Bodgers

Astro Pi Mission Zero Challenge

The European Astro Pi Challenge is an ESA Education
project run in collaboration with the Raspberry Pi
Foundation. It offers young people the amazing opportunity to write computer programs that run on Raspberry Pi computers on board the International Space Station (ISS).
Mission Zero offers participants up to 14 years old the chance to have their code run on the ISS!
Teams write a simple program to display a message and the temperature readings on the Astro Pi computer, available for the astronauts to see as they go about their daily tasks. No special hardware or prior coding skills are needed and all entries that follow program rules are guaranteed to have their program run in space!

Astro Pi Mission Zero Challenge

To take part, teams must:

- Be made up of students/young people who each are 14 years or younger
- Have at least two and at most four students/young people as members
- Be supervised by a teacher, mentor, or educator, who will be the point of contact with the Astro Pi team
- Be made up of at least 50% team members who are citizens of an ESA Member/Associate Member State

In addition, each team member must be at least one of the following:

- Enrolled full-time in a primary or secondary school located in an ESA Member/Associate Member State
- Home-schooled (certified by the National Ministry of Education or delegated authority in an ESA Member or Associate Member State)
- A member of a club or after-school group, such as Code Club, CoderDojo, or Scouts

What Will Happen

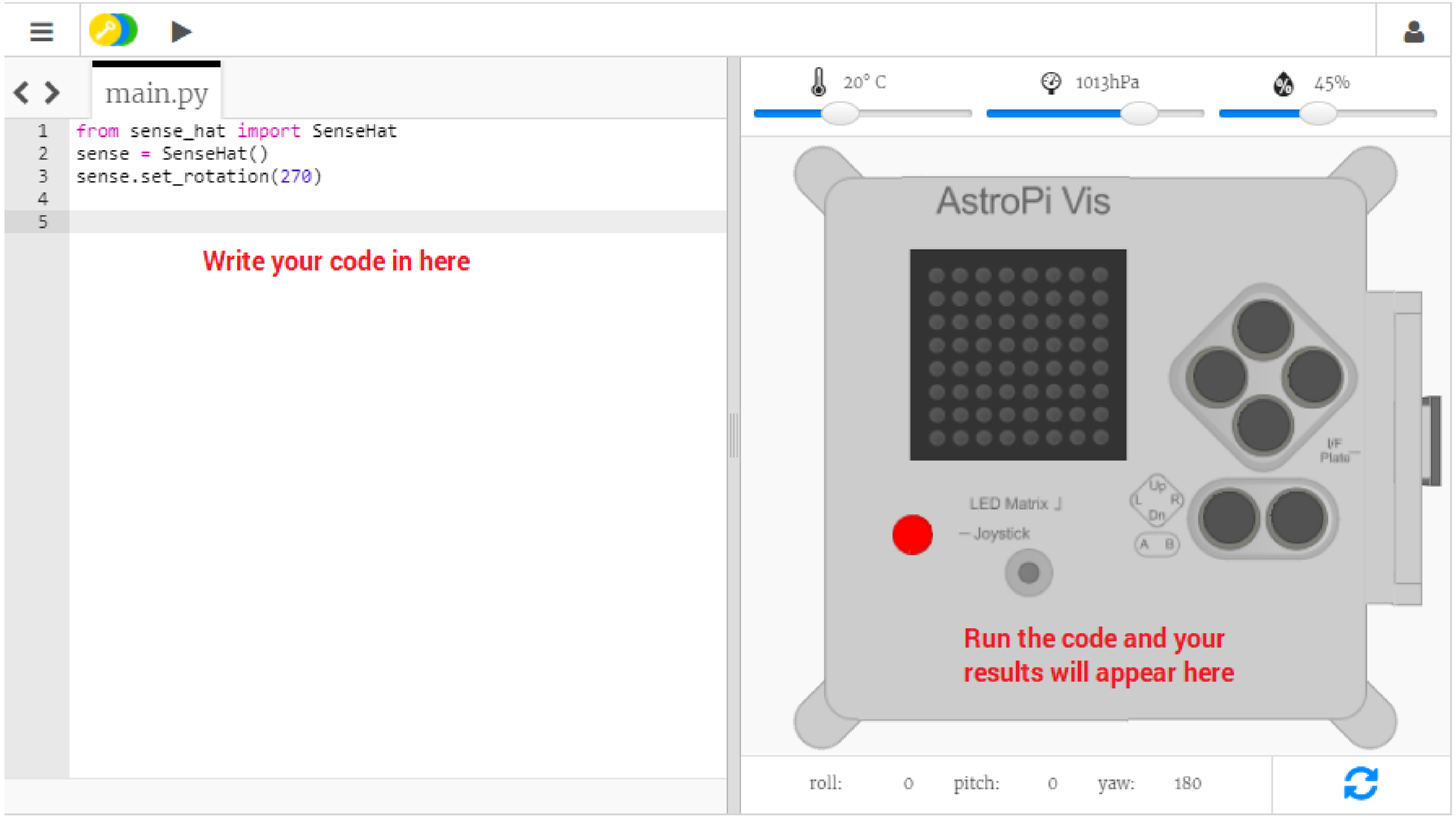
Provided the team’s program doesn’t contain any bad language or unpleasantness it’s guaranteed to run on the International Space Station for 30 seconds in May 2019.

Each team member will then receive an electronic certificate recording the exact start and end of their program’s run — their piece of space science history to keep!

Let's Get Started

We will use the online Sense HAT emulator to create our program, so no extra hardware is needed — everything is done in a web browser. Go to https://trinket.io/mission-zero

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



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

You will see that three lines of code have been added for you automatically:

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

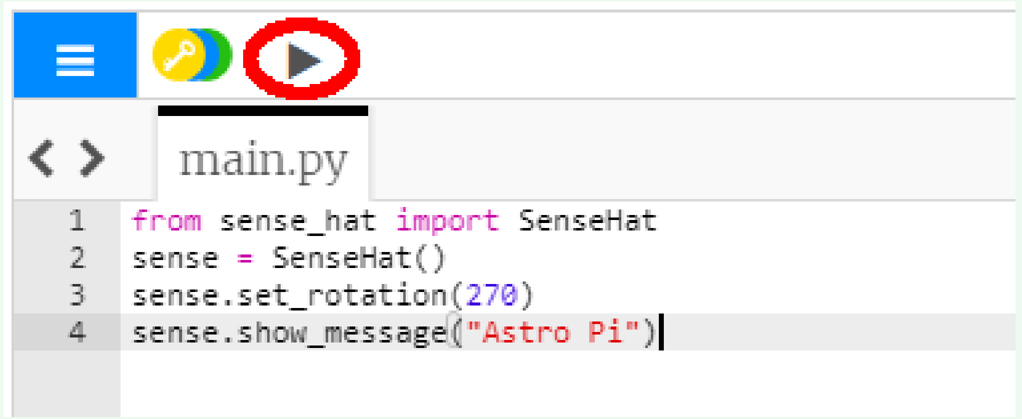
This code connects to the Astro Pi and makes sure the Astro Pi's LED display is shown the correct way around. Leave the code there, because you'll need it.

Perhaps you could leave a nice greeting for the astronauts on the ISS who are working near the Astro Pi? Let's scroll a message across the display.

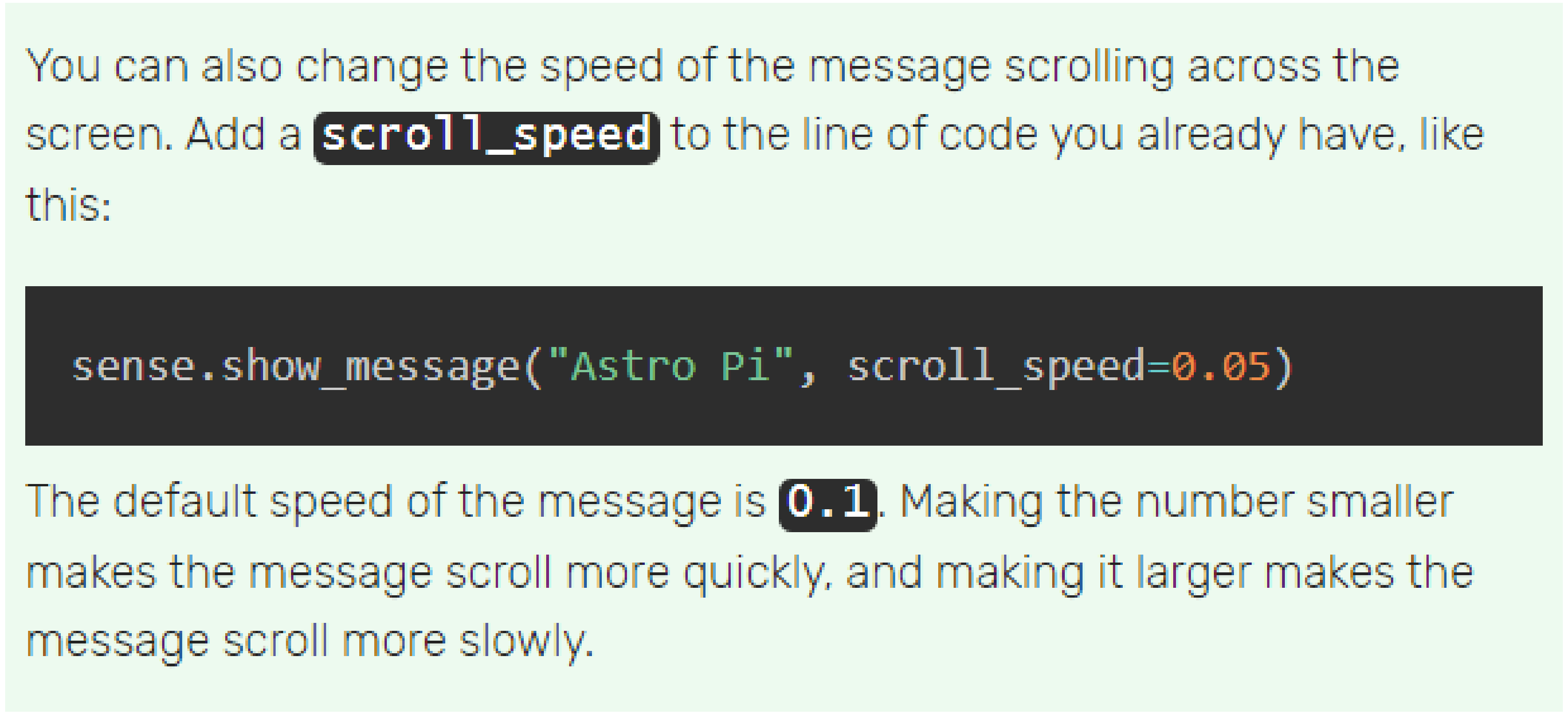
Add this line below the other code:

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

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



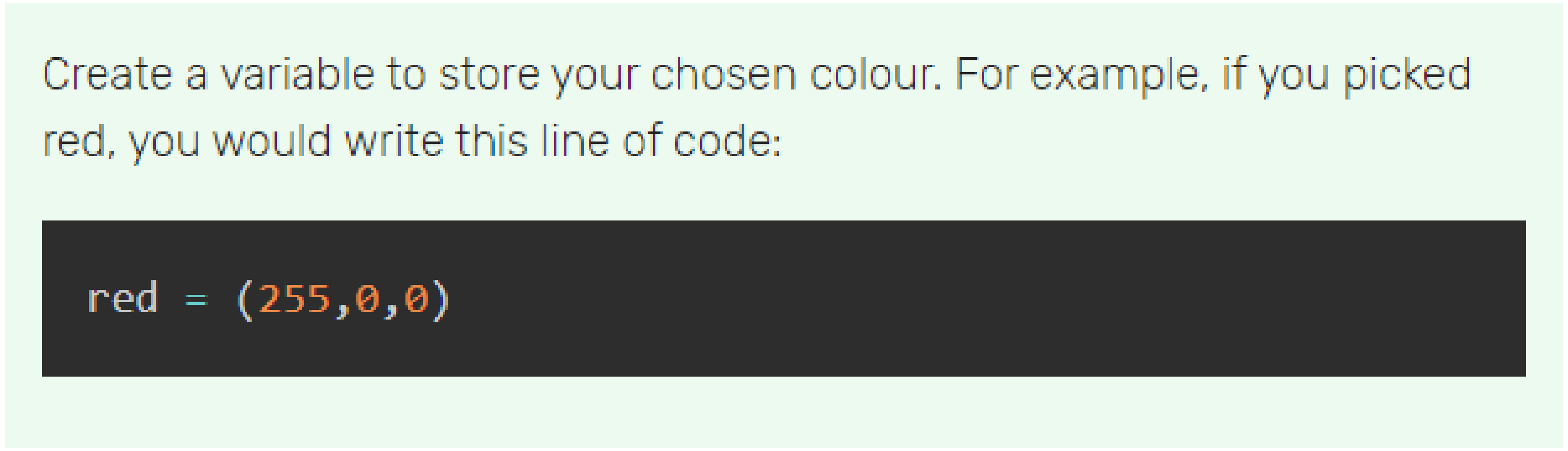
```
from sense_hat import SenseHat
sense = SenseHat()
sense.set_rotation(270)
sense.show_message("Astro Pi")
```



You can also change the speed of the message scrolling across the screen. Add a `scroll_speed` to the line of code you already have, like this:

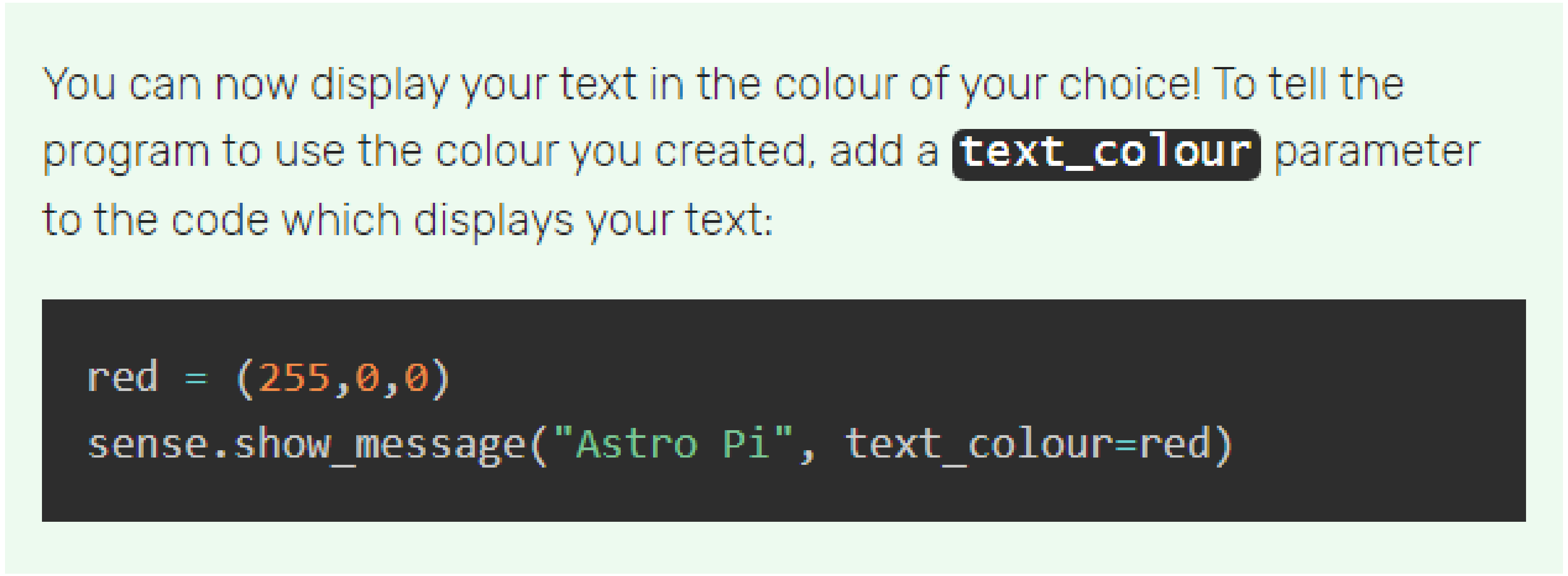
```
sense.show_message("Astro Pi", scroll_speed=0.05)
```

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)
```



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:

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

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)
```

Change the greeting text and colour — what message will you send to the Astronauts aboard the ISS?

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)
```

Note: This time, it's a good idea to give the colour variables single-letter names, because that will save time in the next step, where you are going to be typing them out many times. Moreover, using single letters will make it easier to see the picture you'll draw.

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()
```

This line will measure the current temperature, and store the measured value in the variable `temp`.

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. In the example we have rounded to one decimal place, but for a different level of precision, change the number `1` to the number of decimal places you would like to see.

```
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" )
```

That's It For Today

Next week we'll go over this in more detail and we'll start to work on our own messages and pictures