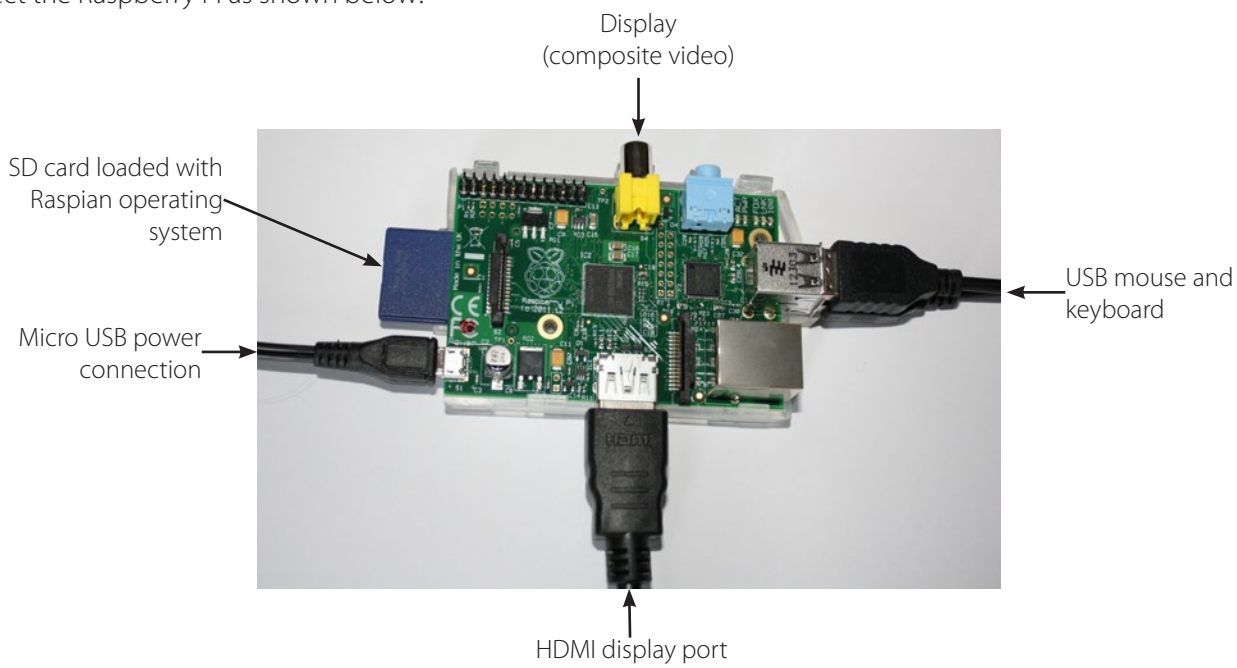
CLASSROOM CHALLENGE

SIMPLE ANIMATION USING THE RASPBERRY PI

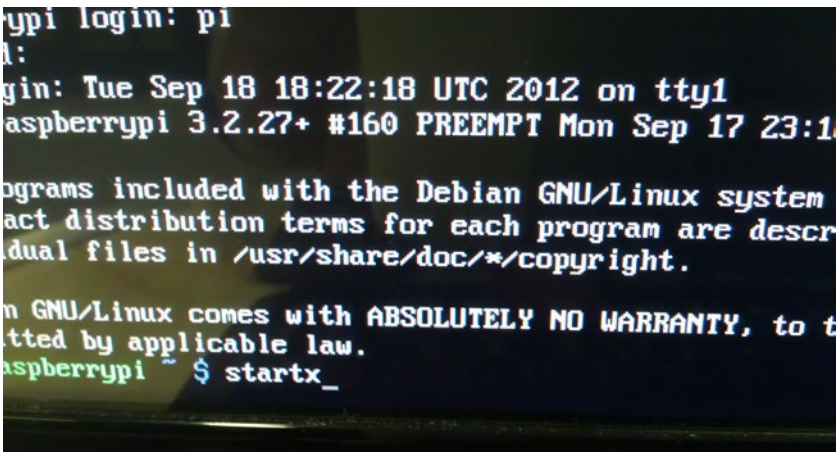
In this lesson you will learn how to create simple animations using Scratch and Python. You are going to create repeating patterns using iteration (loop) constructs. This activity will allow you to develop your programming skills, solve problems and be creative.

TASK 1 (5 minutes)

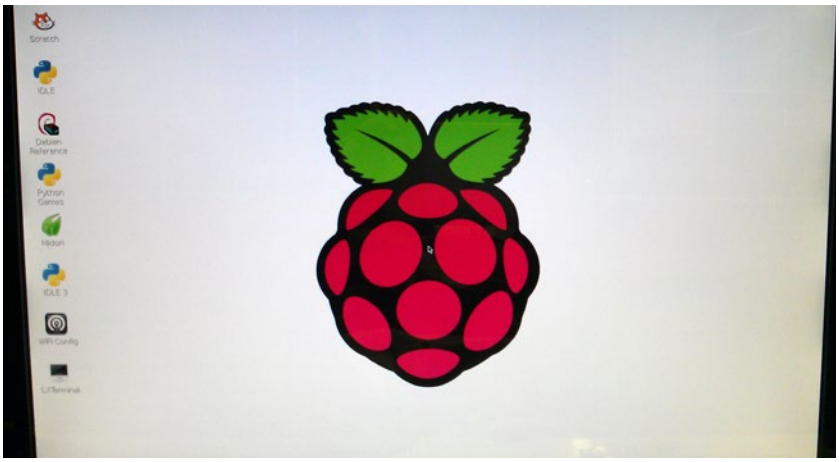
Connect the Raspberry Pi as shown below:



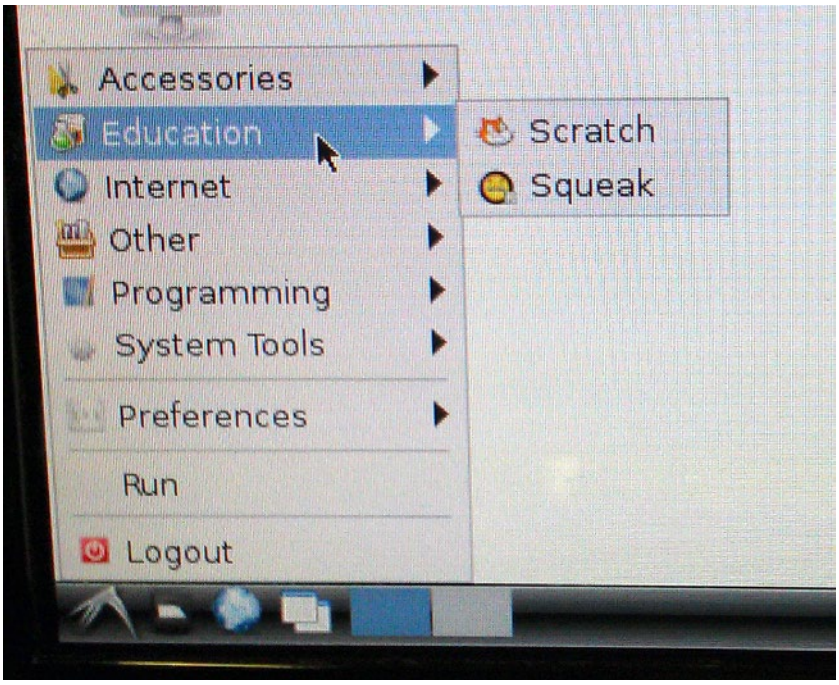
Boot up the Pi and enter the username and password. Ask your teacher for this. At the next prompt, type: **startx** to open the Graphic User Interface (GUI).



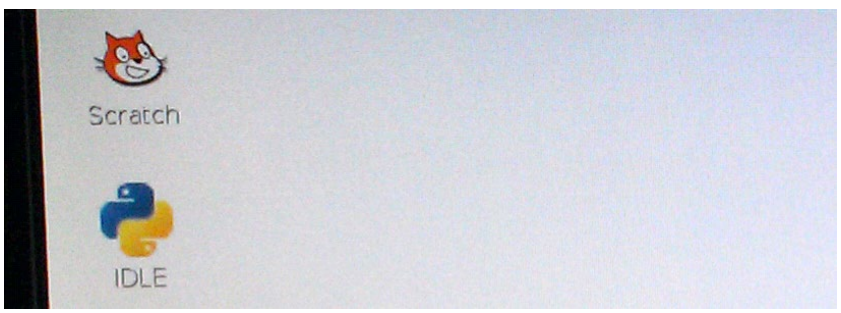
Terminal login screen showing **startx** command



Desktop view



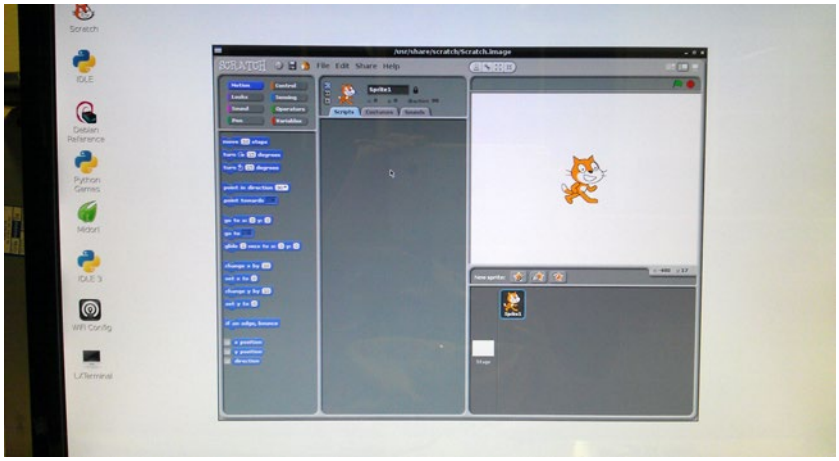
Menu showing Scratch



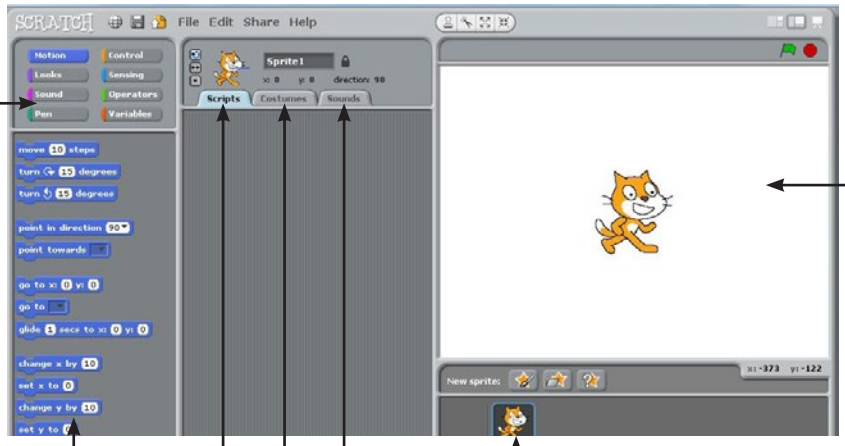
Desktop icon for Scratch

TASK 2 (10 minutes)

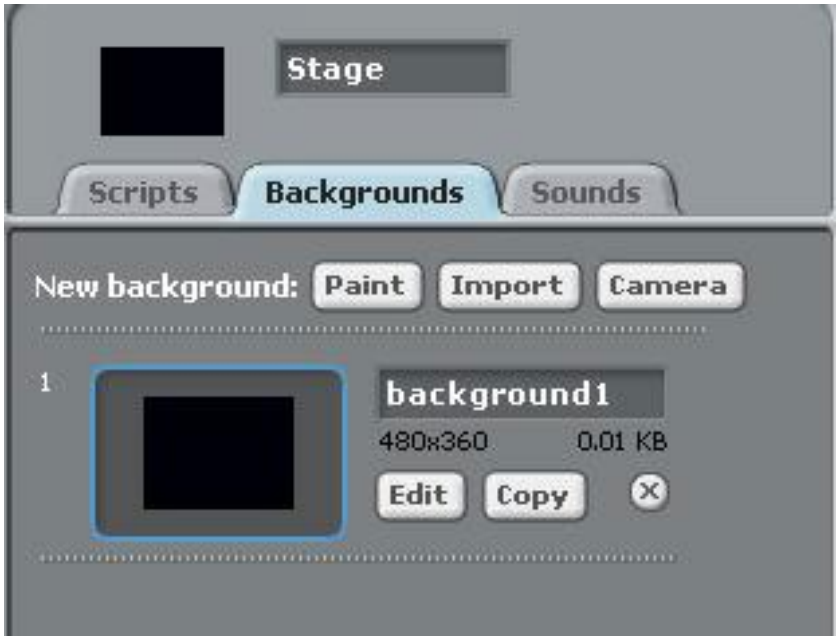
1) Load Scratch:



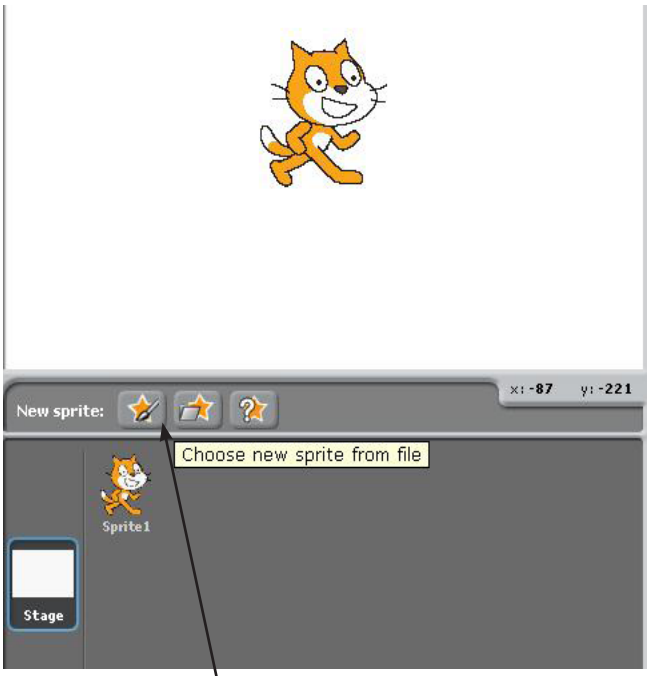
2)



3) Double click on the stage to bring up the stage script view. Click on the backgrounds tab. Set a colour for the background of your animation. You could use white but black or another dark colour may be more effective.



4) Next select your sprite or sprites. This example uses Scratch but you could choose another from the Sprite folder.



We are going to create some shapes on the screen using code. The commands that we will use are based on the programming language **Logo**. This was created in the 1960s and uses a turtle robot to draw lines on the screen. The commands control the turtle to draw shapes.

You can read more about Logo here: http://el.media.mit.edu/logo-foundation/logo/index.html

Here is some pseudocode for drawing a shape using Logo. Can you work out what the shape is?

START

SET START POSITION

FOR COUNTER 1 TO 4

  PEN DOWN

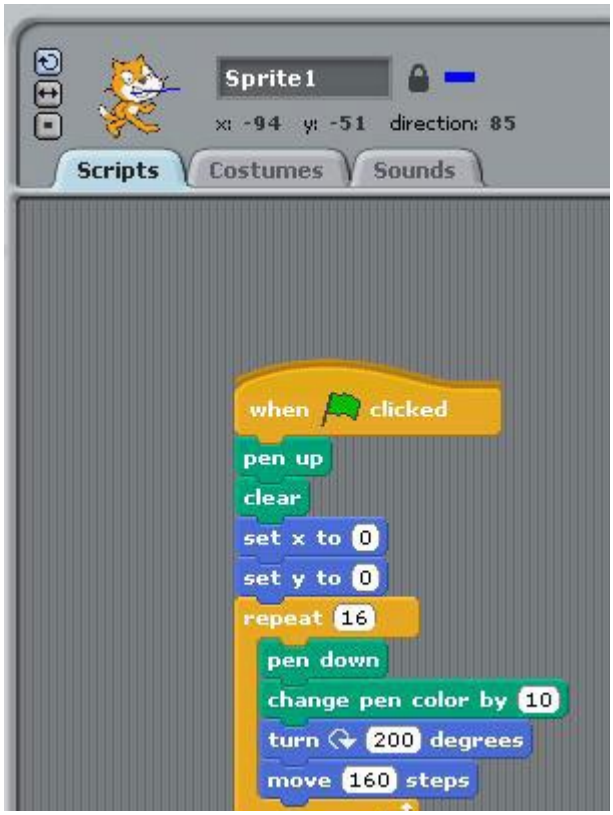
  MOVE FORWARD 100

  TURN 90

  NEXT COUNTER

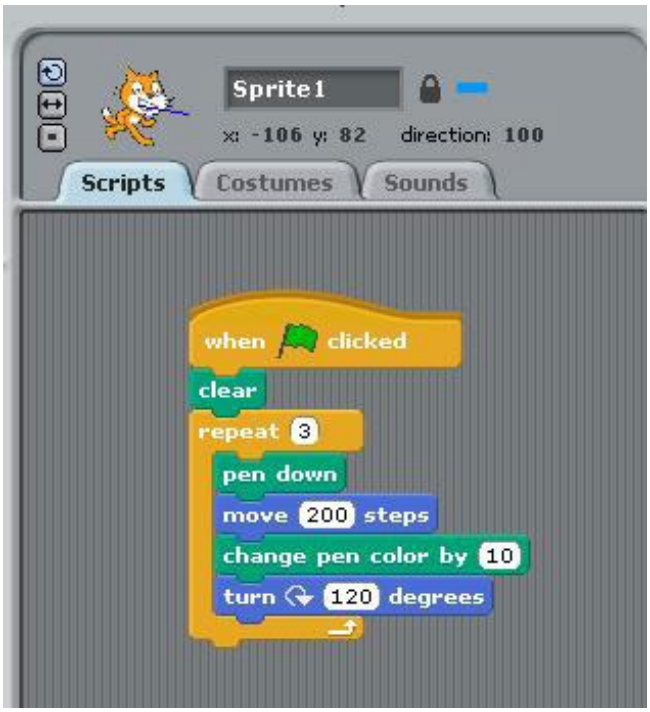
END

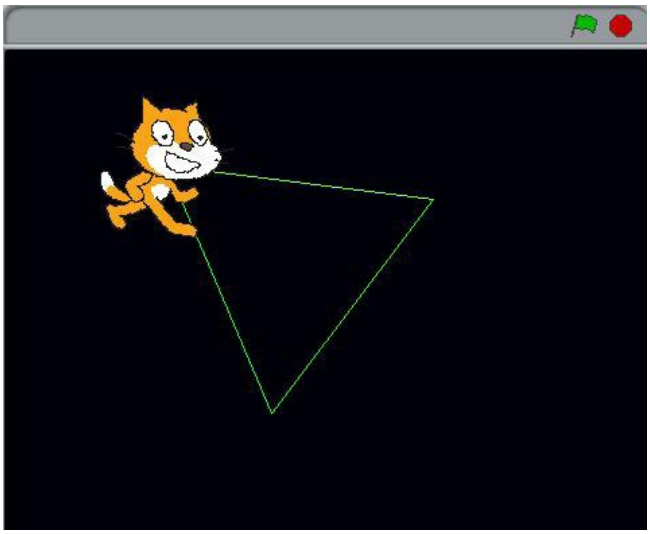
5. This script uses the PEN function which allows the sprite to "draw" a line. Copy this script to see what it does.



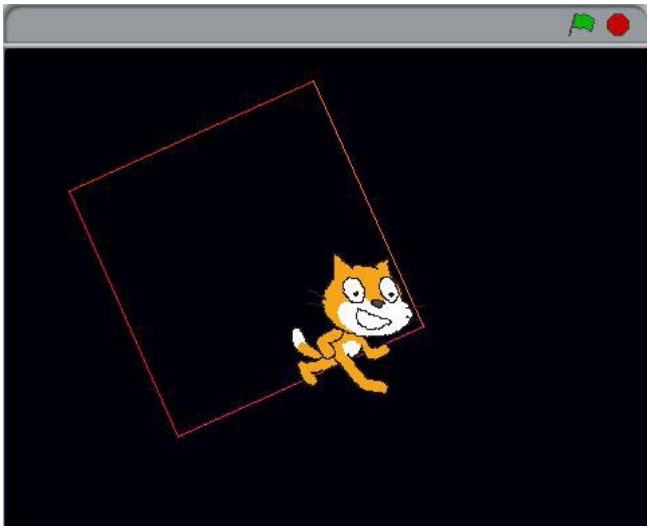
TASK 3 (20 minutes)

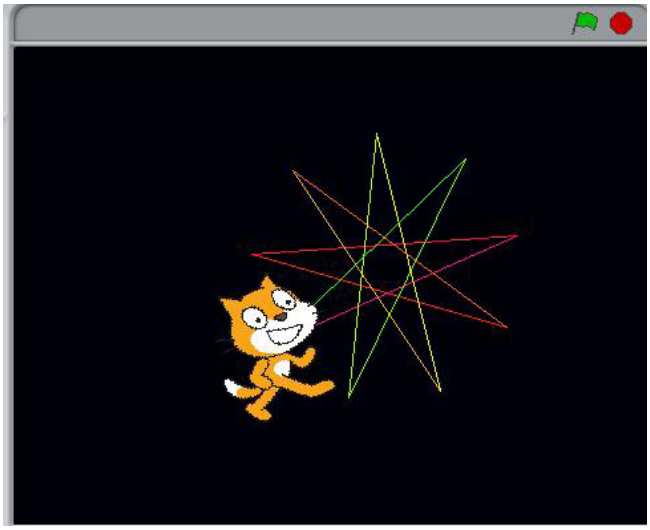
By combining together **moving**, **turning** and **changing pen colour**, different coloured shapes can be produced. Here I have written a script to "draw" a triangle.





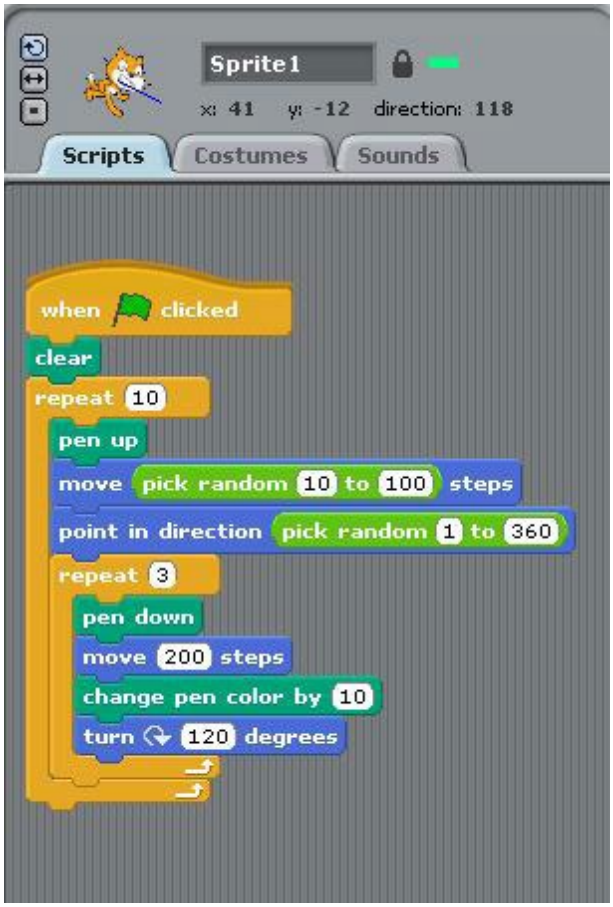
Write scripts to draw other shapes. Produce **pseudocode** or a **flowchart** to show your algorithm. Think before you do this about the angle that the sprite should turn to create the shape.



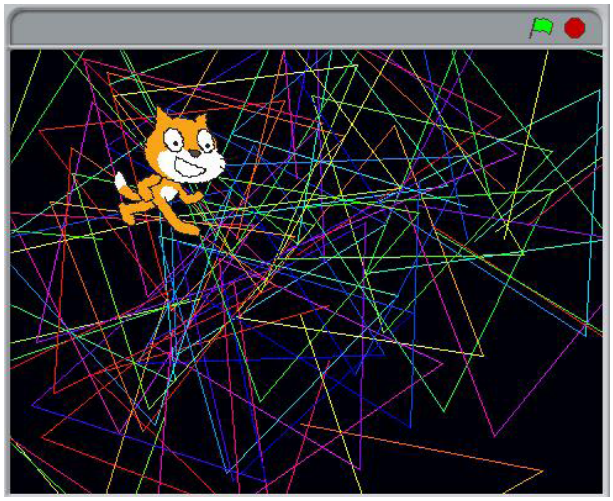


TASK 4 (20 minutes)

We will now use a **nested loop** to move the sprite before we draw the shape. The outer loop will move the sprite to a new starting position each time the shape is drawn. The inner loop will draw the shape at the new starting position. Think of it as a wheel turning inside a larger wheel. If we choose a random number of steps to move and a random direction to face, the shape will be drawn in different positions on the screen. Increasing the number of "turns" of the loop will increase the number of shapes drawn.



You can now design your own pattern by using repeating shapes. Use **pseudocode** or a **flowchart** to plan out your script before you begin.



TASK 5 (5 minutes)

Fill in the table to explain the purpose of each Scratch command:

Command	Purpose
Clear	
Repeat	
Pen up	
Pen down	
Turn	
Move	

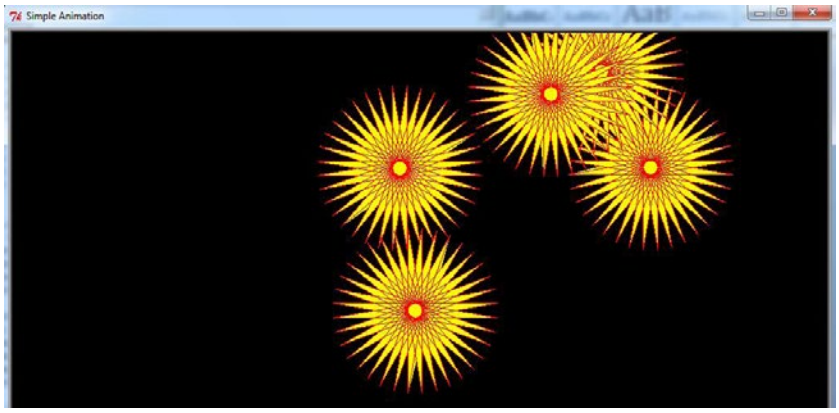
TASK 6 (20 minutes)

In this task you will investigate creating a simple animation using the programming language Python which is also provided with the Raspberry Pi Raspian Wheezy operating system.

1) Load **IDLE3** from the desktop which is the Python Integrated Development Environment.

2) Load the program called **animation.py**

3) Run the program. You should see the five shapes drawn on the screen:



4) Look through the programming code line by line.

```
""" Simple animation program"""
import turtle #import the turtle and random libraries so that they can be used in the program
from random import *

x = int() #This variable will store the x coordinate
y = int() #This variable will store the y coordinate
i = int() #An index for the outer loop
j = int() #Index for the inner loop

myturtle = turtle.Turtle() # create a turtle
mywindow = turtle.setup(1000,500) #setup a window of width 1000 and height 500
mywindow = turtle.Screen() #create the window
mywindow.title("Simple Animation") #The title of the window
mywindow.bgcolor("black") # Background colour
myturtle.color("red", "yellow") #This specifies the pen colour and the fill colour for the turtle
myturtle.penup()#Don't draw while we are moving the turtle to the centre
myturtle.setpos(500,0) # Place the turtle into the centre of the screen

for i in range(5): #Outer loop to move the turtle to the drawing position
    myturtle.penup()
    x = randint(-400,400) #Select a random x coordinate between this range
    y = randint(-200,200) #Select a random y coordinate
    myturtle.setpos(x,y) # Setthe position of the turtle
    myturtle.begin_fill() #This function is called before we start to draw the shape

    for j in range(52): #How many points we are going to draw
        myturtle.pendown() #Start drawing
        myturtle.forward(200)#Draw a line
        myturtle.left(170)#Turn through this angle
    myturtle.end_fill() #Called after drawing - now fill in the shape

mywindow.mainloop() #This is the event handling part of the program
myturtle.done() #This must be the final line of the program
```

This is a useful activity. Very often programmers need to see what another programmer has produced, break down the algorithm and then improve it. Document the program algorithm using a flowchart or pseudocode.

5) Comments have been added to explain what each block or line does. Write some new algorithms to create different patterns and shapes. Here are some ideas to get you started:

   a) Change the background colour (**bgcolor**) of the screen. (Hint: think about what colour names are most likely to be recognised).

   b) Change the **pencolor** and **fillcolor** values to create patterns using different colours.

   c) Alter the number of shapes that are drawn on the screen.

   d) The program uses two **loop** constructs. These are **nested**. Use what you have learned by carrying out the Scratch activity to draw other shapes on the screen by changing the **forward** and **left** values and creating different numbers of points.

ADDITIONAL ACTIVITIES (30 minutes)

a) Allow the user to customise the animation by inputting:
   i) The number of shapes to draw
   ii) The colours to use
   iii) The shape to draw

b) Change the **pen colour** while the program is running to draw shapes of varying colour.

c) Repeat this to include various **fill colours**.

d) Investigate the **stamp()** function and see if this could be used to create more interesting animations.

Remember to document your algorithms. You will also need to work carefully to solve issues in your code. You should do this by performing a dry-run with your flowchart or pseudocode. You can do this by drawing the shape yourself on a sheet of paper to test that your algorithm will work. When you have created your finished program, test it carefully to make sure that it works correctly. Create a test plan and use this to ensure that all aspects are fully tested.