COMP1021
Introduction to Computer Science

# Understanding Colours

David Rossiter and Gibson Lam

# Outcomes

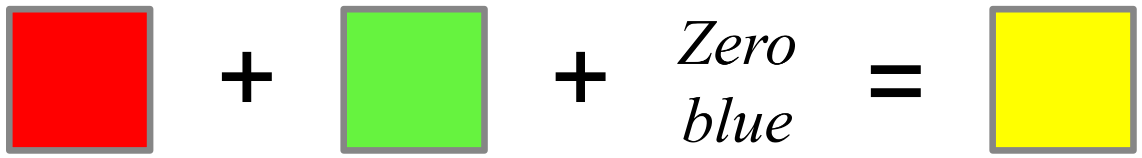
- After completing this presentation, you are expected to be able to:
    1. Explain the concept of the RGB representation of colour
    2. Make colours using the RGB colour system

## This Presentation

- This presentation considers some new things:
  - How colours work in a computer
  - How to keep values in a particular range
- In addition, several things that were discussed previously are used:
  - Using a more appropriate coordinate system
  - Creating and using turtle objects
  - Using event handling (dragging)

# How Colours are Made in Computers

- For computers, a colour is actually a combination of **r**ed, **g**reen and **b**lue (RGB) light
- 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, with no blue



- This is called the RGB colour system

## Making an RGB Colour

- To make a colour, you use three numbers to represent the amount of red, green and blue light
- Usually, each of the numbers is stored in a *byte* (we will not look at what a byte is in detail)
- A byte stores an integer in the range 0-255 inclusive
    - A byte cannot store a number higher than 255
- Example: to make yellow (see last slide) you use red=255, green=255 and blue=0

# Example RGB Values

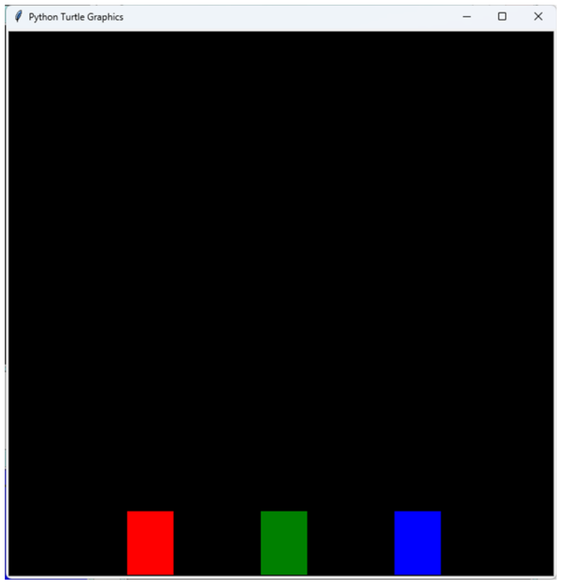
| | |
|---|---|
| • White is | 255, 255, 255 |
| • Middle gray is | 127, 127, 127 |
| • Black is | 0, 0, 0 |
| • Red is | 255, 0, 0 |
| • Green is | 0, 255, 0 |
| • Blue is | 0, 0, 255 |
| • Orange is | 255, 165, 0 |
| • Brown is | 150, 75, 0 |
| • Cyan is | 0, 255, 255 |
| • Magenta | 255, 0, 255 |
| • Yellow is | 255, 255, 0 |
| • Purple is | 160, 32, 240 |
| • Pink is | 255, 192, 203 |

## A Turtle RGB Colour Program

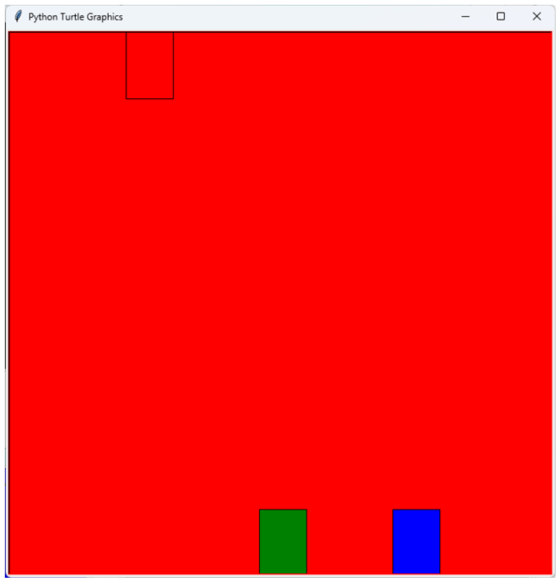
- Let’s use a turtle program which illustrates how a single RGB colour is created
- The program uses a red turtle, a green turtle and a blue turtle to control the level of red, green and blue (RGB) components, which make a colour
- You drag the turtles up and down to adjust the contribution of each RGB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

# Some Examples

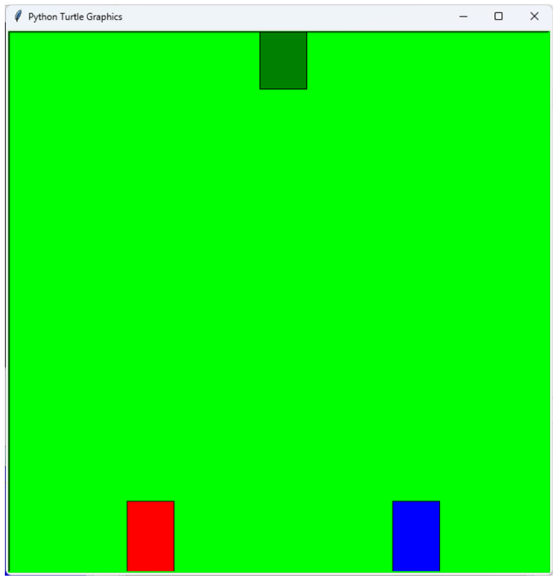
Black



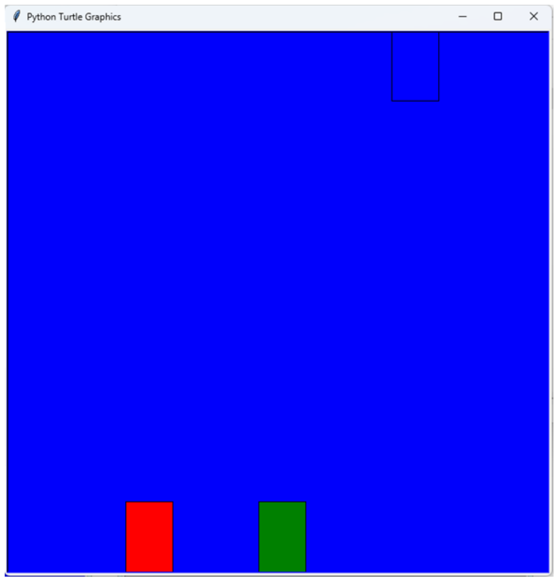
Red



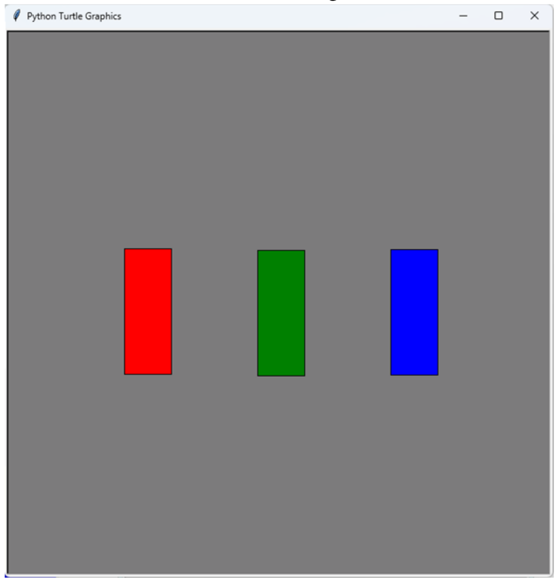
Green



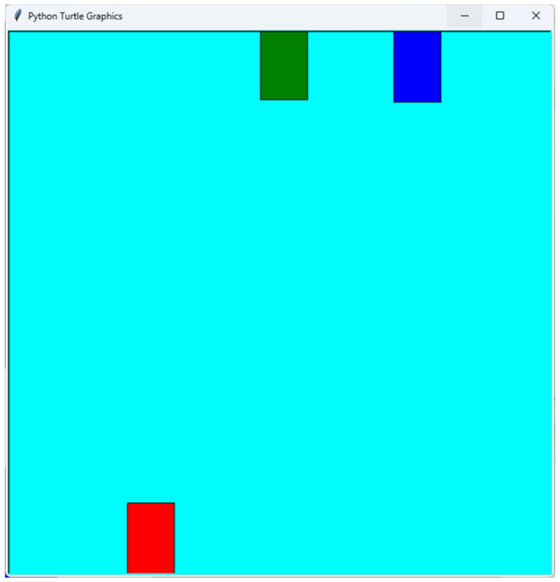
Blue



Gray



Cyan



# Setting Up

- In our example the following code is used to set things up:

```
turtle.colormode(255)

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

turtle.hideturtle()
```

*Tell the system we will use 0…255 for the three values*

*Min x* *Max x*

*Min y* *Max y*

*We will make 3 new turtle objects and use those, no need to see the default turtle*

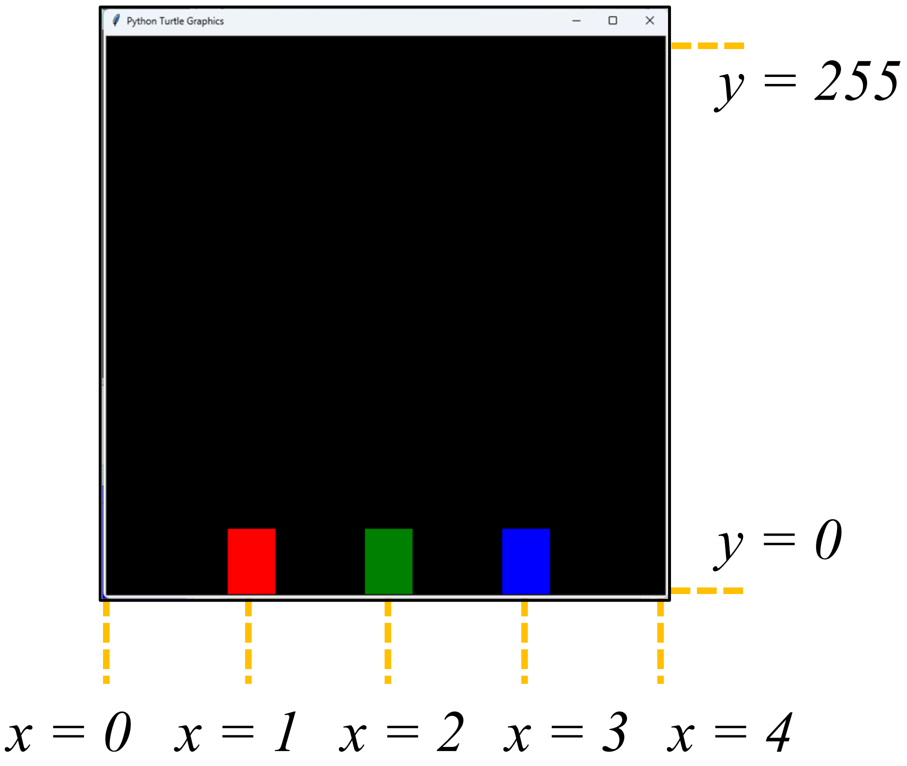
*With this coordinate system we can simply use the y position of the 3 turtles for the red/green/blue values*

# Turning on RGB

- You need to tell the turtle system you will use RGB by doing `turtle.colormode(255)`
- After doing that, any colour can be represented by the 3 numbers

# The Coordinate System

- For this example we use a clever coordinate system
- The y axis range covers the range 0 to 255 (=the range of each number)
- We choose an x axis range so that we have three x values in the middle (one for each of the turtles)
- The code used is:

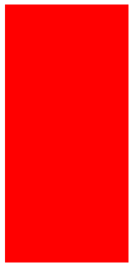


```
turtle.setworldcoordinates(0, 0, 4, 255)
```

# Creating the Turtles

- Here is the code to create the red turtle object:

```
# Set up the red turtle object
red_turtle = turtle.Turtle()
red_turtle.fillcolor("red")
red_turtle.shape("square")
red_turtle.shapesize(3, 8)
red_turtle.speed(0)
red_turtle.up()
red_turtle.goto(red_turtle_x, 0)
red_turtle.left(90)
red_turtle.ondrag(red_turtle_drag)
```



*The turtle looks like a rectangle*

*The x position of the red turtle is always set to* `red_turtle_x` which is 1

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

# Handling the Turtle Dragging

- This is the function which handles the red turtle dragging:

```
def red_turtle_drag(x, y):
    # Ignore any dragging
    red_turtle.ondrag(None)

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

    # Handle any dragging once again
    red_turtle.ondrag(red_turtle_drag)
```

*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*

- Very similar functions have been used for the green and blue turtles

# Safer Event Handling Code

1. *Make sure that the function won't be run a second time even if the user drags the turtle while we are in the middle of this function*
2. *We have finishing doing what we want, so turn on the drag behaviour again*

```
def red_turtle_drag(x, y):
    # Ignore any dragging
    red_turtle.ondrag(None)

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

    # Handle any dragging once again
    red_turtle.ondrag(
        red_turtle_drag)
```

- Python may run the function *again* while it is already in the middle of being executed, we make sure that doesn't happen by adding the two lines of code highlighted above

# Updating the Background Colour

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

*Get the y value from the turtle object*

```
def update_bg_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 window background colour using RGB
  turtle.bgcolor(int(red), int(green), int(blue))
```

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

## Using `min()`

- Each of the three RGB values must be in the range 0 to 255 inclusive
- This code makes sure the red value is not > 255

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

- Using `min()` does the same thing, but less typing:
`red = min(red_turtle.ycor(), 255)`

# Using `max()`

- Similarly, we use `max()` to make sure the value doesn't go below zero e.g. this:

```
if red_turtle.ycor() < 0:
    red = 0
else:
    red = red_turtle.ycor()
```

is equal to `red = max(red_turtle.ycor(), 0)`