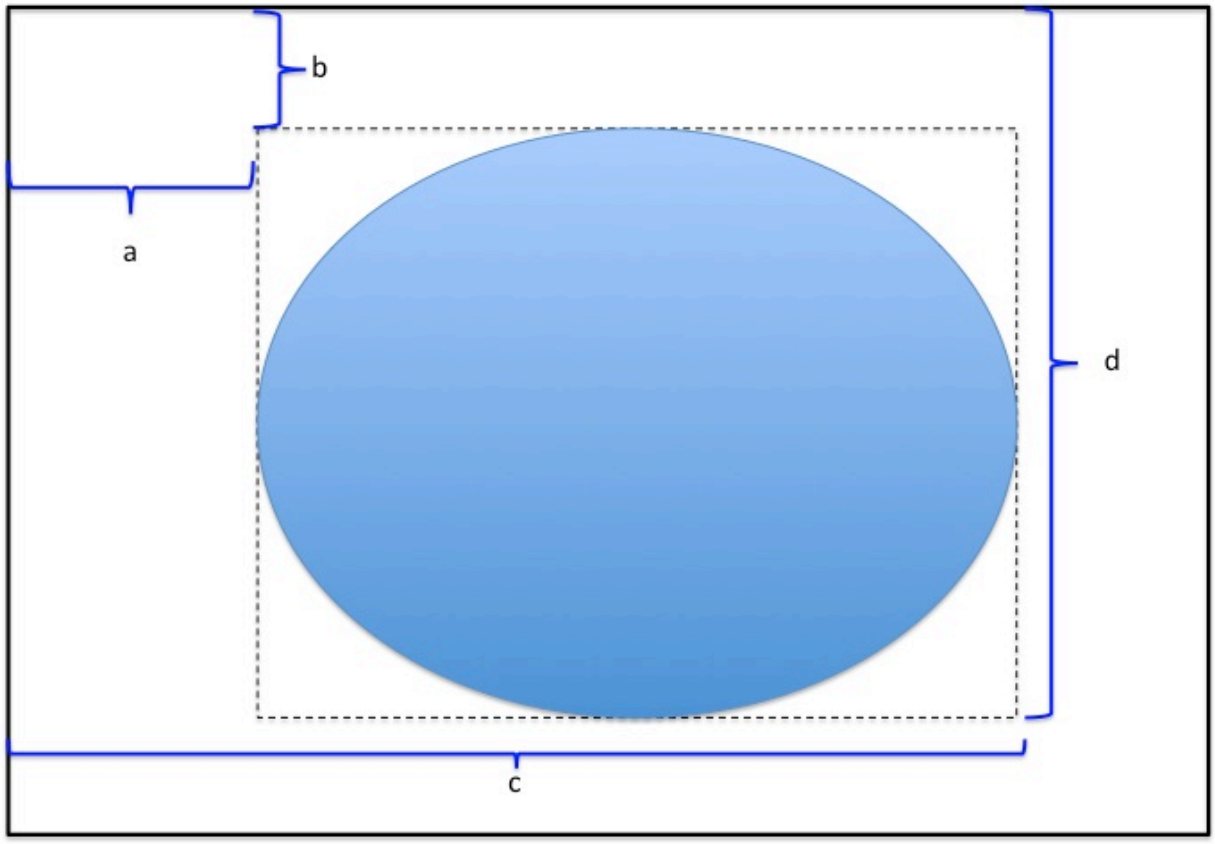
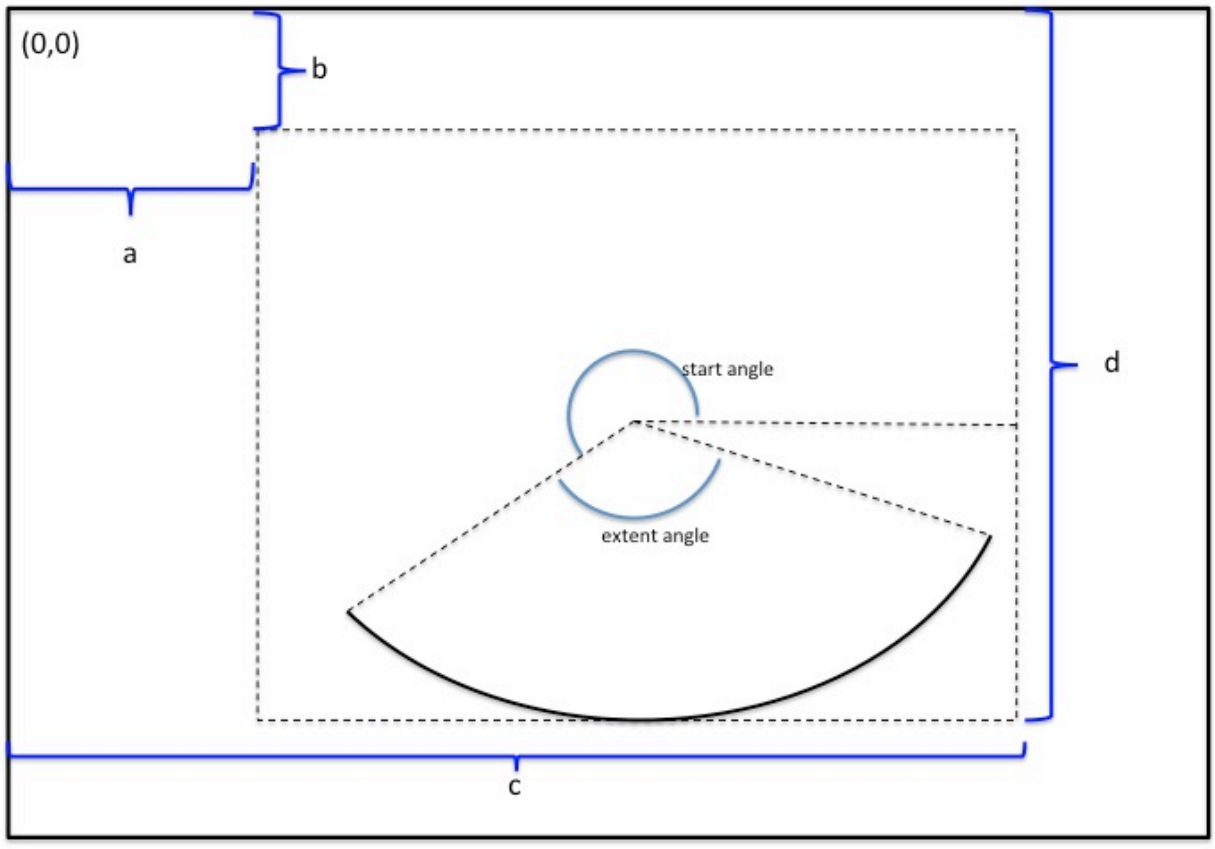
Notes for Part II of PSet 2: *manyfaces*

The Python installation comes with a graphics library called `Tkinter`. To complete the last part of PSet 2, we should first understand how the graphics coordinate system works, and what the four drawing statements in the program are doing. The 'canvas' on which this program draws is 500 units wide and 500 units high. The coordinates of the top left corner of the canvas are (0, 0), and of the bottom right (500, 500). Note that unlike the standard coordinate geometry we use in math classes, the y-coordinate of a point increases *downward*. The function call `ps2_canvas.create_oval(a, b, c, d,....)` inscribes an *ellipse* inside a *rectangle* (which is not drawn) whose top left corner has coordinates (a, b) and whose bottom right corner has coordinates (c, d). If you don't specify a fill color, only the black outline of the ellipse will be drawn.



The posted program draws one large yellow-filled oval for the head, and two tiny black-filled ovals for the eyes.

The function call `ps2_canvas.create_arc(125, 125, 175, 175, start=225, extent=90, style=Tkinter.ARC)` which creates the smile, is slightly more complicated: The first four arguments again specify the corners of a bounding rectangle of an ellipse; the next two specify what portion of the ellipse to draw. In our program, the arc starts at an angle 225 degrees counterclockwise from the positive x-direction, and extends counterclockwise for 90 degrees. To complete this part of PSet 2, we need to change the smile to a frown to show sadness, and change the smile to straight lips to show pensiveness. Please be as creative as possible and change any other features of the face if you like to; you have *carte blanche*. The function for drawing a rectangle is `ps2_canvas.create_rectangle`, with arguments that have the same meaning as those of `ps2_canvas.create_oval`.



You can also try `ps2_canvas.create_line(a, b, c, d)` which draws a line segment between the points (a, b) and (c, d).