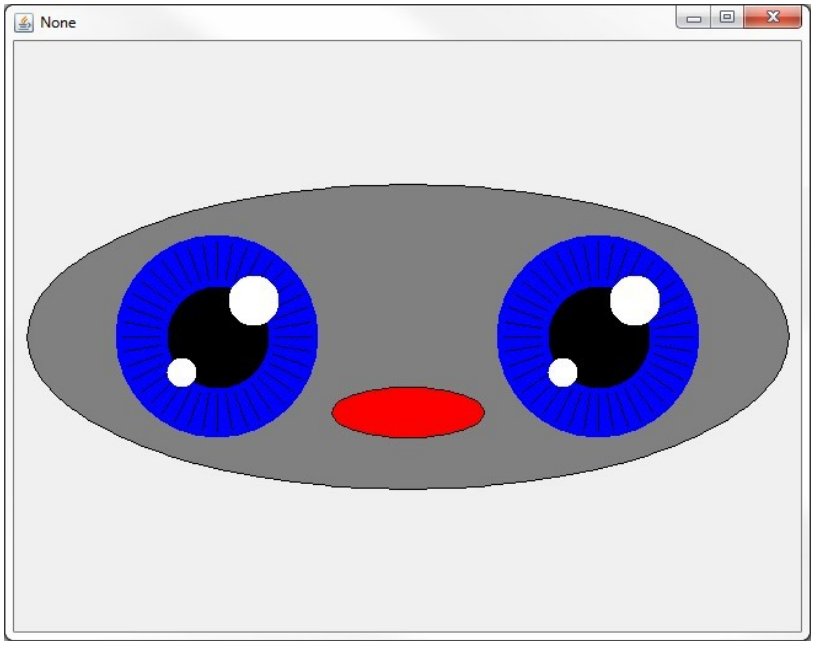
CMPSCI 119
LAB #2 – Greebles / Anime Eyes
Professor William T. Verts

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 little creature called a Greeble containing a head, mouth, and a pair of staring eyes (each of which has 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 Greeble perfectly fits the screen and does 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 few 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 - Greebles / Anime Eyes

#---------------------------------------------------------------
# Helper function to round coordinates to the nearest pixel.
#---------------------------------------------------------------

def INT(N):
    return int(round(N))

#---------------------------------------------------------------
# These functions paint ellipses or filled ellipses at center
# <Xc,Yc> with X-radius Xr and Y-radius Yr and color NewColor,
# as well as filled circles of radius R.
# Xc, Yc, Xr, Yr, and R may all be floats.
#---------------------------------------------------------------

def addEllipse (Canvas,Xc,Yc,Xr,Yr,NewColor=black):


    return

#---------------------------------------------------------------

def addEllipseFilled (Canvas,Xc,Yc,Xr,Yr,NewColor=black):


    return

#---------------------------------------------------------------

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


    return

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

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, and highlights are derived
# from center point <Xc,Yc> and the radius of the iris R.
#---------------------------------------------------------------

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


    return

#---------------------------------------------------------------
# This function paints a Greeble on the Canvas at center
# <Xc,Yc> with Xradius indicating the size of the Greeble
# (everything else is computed from those three values).
# The Greeble has ellipses for the body and mouth, and calls
# addEyeball TWICE (once for each eye).
#---------------------------------------------------------------

def addGreeble (Canvas,Xc,Yc,Xradius,NewColor):


    return

#---------------------------------------------------------------
# This function computed the optimal X radius for the Greeble
# based on the size of the Canvas, then plots the Greeble at
# the center of the screen with that X radius.
#---------------------------------------------------------------

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 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 a Color")
    Canvas = makeEmptyPicture(W,H,makeColor(240))
    Stare(Canvas,C)
    show(Canvas)
    return

#--------------------------------------------------------------
# Test programs to exercise the various functions as they are
# being developed.
#--------------------------------------------------------------

def TestEllipse ():
    Canvas = makeEmptyPicture(400,400)
    addEllipse(Canvas,getWidth(Canvas)/2,getHeight(Canvas)/2,150,50,blue)
    show(Canvas)
    return

def TestEllipseFilled ():
    Canvas = makeEmptyPicture(400,400)
    addEllipseFilled(Canvas,getWidth(Canvas)/2,getHeight(Canvas)/2,150,50,blue)
    show(Canvas)
    return

def TestCircleFilled ():
    Canvas = makeEmptyPicture(400,400)
    addCircleFilled(Canvas,getWidth(Canvas)/2,getHeight(Canvas)/2,150,blue)
    show(Canvas)
    return

def TestStarburst ():
    Canvas = makeEmptyPicture(400,400)
    addStarburst(Canvas,getWidth(Canvas)/2,getHeight(Canvas)/2,150,100,40)
    show(Canvas)
    return

def TestEyeball ():
    Canvas = makeEmptyPicture(400,400)
    addEyeball(Canvas,getWidth(Canvas)/2,getHeight(Canvas)/2,150,blue)
    show(Canvas)
    return
```

Quick Test

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` or `requestColor`, but we will fill in code for the other functions.

Notice that below `Main` are special functions that test five of the functions that you are to complete. Make no changes to these functions. These are provided to you to aid in the debugging process.

Functions that draw ellipses and circles

```
def addEllipse       (Canvas,Xc,Yc,Xr,Yr,NewColor=black):
def addEllipseFilled (Canvas,Xc,Yc,Xr,Yr,NewColor=black):
def addCircleFilled  (Canvas,Xc,Yc,R,NewColor=black):
```

In the blank spaces in the `addEllipse`, `addEllipseFilled`, and `addCircleFilled` functions, ***write new code*** to paint the appropriate objects. In each function the center of the object is at location <`Xc`,`Yc`> and the object is painted with color `NewColor`. In the ellipse functions parameter `Xr` is the radius in the X direction, and `Yr` is the radius in the Y direction. In the circle function `R` is the radius.

All these parameters (`Xc`, `Yc`, `Xr`, `Yr`, and `R`) are allowed to be floats, so your code must take this into consideration when calling JES functions `addOval` or `addOvalFilled`, which require int parameters and will crash if given a float; the `INT` function is provided for your convenience to address this issue.

The `addOval` and `addOvalFilled` functions have the following formal parameters:

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

In these JES functions, `startX` and `startY` are the coordinates of the upper-left corner of the bounding box of an oval, and `width` and `height` describe the size of the bounding box. You must convert the center and radius parameters given to the ellipse functions into the upper-left corner, width, and height values needed by the oval functions.

The `addCircleFilled` function should call the `addEllipseFilled` function instead of calling `addOvalFilled` directly.

Test your completed functions with the provided `TestEllipse`, `TestEllipseFilled`, and `TestCircleFilled` functions.

Do not proceed until your three functions work correctly.

Function `addStarburst`

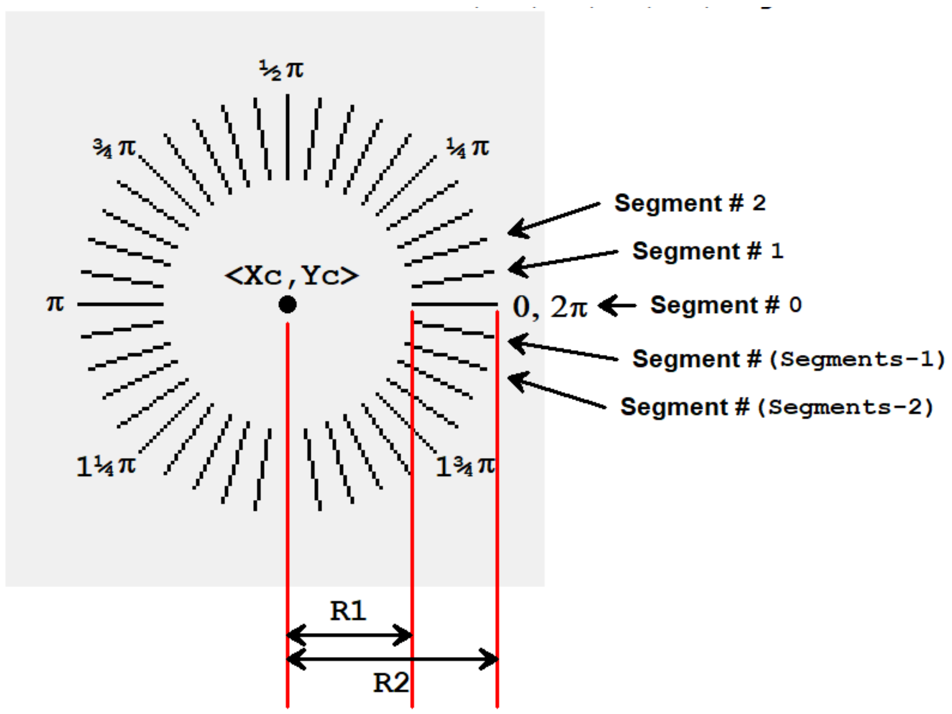
```
def addStarburst (Canvas,Xc,Yc,R1,R2,Segments):
```

NOTE: You can defer completing `addStarburst` until after `addEyeball` is working. Indeed, you can complete this function last, after everything else is working entirely.

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 2π radians. That is, the increment (how much the angle changes between any two lines) is computed by dividing 2π 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.



Use the provided `TestStarburst` function to test your code.

Function `addEyeball`

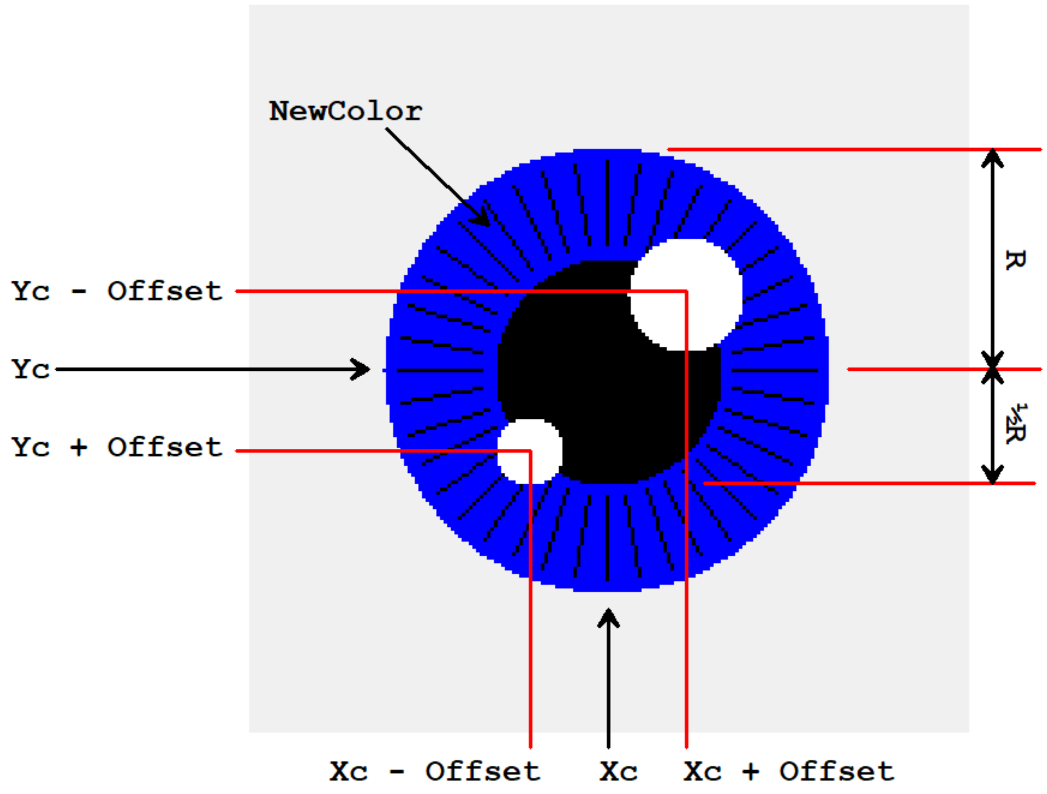
```
def addEyeball (Canvas,Xc,Yc,R,NewColor):
```

In the `addEyeball` 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:

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

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.



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 $^{1}/_{15}$TH of R, and the outer radius is R minus $^{1}/_{15}$TH of R. There are 40 segments in the starburst.

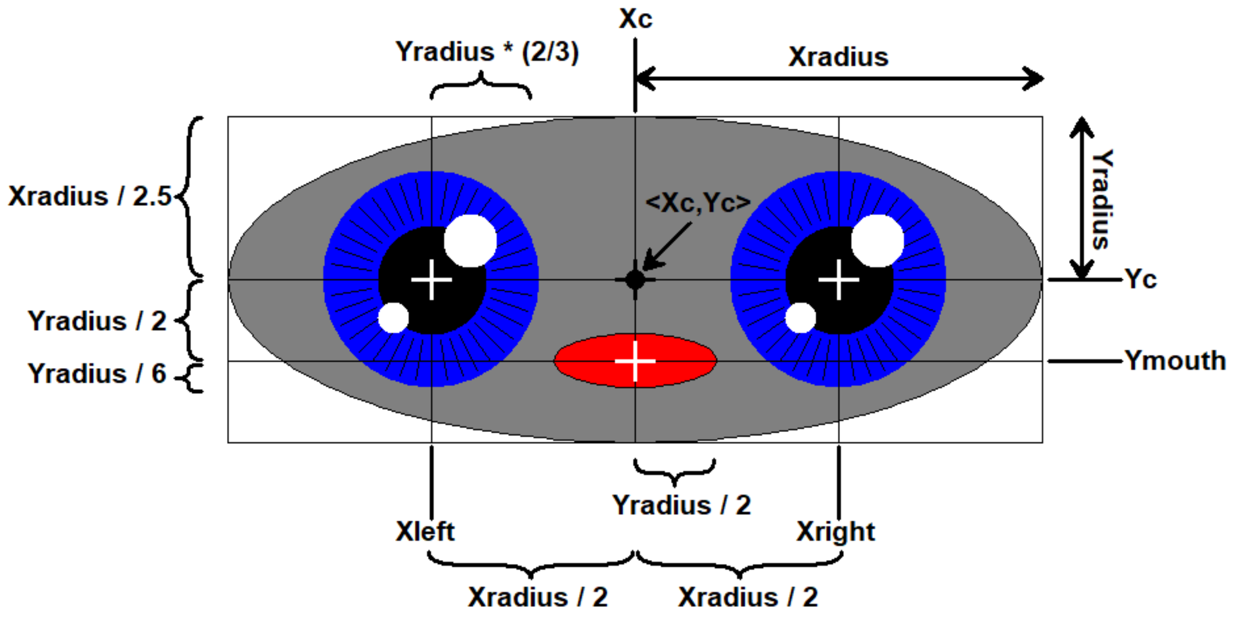
Use the `TestEyeball` function to test your code.

Function `addGreeble`

```
def addGreeble (Canvas,Xc,Yc,Xradius,NewColor):
```

The `addGreeble` function is given only three numbers to determine the location and size of the Greeble: the `Xc` and `Yc` parameters determine the center of the Greeble, and `Xradius` determines the radius in the X direction. All other information can be derived from those three values:

> The `Yradius` of the head is ⅖ of `Xradius`,
> The center of the mouth is ½ of `Yradius` below the center of the Greeble,
> The x radius of the mouth is ½ of `Yradius`,
> The y radius of the mouth is ⅙ of `Yradius`,
> The center of the left eyeball is ½ `Xradius` left of the center of the Greeble,
> The center of the right eyeball is ½ `Xradius` right of the center of the Greeble,
> The radius of the eyeballs is ⅔ of `Yradius`.



In the `addGreeble` function, *write new code* to paint the head, the mouth, and two eyeballs. The head is a gray filled ellipse followed by a black ellipse of the same size and location. The mouth is a red filled ellipse followed by a black ellipse of the same size and location. The eyes are drawn by two calls to `addEyeball`, and the iris color of the eyeballs is determined by `NewColor`.

Function `Stare`

```
def Stare (Canvas,NewColor):
```

In the `Stare` function, ***write new code*** to figure out the optimum X radius size of the Greeble based on the size of the canvas, and then call `addGreeble` to paint the Greeble in the center of the canvas with the proper X radius value.

First compute the center coordinates of the canvas `<Xmid,Ymid>`. That will be the location of the Greeble. The optimum `Xradius` will be whichever is smaller: `Xmid-10` or `(Ymid-10)*2.5` (the constant 10 is to allow some empty space between the Greeble and the edges of the canvas, and the 2.5 is the ratio of the X radius to the Y radius).

Call `addGreeble` with `Xmid` and `Ymid` as the center location, the `Xradius` that was just computed, and the value of `NewColor` that was passed in from `Main`.

Finishing Up

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

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