Computing

Lesson 7: Sense HAT I

Programming Part 5: Strings and Lists

Rebecca Franks

This resource contains colour images which may be printer-intensive

First steps

Task: Getting started

Step 1

Open the Sense HAT emulator using this shortlink:
oaknat.uk/comp-ks4-senseHAT-emulator

Tip: if the emulator does not appear on the right hand side then run the code.

Step 2

Now copy the following code into line 3.

This piece of code is a function that takes a message, formats it so that it can be shown on LED display and scrolls it smoothly across.

```
3 sense.show_message("Hello world!")
```

Task: Getting started

Step 3

Save and run your program. You should see the words “`Hello world!`” scroll across the LED matrix.

Important: If you are using a physical sense HAT for this activity then do not save your file as `sense_hat.py` as it won’t work because it uses the same name as the module used for the Sense HAT.

Task: Using colour

Step 1

The `show_message` function can take more than 1 argument. You can also specify the `text_colour`, `back_colour` and the `scroll_speed`.

Copy the code below to see what happens:

```
sense.show_message("Hello world!", text_colour = (0, 0, 255))
```

Task: Using colour

Step 2

The scroll text should have appeared in **blue**. It appears in **blue** due to the three values that have been used between the brackets. The first value is for **R**ed, the second is for **G**reen and the third is for **B**lue.

```
                                                # R, G, B
sense.show_message("Hello world!", text_colour = (0, 0, 255))
```

Modify the code so that the text appears in **Red** instead of **Blue**.

Task: Using colour

Step 3

For each colour you can use a value from **0 - 255**. **0** is no colour and **255** is full brightness.

Investigate the different colour options on this website and try a different colour with your scroll message:

oaknat.uk/comp-w3-colour

Task: Using colour

Step 4

Instead of entering the values each time you can hold the colour values in **variables**.

```
red = (255, 0, 0)
green = (0, 255, 0)
blue = (0, 0, 255)
```

Create a **variable** with your chosen colour and use the **variable name** inside the brackets instead of the values.

Make sure that you **create the variable** before the `show_message` code.

Task: Using colour

Step 5

To change the `back_colour` you need to add another argument.

```
back_colour = blue
```

Modify your code so that it changes the background colour of the message as well.

Explorer Task (Optional)

Investigate the purpose of this code snippet:

```
sense.set_pixel(1, 5, red)
```

See if you can use it to draw a simple picture.

Note: The pixels will remain on the LED matrix until they are cleared. If you need to clear the screen then you can use `sense.clear()`

Create your character

Introduction: Your colour palette

You have a limited amount of colours that you can use for this activity. These are shown in the **colour palette** opposite.

Colour palette	RGB Values	Colour
a	255, 0, 0	Red
b	0, 255, 0	Green
c	0, 0, 255	Blue
d	255, 255, 0	Yellow
e	183, 183, 183	Grey
f	0, 0, 0	Black
g	255, 0, 255	Pink
h	255, 153, 0	Orange
i	255, 255, 255	White

Introduction: Creating a character

To design a character, create an **8 x 8 grid** and enter the **letters** that match the **colour** that you would like to use from the **colour palette.**

You can also colour in your character if you have some felt tip pens or coloured pencils.

An example is shown opposite.

Frame 1

i	i	i	i	i	i	i	i
i	i	i	i	i	i	i	i
i	i	b	i	i	b	i	i
i	b	b	b	b	b	b	i
i	b	f	b	b	f	b	i
i	b	b	b	b	b	b	i
i	i	b	i	b	i	i	i
i	i	b	i	b	i	i	i

Introduction: Creating a character

Here is an example of 3 completed frames. Notice how each frame has a slight change which will allow the character to look like it is moving.

Frame 1

i	i	i	i	i	i	i	i
i	i	i	i	i	i	i	i
i	i	b	i	i	b	i	i
i	b	b	b	b	b	b	i
i	b	f	b	b	f	b	i
i	b	b	b	b	b	b	i
i	i	b	i	b	i	i	i
i	i	b	i	b	i	i	i

Frame 2

i	i	i	i	i	i	i	i
i	i	i	i	i	i	i	i
i	i	b	i	i	b	i	i
i	b	b	b	b	b	b	i
i	b	f	b	b	f	b	i
i	b	b	b	b	b	b	i
i	i	i	b	i	b	i	i
i	i	i	b	i	b	i	i

Frame 3

i	i	i	i	i	i	i	i
i	i	i	i	i	i	i	i
i	i	b	i	i	b	i	i
i	b	b	b	b	b	b	i
i	b	f	b	b	f	b	i
i	b	b	b	b	b	b	i
i	i	i	i	b	i	b	i
i	i	i	i	b	i	b	i

Task: Creating a character

Either **print** this page to complete your design or **copy** it out onto some lined/squared paper.

Frame 1	Frame 2	Frame 3

Task: Program your character

Worked example .

Display a smiley face on the LED matrix

```
1  from sense_hat import SenseHat
2  sense = SenseHat()
3
4  r = (255, 0, 0) # Red
5  b = (0, 0, 180) # Blue
6
7  smile = [
8    b, b, b, b, b, b, b, b,
9    b, b, b, b, b, b, b, b,
10   b, b, r, b, b, r, b, b,
11   b, b, b, b, b, b, b, b,
12   b, r, b, b, b, b, r, b,
13   b, b, r, r, r, r, b, b,
14   b, b, b, b, b, b, b, b,
15   b, b, b, b, b, b, b, b
16 ]
17
18 sense.set_pixels(smile)
```

Task: Program your character

Step 1

Using the worked example as a guide, create three lists that will hold your three frames for your animated character.

Make sure that you create the variables for each colour. Use the RGB codes to help with this.

Step 2

Run your program to see what happens. If your program works correctly then you should just see the final frame on the screen when you execute the code.

Task: Program your character

Common errors:

- Missing commas - study the worked example.
- A comma at the end of the last line - study the worked example, there should not be a comma on the last line
- Variables not declared
- Missing square bracket at the end - study the worked example
- Missing instruction to set the pixels - there should be a line of code (like line 18) for each frame.

Task: Program your character

Step 3

At the moment your program is too quick so it **appears** that **only frame 3** is ever executed. Use `sleep()` to slow the program down.

Remember the `import` statement for `time` and `sleep`

Step 4

Use a `while True:` loop to make your animated character loop forever.

Explorer Task (Optional)

This code snippet makes use of the temperature sensor.

Incorporate this into your `while True:` loop so that the character is slow when it is hot and fast when it is cool.

```
while True:
    temperature = sense.get_temperature()

    if temperature > 20:
        speed = 2
    else:
        speed = 0.5
```

Tip: If you are using a physical Sense HAT then you will need to know the temperature of the room in order to test your code. Print the temperature to find out what it is.