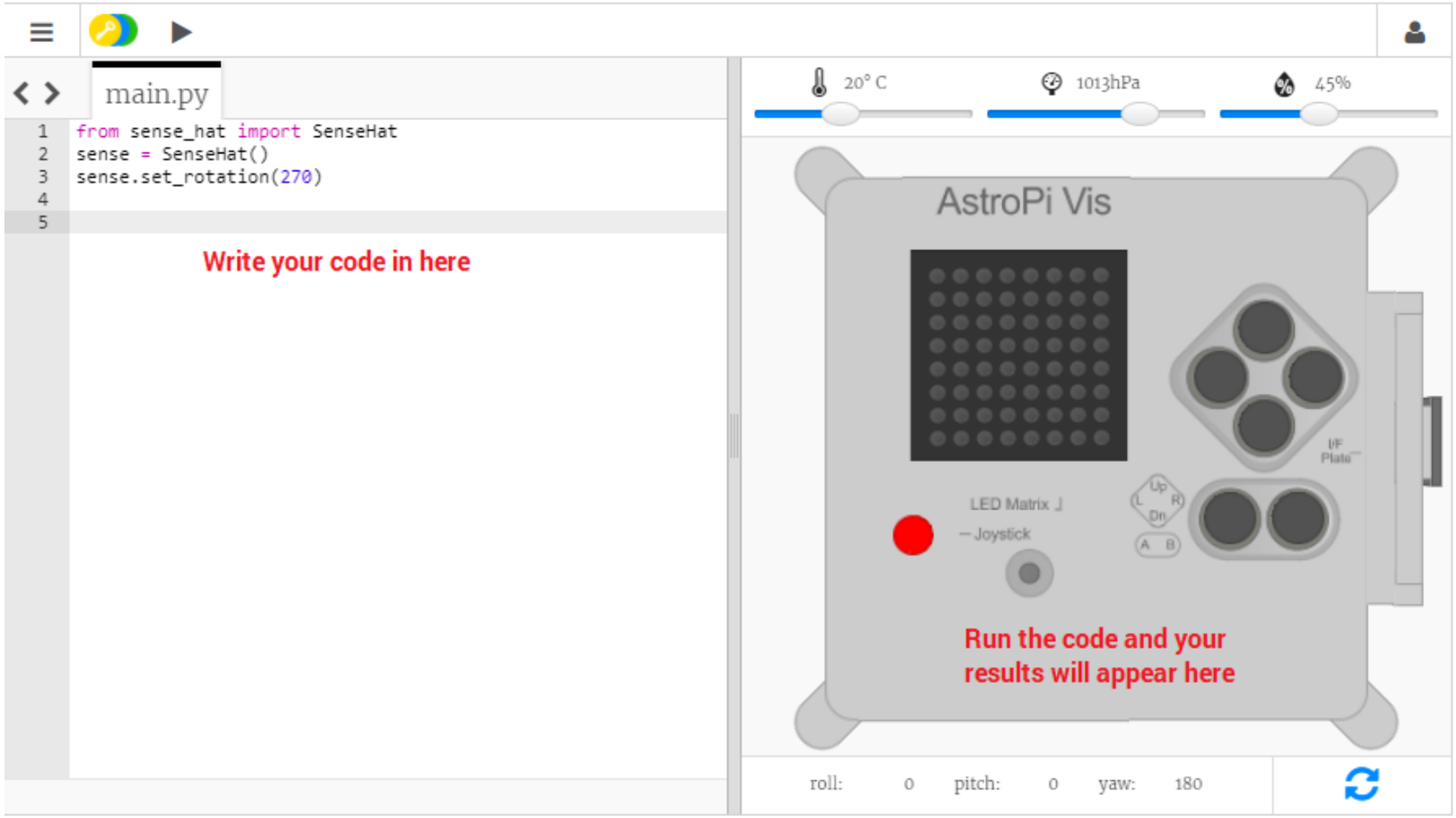

Bodgers

Last Week

We used the online Sense HAT emulator to create our program, https://trinket.io/mission-zero

We followed the tutorial at https://projects.raspberrypi.org/en/projects/astro-pi-mission-zero



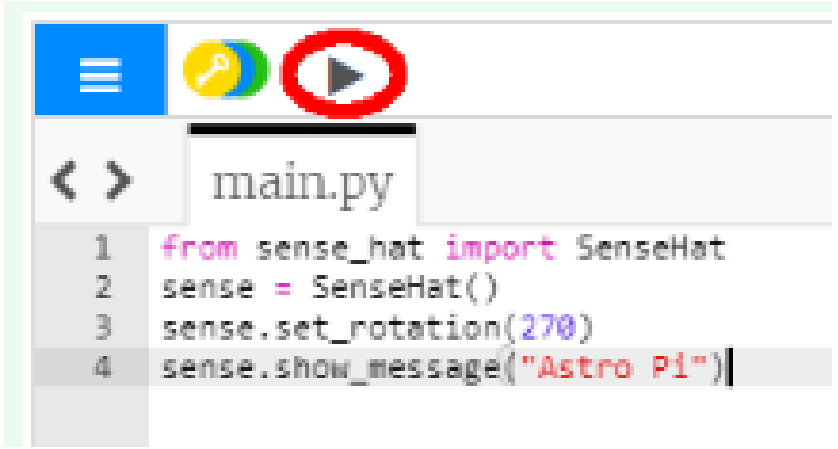
https://trinket.io/python/04b90b70cf

```
from sense_hat import SenseHat
sense = SenseHat()
sense.set_rotation(270)
```

This code connects to the Astro Pi and makes sure the Astro Pi's LED display is shown the correct way around.

```
sense.show_message("Astro Pi")
```

Press the Run button and watch as the message Astro Pi scrolls across the LED display.



```
sense.show_message("Astro Pi", scroll_speed=0.05)
```

The default speed of the message is 0.1. Making the number smaller makes the message scroll more quickly, and making it larger makes the message scroll more slowly.

Create a variable to store your chosen colour. For example, if you picked red, you would write this line of code.

```
red = (255,0,0)
sense.show_message("Astro Pi", text_colour=red)
```

You can now display your text in the colour of your choice! To tell the program to use the colour you created, add a text_colour parameter to the code which displays your text.

You can also change the background colour of the display. Pick another colour, and create another variable to store that colour. To tell the program to use your chosen background colour, add the back_colour parameter to your code:

```
red = (255,0,0)
green = (0,255,0)
sense.show_message("Astro Pi", text_colour=red, back_colour=green)
```

At the bottom of your program, create some colour variables to define the colours with which you want to to draw your picture. You can use as many colours as you like, but in this example we'll stick to only two — white (w) and black (b).

```
w = (255, 255, 255)
b = (0, 0, 0)
```

Below your new variables, create a list of 64 items. Each item represents one pixel on the LED matrix, and corresponds to one of the colour variables you defined.
Draw your picture by putting a variable where you want its assigned colour to appear. We have drawn an astronaut by using the black (b) pixels as the background and the white (w) pixels to draw the astronaut's space suit:

```
picture = [
    b, b, w, w, w, w, b, b,
    b, w, b, b, b, b, w, b,
    b, w, b, w, w, b, w, b,
    b, w, b, b, b, b, w, b,
    b, b, w, w, w, w, b, b,
    b, b, w, w, w, w, b, b,
    b, w, w, w, w, w, w, b,
    b, w, w, w, w, w, w, b
]
```

Add a line of code to display your picture on the LED display.

```
sense.set_pixels(picture)
```

Press Run to see your picture displayed.

You might want to add some code to include a short wait (or sleep) after the picture is displayed. This will give the astronauts time to see your picture before the next part of your message appears. At the top of your program, add:

```
from time import sleep
```

Then, on the line after the one that displays your picture, add this code to wait for two seconds:

```
sleep(2)
```

Add this code to take a temperature reading:

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

The temperature is recorded very precisely, i.e. the stored value will have a large number of decimal places. You can round the value to any number of decimal places.

```
temp = round( sense.get_temperature(), 1 )
```

To display the current temperature as a scrolling message on the display, add this line of code:

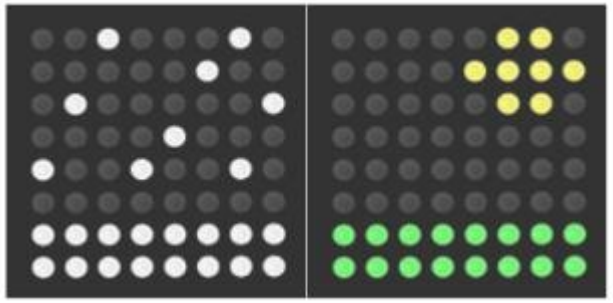
```
sense.show_message( str(temp) )
```

The str() part converts the temperature from a number into text so that the Astro Pi can display it.

You can also display the temperature as part of another message by joining the parts of your message together with a +.

```
sense.show_message( "It is " + str(temp) + " degrees" )
```

Display The Temperature



You could combine your temperature reading with a picture to show the temperature. For example, you might display a snowstorm for cold temperatures, and a sunny day for hot temperatures.

At the bottom of your program, create more colour variables for any colours you want to use in your pictures. You may already have defined some of them in a previous step. In our examples we will use white (w), yellow (y), green (g), and black/blank (b).

```
w = (255, 255, 255)
y = (255, 255, 0)
g = (0, 255, 0)
b = (0, 0, 0)
```

Just like earlier, draw your pictures by first creating a list for each of them, and then setting the list items to the colours you want your pixels to be.

```
hot = [
  b, b, b, b, b, y, y, b,
  b, b, b, b, y, y, y, y,
  b, b, b, b, b, y, y, b,
  b, b, b, b, b, b, b, b,
  b, b, b, b, b, b, b, b,
  b, b, b, b, b, b, b, b,
  g, g, g, g, g, g, g, g,
  g, g, g, g, g, g, g, g
]


cold = [
  b, b, w, b, b, b, w, b,
  b, b, b, b, b, w, b, b,
  b, w, b, b, b, b, b, w,
  b, b, b, b, w, b, b, b,
  w, b, b, w, b, b, w, b,
  b, b, b, b, b, b, b, b,
  w, w, w, w, w, w, w, w,
  w, w, w, w, w, w, w, w
]
```

Now decide which picture to display. For this example, we will display the hot image if the temperature reading is 20 degrees or above, and the cold image if the temperature is below 20 degrees.

```
temp = sense.get_temperature()
if temp >= 20:
    sense.set_pixels(hot)
else:
    sense.set_pixels(cold)
```

Use the temperature slider to set a temperature on the emulator. Run your program and check that the image you've selected for that temperature is correctly displayed.

Your own version?