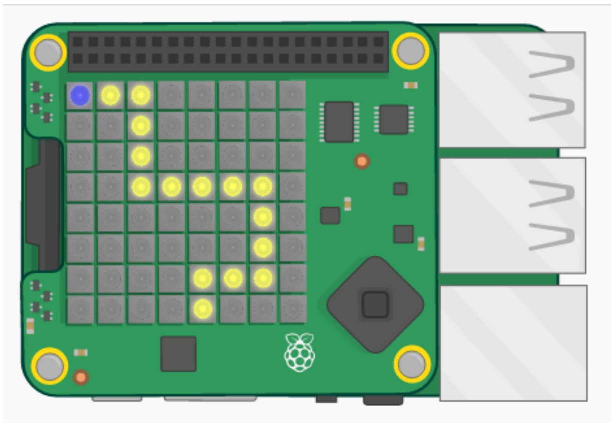
Raspberry Pi Projects

Tightrope

Introduction:

In this project you will create a game in which you have to tilt your Sense HAT to guide a character along a path. If you fall off the path, you have to start again from the beginning!



Additional information for club leaders

If you need to print this project, please use the Printer friendly version (https://projects.raspberrypi.org/en/projects/tightrope/print).

Club leader notes

Introduction:

In this project, children will learn about the Sense HAT orientation sensor by creating a line-follownig game. The player tilts the Sense HAT to move the character along a path. Deviating from the path sends the player back to the start!

Online Resources

This project uses Python 3. We recommend using Trinket (https://trinket.io/) to write Python online. This project contains the following Trinkets:

- 'Tightrope' Starter Trinket – jumpto.cc/tightrope-go (http://jumpto.cc/tightrope-go)

There is also a trinket containing the completed project:

- 'Tightrope' Finished – trinket.io/python/790adaa749 (https://trinket.io/python/790adaa749)

Offline Resources

This project can also be completed offline (https://www.codeclubprojects.org/en-GB/https://projects-static.raspberrypi.org/projects/tightrope/20c6b2928aff3335c809a3dcbb874998ec2f7c68/en/resou sense-hat/) on a Raspberry Pi computer with a Sense HAT. You can access the project resources by clicking the 'Project Materials' link for this project. This link contains a 'Project Resources' section, which includes resources that children will need to complete this project offline. Make sure that each child has access to a copy of these resources. This section includes the following files:

- tightrope/main.py
- tightrope/snippets.py

You can also find a completed version of this project in the 'Volunteer Resources' section, which contains:

- tightrope-finished/main.py
- tightrope-finished/snippets.py

(All of the resources above are also downloadable as project and volunteer `.zip` files.)

Learning Objectives

- Sense HAT orientation (roll, pitch and yaw);
- Sense HAT display;
- RGB colours;

This project covers elements from the following strands of the Raspberry Pi Digital Making Curriculum (http://rpf.io/curriculum):

- Combine programming constructs to solve a problem. (https://www.raspberrypi.org/curriculum/programming/builder)

Challenges

- "Create your own path" - creating a image using a list of pixels;
- "Moving up!" - moving the character up in response to changing `roll` values.
- "Changing the difficulty" - testing the finished game and making changes based on player feedback.

Project materials

Project resources

- .zip file containing all project resources (https://projects-static.raspberrypi.org/projects/tightrope/20c6b2928aff3335c809a3dcbb874998ec2f7c68/en/re project-resources.zip)
- Tightrope starter project (http://jumpto.cc/tightrope-go)
- Offline starter Python file (https://projects-static.raspberrypi.org/projects/tightrope/20c6b2928aff3335c809a3dcbb874998ec2f7c68/en/re main.py)
- Offline Python file containing useful code (https://projects-static.raspberrypi.org/projects/tightrope/20c6b2928aff3335c809a3dcbb874998ec2f7c68/en/re snippets.py)

Club leader resources

- .zip file containing all completed project resources (https://projects-static.raspberrypi.org/projects/tightrope/20c6b2928aff3335c809a3dcbb874998ec2f7c68/en/re volunteer-resources.zip)
- Online completed Trinket Tightrope project (https://trinket.io/python/790adaa749)
- tightrope-finished/main.py (https://projects-static.raspberrypi.org/projects/tightrope/20c6b2928aff3335c809a3dcbb874998ec2f7c68/en/re finished-main.py)
- tightrope-finished/snippets.py (https://projects-static.raspberrypi.org/projects/tightrope/20c6b2928aff3335c809a3dcbb874998ec2f7c68/en/re finished-snippets.py)

Draw a path

First let's draw the path that your character must follow.

- Open the Tightrope Starter Trinket: jumpto.cc/tightrope-go (http://jumpto.cc/tightrope-go).

  The code to set up the Sense HAT has been included for you.

- Let's start by creating variables to store the colours you want to use. Remember that to set the colour of an individual LED, you need to say how much red, green and blue it should have.

  To create yellow, you'll need maximum red and green, and no blue:

```
#!/bin/python3

from sense_hat import SenseHat
from time import sleep

sense = SenseHat()

y = [255, 255, 0]
```

(If you prefer, you can go to jumpto.cc/colours (http://jumpto.cc/colours) and choose any colour you like!

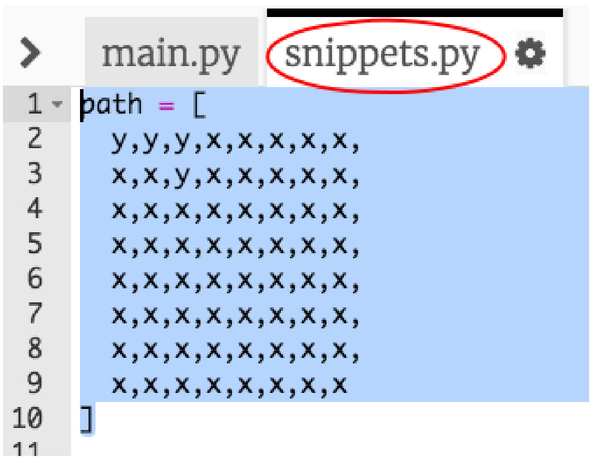
- You'll also need black pixels (or any colour you like) around the path.

```
from time import sleep

sense = SenseHat()

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

- To draw your path, you first need to create a list containing the colour of each pixel.

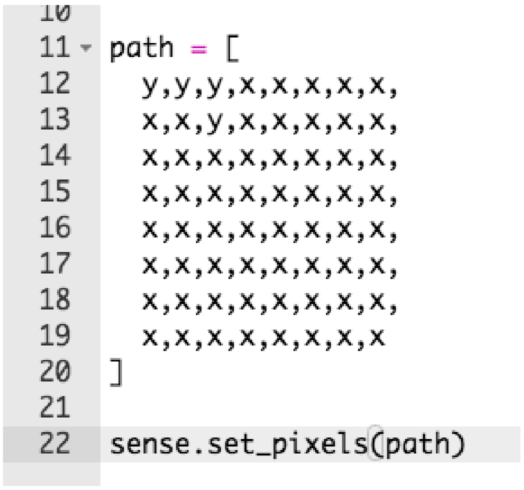
```
y = [255, 255, 0]
x = [0, 0, 0]

path = [
  y,y,y,x,x,x,x,x,
  x,x,y,x,x,x,x,x,
  x,x,x,x,x,x,x,x,
  x,x,x,x,x,x,x,x,
  x,x,x,x,x,x,x,x,
  x,x,x,x,x,x,x,x,
  x,x,x,x,x,x,x,x,
  x,x,x,x,x,x,x,x
]
```

To save typing, you can copy the code from `snippets.py` in your project.



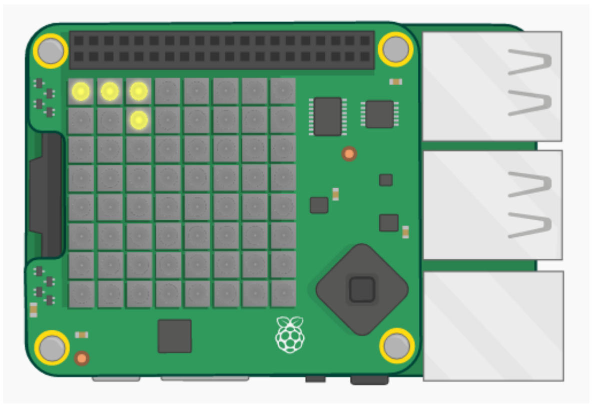
```
path = [
  y,y,y,x,x,x,x,x,
  x,x,y,x,x,x,x,x,
  x,x,x,x,x,x,x,x,
  x,x,x,x,x,x,x,x,
  x,x,x,x,x,x,x,x,
  x,x,x,x,x,x,x,x,
  x,x,x,x,x,x,x,x,
  x,x,x,x,x,x,x,x
]
```

- Next, you need to call `set_pixels` to display your path image on the Sense HAT.



```

path = [
  y,y,y,x,x,x,x,x,
  x,x,y,x,x,x,x,x,
  x,x,x,x,x,x,x,x,
  x,x,x,x,x,x,x,x,
  x,x,x,x,x,x,x,x,
  x,x,x,x,x,x,x,x,
  x,x,x,x,x,x,x,x,
  x,x,x,x,x,x,x,x
]

sense.set_pixels(path)
```

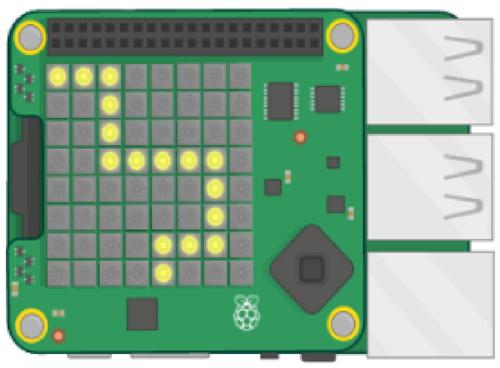
- Click 'Run' to test your code. You should see a yellow pixel in the places that you've used your **y** variable, and no colour in the places that you've used **x**.



Challenge: Create your own path

Can you edit your `path` variable to create a path to follow?

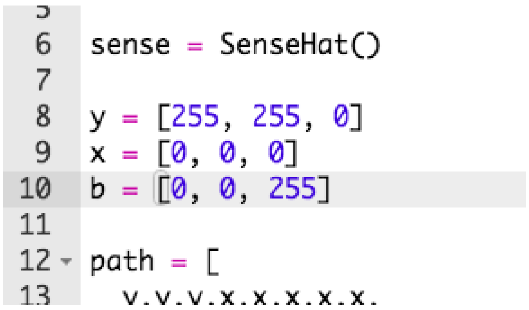
Make sure that your path starts at the top-left of the display.



Draw your player

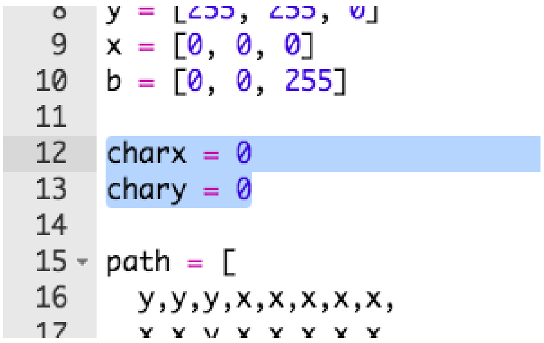
Let's add the character to your game.

- First, create another colour variable for your character. Here's how to create blue:



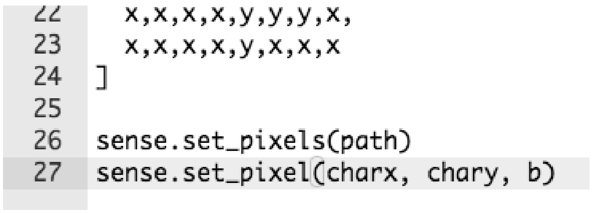
```
sense = SenseHat()

y = [255, 255, 0]
x = [0, 0, 0]
b = [0, 0, 255]

path = [
  y,y,y,x,x,x,x,x,
```

- Next you need to create variables to store your character’s x and y position. To start with, we’ll set these both to `0`, which is the top-left of the Sense HAT.



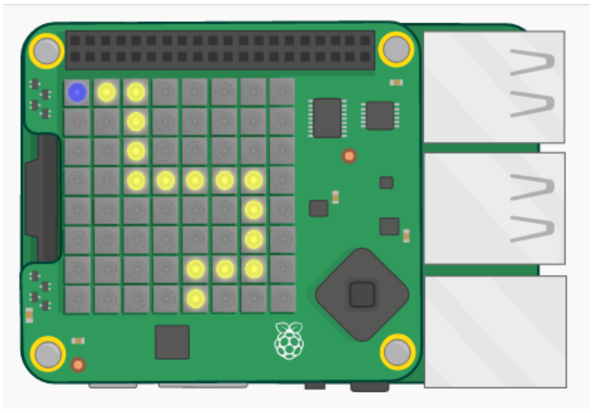
```
y = [255, 255, 0]
x = [0, 0, 0]
b = [0, 0, 255]

charx = 0
chary = 0

path = [
  y,y,y,x,x,x,x,x,
  x,x,y,x,x,x,x,x
```

- To display your character, use `set_pixel`. You need to tell `set_pixel` the x and y position of the pixel to set, as well as the colour.



```
  x,x,x,x,y,y,y,x,
  x,x,x,x,y,x,x,x
]

sense.set_pixels(path)
sense.set_pixel(charx, chary, b)
```

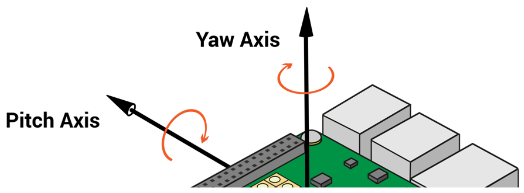
- Test your code, and you should now see your character in the top-left of the screen.



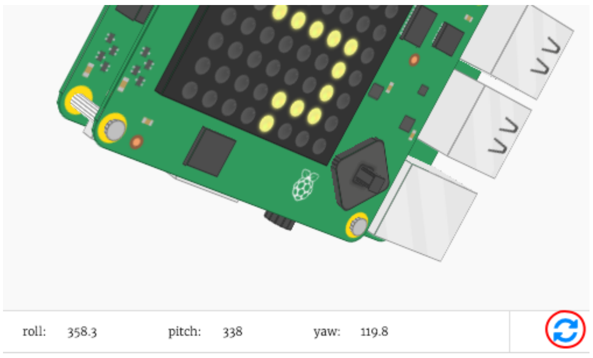
Roll, pitch and yaw

You’ll be tilting the Sense HAT to move your character. Let’s start by finding out the **orientation** (the position) of your Sense HAT.

- The Sense HAT can detect its **roll**, **pitch** and **yaw**.



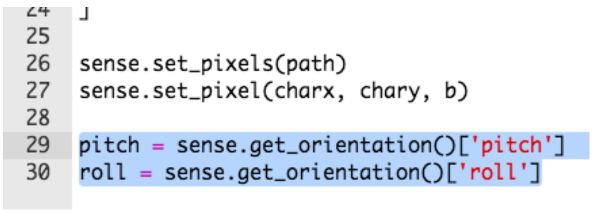
- Try dragging the Sense HAT to change its roll, pitch and yaw values to see how it moves.



roll: 358.3 pitch: 338 yaw: 119.8

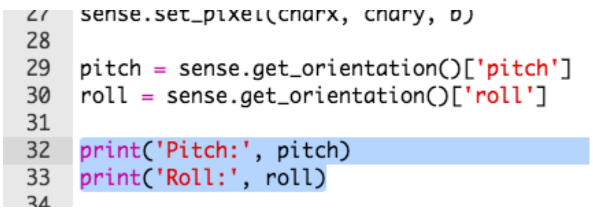
Press the reset button to put the Sense HAT back to the starting position when you've finished testing.

- We only need the pitch and the roll for this project, so add 2 lines of code to get these values from the Sense HAT.



```
]

sense.set_pixels(path)
sense.set_pixel(charx, chary, b)

pitch = sense.get_orientation()['pitch']
roll = sense.get_orientation()['roll']
```

- Print the pitch and roll to test them out.



```
sense.set_pixel(charx, chary, b)

pitch = sense.get_orientation()['pitch']
roll = sense.get_orientation()['roll']

print('Pitch:', pitch)
print('Roll:', roll)
```

- Run your code to test it, and change the pitch of the Sense HAT to tilt it to the right. You'll notice that the printed `pitch` value doesn't change!

- The problem is that you are only getting and printing the `pitch` and `roll` **once**.

  To do this repeatedly, you'll first need to indent all of your code for setting the pixels, as well as getting and printing the `pitch` and `roll` values.

- You can then add `while True:` above the indented code to run it forever.

- Test your code again, and this time you should see the printed `pitch` value change.

Moving the character

Now let's move your character when the Sense HAT is tilted.

- Let's move your character to the right if the Sense HAT's `pitch` is between `270` and `315` degrees.
- Add this code to change the character's x position if the pitch is between `270` and `315`:

  Make sure that this code is indented, so that the character moves repeatedly if the Sense HAT is tilted.
- Tilt your Sense HAT so that the `pitch` is between `270` and `315` degrees. You should see that your character moves to the right, but keeps going off the display!

  You will also see an error, because the character's x position goes above 7, which is not a valid position on the display.
- To fix this, you only want to move your character to the right if its current position is less than 7.
- Test your improved code, and you should now see that your character moves **until it gets to the right side of the display**.
- We also want to move your character to the left when the `pitch` is between `45` and `90` degrees.
- Add this code to move your character to the left if the `pitch` is between `45` and `90`, but **only if the character isn't already at the far left of the display**.
- Test your code to make sure that you can tilt your character back to the left.
- Next, let's add code to change your character's **y-position**, moving it down when the **`roll`** value is between `45` and `90`.
- test this code to see if you can tilt the Sense HAT to move your character down.
- If you want to slow your game down, you can add short `sleep` at the end of your `while True:` loop.

Challenge: Moving up!

Can you add code to move your character up (by changing your `chary` variable by `-1`) when the `roll` is between `270` and `315`?

The code you'll need to add will be **very** similar to the code for moving your character to the left.

Going back to the start

Let's put your character back to the start when it falls off the path.

- You may have noticed that nothing happens when the character falls off the path.

- To fix this, we're going to send the character back to the start if they're standing on a black pixel.

  Let's start by getting the colour of the pixel the character has moved to.

- If the colour of the current pixel is black, then send the character back to the start.

- Test your code and you should see your character move back to the start if they fall off the path.

Challenge: Changing the difficulty

Ask some friends to test your game. Did they find it too easy or too hard? If so, you could make some changes:

- Change the path the character has to follow;
- Change the `pitch` and `roll` values that move your character;
- Add a short or longer `sleep` at the end of your loop.