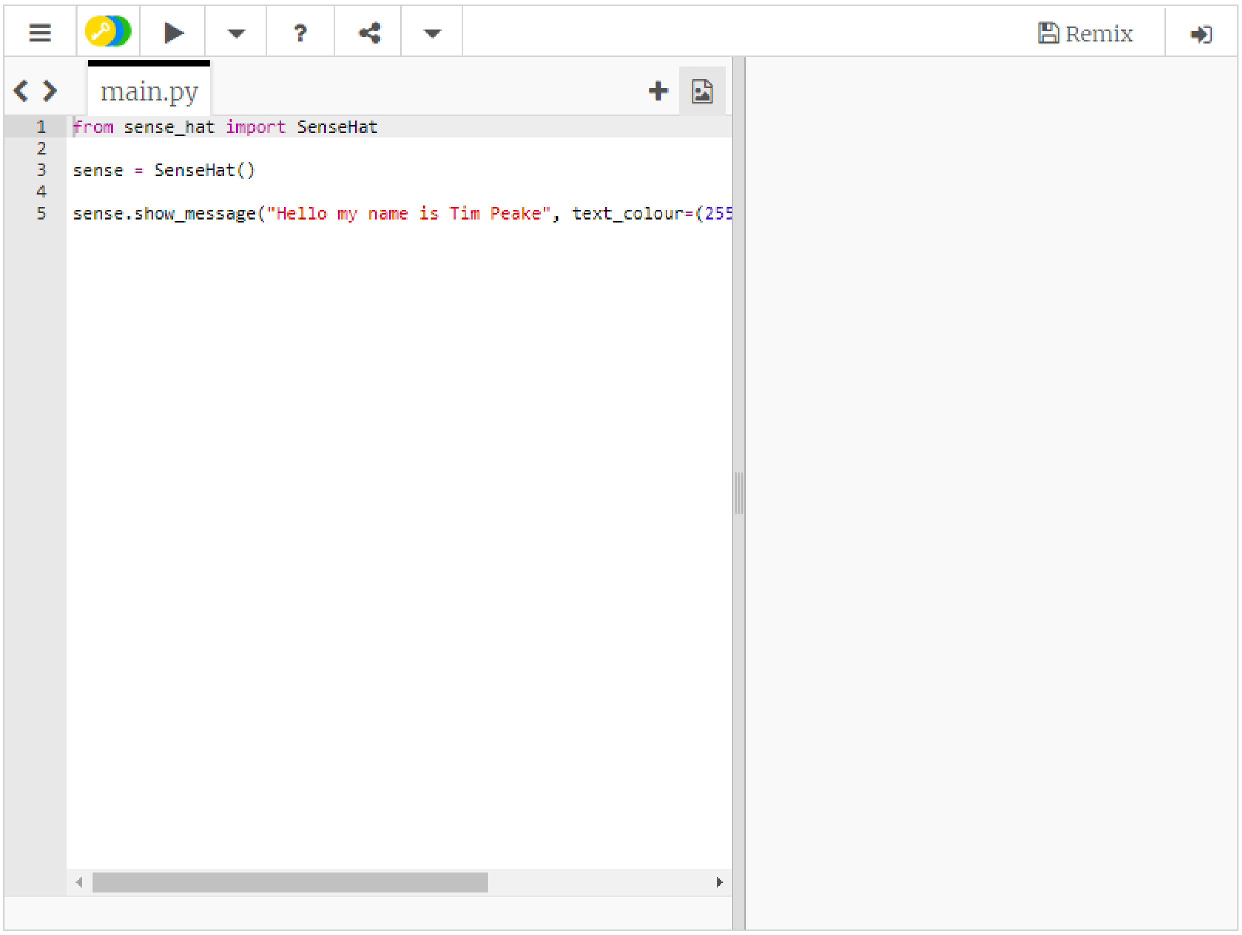
```
from sense_hat import SenseHat

sense = SenseHat()

sense.show_message("Hello my name is Tim Peake", text_colour=(255, 0, 0))
```

Note the syntax `, text_colour=(255, 0, 0)`. *Don't forget the comma!*

2. You can also modify the background colour for the message like so:

```
from sense_hat import SenseHat

sense = SenseHat()

sense.show_message("Hello my name is Tim Peake", text_colour=(255, 255, 0),
back_colour=(0, 0, 255))
```

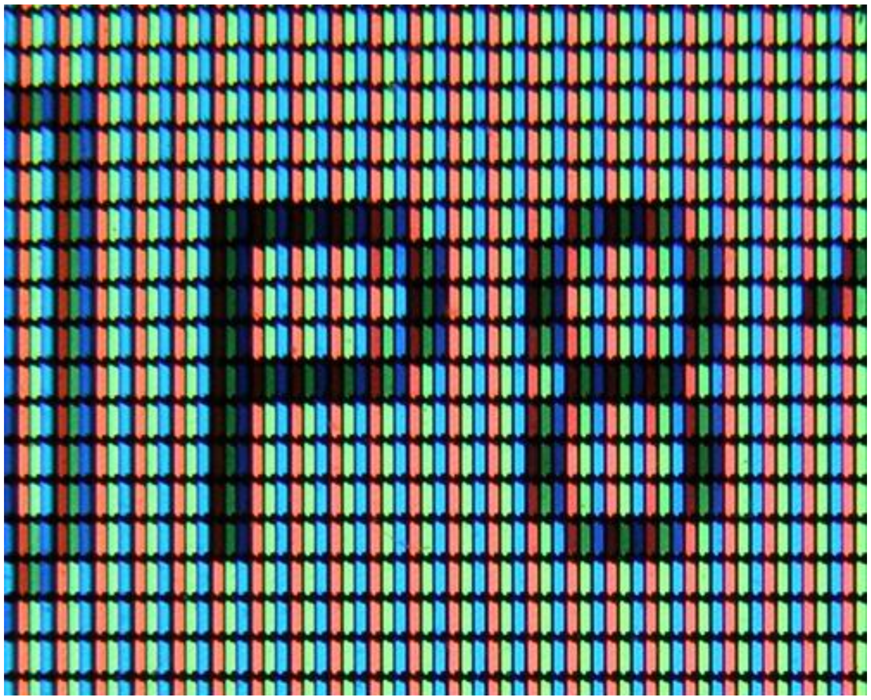
Note: The comma is important, don't forget it!



```
from sense_hat import SenseHat

sense = SenseHat()

sense.show_message("Hello my name is Tim Peake", text_colour=(255
```

Pixels

This image shows the pixels on a laptop LCD screen. You can see that the pixels are turned on and off to form the pattern of letters and numbers.

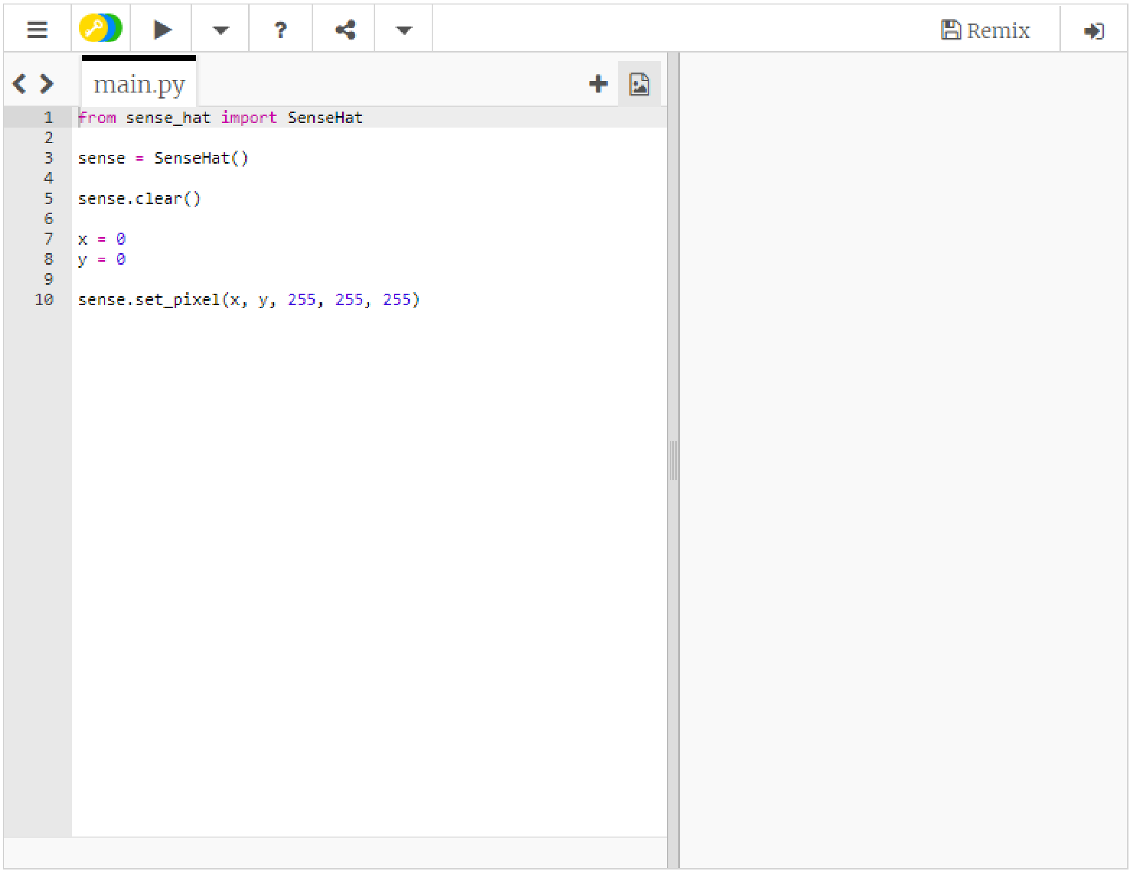


This is how all computer and smartphone screens work. If you want to make recognisable shapes on the LED matrix this is what you also need to do. You only have a resolution of 8 by 8 pixels to work with on the Sense HAT LED matrix though, so you must make shapes and icons that will look quite blocky. This can be a nice challenge!

1. Select `File > New Window`.
2. Type in the following code:

```
from sense_hat import SenseHat

sense = SenseHat()

sense.clear()

x = 0
y = 0

sense.set_pixel(x, y, 255, 255, 255)
```

3. Select `File > Save` and choose a file name for your program.
4. Then select `Run > Run module`.
5. This will turn one LED in the corner white.

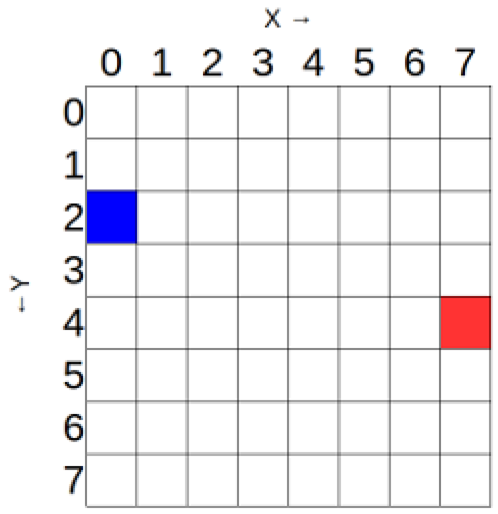


```
from sense_hat import SenseHat

sense = SenseHat()

sense.clear()

x = 0
y = 0

sense.set_pixel(x, y, 255, 255, 255)
```

6. Remember that you can change the colour if you wish.

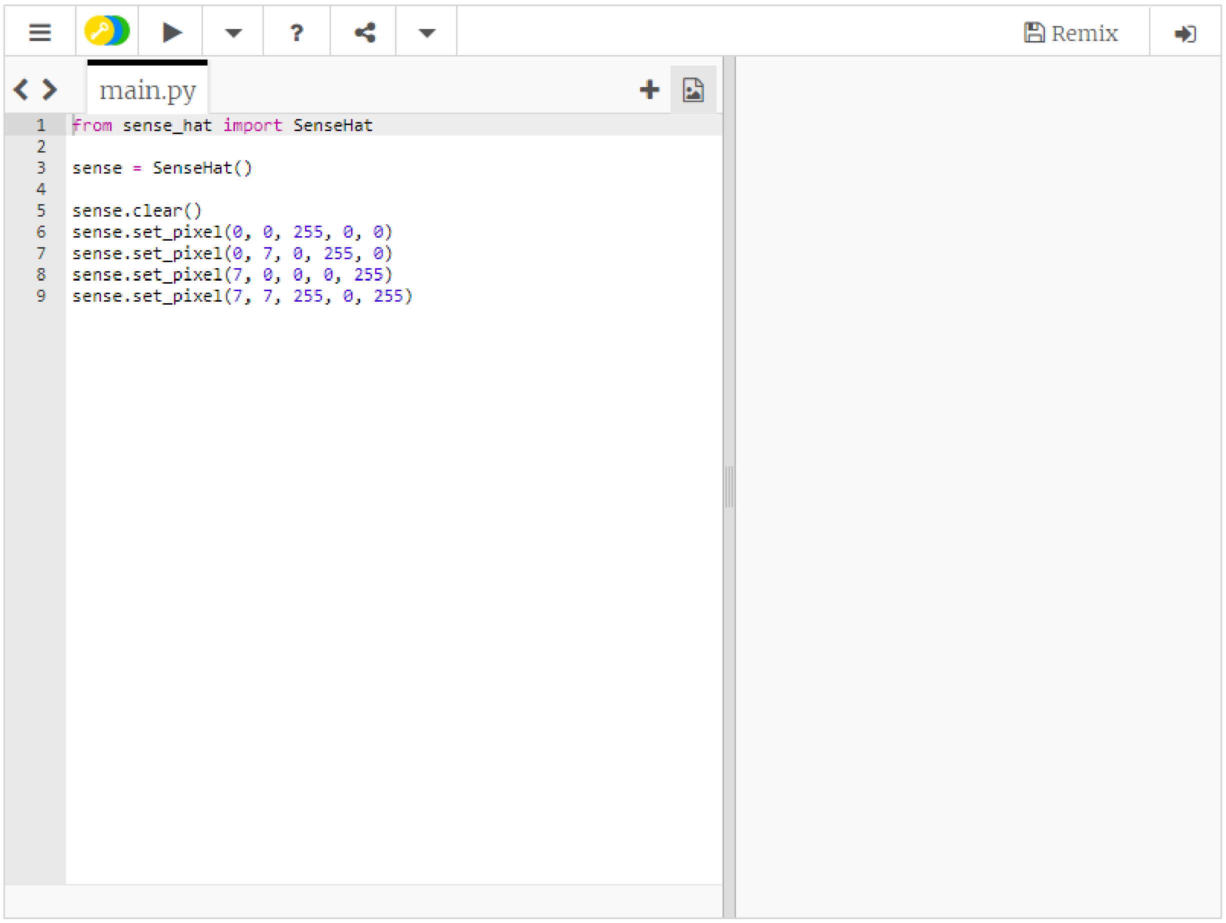
Using coordinates to set pixels

The `x` and `y` variables can be used to control which individual LED the `set_pixel` command should change. **X** is horizontal and ranges from `0` on the *left* to `7` on the *right*. **Y** is vertical and ranges from `0` at the *top* to `7` on the *bottom*. Therefore, an `x, y` coordinate of `0, 0` is the *top left*and an `x, y` coordinate of `7, 7` is the *bottom right*.



You can get a different colour in each corner of the LED matrix. You will need to use the `set_pixel`command multiple times in your code like this:

```
from sense_hat import SenseHat

sense = SenseHat()

sense.clear()
sense.set_pixel(0, 0, 255, 0, 0)
sense.set_pixel(0, 7, 0, 255, 0)
sense.set_pixel(7, 0, 0, 0, 255)
sense.set_pixel(7, 7, 255, 0, 255)
```



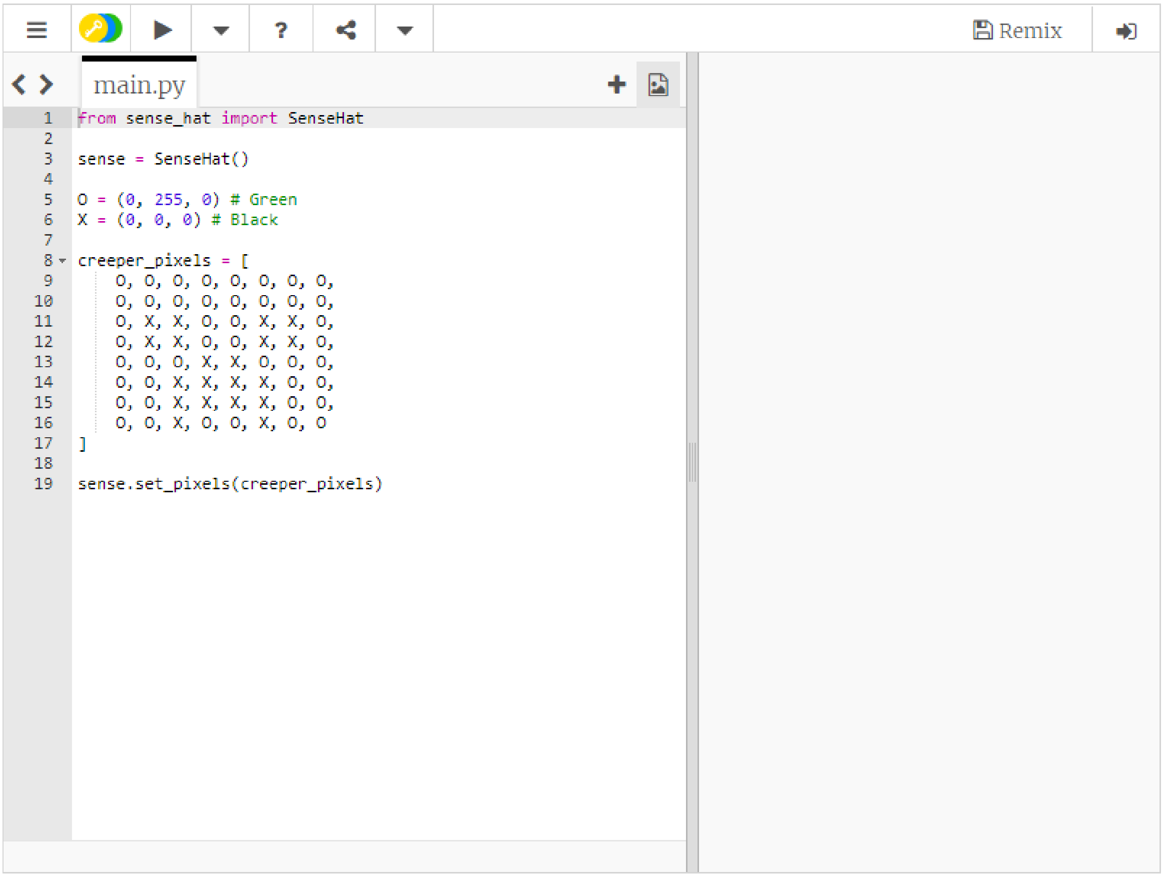
```
from sense_hat import SenseHat

sense = SenseHat()

sense.clear()
sense.set_pixel(0, 0, 255, 0, 0)
sense.set_pixel(0, 7, 0, 255, 0)
sense.set_pixel(7, 0, 0, 0, 255)
sense.set_pixel(7, 7, 255, 0, 255)
```

Drawing shapes and patterns on the LED Matrix

You may be tempted to try and draw shapes or patterns using the `set_pixel` command over and over in your code. There is a `set_pixels` command though, and with it you can change all 64 LEDs using one line of code! For example, you could draw a Minecraft creeper face on the LED Matrix:

```
from sense_hat import SenseHat

sense = SenseHat()

O = (0, 255, 0) # Green
X = (0, 0, 0) # Black

creeper_pixels = [
    O, O, O, O, O, O, O, O,
    O, O, O, O, O, O, O, O,
    O, X, X, O, O, X, X, O,
    O, X, X, O, O, X, X, O,
    O, O, O, X, X, O, O, O,
    O, O, X, X, X, X, O, O,
    O, O, X, X, X, X, O, O,
    O, O, X, O, O, X, O, O
]
```

```
sense.set_pixels(creeper_pixels)
```



```
from sense_hat import SenseHat

sense = SenseHat()

O = (0, 255, 0) # Green
X = (0, 0, 0) # Black

creeper_pixels = [
    O, O, O, O, O, O, O, O,
    O, O, O, O, O, O, O, O,
    O, X, X, O, O, X, X, O,
    O, X, X, O, O, X, X, O,
    O, O, O, X, X, O, O, O,
    O, O, X, X, X, X, O, O,
    O, O, X, X, X, X, O, O,
    O, O, X, O, O, X, O, O
]

sense.set_pixels(creeper_pixels)
```

You can even use more than two colours, like in this example of Steve from Minecraft:

```
from sense_hat import SenseHat

sense = SenseHat()

B = (102, 51, 0)
b = (0, 0, 255)
S = (205,133,63)
W = (255, 255, 255)

steve_pixels = [
    B, B, B, B, B, B, B, B,
    B, B, B, B, B, B, B, B,
    B, S, S, S, S, S, S, B,
    S, S, S, S, S, S, S, S,
    S, W, b, S, S, b, W, S,
    S, S, S, B, B, S, S, S,
    S, S, B, S, S, B, S, S,
    S, S, B, B, B, B, S, S
]
```

```
sense.set_pixels(steve_pixels)
```

Loading images from files

Instead of setting the LED matrix, you may wish to use images which are loaded from files. This is a convenient option if you want to have lots of stock images, for example international flags.

1. Use any graphics editing tool (on Windows, OS X or Linux) to create the files. As long as they are saved onto the Raspberry Pi's SD card as `JPEG` or `PNG`, and are 8 x 8 pixels in size, then they can be loaded directly to the LED matrix with a single command.
2. Open the *File Manager* on the Raspberry Pi using `Menu > Accessories > File Manager`.
3. Browse into the `astro-pi-hat` folder followed by `examples`. There should be a file named `space_invader.png` which you can double-click.
4. Load the image onto the Sense Hat LED matrix by using the `load_image` function, which needs the file system path to the file you want to load. So for `space_invader.png` the full path is `/home/pi/astro-pi-hat/examples/space_invader.png`.

Here is the code:

```
from sense_hat import SenseHat

sense = SenseHat()

sense.load_image("/home/pi/astro-pi-hat/examples/space_invader.png")
```