

# 2D Graphics

Michael C. Hackett

Assistant Professor, Computer Science

Community  
College  
*of* Philadelphia

# Lecture Topics

- 2-D Graphics (Python)

- Canvases
- Drawing Lines
- Drawing Rectangles
- Drawing Ovals
- Drawing Arcs
- Drawing Polygons
- Drawing Text

- 2-D Graphics (Java)

- Canvas and Graphics objects
- Drawing Lines
- Drawing Rectangles
- Drawing Ovals
- Drawing Arcs
- Drawing Polygons
- Drawing Text

# Colors/Fonts

• Global Variable Names	—	Brown
• Local Variable Names	—	Lt Blue
• Literals	—	Blue
• Keywords	—	Orange
• Operators/Punctuation	—	Black
• Functions	—	Purple
• Parameters	—	Gold
• Comments	—	Gray
• Modules	—	Pink
• Object/Class Names	—	Green

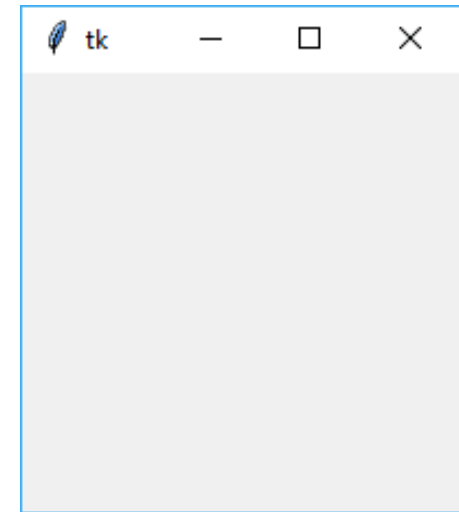
Source Code — **Consolas**  
Output — Courier New

# The tkinter module

- Python does not have built-in GUI capabilities.
- The tkinter module (which comes with Python) allows us to create graphical user interfaces.
  - Short for “Tk interface”
  - Other languages use the Tk framework for creating windowed applications.
- There are other GUI-development libraries for Python.

# Creating a Window

- A ***window*** is the general term for a rectangular container of graphical components.
- Windows are normally decorated with a title and minimize/maximize/close buttons.



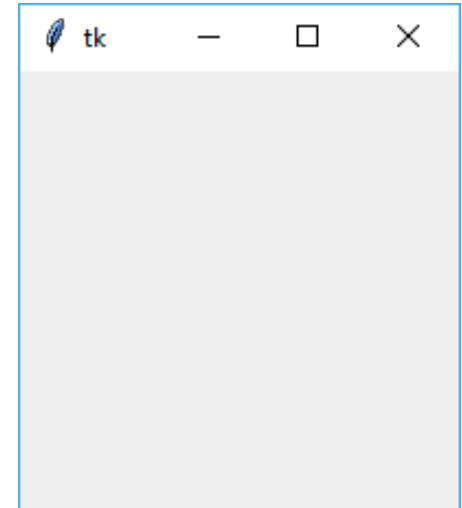
# Creating a Window

```
import tkinter

def main() :
    #Creates the window
    test_window = tkinter.Tk()

    #Enters the main loop, displaying the window
    #and waiting for events
    tkinter.mainloop()

main()
```



# Setting the Window's Title

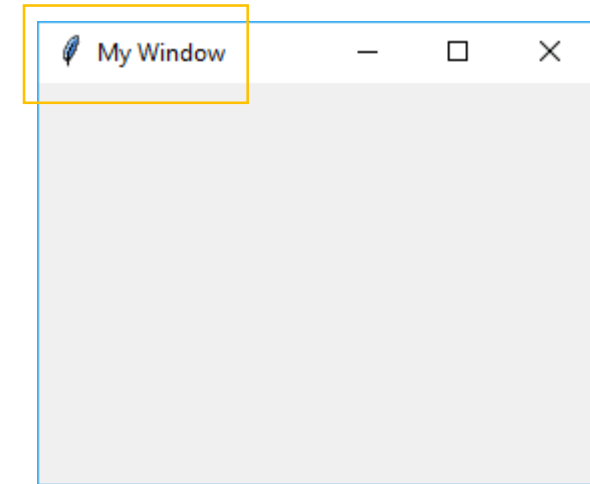
```
import tkinter

def main() :
    #Creates the window
    test_window = tkinter.Tk()

    #Sets the window's title
    test_window.wm_title("My Window")

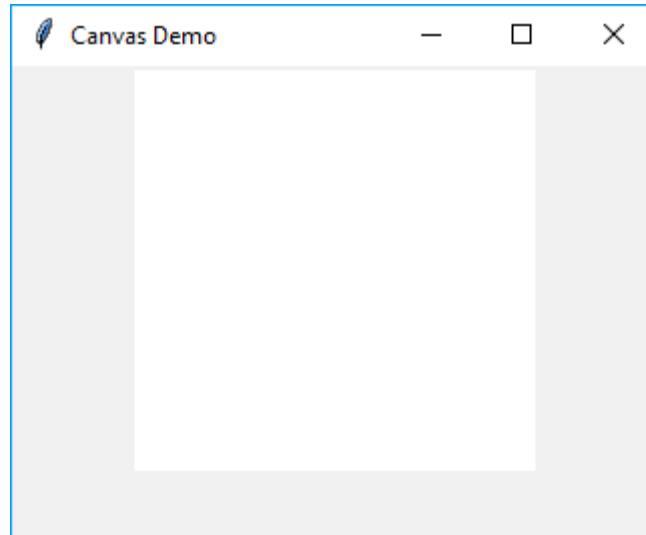
    #Enters the main loop, displaying the window
    #and waiting for events
    tkinter.mainloop()

main()
```



# Creating a Canvas

- A ***canvas*** is a graphical component that allows the drawing of lines and shapes.





# Creating a Canvas

- When creating a canvas, you must specify:
  - The window or frame it belongs to.
  - The canvas' width and height.
- You may optionally set the canvas' background color.
  - The background will be transparent if no background is set.

```
canvas = tkinter.Canvas(test_window,  
                          width=200,  
                          height=200,  
                          background="white")
```

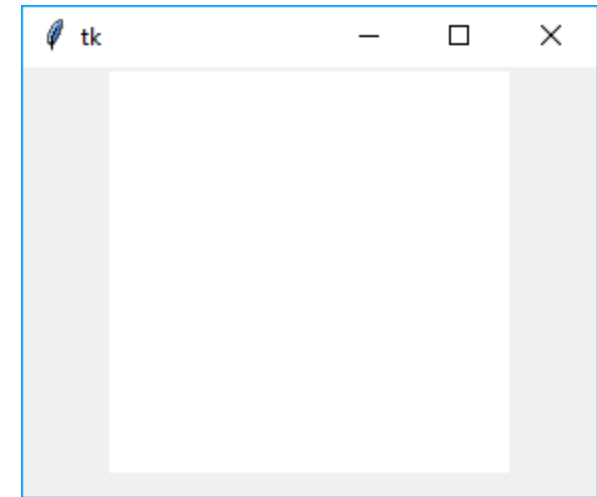
# Creating a Canvas

```
import tkinter

def main() :
    #Creates the window
    test_window = tkinter.Tk()
    #Creates the Canvas widget.
    canvas = tkinter.Canvas(test_window,
                             width=200,
                             height=200,
                             background="white")

    #Packs the canvas onto the window
    canvas.pack()
    #Enters the main loop, displaying the window
    tkinter.mainloop()

main()
```



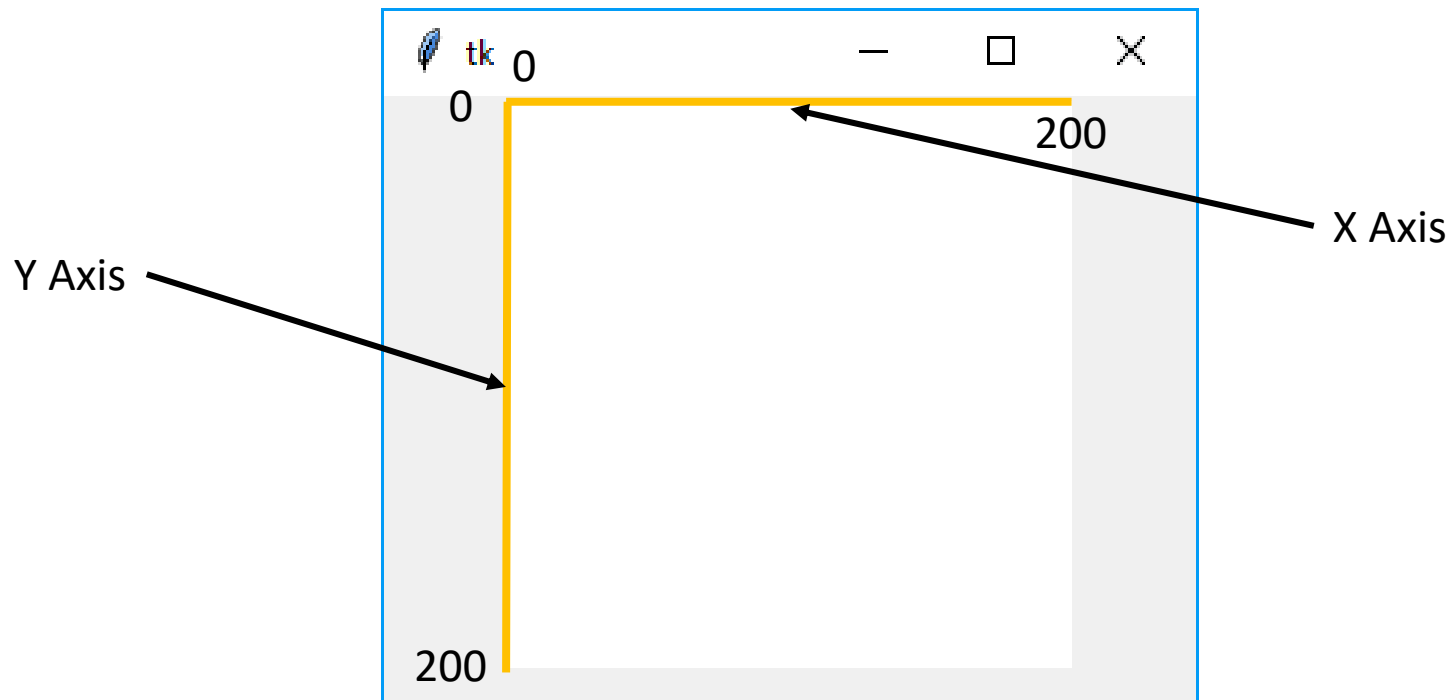
# Tk Named Colors

snow	deep sky blue	gold	seashell3	SlateBlue2	LightBlue3	SpringGreen2	DarkGoldenrod1	brown4	pink3	purple1	gray26	gray64
ghost white	sky blue	light goldenrod	seashell4	SlateBlue3	LightBlue4	SpringGreen3	DarkGoldenrod2	salmon1	pink4	purple2	gray27	gray65
white smoke	light sky blue	goldenrod	AntiqueWhite1	SlateBlue4	LightCyan2	SpringGreen4	DarkGoldenrod3	salmon2	LightPink1	purple3	gray28	gray66
gainsboro	steel blue	dark goldenrod	AntiqueWhite2	RoyalBlue1	LightCyan3	green2	DarkGoldenrod4	salmon3	LightPink2	purple4	gray29	gray67
floral white	light steel blue	rosy brown	AntiqueWhite3	RoyalBlue2	LightCyan4	green3	RosyBrown1	salmon4	LightPink3	MediumPurple1	gray30	gray68
old lace	light blue	indian red	AntiqueWhite4	RoyalBlue3	PaleTurquoise1	green4	RosyBrown2	LightSalmon2	LightPink4	MediumPurple2	gray31	gray69
linen	powder blue	saddle brown	bisque2	RoyalBlue4	PaleTurquoise2	chartreuse2	RosyBrown3	LightSalmon3	PaleVioletRed1	MediumPurple3	gray32	gray70
antique white	pale turquoise	sandy brown	bisque3	blue2	PaleTurquoise3	chartreuse3	RosyBrown4	LightSalmon4	PaleVioletRed2	MediumPurple4	gray33	gray71
papaya whip	dark turquoise	dark salmon	bisque4	blue4	PaleTurquoise4	chartreuse4	IndianRed1	orange2	PaleVioletRed3	thistle1	gray34	gray72
blanched almond	medium turquoise	salmon	PeachPuff2	DodgerBlue2	CadetBlue1	OliveDrab1	IndianRed2	orange3	PaleVioletRed4	thistle2	gray35	gray73
bisque	turquoise	light salmon	PeachPuff3	DodgerBlue3	CadetBlue2	OliveDrab2	IndianRed3	orange4	maroon1	thistle3	gray36	gray74
peach puff	cyan	orange	PeachPuff4	DodgerBlue4	CadetBlue3	OliveDrab4	IndianRed4	DarkOrange1	maroon2	thistle4	gray37	gray75
navajo white	light cyan	dark orange	NavajoWhite2	SteelBlue1	CadetBlue4	DarkOliveGreen1	sienna1	DarkOrange2	maroon3		gray38	gray76
lemon chiffon	cadet blue	coral	NavajoWhite3	SteelBlue2	turquoise1	DarkOliveGreen2	sienna2	DarkOrange3	maroon4		gray39	gray77
mint cream	medium aquamarine	light coral	NavajoWhite4	SteelBlue3	turquoise2	DarkOliveGreen3	sienna3	DarkOrange4	VioletRed1	gray40	gray78	
azure	aquamarine	tomato	LemonChiffon2	SteelBlue4	turquoise3	DarkOliveGreen4	sienna4	coral1	VioletRed2	gray41	gray79	
alice blue	dark green	orange red	LemonChiffon3	DeepSkyBlue2	turquoise4	khaki1	burlywood1	coral2	VioletRed3	gray42	gray80	
lavender	dark olive green	red	LemonChiffon4	DeepSkyBlue3	cyan2	khaki2	burlywood2	coral3	VioletRed4	gray43	gray81	
lavender blush	dark sea green	hot pink	cornsilk2	DeepSkyBlue4	cyan3	khaki3	burlywood3	coral4	magenta2	gray44	gray82	
misty rose	sea green	deep pink	cornsilk3	SkyBlue1	cyan4	khaki4	burlywood4	tomato2	magenta3	gray45	gray83	
dark slate gray	medium sea green	pink	cornsilk4	SkyBlue2	DarkSlateGray1	LightGoldenrod1	wheat1	tomato3	magenta4	gray46	gray84	
dim gray	light sea green	light pink	ivory2	SkyBlue3	DarkSlateGray2	LightGoldenrod2	wheat2	tomato4	orchid1	gray47	gray85	
slate gray	pale green	pale violet red	ivory3	SkyBlue4	DarkSlateGray3	LightGoldenrod3	wheat3	OrangeRed2	orchid2	gray48	gray86	
light slate gray	spring green	maroon	ivory4	LightSkyBlue1	DarkSlateGray4	LightGoldenrod4	wheat4	OrangeRed3	orchid3	gray49	gray87	
gray	lawn green	medium violet red	honeydew2	LightSkyBlue2	aquamarine2	LightYellow2	tan1	OrangeRed4	orchid4	gray50	gray88	
light grey	medium spring green	violet red	honeydew3	LightSkyBlue3	aquamarine4	LightYellow3	tan2	red2	plum1	gray51	gray89	
midnight blue	green yellow	medium orchid	honeydew4	LightSkyBlue4	DarkSeaGreen1	LightYellow4	tan4	red3	plum2	gray52	gray90	
navy	lime green	dark orchid	LavenderBlush2	SlateGray1	DarkSeaGreen2	yellow2	chocolate1	red4	plum3	gray53	gray91	
cornflower blue	yellow green	dark violet	LavenderBlush3	SlateGray2	DarkSeaGreen3	yellow3	chocolate2	DeepPink2	plum4	gray54	gray92	
dark slate blue	forest green	blue violet	LavenderBlush4	SlateGray3	DarkSeaGreen4	yellow4	chocolate3	DeepPink3	MediumOrchid1	gray55	gray93	
slate blue	olive drab	purple	MistyRose2	SlateGray4	SeaGreen1	gold2	firebrick1	DeepPink4	MediumOrchid2	gray56	gray94	
medium slate blue	dark khaki	medium purple	MistyRose3	LightSteelBlue1	SeaGreen2	gold3	firebrick2	HotPink1	MediumOrchid3	gray57	gray95	
light slate blue	khaki	thistle	MistyRose4	LightSteelBlue2	SeaGreen3	gold4	firebrick3	HotPink2	MediumOrchid4	gray58	gray96	
medium blue	pale goldenrod	snow2	azure2	LightSteelBlue3	PaleGreen1	goldenrod1	firebrick4	HotPink3	DarkOrchid1	gray59	gray97	
royal blue	light goldenrod yellow	snow3	azure3	LightSteelBlue4	PaleGreen2	goldenrod2	brown1	HotPink4	DarkOrchid2	gray60	gray98	
blue	light yellow	snow4	azure4	LightBlue1	PaleGreen3	goldenrod3	brown2	pink1	DarkOrchid3	gray61	gray99	
dodger blue	yellow	seashell2	SlateBlue1	LightBlue2	PaleGreen4	goldenrod4	brown3	pink2	DarkOrchid4	gray62	gray63	//

# Coordinate System

- Shapes will be drawn on a canvas using coordinates.

```
canvas = tkinter.Canvas(test_window, width=200, height=200, background="white")
```

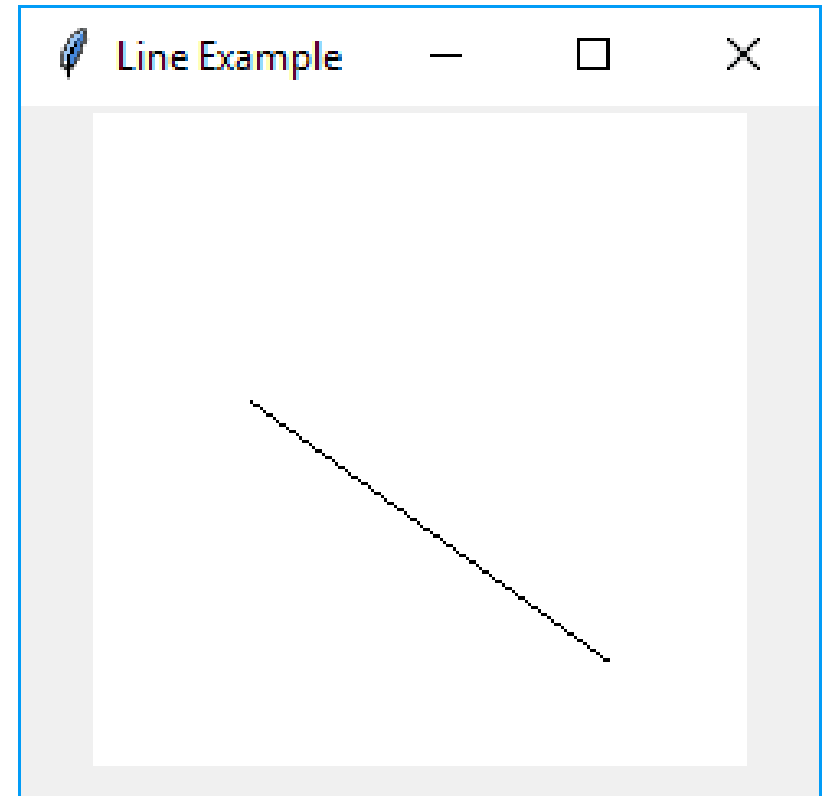


# Drawing Lines

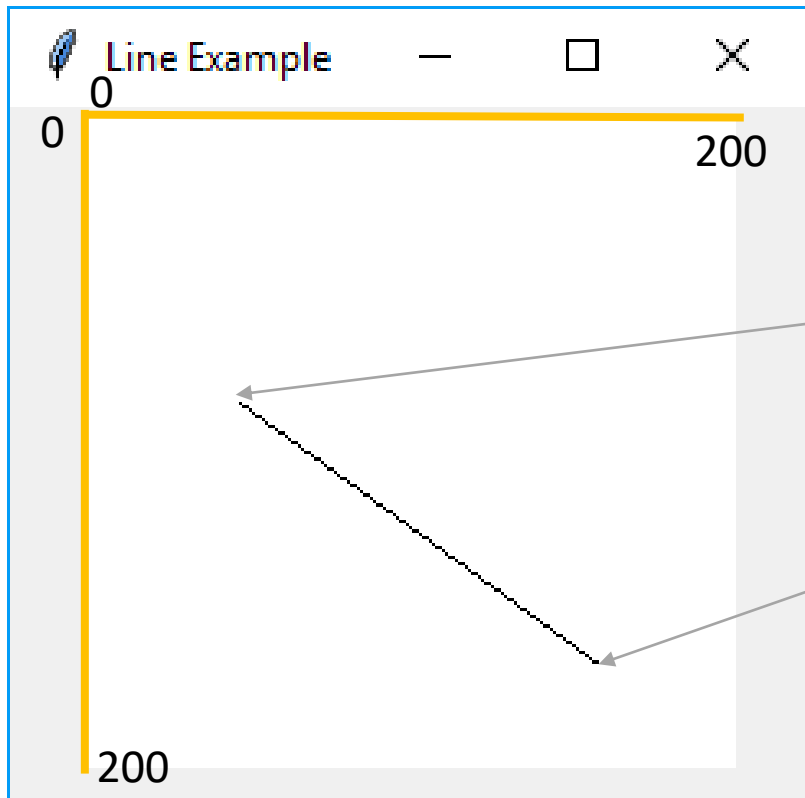
- The canvas object's `create_line` function will draw a straight line.
- A pair of coordinates is required.

```
canvas = tkinter.Canvas(test_window,  
                        width=200,  
                        height=200,  
                        background="white")
```

```
canvas.create_line(50, 90, 160, 170)
```



# Drawing Lines

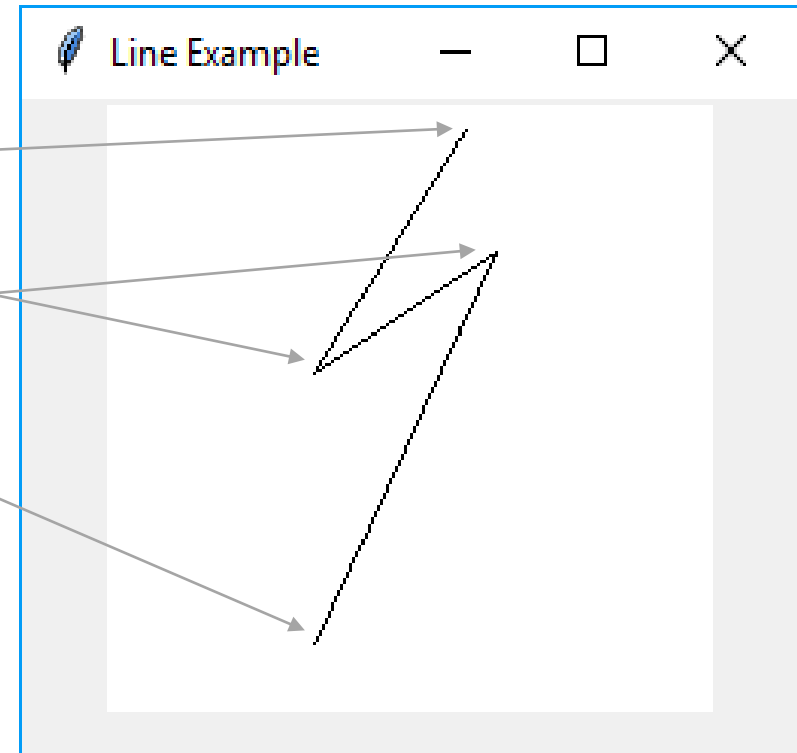


X , Y  
`canvas.create_line(50, 90, 160, 170)`

# Drawing Lines

- Supplying additional coordinates will continue the line.

```
canvas.create_line(120, 10,  
                  70, 90,  
                  130, 50,  
                  70, 180)
```

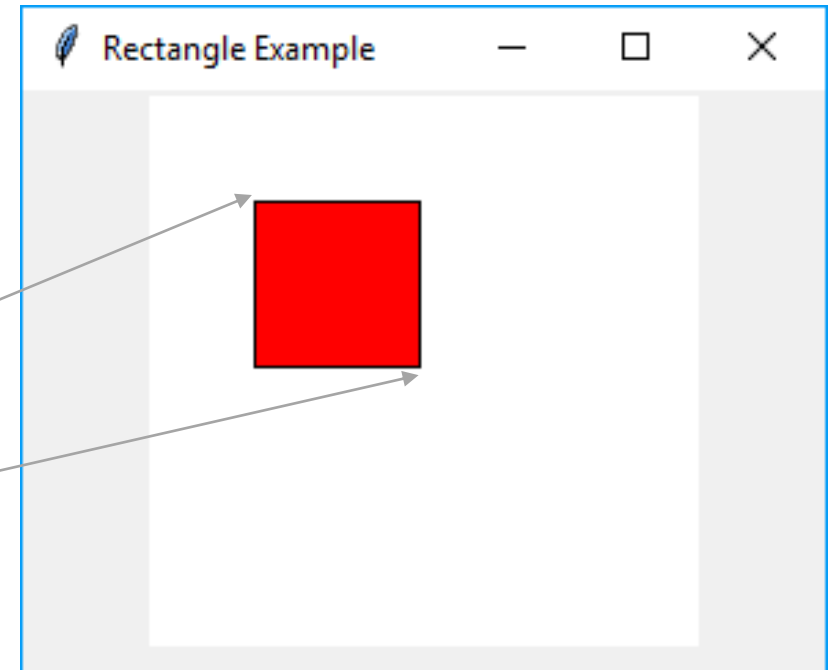


# Drawing Rectangles and Squares

- The canvas object's `create_rectangle` function will draw rectangles/squares.
- Two pairs of coordinates are required.
  - The fill argument is optional.

```
canvas = tkinter.Canvas(test_window,  
                        width=200,  
                        height=200,  
                        background="white")
```

```
canvas.create_rectangle(40, 40,  
                       100, 100,  
                       fill="red")
```



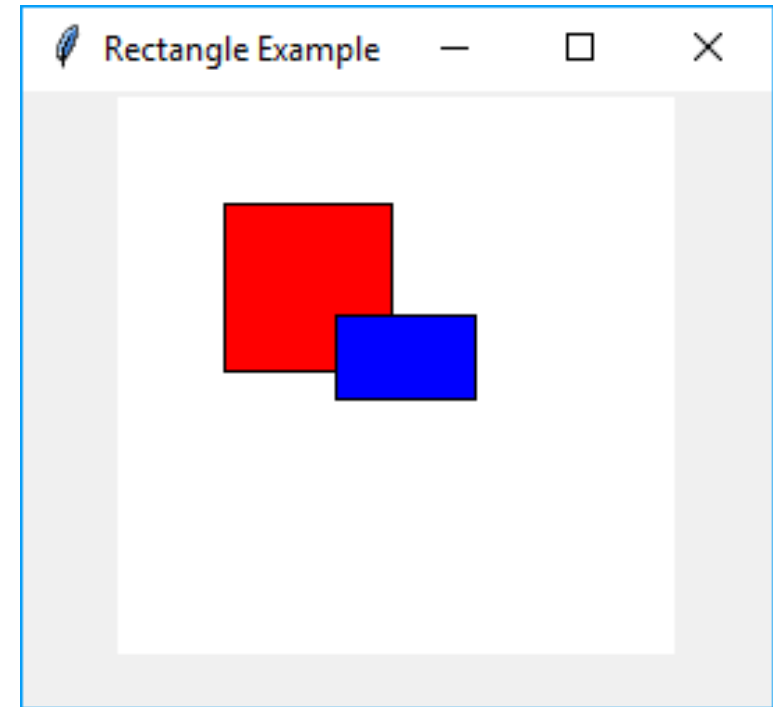


# Drawing Rectangles and Squares

- Lines and Shapes are drawn in the order they are created.
  - This allows drawing on top of previously drawn lines and shapes.

```
canvas.create_rectangle(40, 40,  
                        100, 100,  
                        fill="red")
```

```
canvas.create_rectangle(80, 80,  
                        130, 110,  
                        fill="blue")
```

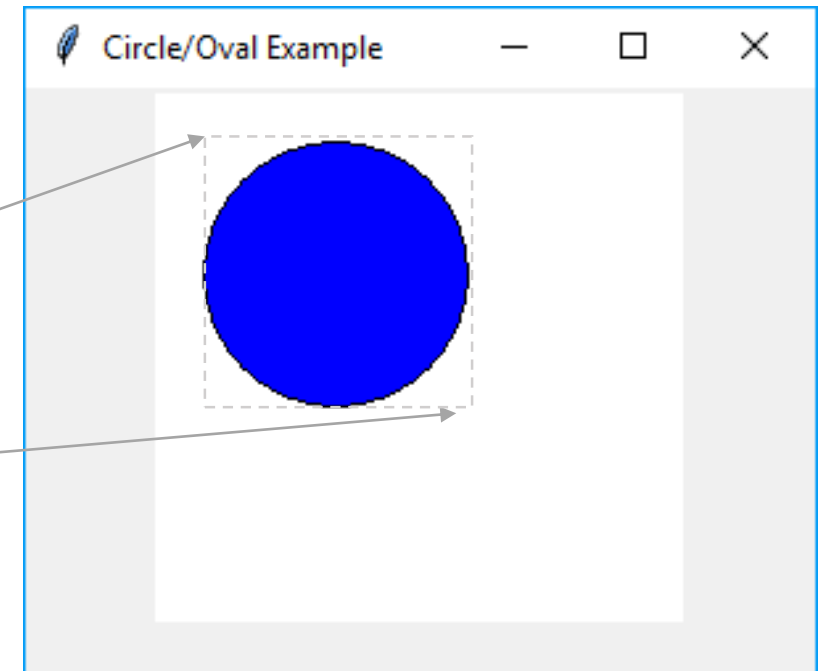


# Drawing Circles and Ovals

- The canvas object's `create_oval` function will draw circles/ovals.
- Two pairs of coordinates are required.
  - The fill argument is optional.

```
canvas = tkinter.Canvas(test_window,  
                          width=200,  
                          height=200,  
                          background="white")
```

```
canvas.create_oval(20, 20,  
                  120, 120,  
                  fill="blue")
```

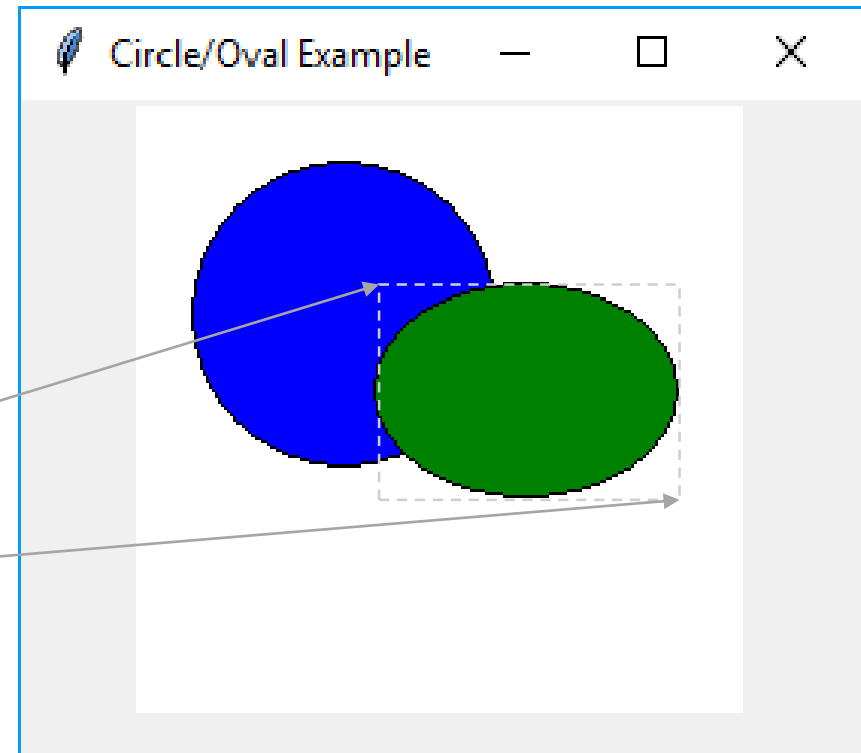


# Drawing Circles and Ovals

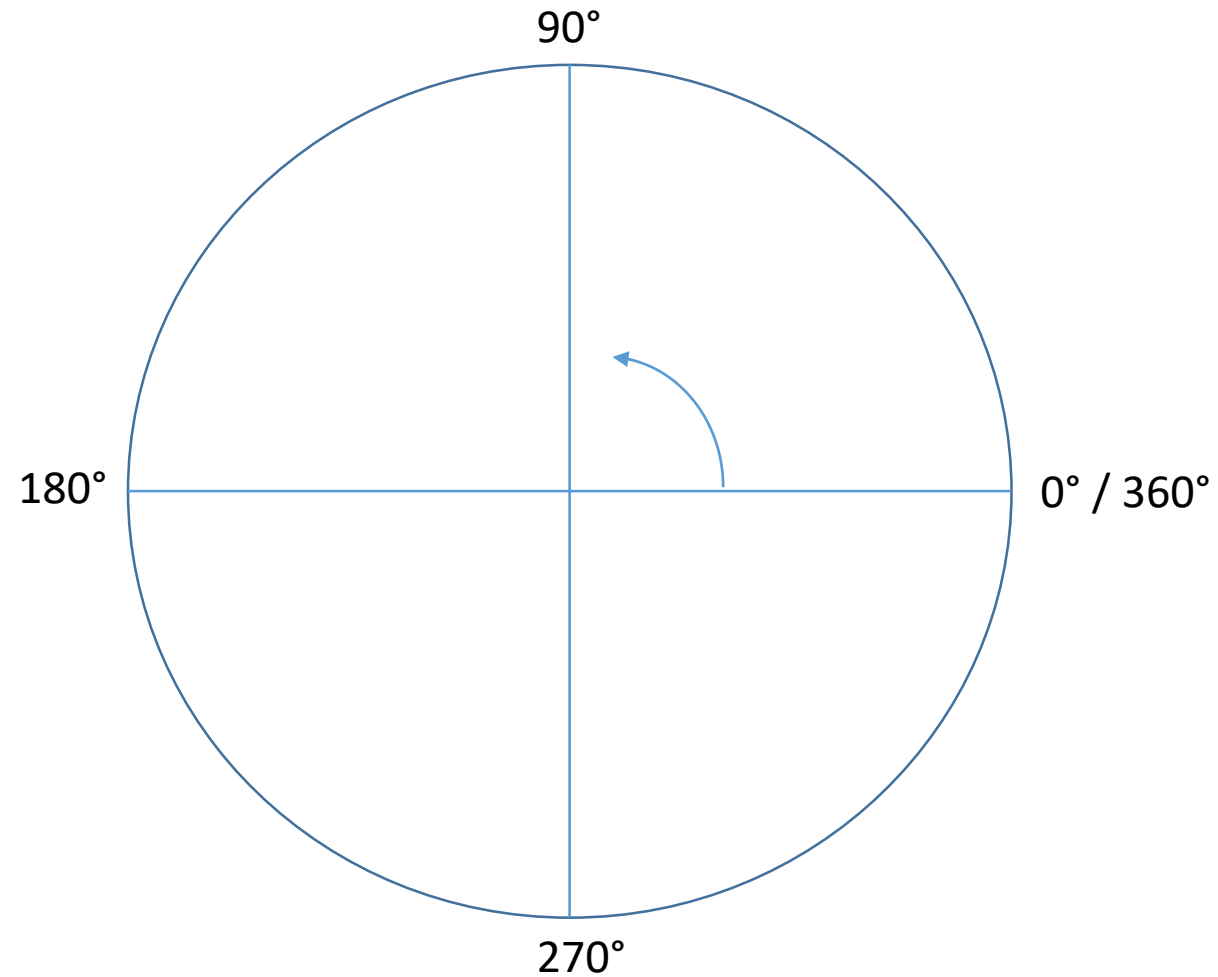
- Lines and Shapes are drawn in the order they are created.
  - This allows drawing on top of previously drawn lines and shapes.

```
canvas.create_oval(20, 20,  
                  120, 120,  
                  fill="blue")
```

```
canvas.create_oval(80, 60,  
                  180, 130,  
                  fill="green")
```



# Drawing Arcs/Slices

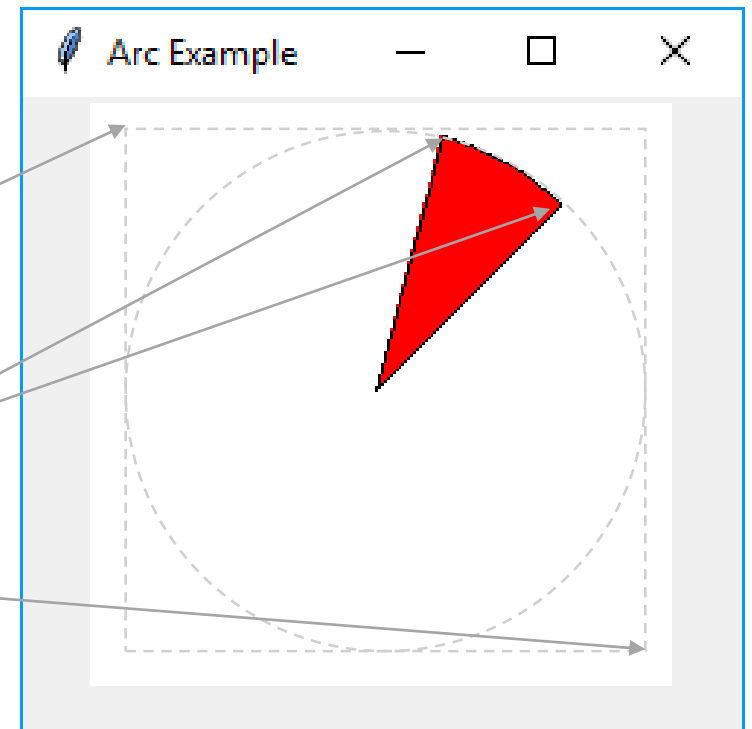


# Drawing Arcs/Slices

- The canvas object's `create_arc` function will draw arcs/slices.
  - Two pairs of coordinates are required.
  - The start argument specifies the start degrees
  - The extent argument specifies degrees from the start
  - The fill argument is optional.

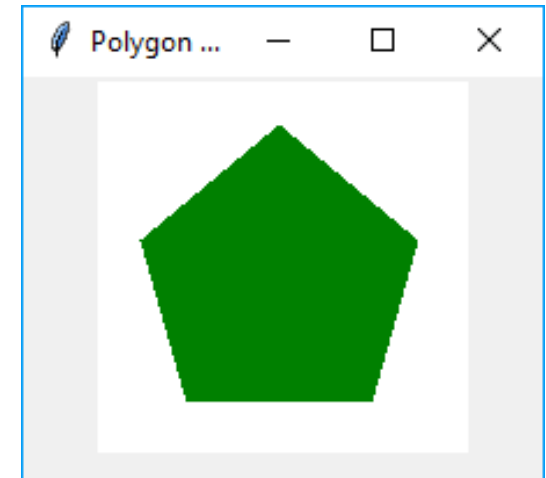
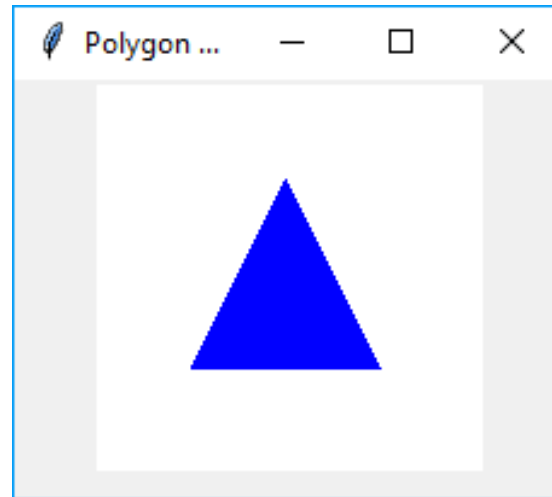
```
canvas = tkinter.Canvas(test_window,  
                          width=200,  
                          height=200,  
                          background="white")
```

```
canvas.create_arc(10, 10,  
                 190, 190,  
                 start=45,  
                 extent=30,  
                 fill="red")
```



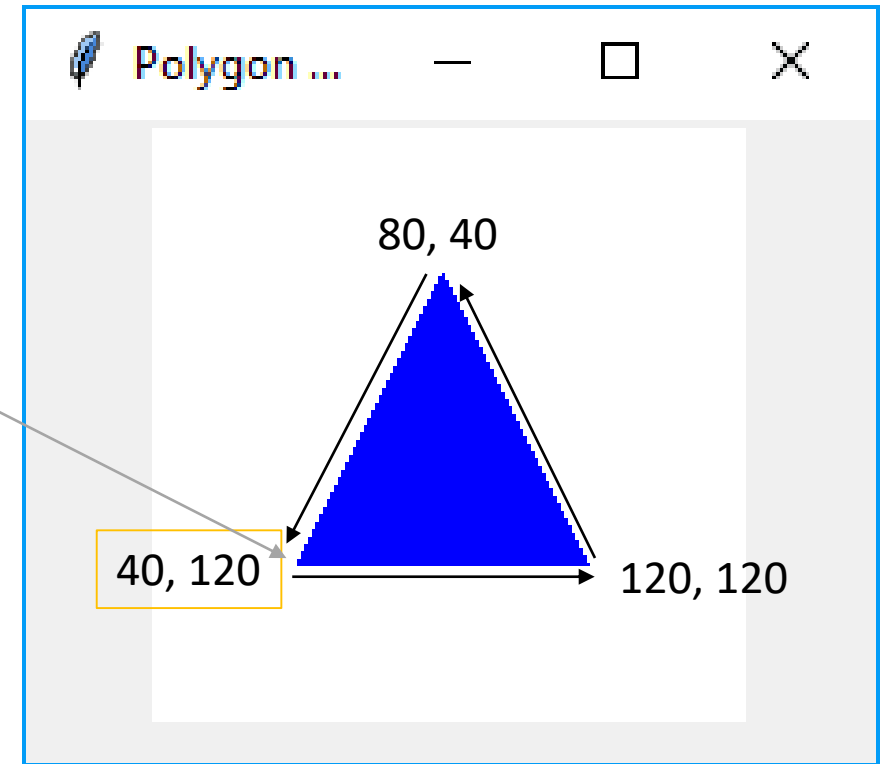
# Drawing Polygons

- The canvas object's `create_polygon` function will draw regular and irregular shapes.
  - Undetermined pairs of coordinates are required.
  - The fill argument is optional.
    - Shape will be filled black if not provided



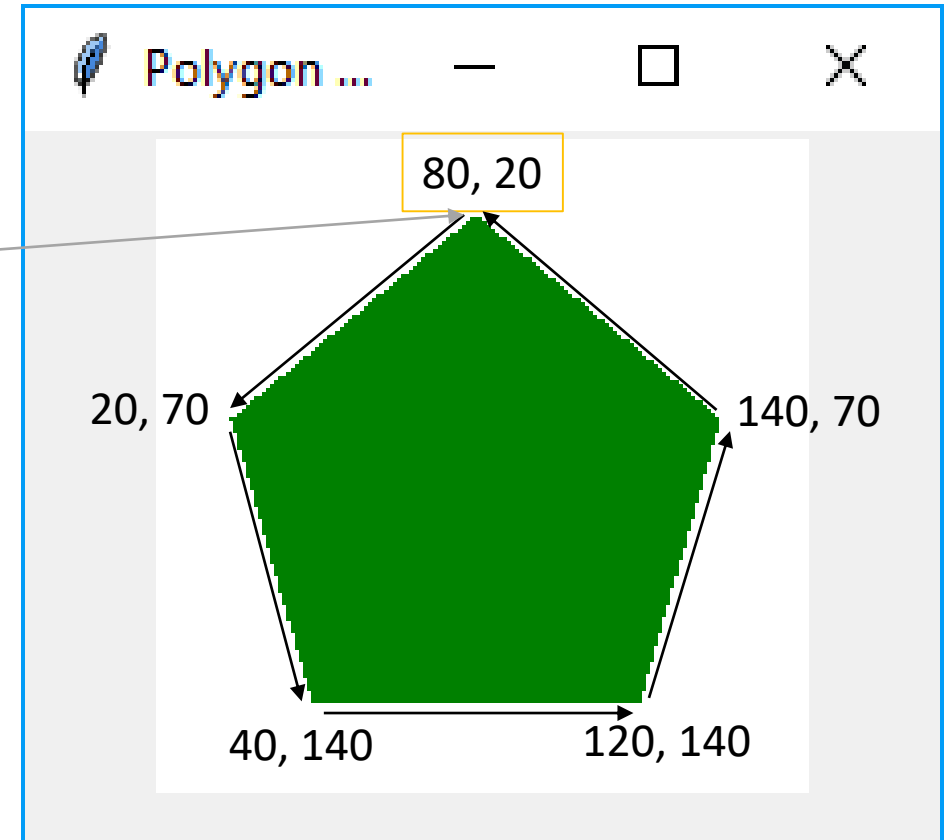
# Drawing Polygons

```
canvas.create_polygon(40, 120,  
                     120, 120,  
                     80, 40,  
                     40, 120,  
                     fill="blue")
```



# Drawing Polygons

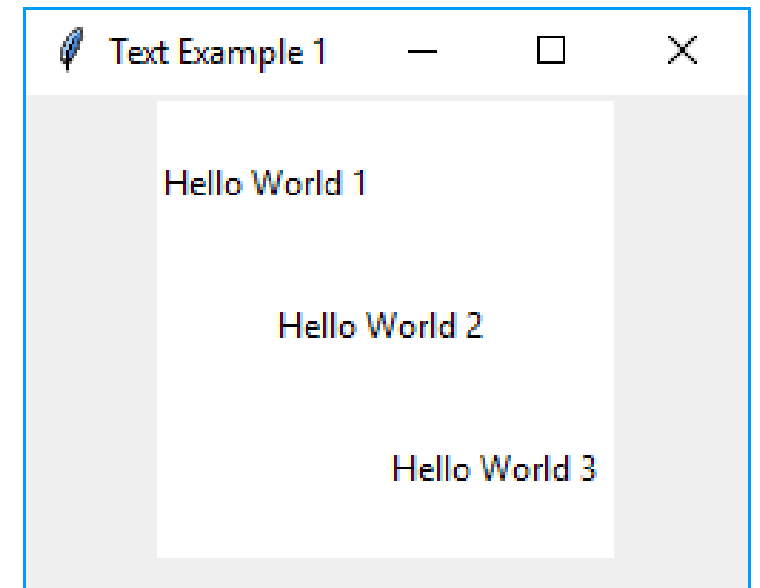
```
canvas.create_polygon(80, 20,  
                     20, 70,  
                     40, 140,  
                     120, 140,  
                     140, 70,  
                     80, 20,  
                     fill="green")
```





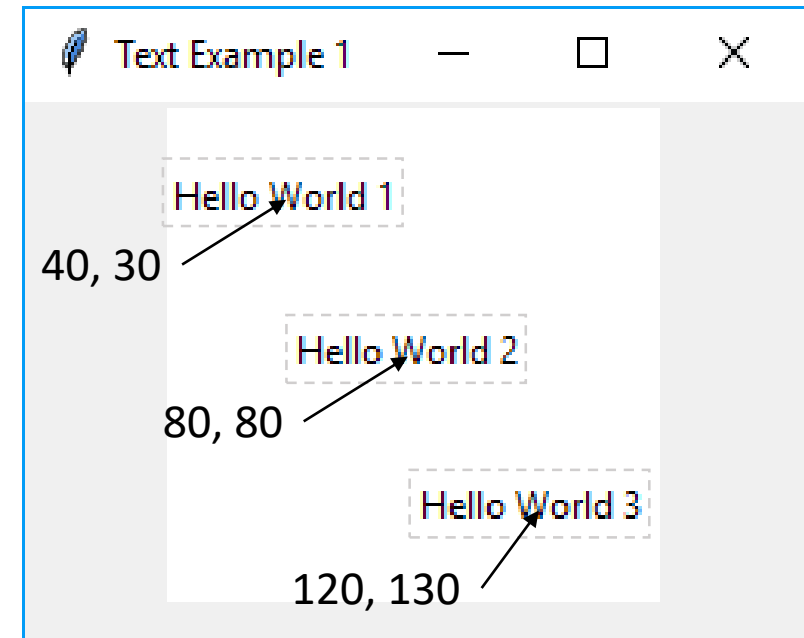
# Drawing Text

- The canvas object's `create_text` function will draw text.
  - One pair of coordinates is required.
    - The **center** of the text.
  - The text argument supplies the text to be drawn.
  - A fill argument is optional.
    - Will set the text's color.
    - Text will be black if not provided



# Drawing Text

```
canvas.create_text(40, 30,  
                  text="Hello World 1")  
  
canvas.create_text(80, 80,  
                  text="Hello World 2")  
  
canvas.create_text(120, 130,  
                  text="Hello World 3")
```



# Creating a Canvas (Java)

- A ***canvas*** is a component that allows the drawing of lines and shapes.
  - The background of a canvas is transparent.
- Using the AWT libraries, canvases are created using the **Canvas** class.
- Must be imported: `import java.awt.Canvas;`

# Creating a Canvas

- The proper way to use a Canvas is to create a subclass of the Canvas class.
- At a minimum, you will need to override the Canvas class's paint method.
  - This method is called when the Canvas is created.
  - This method handles drawing any shapes/graphics on the canvas.

# Creating a Canvas

- The Canvas class's paint method accepts a Graphics argument.
  - This parameter is what is used to draw the shapes/graphics.
  - The Graphics argument is passed to this method when the Canvas (or a subclass of the Canvas class) is instantiated.
- Needs to be imported: **`import java.awt.Graphics;`**

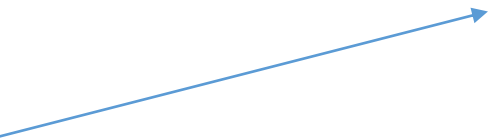
# Creating a Canvas

```
import java.awt.Canvas;
import java.awt.Graphics;

public class MyCanvas() extends Canvas {

    public void paint(Graphics g) {
        //Called when this class is instantiated.
        //All code for drawing on this canvas
        //goes here.
    }
}
```

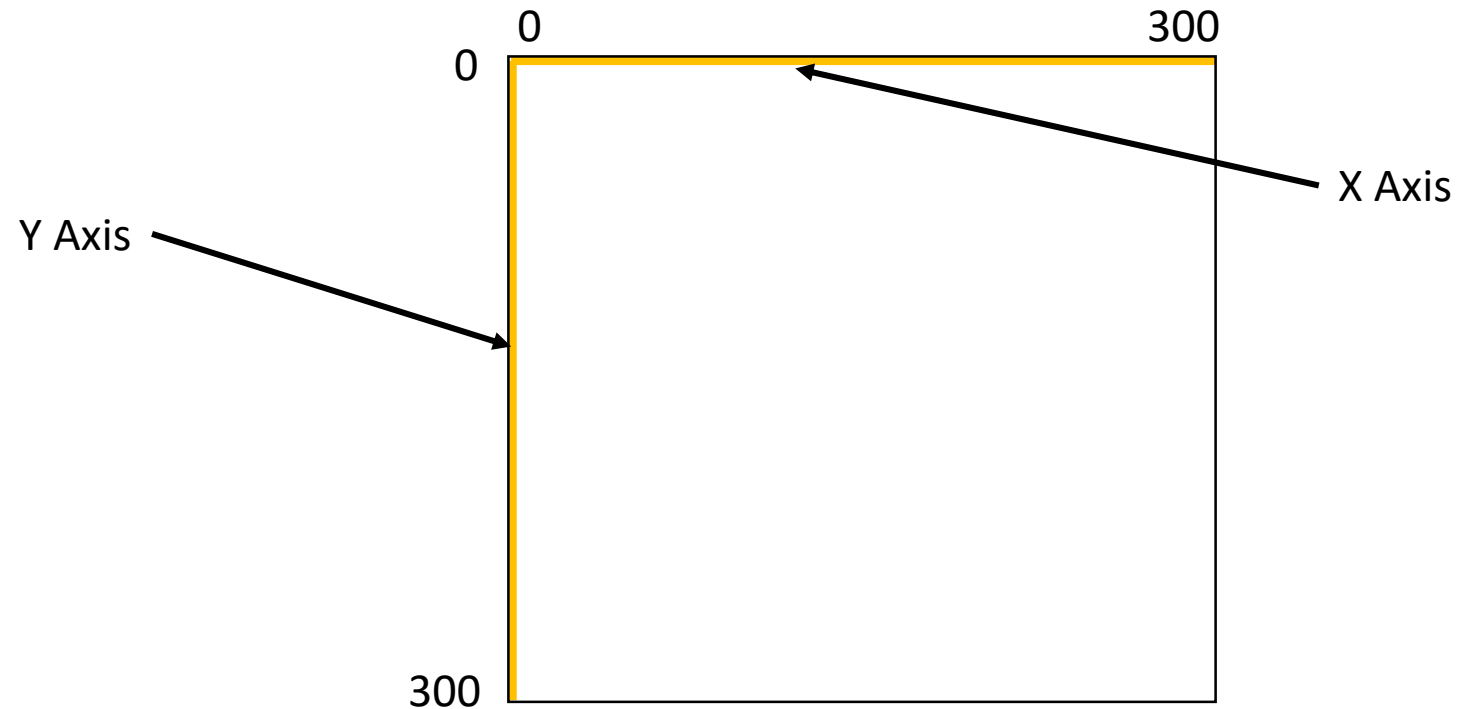
Overrides the superclass's  
paint method.



(No constructor is needed)

# Coordinate System

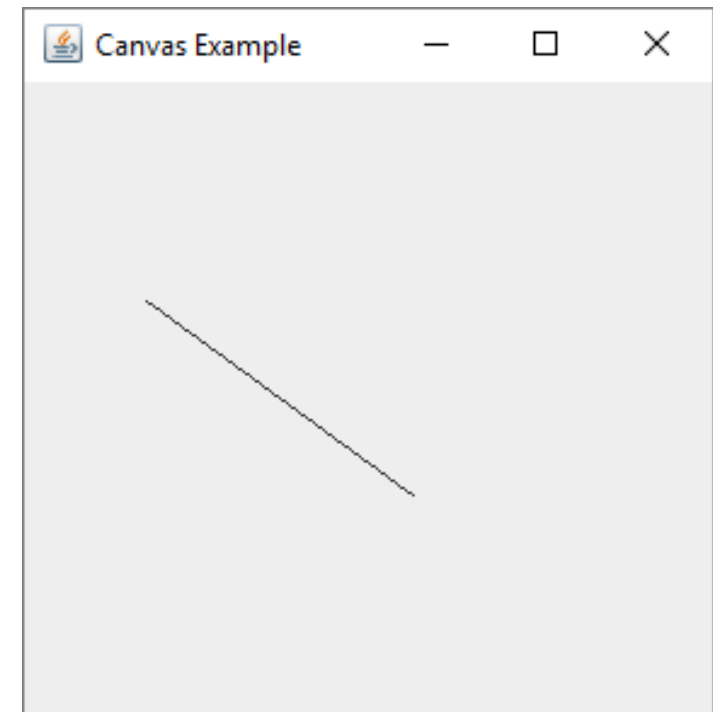
- Shapes and graphics will be drawn on a canvas using coordinates.



# Drawing Lines

- The Graphics object's drawLine method will draw a straight line.
  - A pair of x and y coordinates (4 ints) are required.

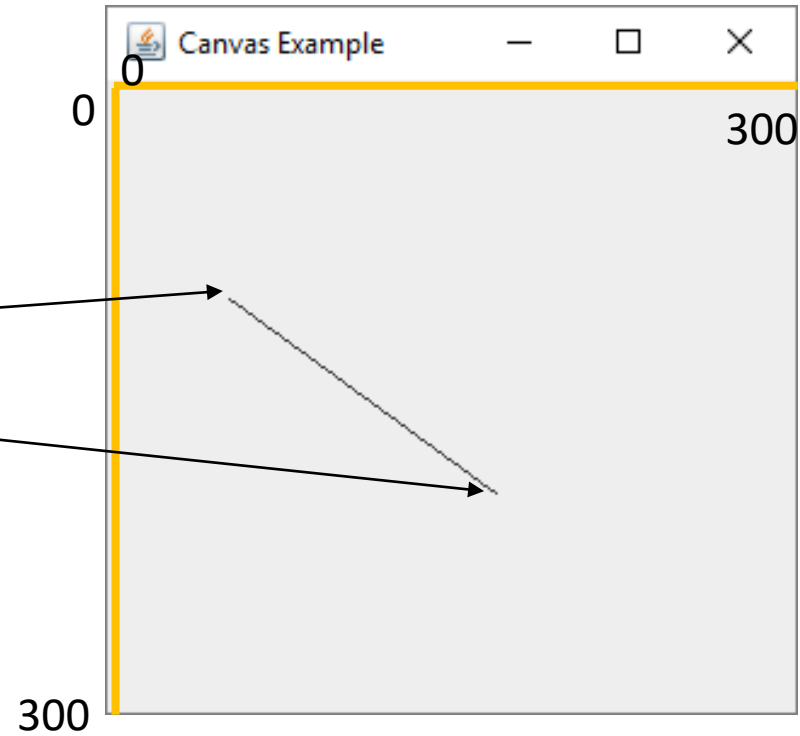
```
import java.awt.Canvas;  
import java.awt.Graphics;  
  
public class MyCanvas() extends Canvas {  
  
    public void paint(Graphics g) {  
        g.drawLine(50, 90,  
                   160, 170);  
    }  
  
}
```





# Drawing Lines

```
g.drawLine(50, 90,  
           160, 170);
```



# Colors

- By default, lines and shapes are drawn in black.
- The current color of the Graphics object being used to draw the lines and shapes can be changed using its setColor method.
  - Accepts a Color class argument.
- Using the AWT libraries, Color objects are created using the **Color** class.
- Must be imported: **import java.awt.Color;**

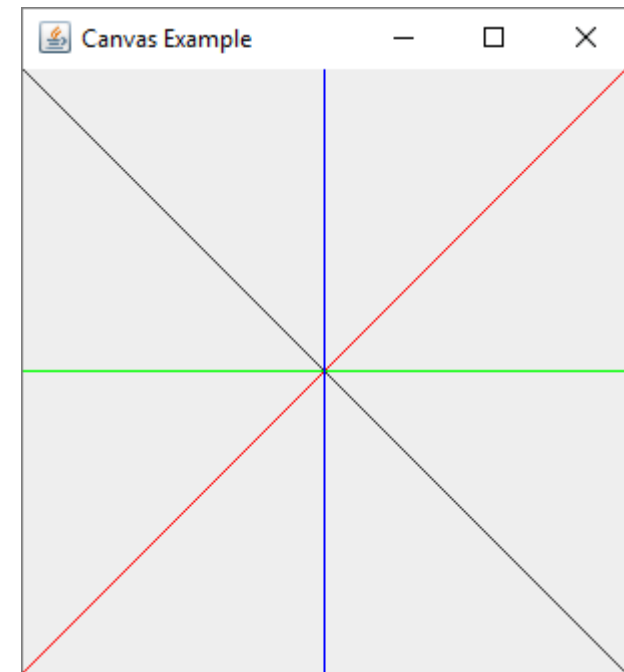
# Colors

- It's possible to create custom colors, but the Color class has some colors/constants already defined.
- BLACK, BLUE, CYAN, DARK\_GRAY, GRAY, GREEN, LIGHT\_GRAY, MAGENTA, ORANGE, PINK, RED, WHITE, and YELLOW.

# Colors

- Once the color is changed, it will remain that color until it is changed again.

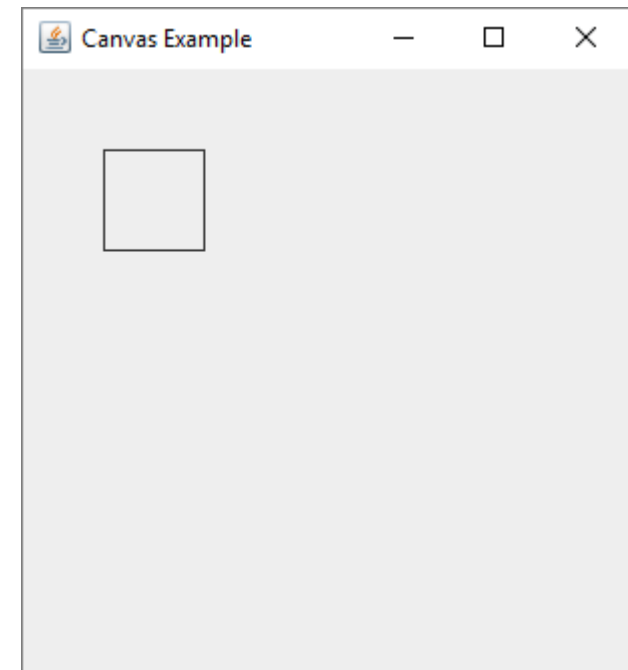
```
public void paint(Graphics g) {  
    g.drawLine(0, 0,  
               300, 300);  
    g.setColor(Color.RED);  
    g.drawLine(0, 300,  
               300, 0);  
    g.setColor(Color.GREEN);  
    g.drawLine(0, 150,  
               300, 150);  
    g.setColor(Color.BLUE);  
    g.drawLine(150, 0,  
               150, 300);  
}
```



# Drawing Rectangles

- The Graphics object's drawRect method will draw an unfilled rectangle.
  - A pair of x and y coordinates (2 ints) of the top left corner are required.
  - The width and height (2 ints) are also required.

```
public void paint(Graphics g) {  
    g.drawRect(40, 40,  
               50, 50);  
}
```



# Drawing Rectangles

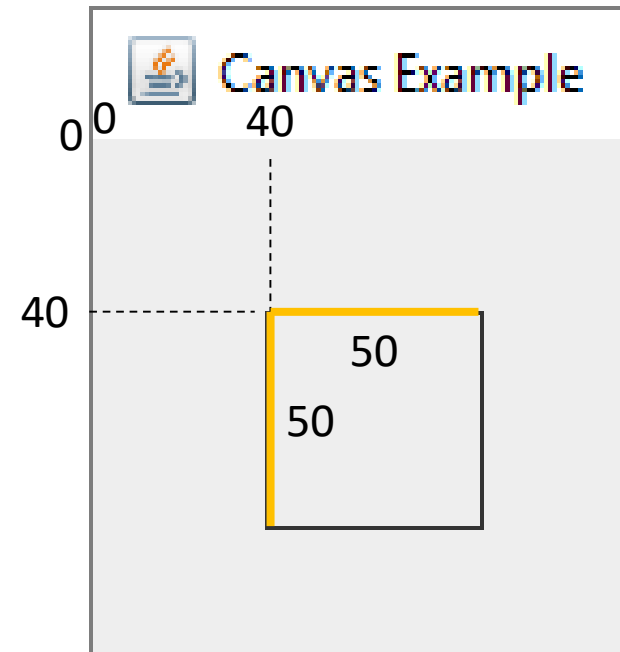
```
public void paint(Graphics g) {
```

```
    g.drawRect(40, 40,  
               50, 50);
```

Diagram illustrating the parameters for `g.drawRect(x, y, width, height)`:

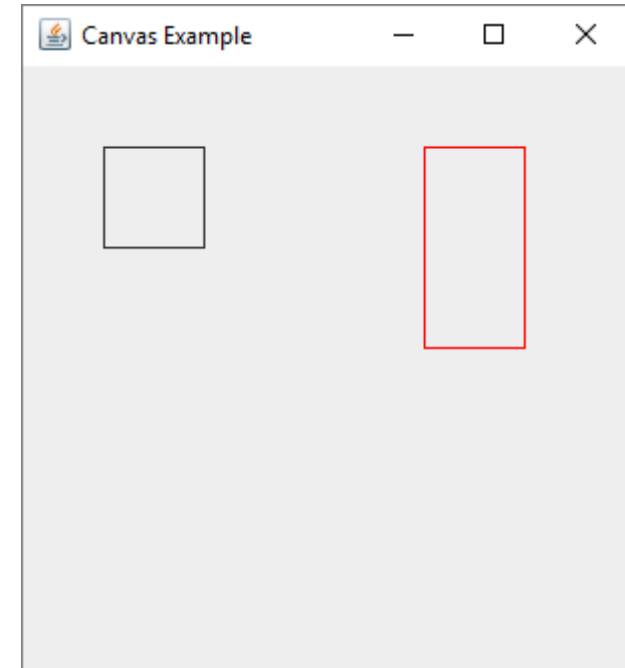
- `x` points to the first `40`.
- `y` points to the second `40`.
- `width` points to the `50` after the first comma.
- `height` points to the `50` after the second comma.

```
}
```



# Drawing Rectangles

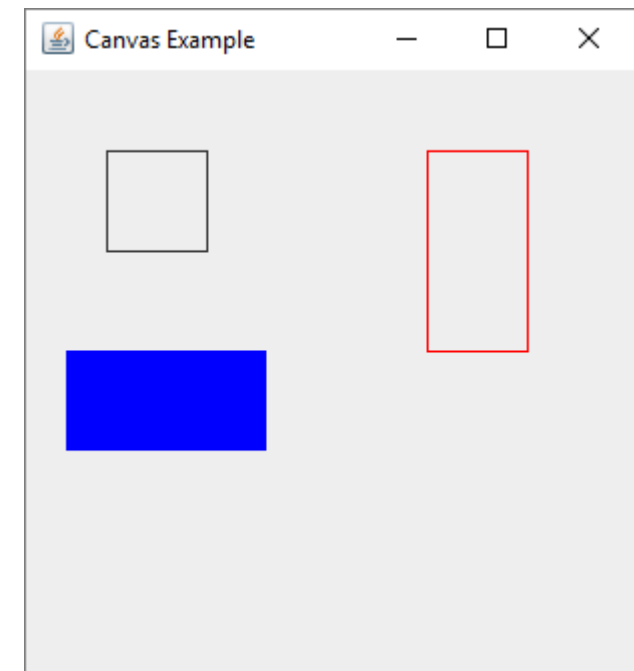
```
public void paint(Graphics g) {  
    g.drawRect(40, 40,  
               50, 50);  
  
    g.setColor(Color.RED);  
    g.drawRect(200, 40,  
               50, 100);  
}
```



# Drawing Rectangles

- The Graphics object's fillRect method will draw a filled rectangle.
  - A pair of x and y coordinates (2 ints) of the top left corner are required.
  - The width and height (2 ints) are also required.

```
public void paint(Graphics g) {  
    g.drawRect(40, 40,  
               50, 50);  
    g.setColor(Color.RED);  
    g.drawRect(200, 40,  
               50, 100);  
    g.setColor(Color.BLUE);  
    g.fillRect(20, 140,  
               100, 50);  
}
```

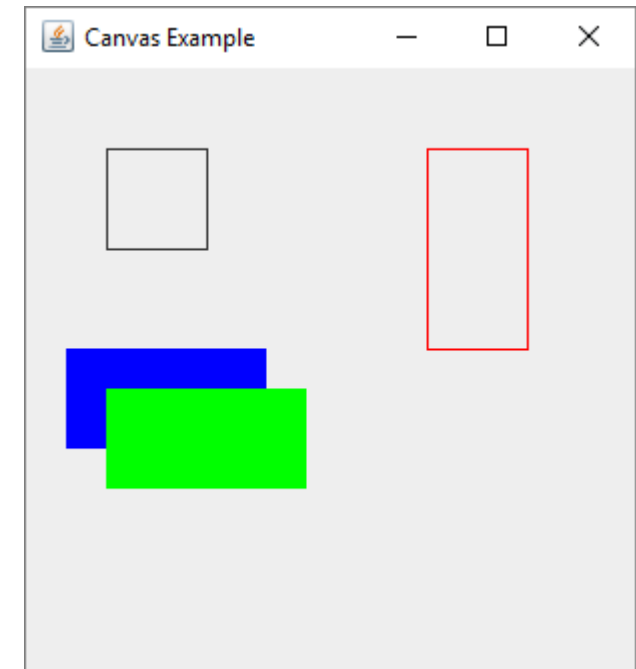




# Drawing Rectangles

- Lines and shapes appear in the order they are drawn.
  - Shapes may overlap ones drawn previously.

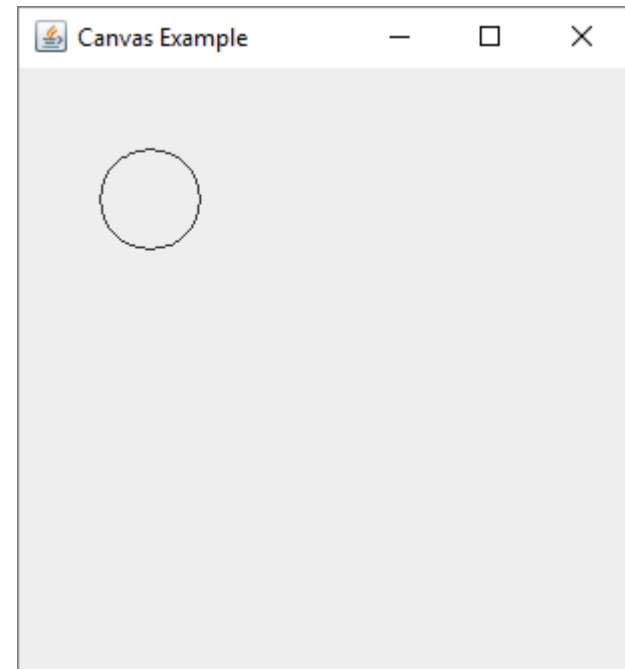
```
public void paint(Graphics g) {  
    g.drawRect(40, 40,  
               50, 50);  
    g.setColor(Color.RED);  
    g.drawRect(200, 40,  
               50, 100);  
    g.setColor(Color.BLUE);  
    g.fillRect(20, 140,  
               100, 50);  
    g.setColor(Color.GREEN);  
    g.fillRect(40, 160,  
               100, 50);  
}
```



# Drawing Circles and Ovals

- The Graphics object's drawOval method will draw unfilled circles and ovals.
  - A pair of x and y coordinates (2 ints) of the top left corner are required.
  - The width and height (2 ints) are also required.

```
public void paint(Graphics g) {  
    g.drawOval(40, 40,  
               50, 50);  
}
```



# Drawing Circles and Ovals

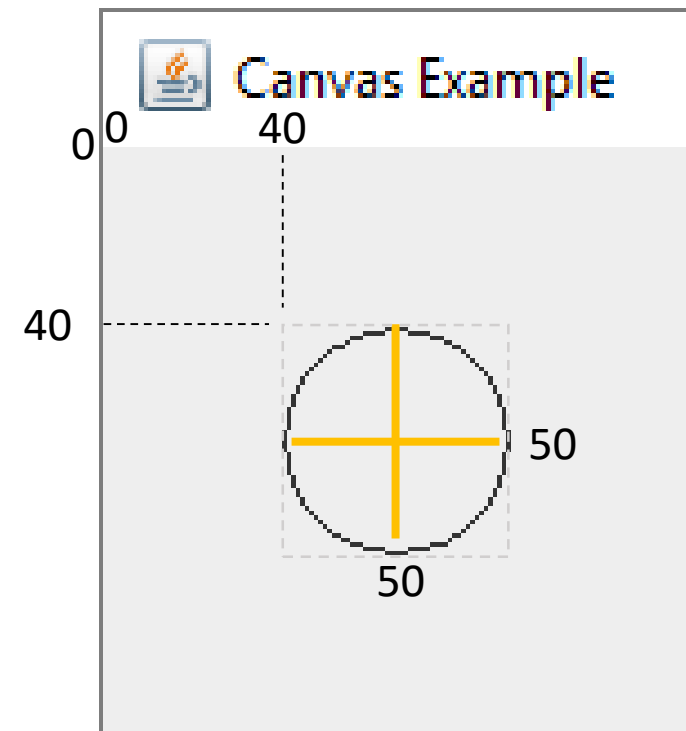
```
public void paint(Graphics g) {
```

```
    g.drawOval(40, 40,  
               50, 50);
```

Diagram illustrating the parameters for `g.drawOval`:

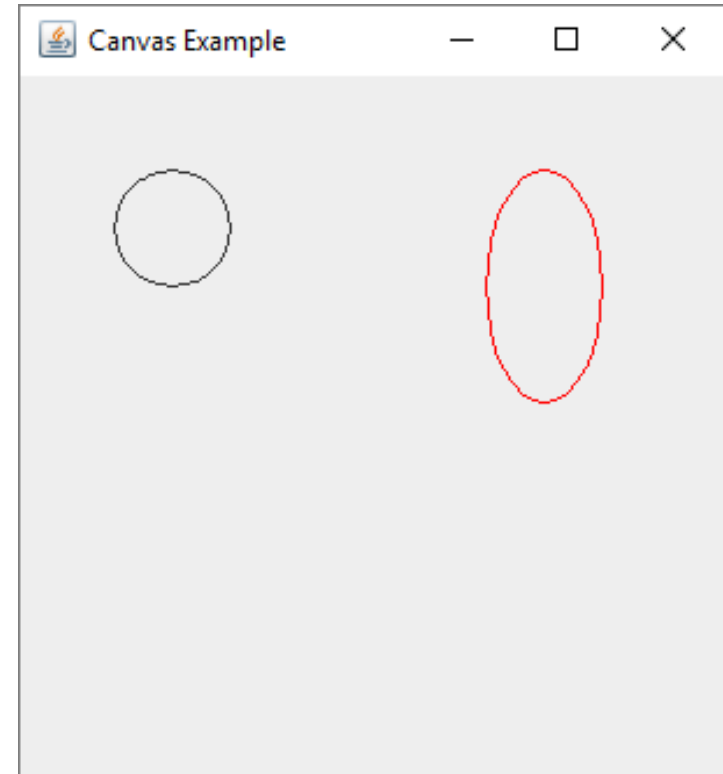
- `40` is the `x` coordinate (top-left corner).
- `40` is the `y` coordinate (top-left corner).
- `50` is the `width`.
- `50` is the `height`.

```
}
```



# Drawing Circles and Ovals

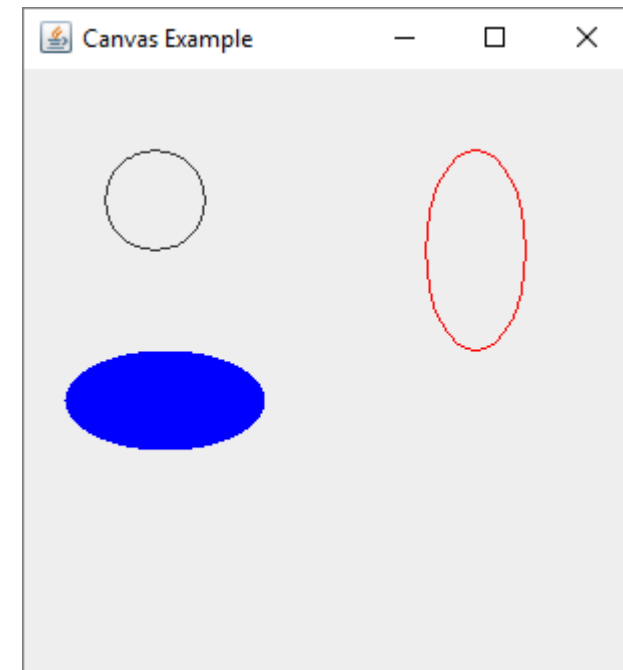
```
public void paint(Graphics g) {  
    g.drawOval(40, 40,  
               50, 50);  
  
    g.setColor(Color.RED);  
    g.drawOval(200, 40,  
               50, 100);  
}
```



# Drawing Circles and Ovals

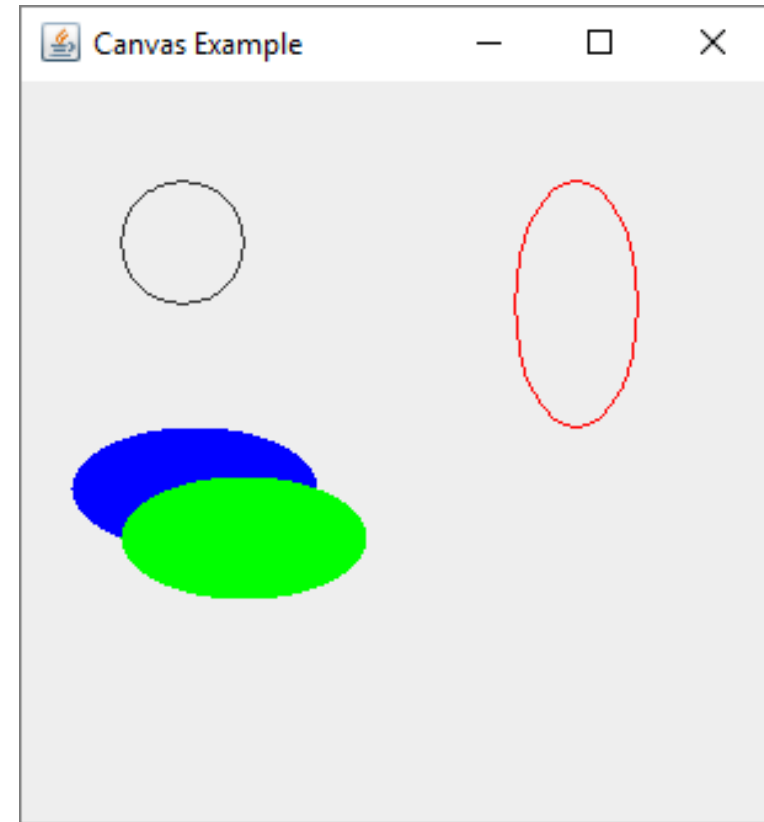
- The Graphics object's fillOval method will draw filled circles and ovals.
  - A pair of x and y coordinates (2 ints) of the top left corner are required.
  - The width and height (2 ints) are also required.

```
public void paint(Graphics g) {  
    g.drawOval(40, 40,  
              50, 50);  
    g.setColor(Color.RED);  
    g.drawOval(200, 40,  
              50, 100);  
    g.setColor(Color.BLUE);  
    g.fillOval(20, 140,  
              100, 50);  
}
```

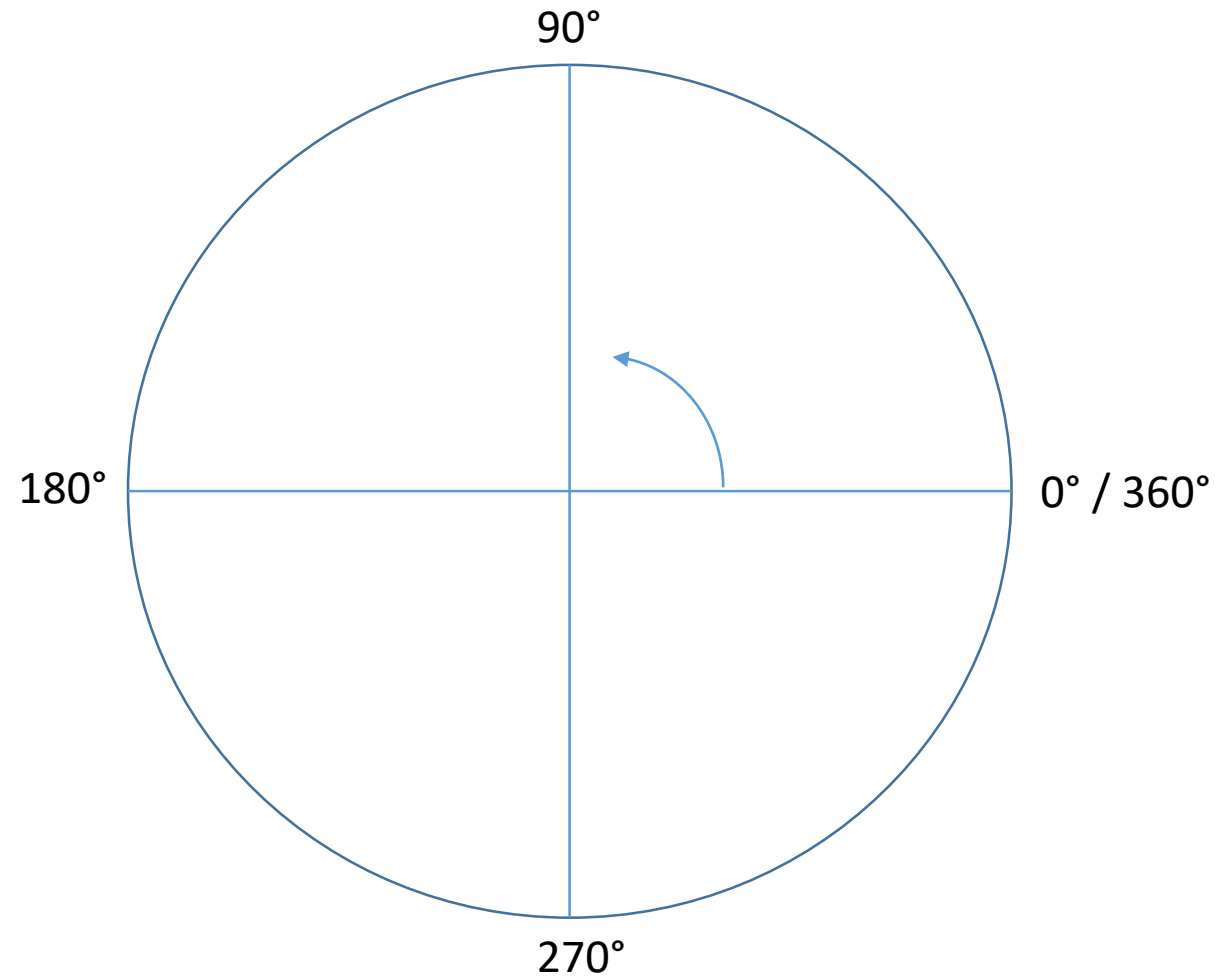


# Drawing Circles and Ovals

```
public void paint(Graphics g) {  
    g.drawOval(40, 40,  
              50, 50);  
    g.setColor(Color.RED);  
    g.drawOval(200, 40,  
              50, 100);  
    g.setColor(Color.BLUE);  
    g.fillOval(20, 140,  
              100, 50);  
    g.setColor(Color.GREEN);  
    g.fillOval(40, 160,  
              100, 50);  
}
```



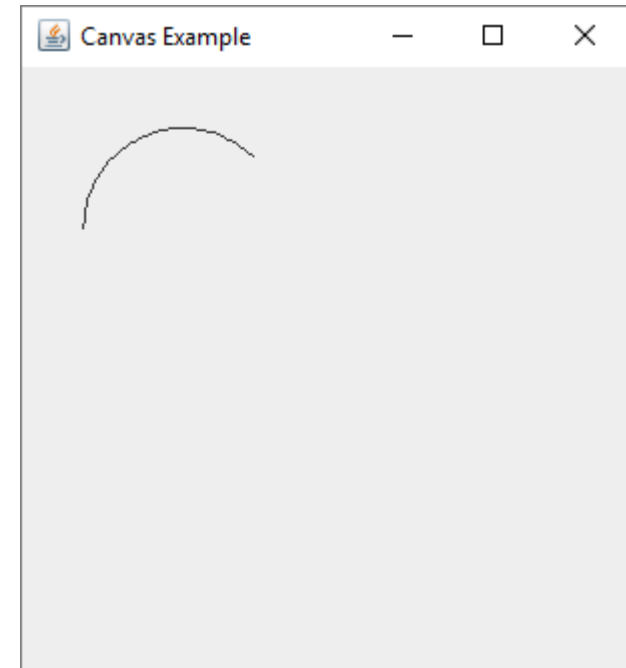
# Drawing Arcs



# Drawing Arcs

- The Graphics object's drawArc method will draw arcs.
  - A pair of x and y coordinates (2 ints) of the top left corner are required.
  - The width and height (2 ints) are also required.
  - The starting degree (an int)
  - The number of degrees from the start (an int)

```
public void paint(Graphics g) {  
    g.drawArc(30, 30,  
              100, 100,  
              45, 135);  
}
```





# Drawing Circles and Ovals

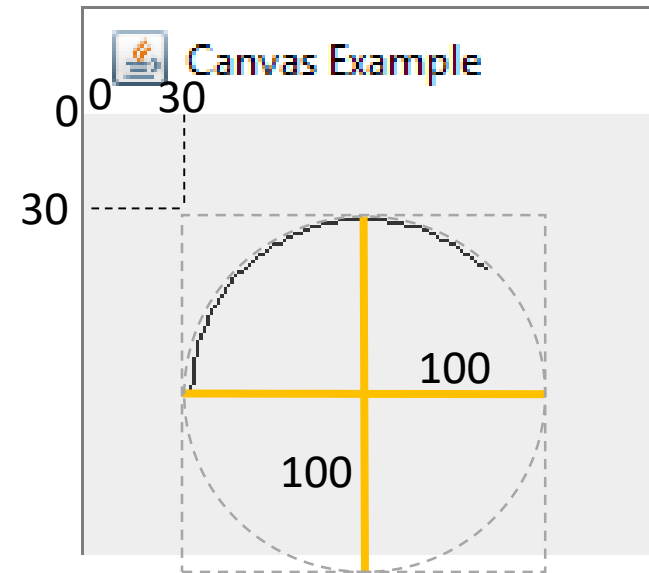
```
public void paint(Graphics g) {
```

```
    g.drawArc(30, 30,  
              100, 100,  
              45, 135);
```

Diagram illustrating the parameters for `g.drawArc(x, y, width, height, start, degrees from start)`:

- `x`: 30
- `y`: 30
- `width`: 100
- `height`: 100
- `start`: 45
- `degrees from start`: 135

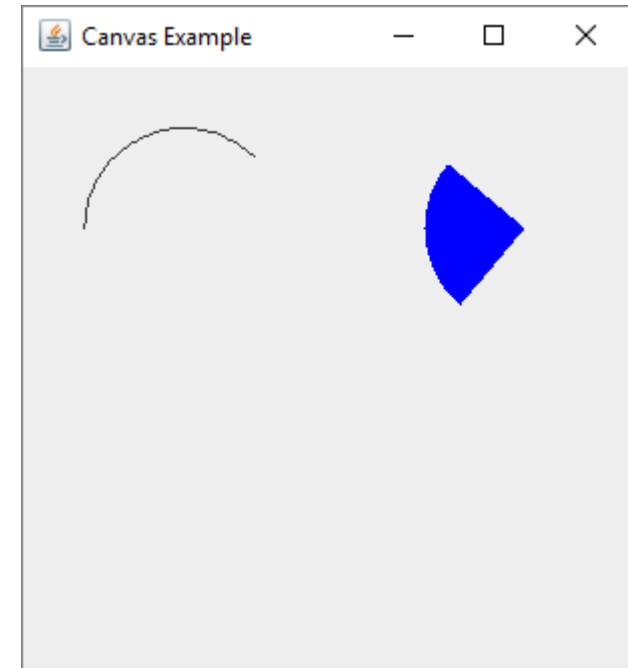
```
}
```



# Drawing Arcs

- The Graphics object's fillArc method will draw filled arcs.
  - A pair of x and y coordinates (2 ints) of the top left corner are required.
  - The width and height (2 ints) are also required.
  - The starting degree (an int)
  - The number of degrees from the start (an int)

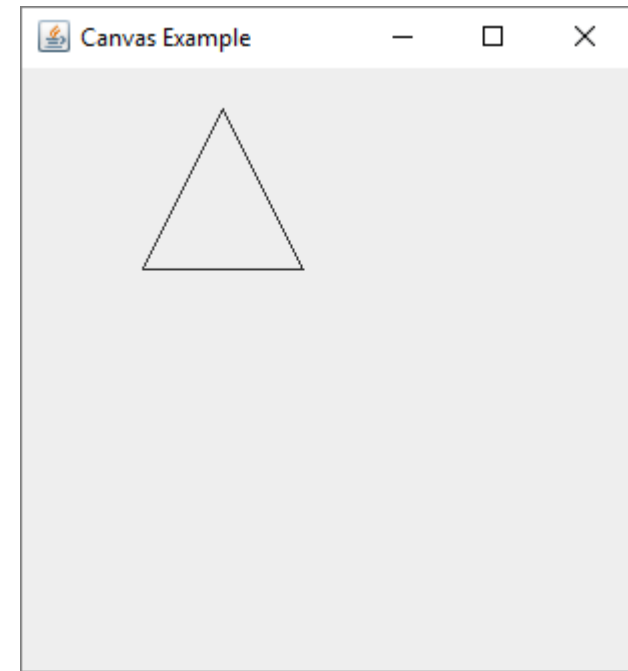
```
public void paint(Graphics g) {  
    g.drawArc(30, 30,  
              100, 100,  
              45, 135);  
    g.setColor(Color.BLUE);  
    g.fillArc(200, 30,  
              100, 100,  
              140, 90);  
}
```



# Drawing Polygons

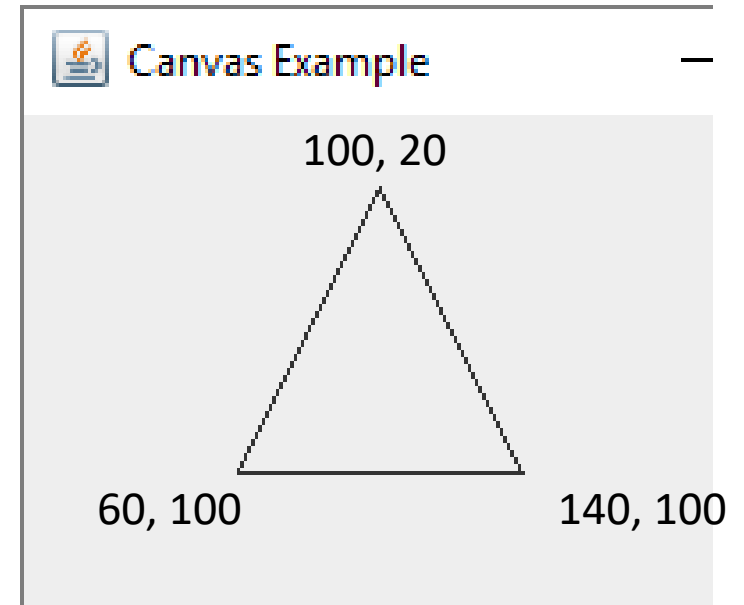
- The Graphics object's drawPolygon method will draw polygons.
  - An array of x coordinates (ints) is required.
  - An array of y coordinates (ints) is required.
  - The number of vertices (an int) is required.

```
public void paint(Graphics g) {  
    int[] shape1_x = {100, 60, 140};  
    int[] shape1_y = {20, 100, 100};  
    g.drawPolygon(shape1_x, shape1_y, 3);  
}
```



# Drawing Polygons

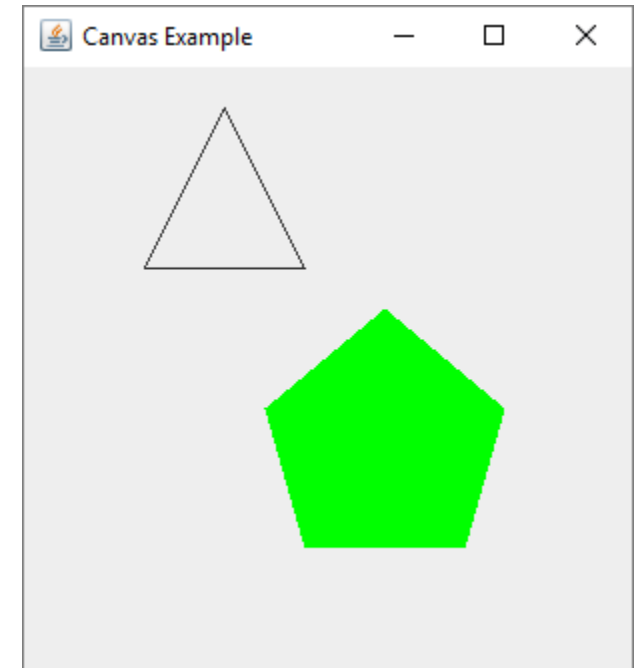
```
public void paint(Graphics g) {  
    int[] shape1_x = {100, 60, 140};  
    int[] shape1_y = {20, 100, 100};  
    g.drawPolygon(shape1_x, shape1_y, 3);  
}
```



# Drawing Polygons

- The Graphics object's fillPolygon method will draw filled polygons.
  - An array of x coordinates (ints) is required.
  - An array of y coordinates (ints) is required.
  - The number of vertices (an int) is required.

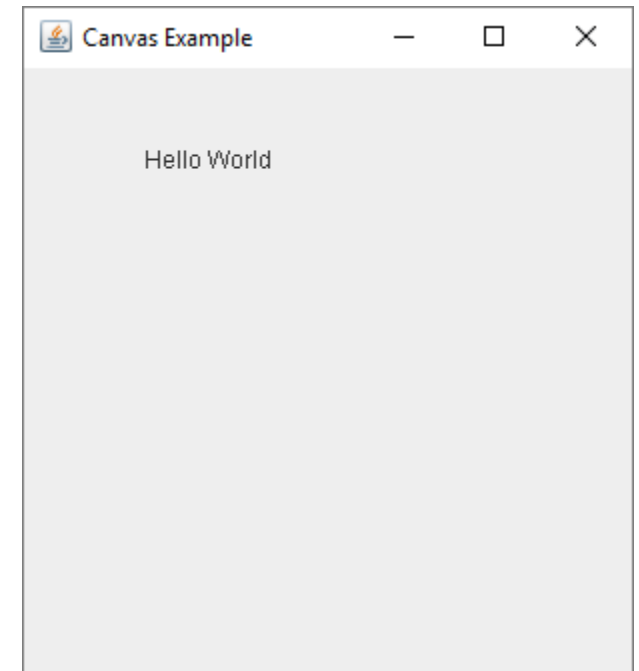
```
public void paint(Graphics g) {  
    int[] shape1_x = {100, 60, 140};  
    int[] shape1_y = {20, 100, 100};  
    g.drawPolygon(shape1_x, shape1_y, 3);  
  
    g.setColor(Color.GREEN);  
    int[] shape2_x = {180, 120, 140, 220, 240};  
    int[] shape2_y = {120, 170, 240, 240, 170};  
    g.fillPolygon(shape2_x, shape2_y, 5);  
}
```



# Drawing Text

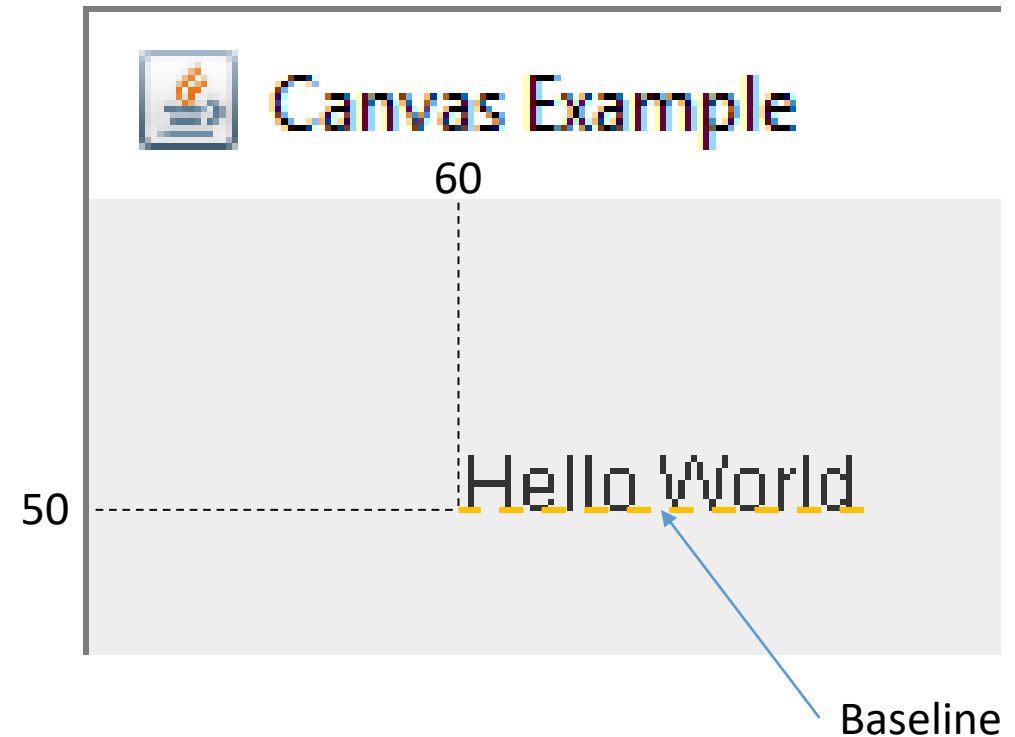
- The Graphics object's drawString method will draw text.
  - The text to draw (a String)
  - A pair of x and y coordinates (2 ints) of the baseline.

```
public void paint(Graphics g) {  
    g.drawString("Hello World", 60, 50);  
}
```



# Drawing Text

```
public void paint(Graphics g) {  
    g.drawString("Hello World", 60, 50);  
}
```



# Colors and Fonts

- By default, text is drawn in black.
- The current color of the Graphics object being used to draw the text can be changed using its setColor method.
  - Just like with lines and shapes.
- By default, the text's font is Dialog.
- The current font of the Graphics object being used to draw the text can be changed using its setFont method.



# Colors and Fonts

```
public void paint(Graphics g) {  
    g.drawString("Hello World", 60, 50);  
  
    g.setColor(Color.GREEN);  
    g.setFont(new Font("Serif", Font.BOLD, 16));  
    g.drawString("Hello World", 200, 50);  
}
```

