Create a canvas of size width, height. Returns the canvas so that you can call canvas methods on it.

```
canvas = Canvas(width, height)
```

Don't forget to import the Canvas function

```
from graphics import Canvas
```

Create Shapes

The following methods can be called on a canvas to create different shapes:

Rectangle
Create a new rectangle defined by the points (left_x,top_y) and (right_x,bottom_y).
First optional parameter is the color of the rectangle.
Second optional parameter is the outline color of the rectangle.
Returns the object id of the new rectangle.

```
# Black rectangle
rect = canvas.create_rectangle(
    left_x,
    top_y,
    right_x,
    bottom_y
)

# Filled rectangle
rect = canvas.create_rectangle(
    left_x,
    top_y,
    right_x,
    bottom_y,
    color
)

# Rectangle with an outline
rect = canvas.create_rectangle(
    left_x,
    top_y,
    right_x,
    bottom_y,
    color,
    outline
)
```

Oval

Create an oval inscribed in the bounding box defined by (left_x,top_y) and (right_x,bottom_y). First optional parameter is the color of the oval.
Second optional parameter is the outline color of the oval.
Returns the object id of the new circle.

```
# Black oval
rect = canvas.create_oval(
    left_x,
    top_y,
    right_x,
    bottom_y
)

# Filled oval
rect = canvas.create_oval(
    left_x,
    top_y,
    right_x,
    bottom_y,
    color
)

# Oval with an outline
rect = canvas.create_oval(
    left_x,
    top_y,
    right_x,
    bottom_y,
    color,
    outline
)
```

Line
Create a new line connecting (x1, y1) to (x2, y2).
Optional extra parameter is the color of the line.
Returns the object id of the new line.

```
# Black line
line = canvas.create_line(
    x1, y1,
    x2, y2
)

# Line with color
line = canvas.create_line(
    x1, y1,
    x2, y2,
    color
)
```

Text

Draw the given text on the screen anchored at the given (x, y) location.
Optional parameter: font,
Optional parameter: font_size
Optional parameter: color

```
# Text drawn at (x,y)
obj = canvas.create_text(
    x,
    y,
    text
)

# "My first text!"
canvas.create_text(
    x,
    y,
    text = 'My first text!',
    font = 'Arial',
    font_size = 50,
    color ='blue'
)
```

Image
First upload an image to your project (see the files tab).
Then you can add that image to the canvas.

```
# Image at a location
image = canvas.create_image(
    left_x,
    top_y,
    filename
)

# Image with size
image = canvas.create_image_with_size(
    left_x,
    top_y,
    width,
    height,
    filename
)
```

Polygon
Create a new polygon by passing through a series of points. Because coordinates is a list of arguments, a call to the function would look like this: polygon = canvas.create_polygon(10, 10, 20, 20, 10, 20, color="RED", outline="BLACK"). This would create a triangle with points at (10, 10), (20, 20), and (10, 20).

```
# Create a polygon
polygon = canvas.create_polygon(
```

```
    coordinates,
    color,
    outline
)

# Example triangle
polygon = canvas.create_polygon(
   10, 10, 20, 20, 10, 20,
   color="red",
)
```

Modify Shapes

The following methods can be called on a canvas to modify a given shape:

Move the object with given id dx pixels to the right and dy pixels down.

```
canvas.move(objectId, dx, dy)
```

Move the object with given id the the location (new_x, new_y)

```
canvas.moveto(objectId, new_x, new_y)
```

Delete an object and remove it from the canvas.

```
canvas.delete(objectId)
```

Set whether a shape is visible. If is_hidden is true, the object will become hidden, if it is false, it will become unhidden.

```
canvas.set_hidden(objectId, is_hidden)
```

Changes the text content of the text object referenced by objectId to the string passed in as new_text

```
canvas.change_text(objectId, new_text)
```

Canvas Helpers

The following methods can be called on a canvas:

Returns the current x coordinate of the mouse, measured in pixels from the left of the canvas.

```
x = canvas.get_mouse_x()
```

Returns the current y coordinate of the mouse, measured in pixels from the top of the canvas.

```
y = canvas.get_mouse_y()
```

Returns the location of the last mouse click since this function was called. Returns null if there have been no clicks

```
click = canvas.get_last_click()
```

Returns the last key which was pressed by the user on the keyboard. Returns null if there have been no key presses.

```
key = canvas.get_last_key_press()
```

Returns a list of objects which overlap with the rectangular region defined by (left_x,top_y) and (right_x,bottom_y)

```
objs = canvas.find_overlapping(
    left_x,
    top_y,
    right_x,
    bottom_y
)
```

Clears all objects from the canvas.

```
canvas.clear()
```

Returns the left most x coordinate of images, lines, rectangles, and ovals.

```
left_x = canvas.get_left_x(obj)
```

Returns the top most y coordinate of images, lines, rectangles, and ovals.

```
top_y = canvas.get_top_y(obj)
```

Returns the width of the object with the specified objectId on the canvas.

```
width = canvas.get_object_width(obj)
```

Returns the height of the object with the specified objectId on the canvas.

```
height = canvas.get_object_height(obj)
```

Sets the fill color of the object with the specified objectId on the canvas to the specified color. The color parameter can be any valid CSS color value.

```
canvas.set_color(obj, color)
```

Sets the outline color of the object with the specified objectId on the canvas to the specified color. The color parameter can be any valid CSS color value.

```
canvas.set_outline_color(obj, color)
```

Pauses the execution of the program until the user clicks somewhere on the canvas. Returns the location of the mouse click as an object with 'x' and 'y' properties, representing the coordinates of the mouse click on the canvas.

```
canvas.wait_for_click()
```

Returns an list of new mouse clicks since this function was last called. Each element in the list is an object with 'x' and 'y' properties, representing the coordinates of the mouse

click on the canvas. Returns an empty list if there have been no new clicks.

```
clicks = canvas.get_new_mouse_clicks()
```

Returns an list of new key presses since this function was last called. Each element in the list is a string representing the key pressed by the user. Returns an empty list if there have been no new key presses.

```
keys = canvas.get_new_key_presses()
```

Returns an list of coordinates of the object that is formatted as [left_x, top_y]

```
coords = canvas.coords(objectId)
```

Thats all folks!