**Outline**

Develop an understanding of how images and graphics are drawn and stored in a computer. Learn about the RGB colour space. Apply Python concepts related to lists and loops.

**Objectives**

* tbd

**Materials**

* tbd

**Level 1: Pixels & RGB**

1. Create a new Repl for Python with Turtle.
   1. Copy and paste “Sample Program #1” from the listing at the end of this module.
   2. Run the program and examine the Turtle output
2. Colours can be specified by using a combination of three numbers. These three numbers together define a “Pixel” point in a graphic image.
   1. What position is the number that controls the amount of red (r) in the pixel?

**The first one**

* 1. What position is the number that controls the amount of green (g) in the pixel?

**The second one**

* 1. What position is the number that controls the amount of blue (b) in the pixel?

**The third one**

1. Colour number values can range from 0 to 255.
   1. What happens when the colour value is less than 255?

**A darker shade of the colour**

* 1. What happens when the colour value is close to 0?

**Becomes black**

1. Other shades of colours can be created using a combination of r,g,b number values.
   1. Create a pixel containing a shade of the colour orange.

**redColor = (255,101,10)**

* 1. Create a pixel containing a shade of the colour yellow.

**redColor = (255,255,0)**

* 1. Create a pixel containing a shade of your favorite colour.

**redColor = (255,100,100)**

1. Black, white, and shades of grey are created using combinations of equal r,g,b number values.
   1. Create a completely white pixel.

**redColor = (255,255,255)**

* 1. Create a completely black pixel.

**redColor = (0,0,0)**

* 1. Create a pixel containing a shade of middle grey.

**redColor = (100,100,100)**

**Level 2: Images Using Pixels**

1. Download the image “Resoultion\_284x177.jpg” from Topic B folder in the class repository.
   1. Open the image in a program like Paint or Photoshop.
   2. What is the size of this image? How many pixels does it contain?

**284x177 is the size of the image**

**It contains 50268 pixels**

* 1. Describe how the image looks (e.g. Can you see the pixels?)

**I cannot see the pixels**

* 1. Zoom in the view to enlarge the image
  2. Describe how the image looks (e.g. Can you see the pixels?)

**I can see the pixels and the picture is a bit more blurry**

1. Download the image “Resoultion\_16x16.jpg” from Topic B folder in the class repository.
   1. Open the image in a program like Paint or Photoshop.
   2. What is the size of this image? How many pixels does it contain?

**Size is 8x8**

**There are 64 pixels**

* 1. Describe how the image looks (e.g. Can you see the pixels?)

**Cannot see the pixels because the image is very small**

* 1. Zoom in the view to enlarge the image
  2. Describe how the image looks (e.g. Can you see the pixels?)

**I can see all the pixels and there it is very blurry**

1. Create a new Repl for Python with Turtle.
2. Copy and paste “Sample Program #2” from the listing at the end of this module.
3. Run the program and examine the Turtle output
4. Compare the program output to the “Resoultion\_16x16.jpg” image in question #2 above.  
   **It is the same as the “Resoultion\_16x16.jpg” image**
5. Explain how the program code in lines 52 to 58 works. (i.e. The main program code.)
6. How the program prints out pixels to produce and 8 by 8 resolution image.

**The numbers that were put in the loop are 8 and 8 resulting in and 8x8 image**

1. How the program decides which colour information to use for each pixel.

**From the pixel memory**

1. Explain the purpose of the code in lines 12 to 21

**It has all the data for the colours used**

1. How this code is related to the pixels produced by the main program.

**The main program uses this data to produce the different colours**

1. The RGB value of the 19th pixel in the image

**(28,28,12)**

1. The RGB value of the pixel in the 5th column on the 4th row.

**(154,140,22)**

1. Modify the main program to print the image upside-down (i.e. pixels in reverse order).
2. Show your modified image to Mr. Nestor.
3. Explain your changes to the program code below.

**pixelAddress += -1**

1. Modify the main program to print the image at a resolution of 12 by 4 pixels.
2. Show your modified image to Mr. Nestor.
3. Explain your changes to the program code below.

**for row in range (12) :**

**for column in range(4) :**

**This makes the resolution 12 by 4 because of the rows and columns values**

**Level 3: Your Custom Image**

1. Use and modify the sample pixel program code to create your own custom image.
   1. Create a larger resolution image than provided in the sample.
   2. Make sure the image is recognizable (or a clear pattern).
   3. Show your image to Mr. Nestor.
2. List and explain your modified image code below.

This is a picture of a barn on a field with the sun out.

import turtle

myPen = turtle.Turtle()

# These variables track the position of the turtle pen

posX = 0

posY = 0

# These variables define the image information.

# Each pixel in the image has a (r,g,b) value

# The complete image is simply a list of pixels

pixelAddress = 0

pixelMemory = [

(135,206,235),(135,206,235),(135,206,235),(135,206,235),(135,206,235),(255,255,0),(255,255,0),(255,255,0),

(135,206,235),(100,69,19),(135,206,235),(135,206,235),(135,206,235),(255,255,0),(255,255,0),(255,255,0),

(100,69,19),(100,69,19),(100,69,19),(135,206,235),(135,206,235),(135,206,235),(255,255,0),(255,255,0),

(178,34,34),(178,34,34),(178,34,34),(135,206,235),(135,206,235),(135,206,235),(135,206,235),(135,206,235),

(178,34,34),(178,34,34),(178,34,34),(135,206,235),(135,206,235),(135,206,235),(135,206,235),(135,206,235),

(34,139,34),(34,139,34),(34,139,34),(34,139,34),(34,139,34),(34,139,34),(34,139,34),(34,139,34),

(0,100,0),(0,100,0),(0,100,0),(0,100,0),(0,100,0),(0,100,0),(0,100,0),(0,100,0),

(0,70,0),(0,70,0),(0,70,0),(0,70,0),(0,70,0),(0,70,0),(0,70,0),(0,70,0)

]

# This user defined function draws a single image pixel

def drawPixel(rgb) :

global posX

myPen.down()

myPen.color(rgb)

myPen.begin\_fill()

myPen.circle(8)

myPen.end\_fill()

myPen.up()

myPen.forward(18)

posX = posX + 18

# This user defined function starts a new row of pixels

def newRow() :

global posX

global posY

myPen.up()

myPen.left(180)

myPen.forward(posX)

myPen.left(90)

myPen.forward(18)

myPen.left(90)

myPen.down()

posX = 0

posY = posY + 18

# THE MAIN PROGRAM CODE STARTS HERE

#

# Draw eight rows of the image.

# Each row contains eight pixels

for row in range (8) :

for column in range(8) :

drawPixel(pixelMemory[pixelAddress])

pixelAddress += 1

newRow()

**I changed the colors of the pixels and if I wanted to I could have increased the resolution**