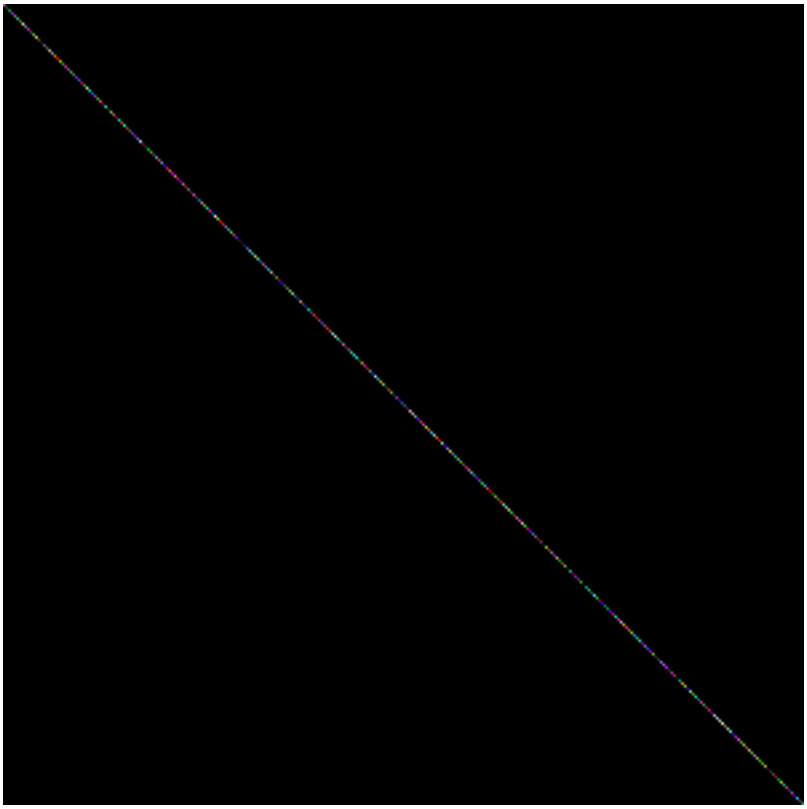
PPiC 6.1 Modify Session 6.5 to create a line with random pixel colors.

```
>>> from cImage import *
>>> myImWin = ImageWin("Line Image",300,300) #create a windown to display images that are 300 pixels wide and 300 pixels high
>>> lineImage = EmptyImage(300,300) #create an empty image that is 300 pixels wide and 300 pixels high
>>> for i in range(300): #for loop: for each number in range(300)
        import random #since we want random pixel colors, we import random module
        r = random.randint(0,255) #Intensities will range from a minimum of 0 to a maximum of 255
        g = random.randint(0,255)
        b = random.randint(0,255)
        colorPixel = Pixel(r,g,b) #create a pixel with r,g,b we just created
        lineImage.setPixel(i,i,colorPixel) #set the pixel of the line at row i, column i, to the colorPixel

>>> lineImage.draw(myImWin) #draw the lineimage in the myImWin image window
>>> lineImage.save("lineImage.gif")
>>>
```



PPiC 6.10 Write a function that takes a color image and displays a black and white image next to it. *Hint*: You may want to start by converting the image to grayscale. Any pixel with gray value less than some threshold will become black. All other pixels will be white.

```
>>> from cImage import *
>>> def blackandwhite(imageFile): #define a function blackandwhite with parameter imageFile
        myImWin = ImageWin("Image Processing",800,250) #create a windown to display images that are 800 pixels wide and 250 pixels high
        oldIm = FileImage(imageFile) #create an image object from a file named imageFile
        oldIm.draw(myImWin) #draw the image oldIm in the myImWin image window. It will default to the upper-left corner.
        width = oldIm.getWidth() #return the width of the oldIm in pixels
        height = oldIm.getHeight() #return the height of the oldIm in pixels
        newim = EmptyImage(width,height) #create an empty image that is width pixels wide and height pixels high
        blackPixel = (0,0,0) #the pixel of black
        whitePixel = (255,255,255) #the pixel of white
        for row in range(height):
                for col in range(width):
                        (r,g,b) = oldIm.getPixel(col,row) #return the pixel from col, and row
                        aveRGB = (r+g+b)//3 #convert to grayscale
                        if aveRGB < 128: #aveRGB less than threshold value 128
                                newim.setPixel(col,row,blackPixel) #set pixel to black
                        else:
                                newim.setPixel(col,row,whitePixel) #set pixel to white
        newim.setPosition(width+1,0) #position the newim next to the oldIm, so add the width by 1
        newim.draw(myImWin) #draw the newim in the myImWin image window
        myImWin.exitOnClick() #click on the window to exit

>>> blackandwhite("LutherBell.jpg")
>>>
```

PPiC 6.17 Sepia tone is a brownish color that was used for photographs in times past. The formula for creating a sepia tone is as follows:

$$newR = (R \times 0.393 + G \times 0.769 + B \times 0.189)$$
$$newG = (R \times 0.349 + G \times 0.686 + B \times 0.168)$$
$$newB = (R \times 0.272 + G \times 0.534 + B \times 0.131)$$

Write an RGB function to convert a pixel to sepia tone. *Hint:* Remember that RGB values must be integers between 0 and 255.

```
>>> from cImage import *
>>> def sepiatone(pixel): #define a function sepiatone with parameter pixel
        r = pixel.getRed() #return the red component intensity
        g = pixel.getGreen() #return the green component intensity
        b = pixel.getBlue() #return the blue component intensity
        newR = int(r*0.393+g*0.769+b*0.189) #calculate the new red intensity
        newG = int(r*0.349+g*0.686+b*0.168) #calculate the new green intensity
        newB = int(r*0.272+g*0.534+b*0.131) #calculate the new blue intensity
        if newR > 255: #Intensities will range from a minimum of 0 to a maximum of 255, so cannot exceed 255
                newR = 255
        if newG > 255:
                newG = 255
        if newB > 255:
                newB = 255
        newPixel = Pixel(newR,newG,newB) #create a new pixel with newR, newG, newB we just created
        return newPixel

>>> sepiatone(Pixel(34,128,75))
(125, 112, 87)
>>>
```

PPiC 6.19 Import the `turtle` module and find out the names defined.

```
>>> from turtle import *
>>> dir()
['Pen', 'RawPen', 'RawTurtle', 'Screen', 'ScrolledCanvas', 'Shape', 'Terminator', 'Turtl
e', 'TurtleScreen', 'Vec2D', '__builtins__', '__doc__', '__loader__', '__name__', '__pac
kage__', '__spec__', 'addshape', 'back', 'backward', 'begin_fill', 'begin_poly', 'bgcolo
r', 'bgpic', 'bk', 'bye', 'circle', 'clear', 'clearscreen', 'clearstamp', 'clearstamps',
 'clone', 'color', 'colormode', 'degrees', 'delay', 'distance', 'done', 'dot', 'down', '
end_fill', 'end_poly', 'exitonclick', 'fd', 'fillcolor', 'filling', 'forward', 'get_poly
', 'get_shapepoly', 'getcanvas', 'getpen', 'getscreen', 'getshapes', 'getturtle', 'goto'
, 'heading', 'hideturtle', 'home', 'ht', 'isdown', 'isvisible', 'left', 'listen', 'lt',
'mainloop', 'mode', 'numinput', 'onclick', 'ondrag', 'onkey', 'onkeypress', 'onkeyreleas
e', 'onrelease', 'onscreenclick', 'ontimer', 'pd', 'pen', 'pencolor', 'pendown', 'pensiz
e', 'penup', 'pos', 'position', 'pu', 'radians', 'register_shape', 'reset', 'resetscreen
', 'resizemode', 'right', 'rt', 'screensize', 'seth', 'setheading', 'setpos', 'setpositi
on', 'settiltangle', 'setundobuffer', 'setup', 'setworldcoordinates', 'setx', 'sety', 's
hape', 'shapesize', 'shapetransform', 'shearfactor', 'showturtle', 'speed', 'st', 'stamp
', 'textinput', 'tilt', 'tiltangle', 'title', 'towards', 'tracer', 'turtles', 'turtlesiz
e', 'undo', 'undobufferentries', 'up', 'update', 'width', 'window_height', 'window_width
', 'write', 'write_docstringdict', 'xcor', 'ycor']
```

PS 4.11.1 Write a recursive function to compute the factorial of a number.

```
>>> def factorial(x):
        if x==0 or x==1: #the factorial of 0 and 1 is 1
                return x
        else:
                return x*factorial(x-1) #factorial equals x*(x-1)*(x-1-1)...
        if x<0:
                print("No factorial for negative number.")

>>> factorial(2)
2
```