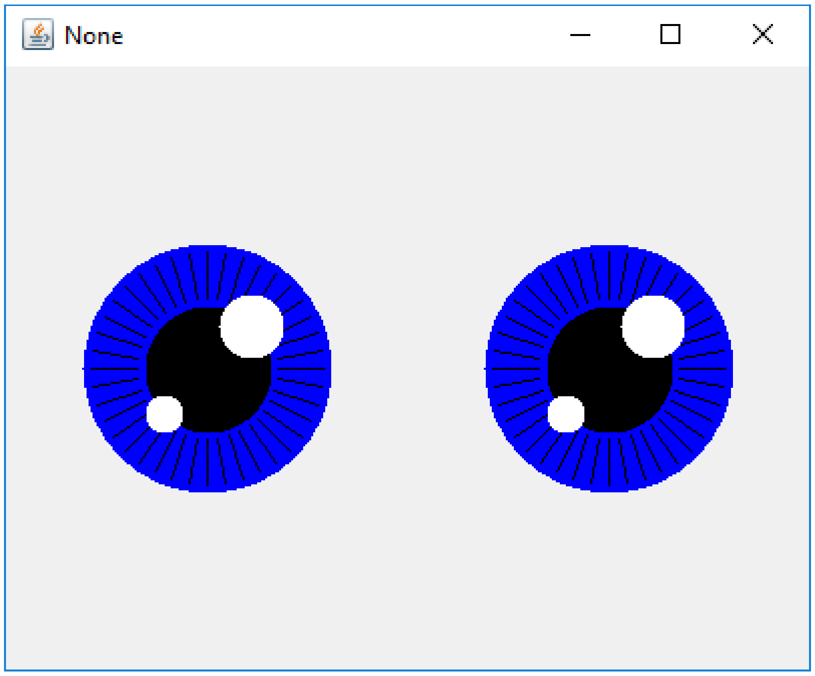
# CMPSCI 119
# LAB #2 – Anime Eyes
# Professor William T. Verts

The goal of this Python programming assignment is to write your own code inside a provided program framework, with some new graphical and mathematical considerations. You must create a pair of staring Anime Eyes, which have an iris, a pupil, a starburst ring of dashes in the iris, and a pair of highlights. Your code must adapt to the width and height of the canvas, so that the eyes perfectly fit the screen and do not overlap any edges, for any arbitrary canvas size. This is shown as follows:



In the JES environment, type in the following program code *exactly as you see it on the next two pages, including the comments*, and save the program as `Lab2.py` in your Python folder. Alternatively, you can download the framework code from the class site.

Where you see my name in the comment code, replace it with *your own name*. As always, be very careful about indentation and capitalization. The blank space in each function is where you will be writing your own code.

```
# William T. Verts - Lab #2 - Anime Eyes

#---------------------------------------------------------------
# This function paints a filled circle of radius R
# and color NewColor on the Canvas, using <Xc,Yc> as
# the coordinates of the center of the circle.
# Xc, Yc, and R are all allowed to be floats.
#---------------------------------------------------------------

def addCircleFilled (Canvas,Xc,Yc,R,NewColor=black):


    return

#---------------------------------------------------------------
# This function paints a bunch of black lines radially around
# <Xc,Yc> where R1 and R2 are the inner and outer radii and
# Segments indicates the number of lines.  Xc, Yc, R1, and R2
# are all allowed to be floats.
#---------------------------------------------------------------

def addStarburst (Canvas,Xc,Yc,R1,R2,Segments):


    return

#---------------------------------------------------------------
# This function paints ONE anime eyeball on the Canvas,
# centered at <Xc,Yc>.  The color of the iris is NewColor, the
# pupil is black, and the highlights are white.  The sizes and
# positions of the iris, pupil, starburst, and highlights are
# derived from center point <Xc,Yc> and the radius R of the
# iris.
#---------------------------------------------------------------

def Eyeball (Canvas,Xc,Yc,R,NewColor):


    return

#---------------------------------------------------------------
# This function CALLS Eyeball TWICE (once for each eye) to
# place the pair of eyeballs on the screen.  Stare must first
# determine the correct radius and center positions for each
# eye before calling Eyeball at all.
#---------------------------------------------------------------

def Stare (Canvas,NewColor):


    return
```

```
#---------------------------------------------------------------
# Prompt the user for a color.  You MAY NOT change ANY code
# in the requestColor function!
#---------------------------------------------------------------

def requestColor(Message):
    while True:
        S = requestString(Message)
        S = S.lower()
        if (S == "red"):     return red
        if (S == "green"):   return green
        if (S == "blue"):    return blue
        if (S == "cyan"):    return cyan
        if (S == "magenta"): return magenta
        if (S == "yellow"):  return yellow

#---------------------------------------------------------------
# This function establishes the size of the canvas and the
# color of the anime eyes.  You MAY NOT change ANY code in
# the Main function!
#---------------------------------------------------------------

def Main():
    W = requestIntegerInRange("Enter Width", 100, 1000)
    H = requestIntegerInRange("Enter Height", 100, 1000)
    C = requestColor("Enter Iris Color")
    Canvas = makeEmptyPicture(W,H,makeColor(240))
    Stare(Canvas,C)
    show(Canvas)
    return
```

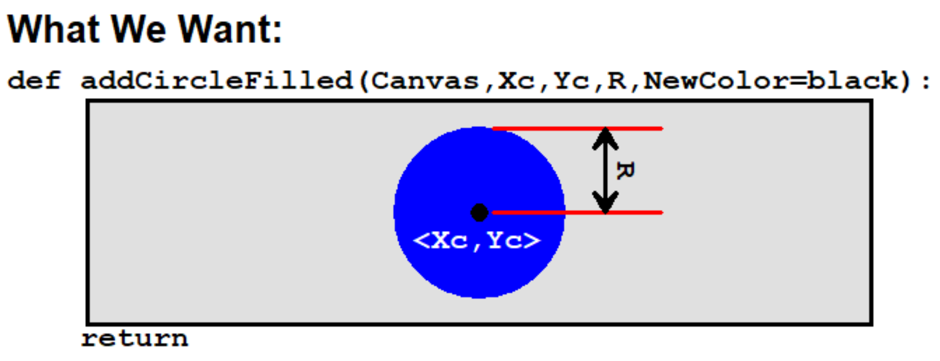
In JES click the Load Program button. At the `>>>` prompt, type `Main()` with the parentheses and press [Enter↵]. The program should run as it is shown here, but should not do anything except put a blank canvas on screen with dimensions selected by the user. Fix any syntax errors or other mistakes. We will not change `Main`, but we will fill in code for the other four functions.

## Function `addCircleFilled`

In the blank space in the `addCircleFilled` function, ***write new code*** to paint a filled circle with the center of the circle at location <`Xc`,`Yc`> with radius `R` and color `NewColor`. You must use the JES function `addOvalFilled`, which has the following formal parameters:

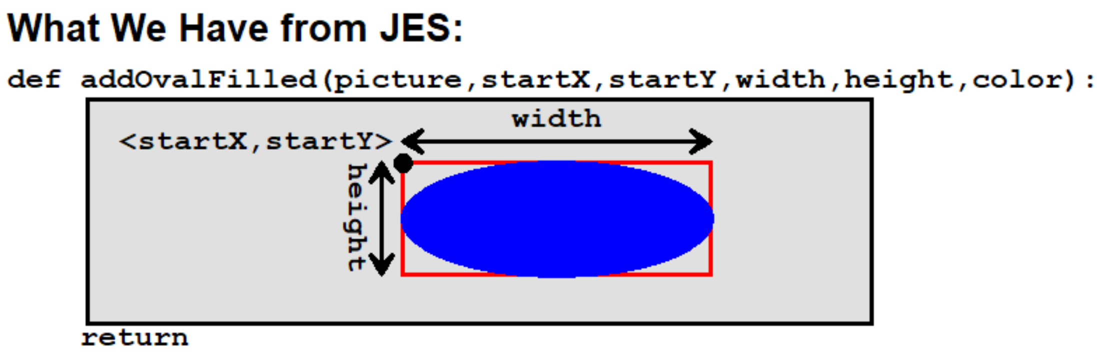
```
addOvalFilled(picture, startX, startY, width, height, color)
```

In this JES function, `startX` and `startY` are the coordinates of the upper-left corner of an oval (not a circle), and `width` and `height` describe the size of the oval's bounding box. We want to use those parameters to create a circle, as shown below:



**What We Want:**

```
def addCircleFilled(Canvas,Xc,Yc,R,NewColor=black):
    return
```



**What We Have from JES:**

```
def addOvalFilled(picture,startX,startY,width,height,color):
    return
```

You must figure out how to convert the `Xc`, `Yc`, and `R` of our new `addCircleFilled` function into the `startX`, `startY`, `width`, and `height` that can be used to call the `addOvalFilled` function of JES. (We've done this in class.)

**HINT:** Remember that `addOvalFilled` requires integer parameters; but many of the values calculated in this program are passed to `addCircleFilled` as floats and must be rounded and converted to integers before they can be used in the call to `addOvalFilled`. You are allowed to type in and use the `INT` function described in the Companion book.

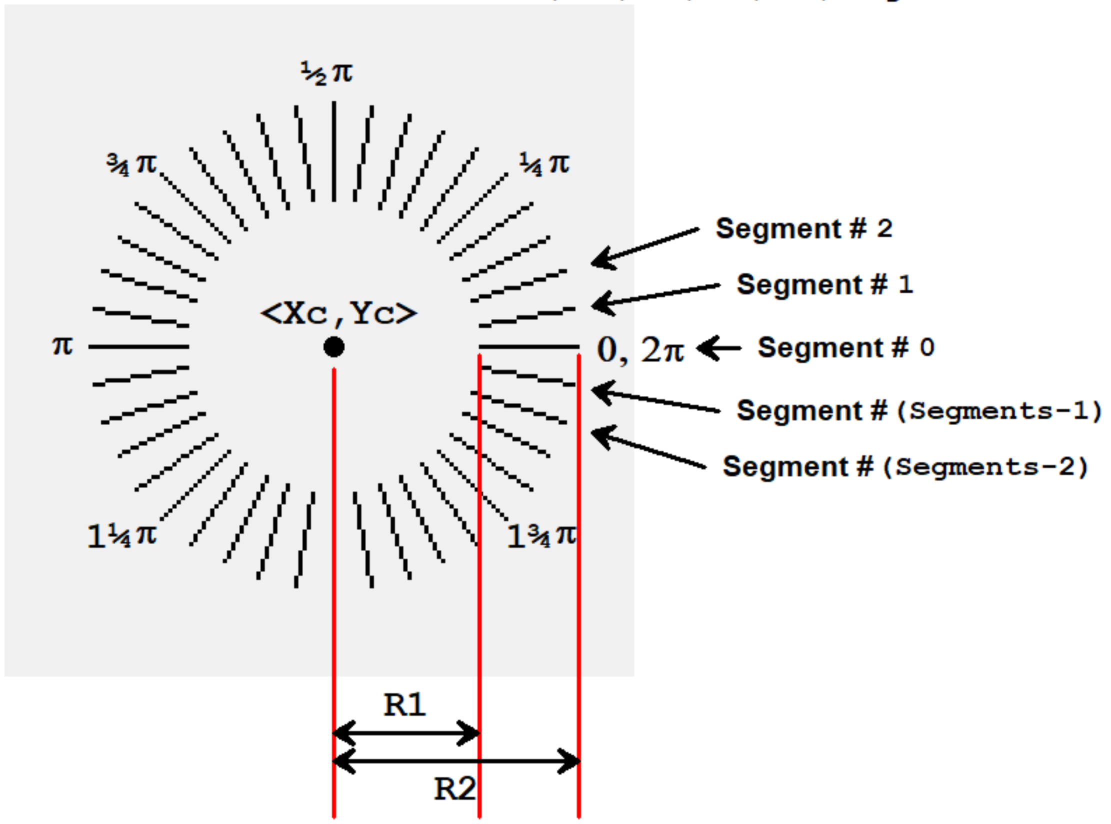
## Function `addStarburst`

In the blank space in the `addStarburst` function, ***write new code*** to plot a bunch of radial lines around the center `<Xc,Yc>`. The number of lines is in parameter `Segments`, and is always a positive integer. For each line, you need to compute its angle as a fraction of the distance around a circle, which is $2\pi$ radians. That is, the increment (how much the angle changes between any two lines) is computed by dividing $2\pi$ by the number of segments. Start with the angle equal to 0, and for each new line add the increment to the old angle to compute the new angle. Once you know the angle for a particular line, its endpoints can be computed with a little trigonometry:

```
X1 = Xc + R1 × cos(angle)
Y1 = Yc - R1 × sin(angle)
X2 = Xc + R2 × cos(angle)
Y2 = Yc - R2 × sin(angle)
```

Use the `addLine` function from JES to draw a line between `<X1,Y1>` and `<X2,Y2>`. Be careful about the data types of your numbers: the coordinates are floats, but `addLine` expects integers.

```
def addStarburst (Canvas,Xc,Yc,R1,R2,Segments):
```



## Function `Eyeball`

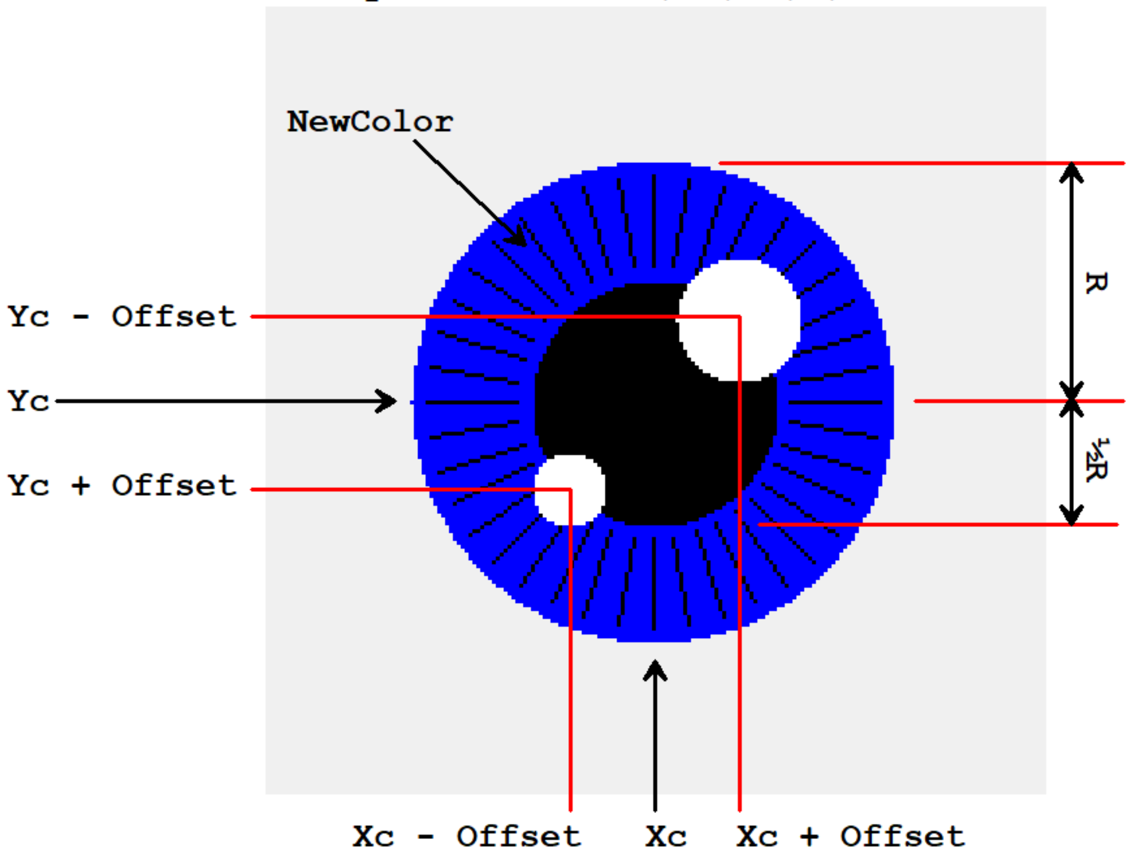
In the `Eyeball` function, *write new code* to paint a single anime eyeball on the canvas, given the center of the circle is at location <`Xc`,`Yc`>, with iris radius `R` and iris color `NewColor` (all of this is passed in through the parameter list). In `Eyeball`, you will need to call the `addCircleFilled` function four times, once for the iris, once for the pupil, and once for each of the two highlights.

The radius of the iris (the colored area) comes directly from parameter `R`. The radius of the pupil (the black center of the eye) is half the radius of the iris, and has the same center coordinates as the iris. The radius of the big highlight (the big white spot) is one-quarter that of the iris. The radius of the smaller highlight is one-seventh that of the iris. The centers of the highlights are located relative to the center of the eyeball at a distance offset in the X and Y directions by an amount calculated as follows:

$$\texttt{Offset} = \tfrac{1}{4} R \sqrt{2}$$

This will put the center of the highlights directly on the boundary between the pupil and the iris, at 45° angles relative to the center of the pupil.

**def Eyeball (Canvas,Xc,Yc,R,NewColor):**



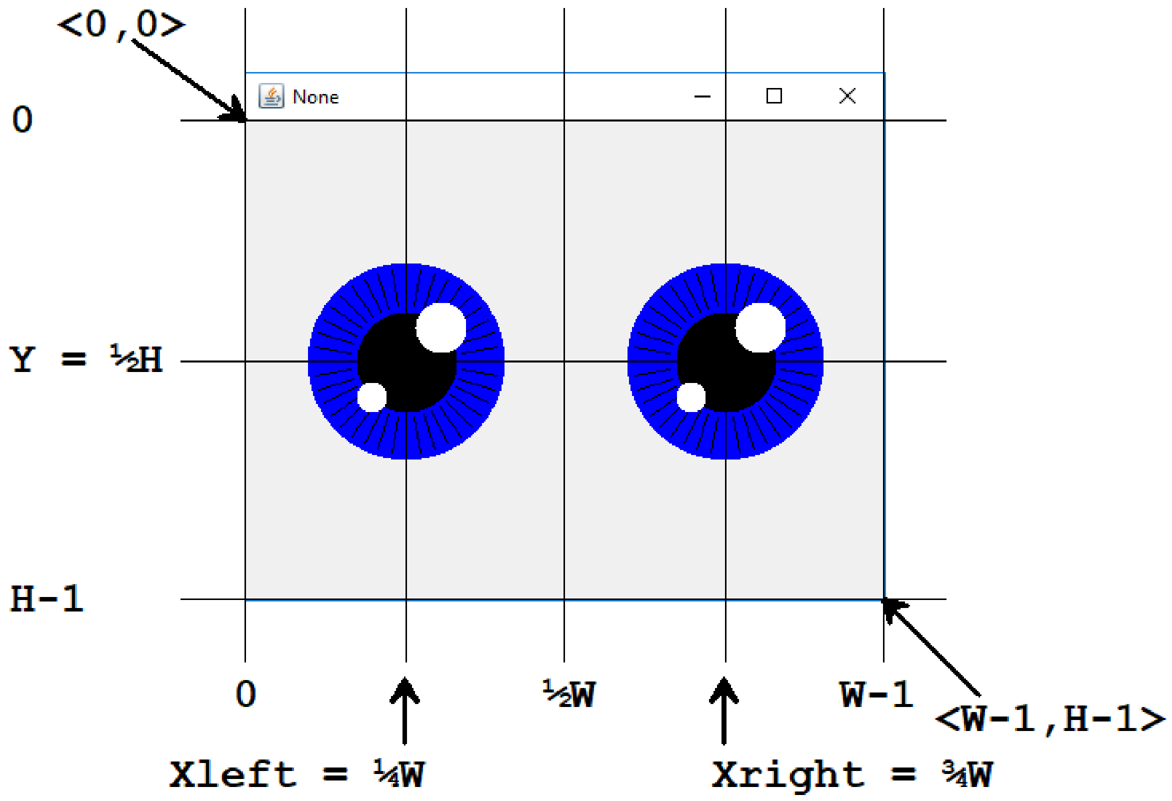
You will also have to call `addStarburst` at the appropriate time. The starburst has the same center coordinates as the pupil and the iris. The inner radius is ½R (the radius of the pupil) plus $\frac{1}{15}$TH of R, and the outer radius is R minus $\frac{1}{15}$TH of R. There are 40 segments in the starburst.

## Function `Stare`

In the `Stare` function, *write new code* to call `Eyeball` twice, once for the left eye and once for the right eye, with the correct radius and coordinates for each eye computed from the size of the `Canvas`. If the width and height of the Canvas are extracted into variables `W` and `H` as follows:

```
W = getWidth(Canvas)
H = getHeight(Canvas)
```

then all the other measurements needed to find the position of the eyeballs are computed as follows:



What remains for `Stare` to compute is the eyeball radius. The radius `R` is computed from the minimum (using Python's `min(…,…)` function) of `Xleft` and `Y`, times ⅔, minus 5 pixels. This may seem overly complicated, but doing it this way guarantees that the eyes will always fit onto the screen.

## Finishing Up

That is enough information for you to figure out how to fill out the four functions and run the program. Once complete, try using different sizes for the screen (the eyeball sizes should track this correctly).

When you are done, turn in your assignment through the on-line code submitter Web form.