Project: Board for Snakes and Ladders

Difficulty level: Medium/Hard

Description

In this project you will create the ***board*** for the classic game snakes and ladders.

You can try out the complete version of the project here:

https://goo.gl/TeQW7T (https://goo.gl/TeQW7T)

Use the up, left, right and left arrows to move the cursor. Press the middle button (ENTER) to place your piece.

Project Manual

This project guide will tell you step-by-step the main things you have to do in order to create a *snakes and ladders board*. For some of the steps, you'll have to use your own creativity to proceed, good luck!

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:
https://goo.gl/rXr3vQ (https://goo.gl/rXr3vQ)

If you can, you should also create and 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. Code section
```

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

import sys
sys.path.insert(1,'/home/pi/Go4Code/g4cSense/graphics')
sys.path.insert(1,'/home/pi/Go4Code/g4cSense/skeleton')
from senseGraphics import *
from snakes_ladders_lib import *
from sense_hat import SenseHat
```

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 later on.

Sec. 1.3 is where the initial variables of the project are set up.

Sec. 2 is where you will set up the board by specifying the board colour, adding snakes, ladders, a player and a cursor.

Sec. 3 is where we'll write code that controls the *cursor*, that places piece on the screen.

Sec. 4 is where we'll write code that changes the checks if the piece is on a snake or ladder.

Understanding the Problem

First let's break down the problem by focussing on all the features that the board should have.

Think about these and list them out below as this will help you plan out your code.

If you are unsure ask a supervisor.

If you have understood the problem and know what the board should do, let's program it!

Writing the code

(Sec. 1.3) Initialise the game

This section is already complete so you don't have to code anything here. The line of code below initialises the Sense Hat and the snakes and ladders game.

```
sense = SenseHat() #initialises the SenseHat
game = snakesLadders(sense) #Initialises the game
```

(Sec. 2.1) Setting the board colour

The colour in which the numbers will be displayed needs to be defined here in *RGB* format.As you might remember, in programming a color is usually represented as *three* numbers *[r, g, b]*. The numbers in the square brackets specify how much red (*r*), green (*g*) and blue (*b*) .By mixing different amounts of red, green and blue light, we can make almost any colour. The amount of red, green or blue you can put in to make a colour goes from 0 to 255. For example, orange is made by mixing 255 red, 127 green and 80 blue. In code this would look like this:

```
#This will set the colour variable to a combination of RGB that gives orange
colour = [255,127,80]
```

Set the variable *board_colour* to the colour you want the board to be. The next line will set the colour of the board to whatever you pick.

Hint: The snakes in the game are set to a red colour and the ladders are a blueish colour. To make sure you can tell all the things on the board clearly, pick a colour that is different from red or blue. Yellow seems to work well. Look up the RGB value for yellow and experiment to find whatever works for you. If you are unsure, ask a supervisor.

(Sec. 2.2) Adding Snakes

In this section we add snakes to the board using the *addSnake()* function on the *game* object. To add a snake onto the board, we must specify the position of the snakes head and the position of the snakes tail. But how do we specify position on the Sense Hat?

Specifying Position (Reminder)

If you know how position and coordinates work on the SenseHat, you might want to skip this section

In programming, we specify positions using coordinates. That might sound complicated but it is actually really easy and useful. The image below illustrates coordinates on the SenseHat.

coordinates on the SenseHat

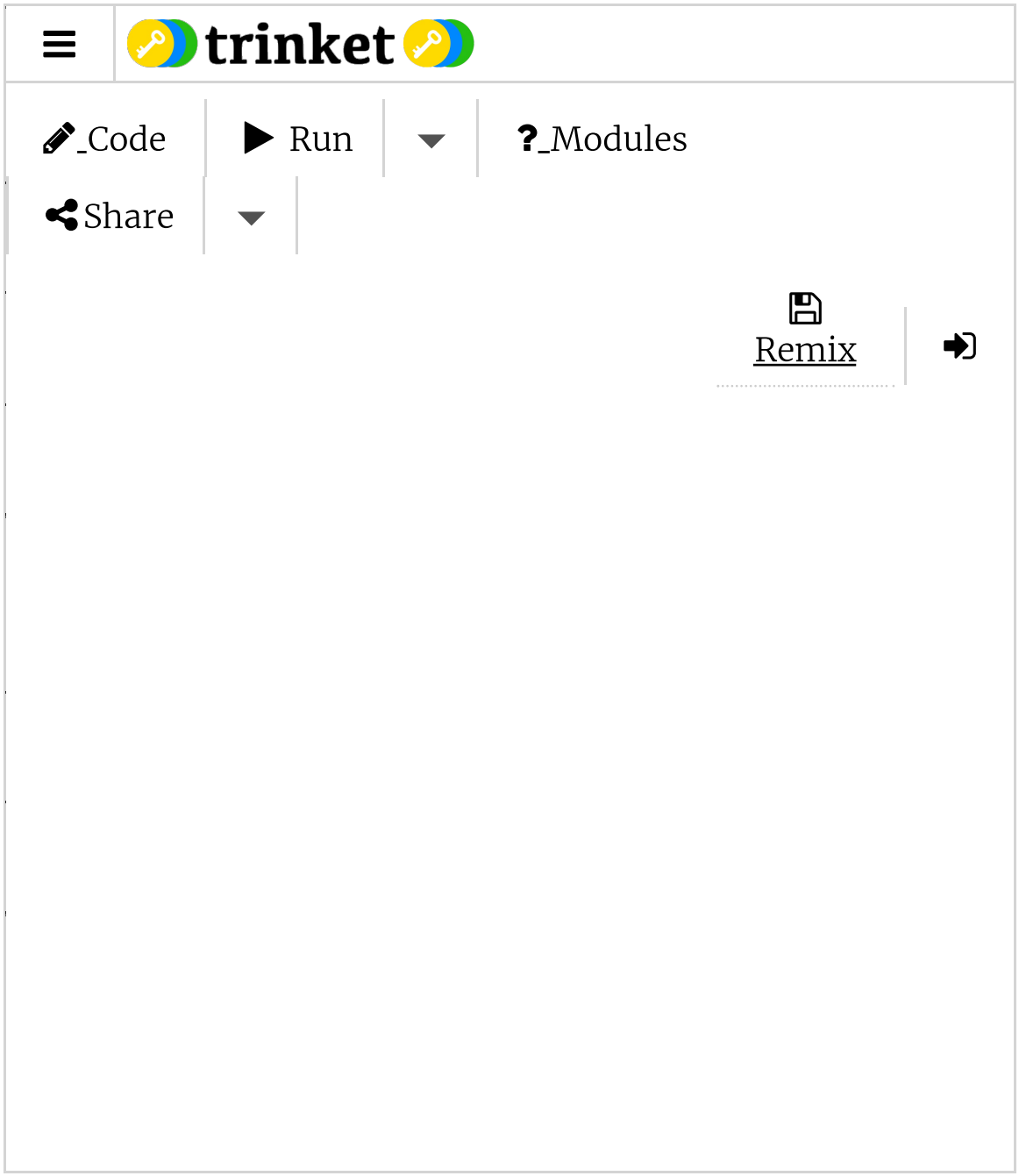
The *x* coordinate specifies how many squares along horizontally we mean and the *y* coordinate specifies how many squares vertically down we mean. We can use these coordinates to set individual pixels to a particular colour.

For example, if I wanted to set the third pixel along and 4th pixel down to green, I would use the following code:

```
green = [0,255,0] #Defining the colour green in RGB format

###Remember that in programming we count from zero!
sense.set_pixel(2,3,green) #Set the third pixel along and 4th pixel down to
green
```

Try it in the Trinket below:



Trinket Emulator

In the *set_pixel()* function, we first specify how many pixels along we mean and then how many pixels down we mean. In coordinate form this would be written like this: *(2,3)* where the first number, *2*, is the x position and the second number *3*, is the y position.

Back to snakes

To add a snake we use the *addSnake()* function. This function takes 4 arguments (arguments are the values we give the function), the first is the x position of the head of the snake, the second is the y position of the head, the third is the x position of the tail and the fourth is the y position of the tail.

If we wanted to add a snake with it's head at the position (4,1) and tail at (3,3) like in the image below:

We would write:

```
game.addSnake(4,1,3,3)
```

In the *skeleton code* add at a few snakes on to your board. Don't forget to make sure the head of the snake is above the tail!

(Sec. 2.2) Adding ladders

Adding ladders on to the board is also very simple. Just like the *addSnake()* function we have a function to add a ladder on to the board. The following code will add a ladder with foot at position (2,7) and top at position (4,4):

```
game.addLadder(4,4,2,7)
```

(Sec. 2.2) Creating Players

The next lines of code create a player and add it to the game. But this code is incomplete. You need to define the colour of the player's piece.

Set the variable pieceColour to the colour you want the player to be on the board. Remember, colour is defined in RGB format.

```
pieceColour = [ ] #Define the piece's colour
```

(Sec. 2.3) Creating a cursor

The cursor starts in the bottom left corner of the Sense Hat's screen. The two lines below set this position as the cursors starting position. Later in the code you will change these variables to make the cursor move up, down, left and right.

```
cursorPositionX = 0
cursorPositionY = 7
```

In the next line specify the colour of the cursor.

Setting up the Game

The next few lines of code will set up the variables before we get down to the nitty gritty coding in the *main-loop*.

```
game.setBoard() ## Setting the board on to the pixels
image = sense.get_pixels() # Save the screen as it is for now
```

The line above takes all the objects we added into the game (snakes, ladders and players) and displays them on the screen. The second line takes an image of the screen as it is after we set it.

The first line in the code-block below takes gets all the snakes in the game and puts them in a *list* called *snakes*. The second line gets a *list* of ladders. These variables will be used in checking whether the player lands on a snake or ladder by *looping* through all the snakes in the game and *looping* through all the ladders.

```
snakes = game.getSnakes() # Getting list of snakes in the game
ladders = game.getLadders() # Getting a list of ladders in the game
```

(Sec. 3) Main program code

Before we start coding, we'll explain a bit what's going on in Sec. 2.

In this part of the code, we'll do all the main programming.
If we look at the skeleton code, we see that the section is within a *while*-loop.
Remember that code inside a while loop runs again and again, until we tell it to stop. The reason why we want our program to be in a loop is because we want to continually run until the user turns it off.

Because the code is in the *while*-loop, remember that you have to add a *Tab* at every line, to make it indented. Like this:

```
while True:
    # write your code like this,
    # with a tab at the start of the line.
```

Structure

The structure of the *main-loop* might look a bit complicated, but it is split up into three clear sections.

The first section is where the user plays their turn by moving the cursor using the joystick and placing their piece by pressing the middle button.

The second section is where we program the game to check if the player has landed on a snake's head. If the player does land on a snake's head the players piece is moved back to the position of the snakes tail.

The third section is where we code the game to check if the player has landed on the foot of a ladder. If they do, the piece will be moved to the top of the ladder!

(Sec 3.0) Moving the cursor

This part is a bit complicated, so pay attention! Don't be afraid to ask a supervisor if you don't understand something.

This section allows the player to place his piece on the board. It will run in a *while-loop* until the player places the piece with the middle button. The skeleton code should look like this:

```
turnPlayed = False #Defines whether the player has played their turn
while not turnPlayed:

    for event in sense.stick.get_events():

        ##Shows the cursor
        sense.set_pixels(image)
        sense.set_pixel(cursorPositionX, cursorPositionY, cursorColour)

        ## Your code goes in the sections below

        if event.action == "pressed":
            if event.direction == "up":
            ## 3.1 Make cursor go up one square

            elif event.direction == "down":
            ## 3.2 Make cursor go down one square

            elif event.direction == "left":
            ## 3.3 Make cursor go left one square

            elif event.direction == "right":
            ## 3.4 Make cursor go right one square

            elif event.direction == "middle":
                ## 3.5 Set player position to cursor position

                ## Refresh the board
                game.setBoard()
                image = sense.get_pixels()

                turnPlayed = True
```

This code will check if the user has pressed either the *up*, *down*,*left*, *right* or the *middle* button on the joystick. Never mind the complicated structure, try to look at the code and figure out how to use it.

Inside the code you find the comments with hints. Your code should go underneath these comments.

If the user presses *up*, *down*,*left* or *right* on the joystick, the variables *cursorx* and *cursory* should be changed. We'll give you a hint to start off with: If the user presses the up button, cursory should *decrease* by 1.

Bonus: One thing that you might want to add is that if the user pushes the brush off the screen, it should pop back up on the other side of the screen. You can do this using further if-statements.

Finally, if the user presses the *middle* button, the cursor should set the players position to the position of the cursor. To set the players position we use the function setPlayerPosition(x,y). Where x and y are the position we want to put the player at. For example if we were to set player1 to the position (3,1) we would write:

```
player1.setPlayerPosition(3,1)
```

This will set the player position to the pixel that is 3 along horizontally and 1 down vertically.

(Sec 4.0) Check if the piece is on a snake

To see if the piece is landed on a snakes head we must check if the position of the piece is the same as any of the snakes in the board. To do this, we loop through the *list* of snakes and check all of the snakes in the game!

```
for snake in snakes:
    snakeHead = snake.getHead() ## The snakes head
    snakeTail = snake.getTail() ## The snakes tail

     ## 4.1 Insert conditions that checks if player is on a snake
    if *put condition here * and * put second condition here *:

        ## 4.2 Insert what happens if the conditions are met


        ## Refresh board
        game.setBoard()
        image = sense.get_pixels()

        checkingSnakes = True
        break # breaks out of the loop
    else:
        checkingSnakes = False
```

The variable snakeHead is a coordinate like this [x,y]. Where x is how far along horizontally the snake's head is and y is how far down it is.

We can get the x position from the snakeHead variable like this:

```
snakeHead = snake.getHead()
headX = snakeHead[0] ## The x position of the head
```

Similarly, we can get the y position from the snakeHead variable like this:

```
snakeHead = snake.getHead()
headY = snakeHead[1] ## The y position of the head
```

The snakeTail works exactly the same way!

```
snakeTail= snake.getTail()
tailY = snakeHead[1] ## The y position of the tail
```

Optional: The snakeHead variable is actually a list. To get the first element in a list we use square brackets. The number inside the square brackets is the position of the element in the list. The first element is at position 0, the second at 1 and so on.

In the ***if*** statement add your conditions to check if the player has landed on the snake's head. You should replace the lines that say *put condition here*.

Try to think how you would test if two things are at the same position.

(Sec 4.2) Bitten by a snake!

If the player lands on a snake's head, where is the piece meant to go?

Remember, you can set the player's position using the setPlayerPosition() function on the player *object*.

(Sec 5.0) Check if the piece is on a ladder

This section is very similar to the one above. The code does very similar things to the one in the previous section.

Getting the x and y position for the ladderTop and ladderBottom is done using square brackets just like the snake heads.

Try to use the code that you completed above and figure out how to complete this section.

Finished!

If it's all done, correctly, the game should now work! 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!

If it works, congratulations! You can either move on to another project 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: Make the game multiplayer

Currently, you only have 1 player in the game. Try to think about how you would change the code to add some more players to the game.

To get a list of players added to the game you can use the following code:

```
players = game.getPlayers()
```

Hint: The solution lies in a loop!

Author: Ishan Khurana

Date: August 15, 2017