Graphics on the Sense HAT

Go4Code

Project: Glitter Lights

Difficulty: Easy

In this project we will be making use of the Raspberry Pi's display panel to make some glittery lights. Ask your supervisor to show you what they should look like!

Introducing the project

The first thing you should do is open the *skeleton code* for the project.
In programming, skeleton code means code that only has the basic elements of a program. It is up to you to fill in the rest!

You can find the skeleton code on Trinket, here:
Glitter Project (https://trinket.io/python/77b6a432af)

If you can, you should also create an account and log in to Trinket. This will allow you to save the Trinket projects. Otherwise you have to copy the code on to your computer to save it.

On Trinket, you'll be able to test your code on a *virtual* Sense HAT, before you try your code on the real thing.

As you might see, the skeleton code is split up into sections, divided by the headlines.
For example:

```
#### 2. Main Loop
```

This guide will go through the various sections (not necessarily in order), and help you write your code. **Very important note:** *You should add the code in the specified section in your skeleton code as you follow this guide.*

The next part of this guide will explain the stuff that's already in the skeleton code when you first open it.

Explanation of the skeleton code

Before we get on to the coding, it's worth looking over the *skeleton code* and make sure you are familiar with it.

The first few lines in the script are:

```
#### 1.1 Import libraries
```

```
#### 1.1 Import Libraries

import sys
sys.path.insert(1,'/home/pi/Go4Code/g4cSense/skeleton')

from sense_hat import SenseHat
from senselib import *
```

Without going into detail, these lines are called *import statements*. They are used to *import* code from other Python files into your own file. This is useful because you can use other people's code to simplify your own.

The next part of the code (Sec. 1.2) creates some important *Objects* (don't worry if you're not sure what that means) that we'll use in the later on.

Section 2 is where you'll actually be coding.

Setting Pixels on the Sense HAT (Recap)

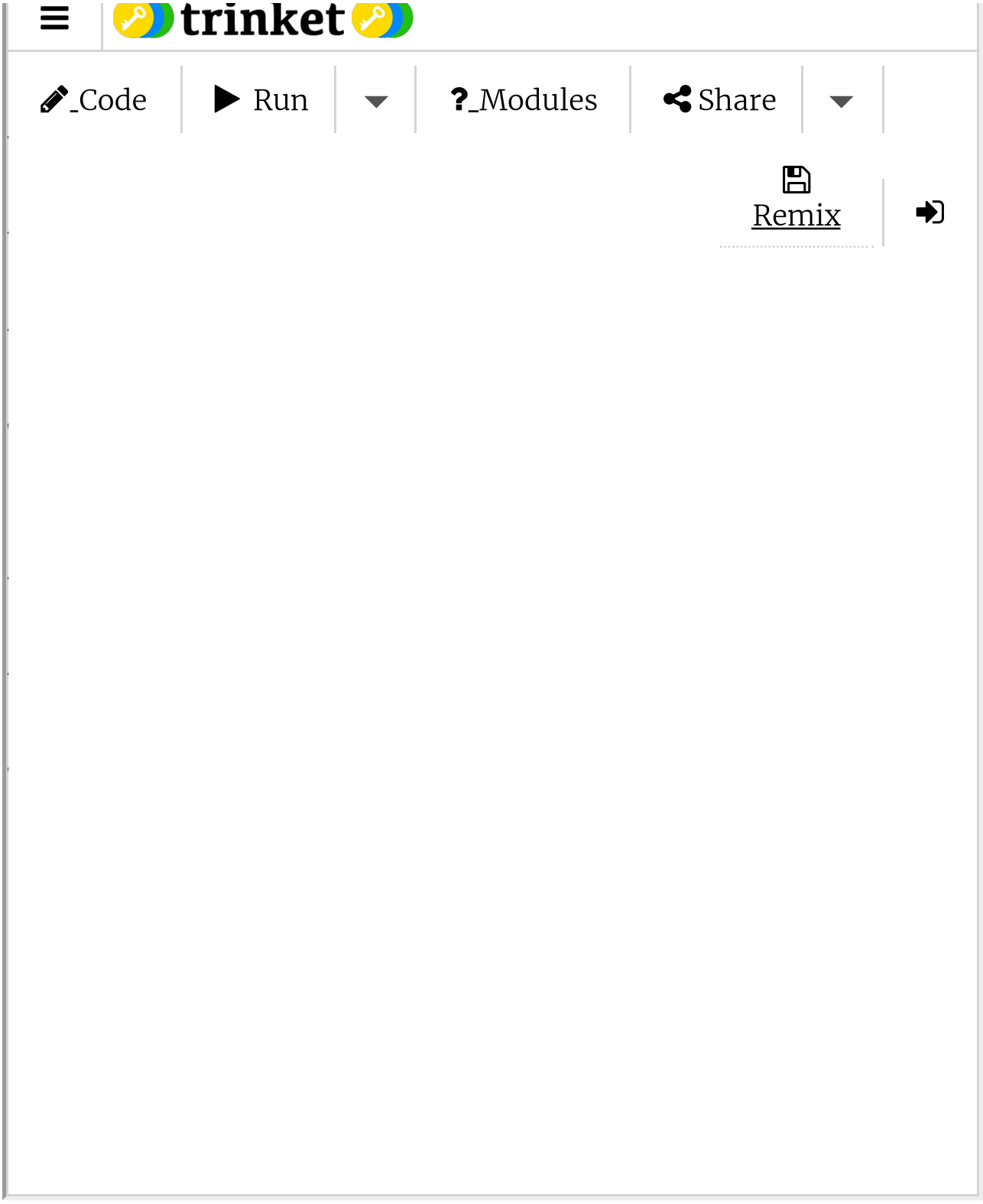
This section is meant to remind you how to set all the pixels on the Sense HAT using a list. If you already know how to do this from a previous project, you may skip this section.

The Sense HAT allows us to use the *set_pixels* function to set all the pixels to certain colours as defined by in a *list*. This is better understood with an example. The code below defines a *list* with 64 values. Each value is a colour. The *list* tells the Raspberry Pi what colour each pixel will be set to.

```
r = [255,0,0]

image = [
  r,r,r,r,r,r,r,r, #This is the first row of pixels
  r,r,r,r,r,r,r,r, #The second
  r,r,r,r,r,r,r,r, #The third
  r,r,r,r,r,r,r,r, #And so on..
  r,r,r,r,r,r,r,r,
  r,r,r,r,r,r,r,r,
  r,r,r,r,r,r,r,r,
  r,r,r,r,r,r,r,r,
]
```

Let's try the code above in the Trinket. ***Read the comments for explanations!***



Trinket Emulator

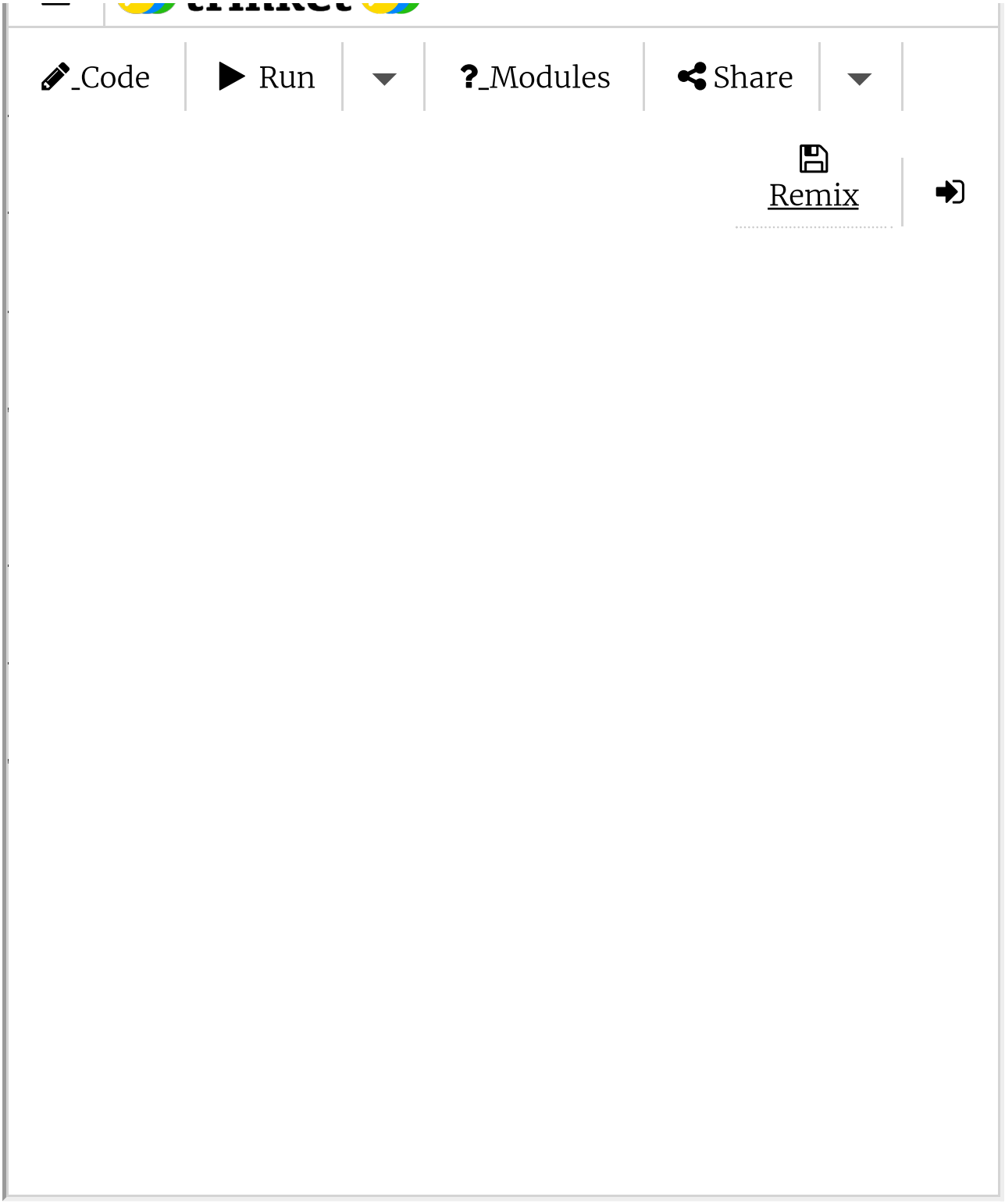
Making a picture

The *set_pixels* function is actually really useful as it let's us make images really easily. The *list* essentially acts as a drawing board that resembles the LED matrix on the Sense HAT. For example if I take the 3rd element along in the first row and set that to yellow in the *list* it will set the 3rd pixel along in the first row to orange.

Remember: In programming, we count from zero not one!

```
r = [255,0,0]
y = [255,255,0]
image = [
  r,r,r,y,r,r,r,r, #Setting the third pixel along to yellow
  r,r,r,r,r,r,r,r, #The second
  r,r,r,r,r,r,r,r, #The third
  r,r,r,r,r,r,r,r, #And so on..
  r,r,r,r,r,r,r,r,
  r,r,r,r,r,r,r,r,
  r,r,r,r,r,r,r,r,
  r,r,r,r,r,r,r,r,
]
```

Let's try it out in the Trinket below:



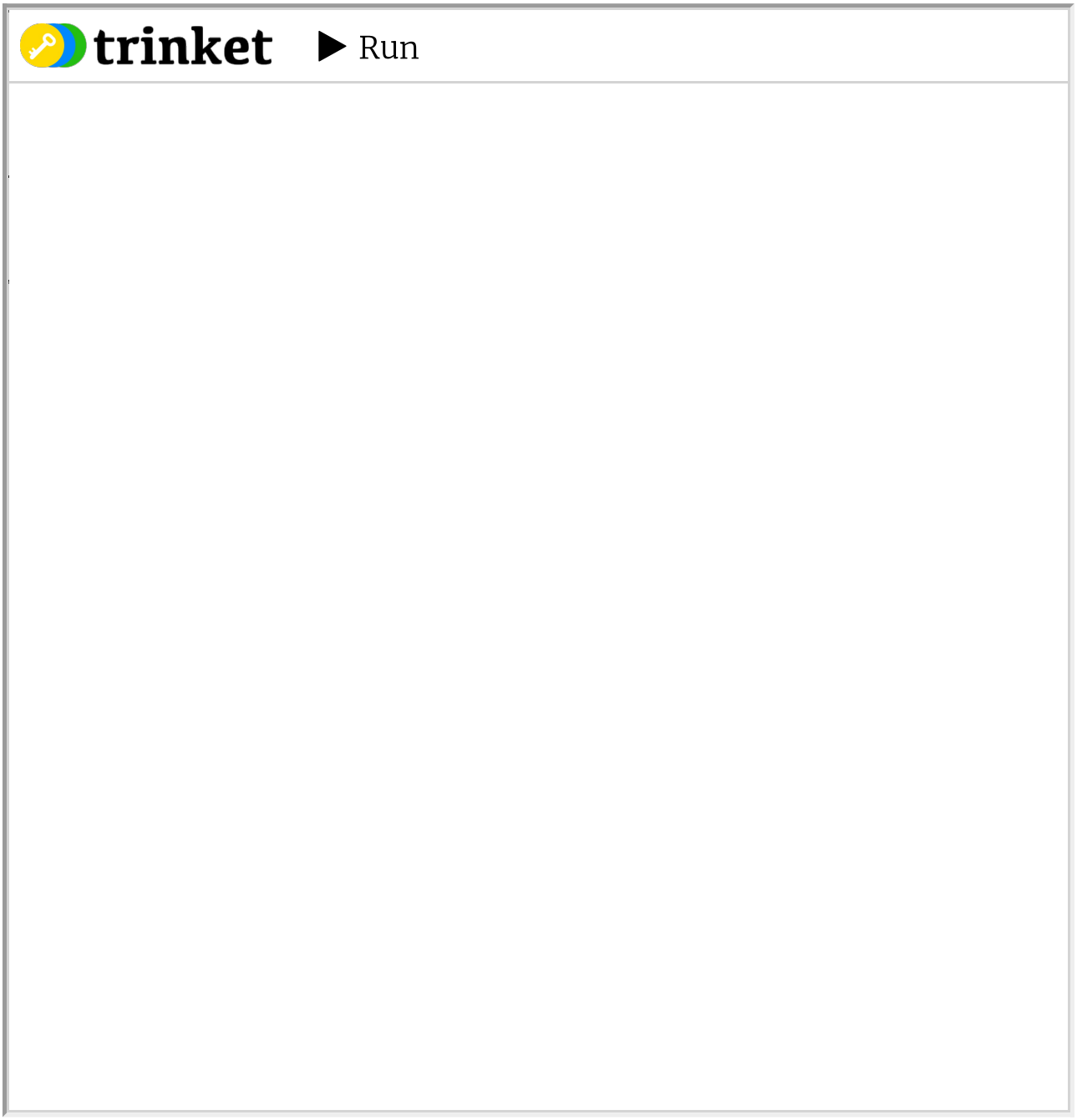
Trinket Emulator

Great! So now that we can see how to use *lists* and the *set_pixels* function to set all the pixels at the same time. We can make some images!

If you are unsure about how the code above works ask your supervisor to explain it to you

Making a plus sign

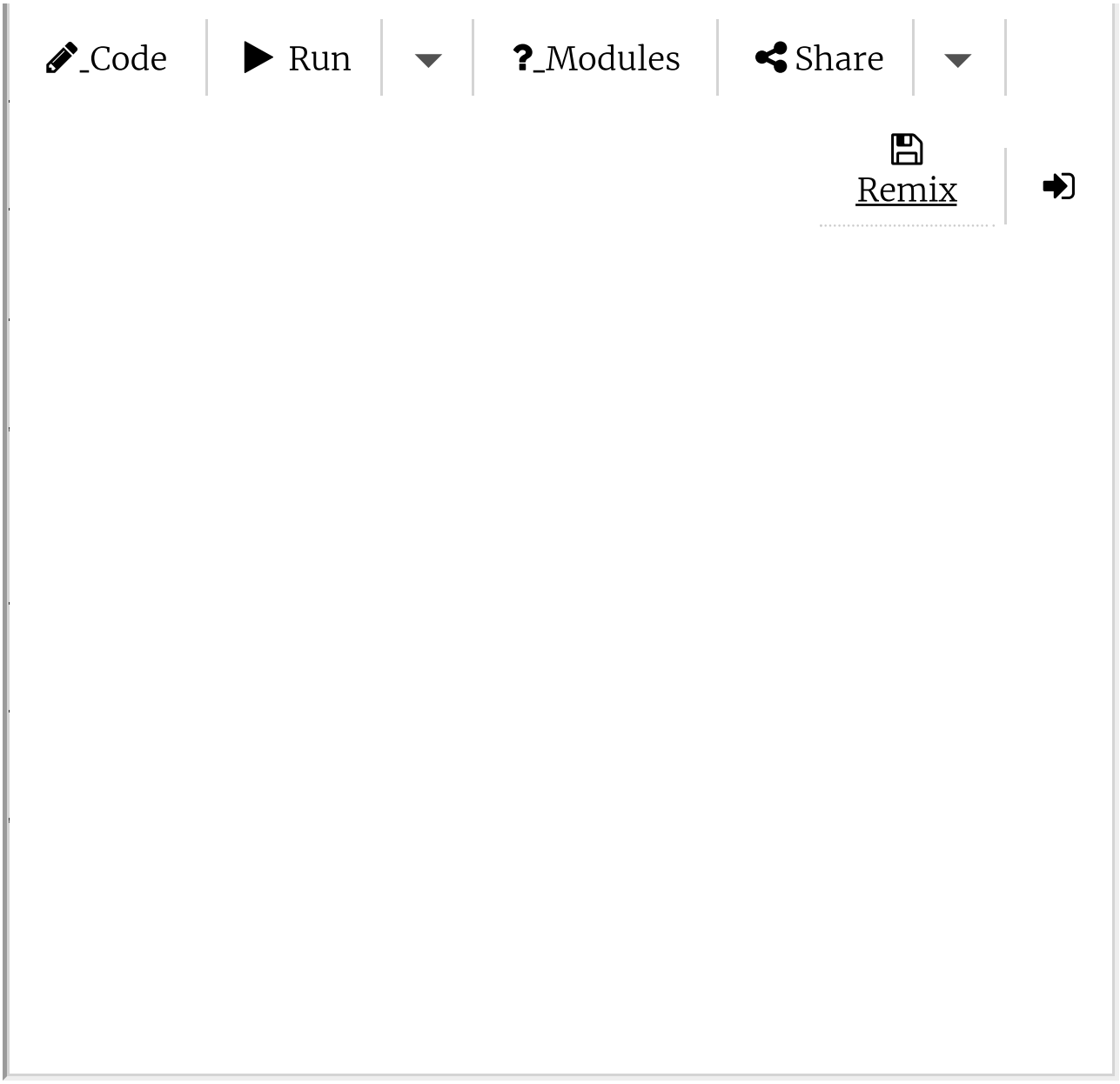
Let's try making a simple image first. We're going to make a plus sign as shown in the Trinket below:



Trinket Emulator

To make that plus sign, we used the *set_pixels* function and a *list* called image as a drawing board to make the drawing.

Exercise 1: Change the values of the *list* in the Trinket below to make a plus sign:



Trinket Emulator

The Raspberry Pi Logo

We can add any colour to our *list*. We can also add no colour to a pixel, this would mean that pixel would be off!

By setting one of the elements to *e = [0,0,0]* we can turn the corresponding pixel off. Using this and what you have just learned, we made a *list* that contains the image for the Raspberry Pi Logo.

```
r = [255,0,0] ## Defining the colour red
g =[0,255,0] ## Defining the colour green
```

```
e = [0,0,0] ## This variable set's the pixel to off

logo = [
    e, g, g, e, e, g, g, e,
    e, e, g, g, g, g, e, e,
    e, e, r, r, r, r, e, e,
    e, r, r, r, r, r, r, e,
    r, r, r, r, r, r, r, r,
    r, r, r, r, r, r, r, r,
    o, r, r, r, r, r, r, o,
    o, o, r, r, r, r, o, o,
    ]
```

Exercise 2: Using the *list* given, try changing your code to set the pixels to display the Raspberry Pi Logo in the Trinket emulator.

Back to Glitter

Creating glitter on the screen is actually easily done by creating a *list* with random shades of a certain colour. To get a random shade of a certain colour we use the *randomShade()* function. For example the code below will set all the pixels to some random shade of green using a *list* and the *set_pixels* function.

```
g = randomShade('green')

image = [
  g,g,g,g,g,g,g,g, #Setting the third pixel along to yellow
  g,g,g,g,g,g,g,g, #The second
  g,g,g,g,g,g,g,g, #The third
  g,g,g,g,g,g,g,g, #And so on..
  g,g,g,g,g,g,g,g,
  g,g,g,g,g,g,g,g,
  g,g,g,g,g,g,g,g,
  g,g,g,g,g,g,g,g,
]

sense.set_pixels(image)
```

Try it in the Trinket Emulator below. Each time you run the code the shade of green will change.

Filling lists using Loops

Typing out *lists* can get quite annoying and it would be easier if we can automate this. We can add elements to a *list* using the *.append* function on a *list*. Let's demonstrate what this means.

First we create an empty *list*.

```
my_list = []
```

If we were to *print* the *list* defined above, we would get an empty *list* because we haven't actually put anything in it. The code bellow will add the colour variable *g* and the variable *r* to the list.

```
g = [0,255,0]
r = [255,0,0]
my_list.append(g)
my_list.append(r)
```

Now if we were to *print* the *list* defined above we would get the following output:

```
[[0,255,0],[255,0,0]]
```

The *list* given above has 2 elements. Both of the elements are also *lists* but that is not important here, as we can make a list of any type of object in python. To create a image for the Sense HAT's screen we need a *list* with 64 elements (all the elements will be also be lists that hold the RGB values to set the colour).

This means we need to *append* to an empty *list* 64 times. But what can we use to repeat code?

Loops!

The *for-loop* below will run 64 times and add the colour green (variable *g*) to a *list*.

```
image = []
g = [0,255,0]

for i in range(64):
    image.append(g) ## Adding the variable g to the list
```

If we were to use the *set_pixels* function with the *image* list after the *for-loop*, we would set all the pixels to green.

Writing Code

If you understood the concepts explained above, you are ready to make glitter! **If you are unsure about anything, ask your supervisor to explain it to you.**

*** Sec 2.1 Picking a colour**

Define a variable with the colour of your glitter. This variable will be used in the *randomShade* function so it must be a *string*. The possible colours are given in the list *colours*.

*** Sec 2.2**

Create an empty list that you will add the random shade of your chosen colour to.

*** Sec 2.3**

Get a random shade of the colour you chose using the *randomShade* function.

*** Sec 2.4**

Add the random shade to the list.

*** Sec 2.5**

Finally set the pixels on the Sense HAT to the list

*** Sec 2.6**

There is nothing to add here for now. The *time.sleep(0.1)* function makes the Raspberry Pi wait for 0.1 seconds before changing the pixels to a new list. Every 0.1 seconds the pixels are set to a random shade of the chosen colour, this is what makes the glittery effect.

Finished!

If it's all done, correctly, your screen will show a glittery pattern! Don't worry if it doesn't, things often go wrong in programming. Errors in code are usually called *bugs*. If you have a bug in your code, you'll have to *debug* it!

Transfer it to your Raspberry Pi to try it out!

If it works, congratulations! You can either move on to another project, add the suggested **bonus features** or try to come up with new things to add to the current project. Use your creativity! You can discuss any ideas you have with a supervisor.

Bonus Features

- Change the Glitter Colour by Shaking

Note: The Trinket Emulator doesn't let you shake the Raspberry Pi, ask your supervisor to help you test your code on your Raspberry Pi

The Sense HAT comes with an accelerometer. This means that you can detect if the Raspberry Pi has been moved! The code below will measure acceleration of a *shake* and if it is above a certain value, all the pixels will be set to the Raspberry Pi Logo

```
r = [255,0,0] ## Defining the colour red
g =[0,255,0] ## Defining the colour green
e = [0,0,0] ## This variable set's the pixel to off

logo = [
```

```
    e, g, g, e, e, g, g, e,
    e, e, g, g, g, g, e, e,
    e, e, r, r, r, r, e, e,
    e, r, r, r, r, r, r, e,
    r, r, r, r, r, r, r, r,
    r, r, r, r, r, r, r, r,
    o, r, r, r, r, r, r, o,
    o, o, r, r, r, r, o, o,
    ]

ac = sense.get_accelerometer_raw()
  x = ac["x"]
  y = ac["y"]
  z = ac["z"]
  shake = x*x + y*y + z*z

  if shake > 5.0:
    sense.set_pixels(logo)
```

Think about how you can use the code above to randomly change the colour of the glitter when the Raspberry Pi is shaken.

You will find the following function useful:

To randomly select something from a list, you can use the *random.choice* function. The code below will randomly choose a number from the list *numbers* and *print* it to the console.

```
numbers = [1,2,3,4,5,6]

print(random.choice(numbers))
```

Author: Ishan Khurana

Date: August 15, 2017