COMP1021
Introduction to Computer Science

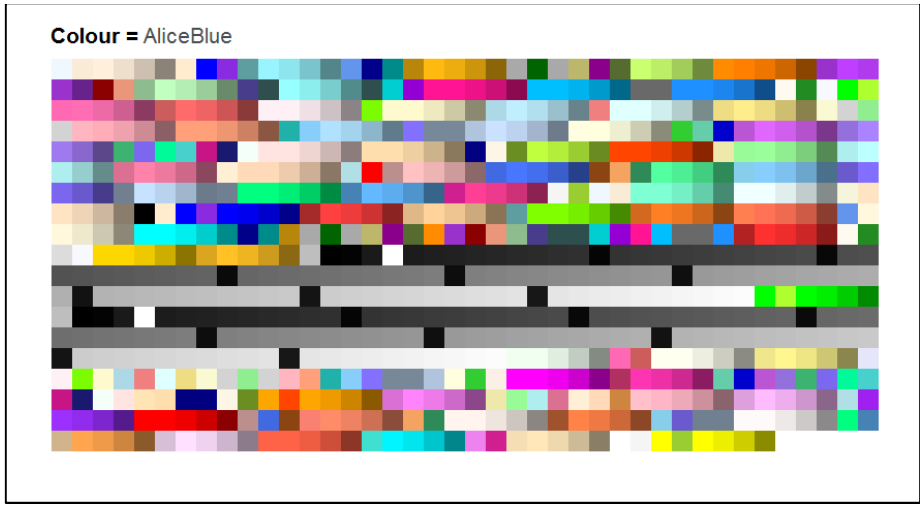
Understanding Colours

David Rossiter and Gibson Lam

Outcomes

- After completing this presentation, you are expected to be able to:
  1. Make colours in Python using the RGB colour system

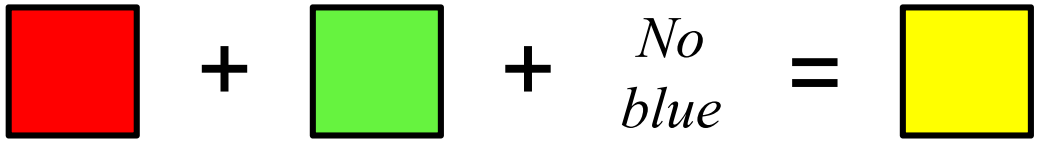
Using Colours in Turtle



- In lab 2, we gave you a program to show the colours that you can use by their names in your turtle graphics program
- These colours are predefined by Python so that you can easily pick suitable colours for your program
- Sometimes, you may not be able to find the colours you want
- In this presentation, we will look at how you can *make* your own colours

How Colours are Made in Computers

- For computers, a colour is actually a combination of red, green and blue (RGB) that gives you a single colour
  - You make one colour by using some amount of red, some amount of green and some amount of blue
- For example, yellow is made of a combination of red and green, without any blue



- Sometimes this is called the RGB colour system

Making an RGB Colour

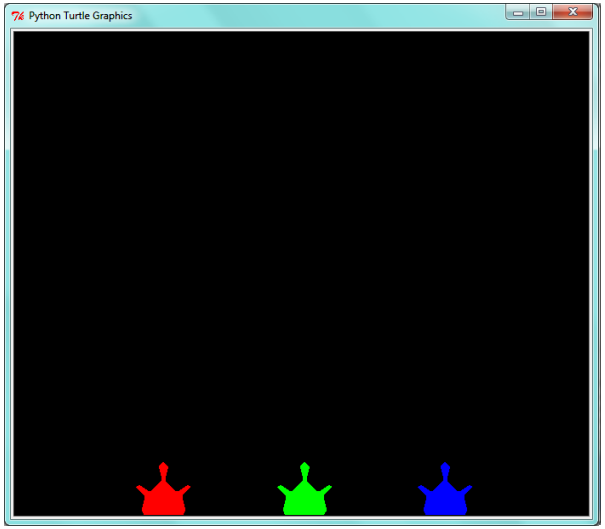
- To make a colour using RGB, you give three numbers to represent the amount of red, green and blue you need to use
- Usually, the three numbers are each stored in a *byte* (we will not look at what a byte is in any detail)
- A byte stores an integer in the range 0-255 inclusive
- For example, to make yellow, you will use 255 of red, 255 of green and 0 of blue
- White has 255 for all three numbers and black has 0 for all of them

A Turtle Colour Program

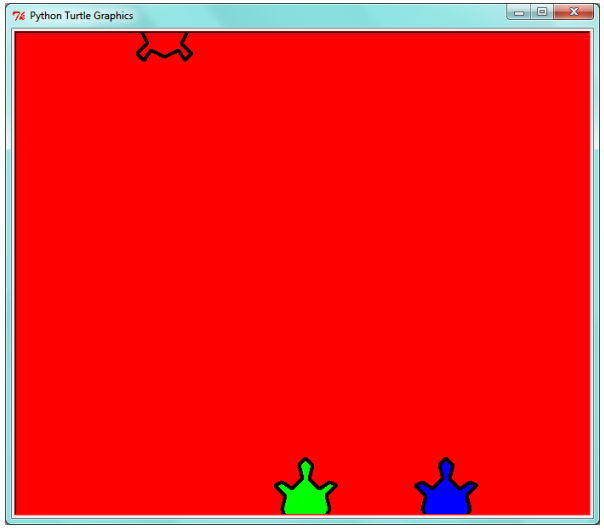
- Let's look at a turtle program which illustrates how a single colour is created
- The program uses a red turtle, a green turtle and a blue turtle to control the level of red, green and blue (RGB) components, which make a colour
- You drag the turtles up and down to adjust the contribution of each colour
- In this example, the three levels of RGB together determine the background colour of the screen

Some Examples

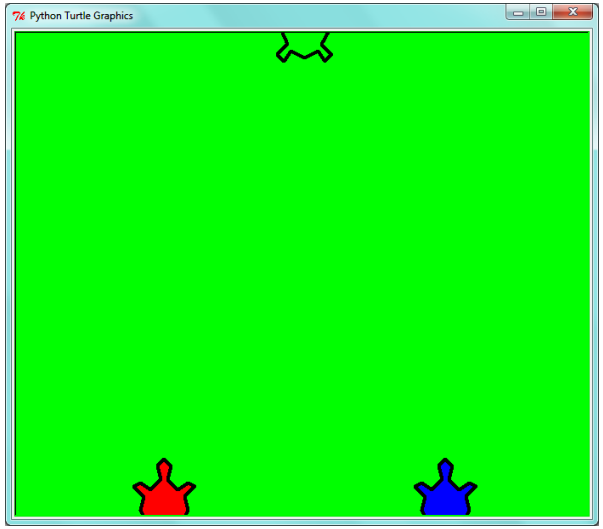
Black



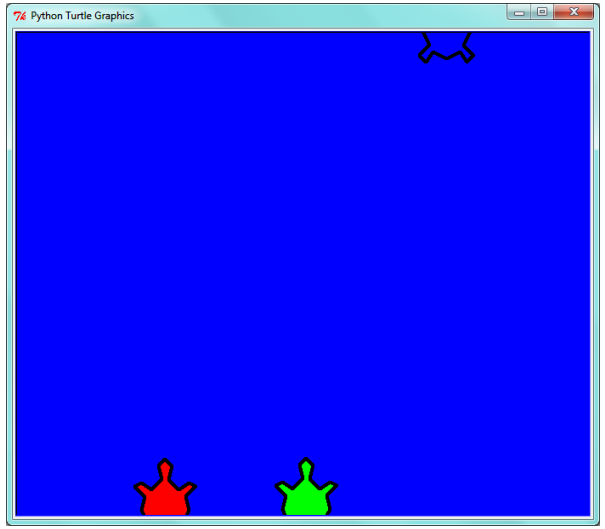
Red



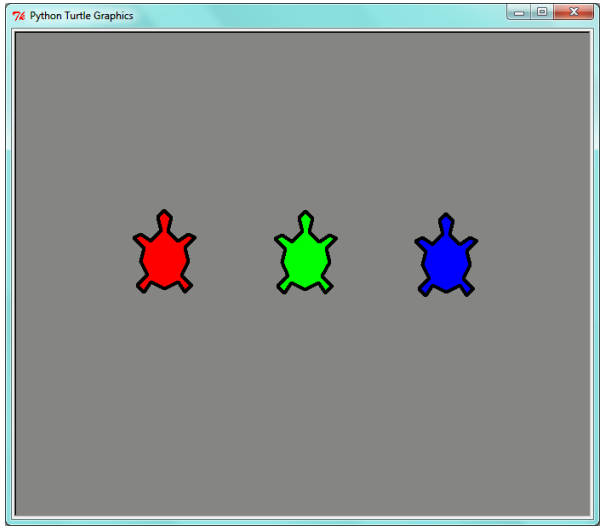
Green



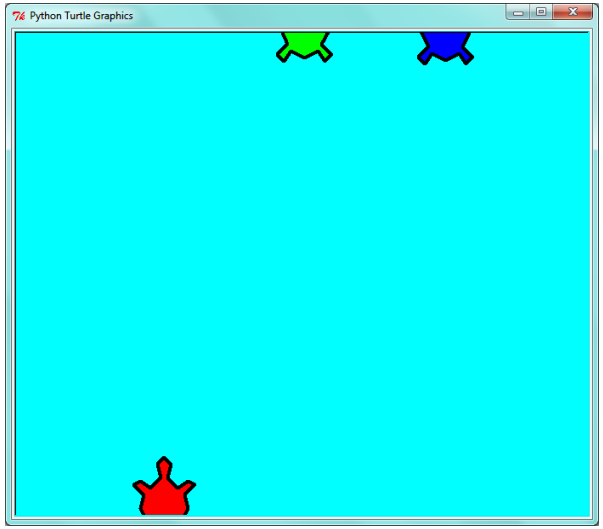
Blue



Gray

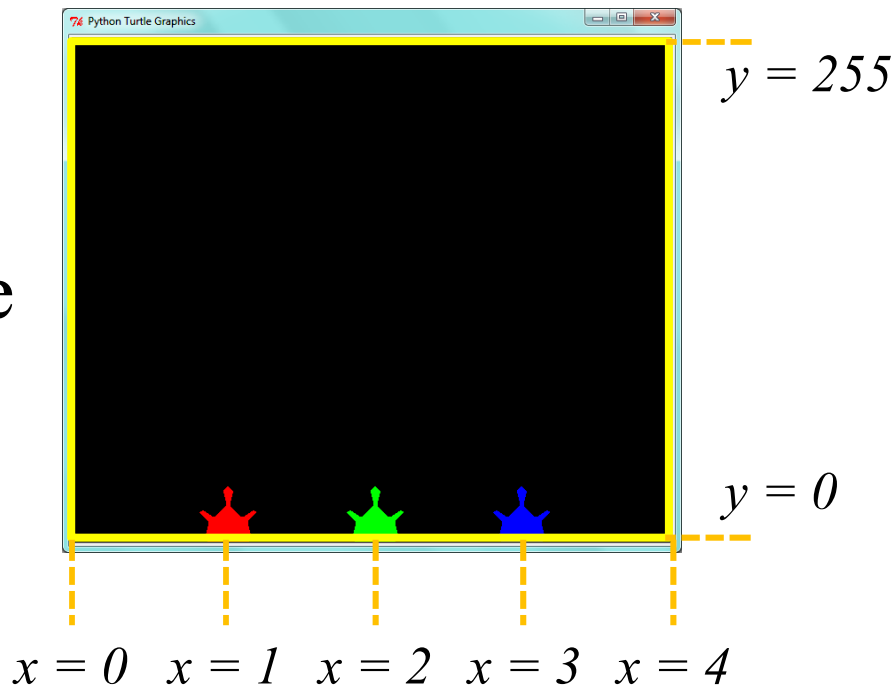


Cyan



The Screen Coordinate System

- In this example, we use a clever coordinate system
- We choose a y axis range so that it covers the range 0 to 255 (for RGB input)
- We choose an x axis range so that we have three x values in the middle (for the three turtles)
- The code to do that is:




```
turtle.setworldcoordinates(0, 0, 4, 255)
```

The Turtle Colour Mode

- The turtle system accepts two different ways of handling RGB colour values:
  - 3 float values from 0.0 to 1.0, or:
  - 3 integer values from 0 to 255 *(more commonly used)*
- You can ask the turtle system to accept a particular range using `turtle.colormode()`
- Our example uses the following line of code to tell the turtle system we will use the integer range 0 to 255:

```
turtle.colormode(255)
```

Setting Up the Turtle Window

- In our example the following code sets up the turtle window:

```
# Set up the turtle window

turtle.colormode(255)
turtle.setworldcoordinates(0, 0, 4, 255)
turtle.hideturtle()
turtle.tracer(False)
```

Use 0...255 for the RGB colours

Min x *Max x*

Min y *Max y*

With this coordinate system we can simply use the y position of the 3 turtles for the red/green/blue values

Creating the Turtles

- The code to create the red turtle is shown below:

```
# Set up the red turtle
red_turtle = turtle.Turtle()
red_turtle.fillcolor("red")
red_turtle.shape("turtle")
red_turtle.shapesize(4, 4, 4)
red_turtle.up()
red_turtle.goto(red_turtle_x, 0)
red_turtle.left(90)
red_turtle.ondrag(red_turtle_drag_handler)
```

The x position of the red turtle is always set to `red_turtle_x` (=1)

- Similar code is used to set up the green and blue turtles

The Turtle Drag Handlers

- The turtle drag handler for the red turtle is shown here:

```
def red_turtle_drag_handler(x, y):
    # Clear the drag handler
    red_turtle.ondrag(None)

    x = red_turtle_x
    red_turtle.goto(x, y)
    update_screen_colour()

    # Reassign the drag handler
    red_turtle.ondrag(red_turtle_drag_handler)
```

See next slide

Update the y position of the turtle by fixing the x position (so it cannot be dragged away from that x position), then update the background colour

- Similar event handler functions have been used for the green and blue turtles

Clearing the Drag Handler

1. *Clear the event handler so that the function won't be run even if the user drags the turtle while we are in the middle of the function*

2. *Use the event handler again after finishing the function code*

```
def red_turtle_drag_handler(x, y):
    # Clear the drag handler
    red_turtle.ondrag(None)

    x = red_turtle_x
    red_turtle.goto(x, y)
    update_screen_colour()

    # Reassign the drag handler
    red_turtle.ondrag(
        red_turtle_drag_handler)
```

- Python may get confused if you drag the turtle *while* the turtle drag event handler code is being executed, so we make sure that doesn't happen by doing the above

Updating the Background Colour

- This function updates the background colour using the turtles' y positions:

```
def update_screen_colour():
  red   = min(red_turtle.ycor(),   255)
  green = min(green_turtle.ycor(), 255)
  blue  = min(blue_turtle.ycor(),  255)

  red   = max(red,   0)
  green = max(green, 0)
  blue  = max(blue,  0)

  # Set the colour of the window
  turtle.bgcolor(int(red), int(green), int(blue))
  turtle.update() # Update the display
```

We want red, green and blue values to be in the range 0..255

Using min() and max()

- We could use `if` statements to check that the RGB values are within the allowed range of 0 to 255 inclusive
- Here is an example to make the red value to be smaller than or equal to 255 based on the y coordinate of the red turtle:

```
if red_turtle.ycor() > 255:
    red = 255
else:
    red = red_turtle.ycor()
```

- This is equal to `red=min(red_turtle.ycor(), 255)`
- We also use `max()` to make sure the value doesn't go below zero e.g. `red=max(red, 0)`