# *tkinter Reference Guide*

### Common Usage and Functions

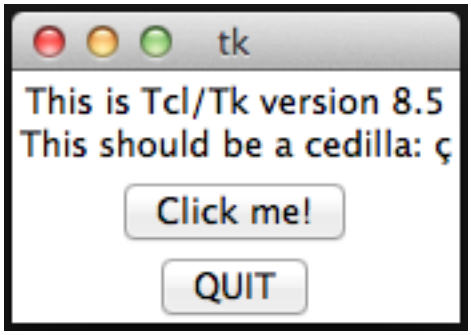

Installation on Your Local Machine . . . . . 2
Canvas Initialization . . . . . 3
The Coordinate System . . . . . 3
Drawing Lines . . . . . 4
Drawing Rectangles . . . . . 5
Drawing Ovals . . . . . 6
Drawing Polygons . . . . . 7
Drawing Text . . . . . 8

## Installation on Your Local Machine

It is very likely that your version of Python 3 came with `tkinter` already installed. You can check if your local machine has a working version of `tkinter` by writing and executing a simple Python program consisting of the two lines of code shown below:

```
import tkinter
tkinter._test()
```

Upon running this program, a window similar to the one pictured below should appear.




If this window appears, then you have a working version of `tkinter` installed and available to use in your programs!

If this window does not appear, then:

- Make sure that the function you called is `tkinter._test()`, with a period followed by an underscore as well as a set of empty parenthesis.
- If you are running a version of Python 2, then upgrade your version to Python 3 (this is the standard version for 15-110).
- If you cannot upgrade your version of Python, then capitalize all instances of `tkinter` in the above program and all programs you may write to `Tkinter` and try again.
- If none of the above have solved your problem, ask a question on Piazza with specific details of how your program is failing.

## Canvas Initialization

A few simple steps must be followed to create a window for your graphics in Python. First, be sure that both `tkinter` and the `Canvas` module within it have been imported to the program:

```
import tkinter
from tkinter import Canvas
```

Then, a display window must be created:

```
my_window = tkinter.Tk()
```

Now, using this window we can create a Canvas where we will draw our graphics:

```
my_canvas = Canvas(my_window, width = 400, height = 500)
```

Now, we will put our Canvas into the window to be displayed:

```
my_canvas.pack()
```

This will create a Canvas with the dimensions 400x500 pixels. These dimensions of the Canvas can be changed simply by changing the numbers used in its initialization.

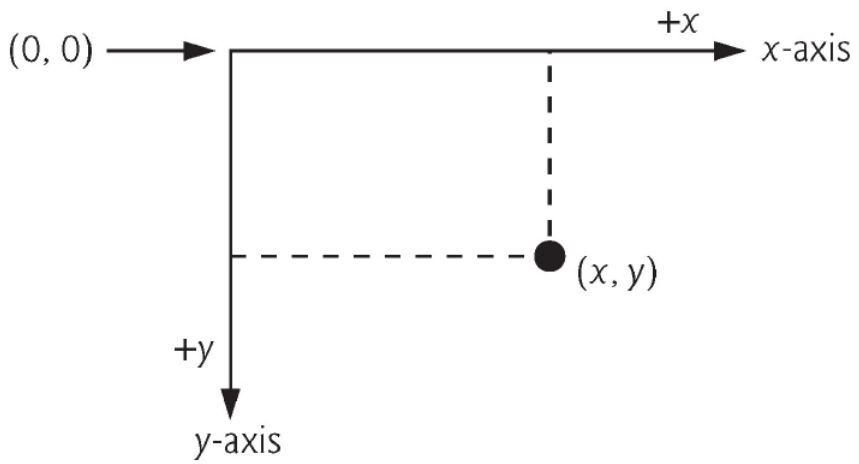
Now this Canvas, `my_canvas`, can be used to display shapes using the functions provided below!

## The Coordinate System

When working with graphics, it is important to remember that the coordinate system has its origin in the **top-left corner** of the screen! This means that the top left corner represents (0,0).

Because of this, x-coordinates increase as expected when moving to the right along the screen, but y-coordinates **increase** when moving **down** the screen.

This is illustrated in the image below:



## Drawing Lines

```
my_canvas.create_line(x0, y0, x1, y1)
```

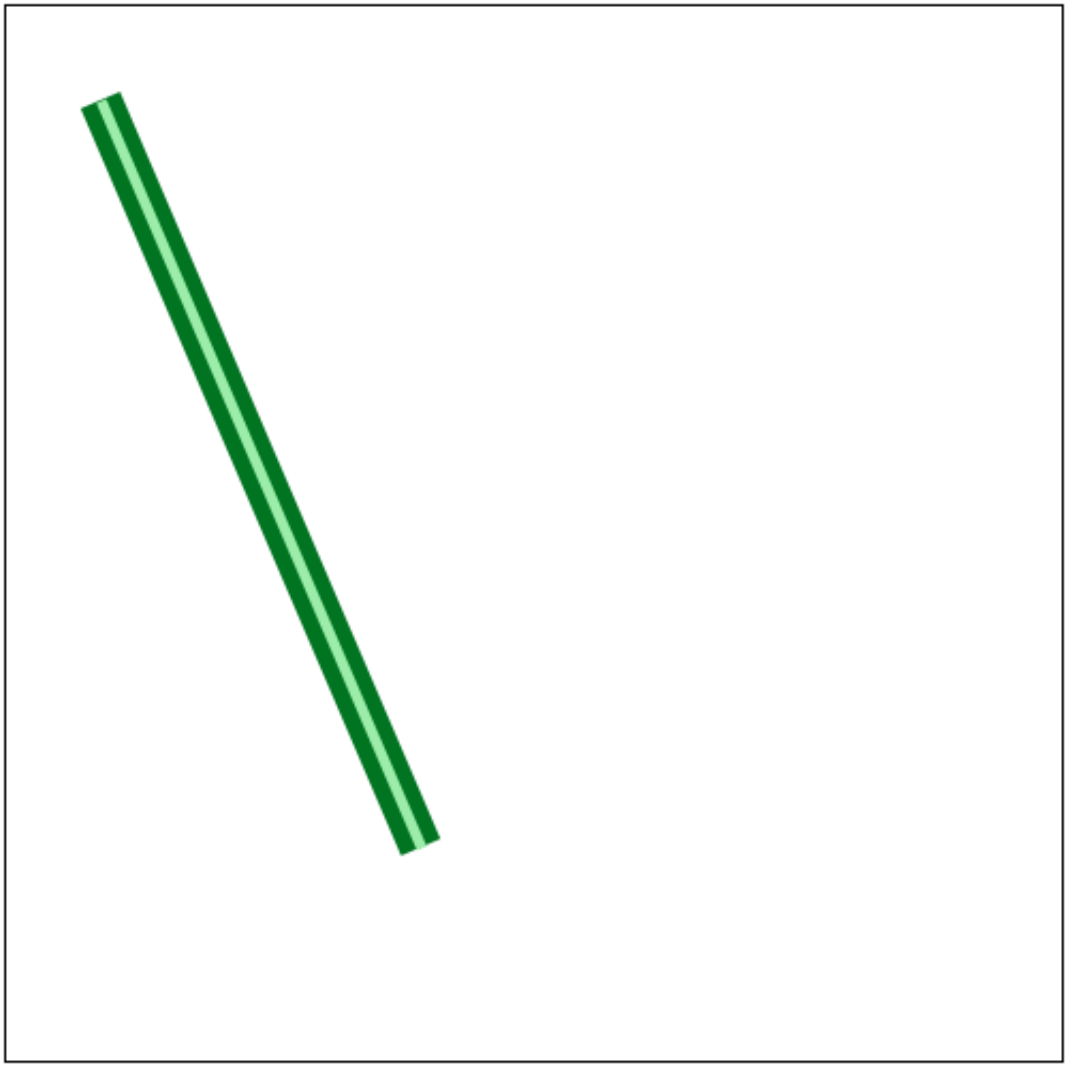
Draws the line connecting point $(x_0, y_0)$ to point $(x_1, y_1)$.

Optional parameters include:

- fill: Draws the line using the specified color. The default value is black.
- width: Sets the thickness of the line to the specified number of pixels. The default value is 1.

Example:

```
my_canvas.create_line(50,50,  200,400,  width=20, fill="dark green")
my_canvas.create_line(200,400,  50,50,  width=5,  fill="light green")
```



## Drawing Rectangles

```
my_canvas.create_rectangle(x0, y0, x1, y1)
```

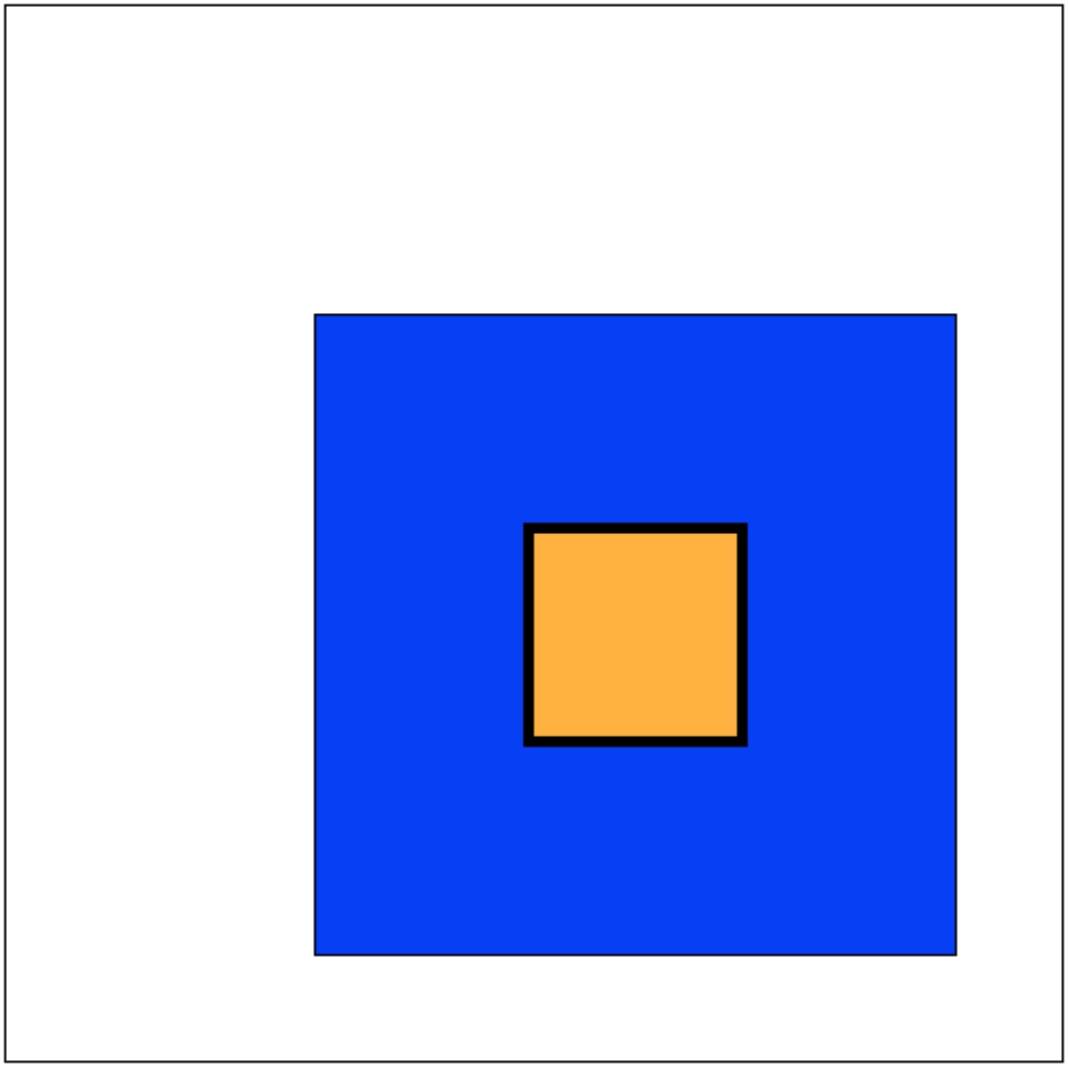
Draws the rectangle with upper left corner at point $(x_0, y_0)$ and lower right corner at point $(x_1, y_1)$.

Optional parameters include:

- fill: Fills the rectangle with the color specified. The default value is transparent.
- outline: Sets the color of the border to the color specified. The default value is black.
- width: Sets the border thickness to the specified number of pixels. The default value is 1.

Example:

```
my_canvas.create_rectangle(150,150,  450,450,  fill="blue", outline="orange")
my_canvas.create_rectangle(250,250,  350,350,  width=5, fill="orange")
```



## Drawing Ovals

```
my_canvas.create_oval(x0, y0, x1, y1)
```

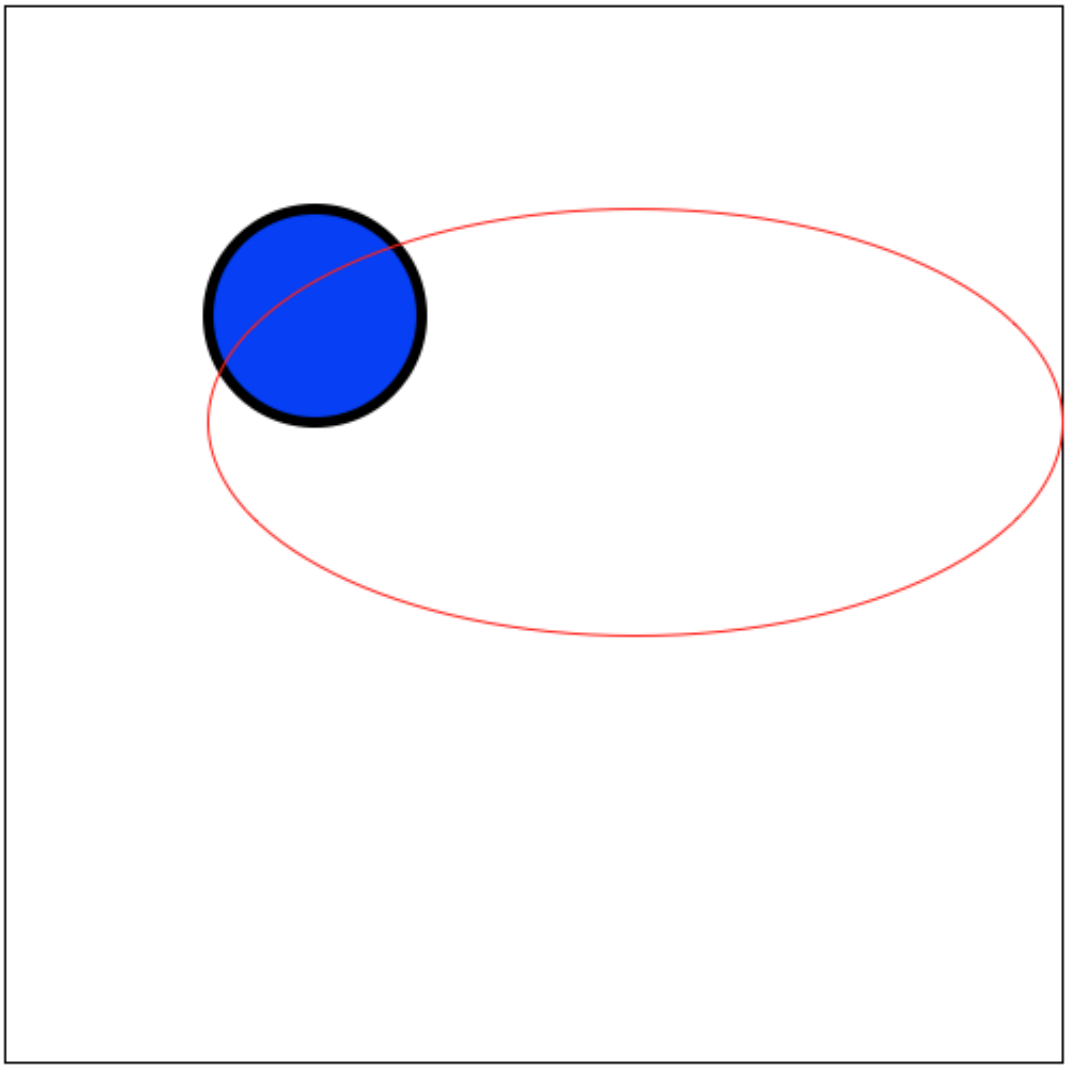
Draws the oval inscribed in the rectangle with upper left corner on the point $(x_0, y_0)$ and lower right corner on the point $(x_1, y_1)$.

Optional parameters include:

- fill: Fills the oval with the color specified. The default value is transparent.
- outline: Sets the color of the border to the color specified. The default value is black.
- width: Sets the border thickness to the specified number of pixels. The default value is 1.

Example:

```
my_canvas.create_oval(100,100,  200,200,  fill="blue", width = 5)
my_canvas.create_oval(100,100,  500,300,  outline = "red")
```



## Drawing Polygons

```
my_canvas.create_polygon(x0, y0, x1, y1, x2, y2, ...)
```

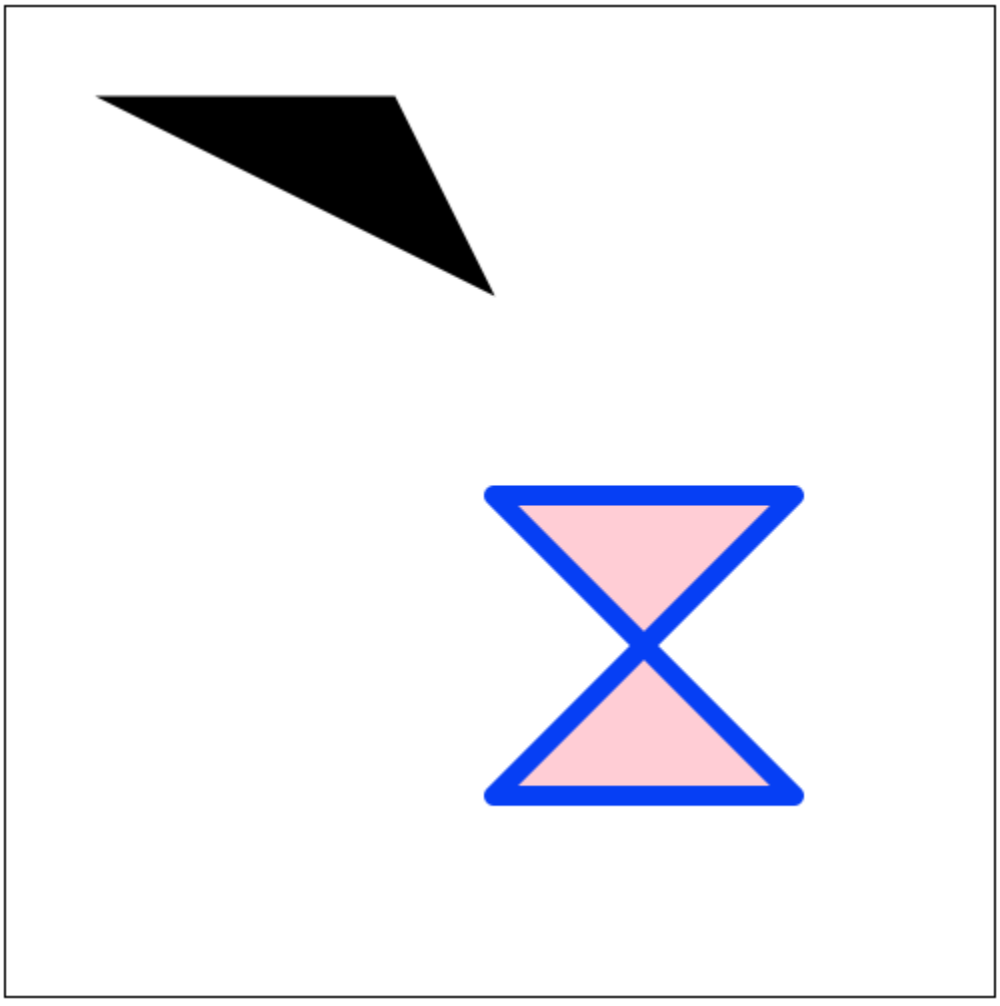
Draws the polygon defined by the lines connecting point $(x_0, y_0)$ to point $(x_1, y_1)$, point $(x_1, y_1)$ to point $(x_2, y_2)$, etc. The final point is then connected back to $(x_0, y_0)$.

Optional parameters include:

- fill: Fills the polygon with the color specified. The default value is **black**.
- outline: Sets the color of the border to the color specified. The default value is black.
- width: Sets the border thickness to the specified number of pixels. The default value is 1.

Example:

```
my_canvas.create_polygon(50,50,  200,50,  250,150)
my_canvas.create_polygon(250,250,  400,250,  250,400, 400,400,
                         fill="pink", outline="blue", width=10)
```



## Drawing Text

`my_canvas.create_text(x0, y0, text=my_text)`

Draws the text specified by `my_text`, centered on point $(x_0, y_0)$. Make sure to always specify the `text` optional parameter!

Optional parameters include:

- text: Defines the text which you want to be drawn. The default value is no text at all.
- anchor: Defines how the text will be positioned relative to the given point, $(x_0, y_0)$.

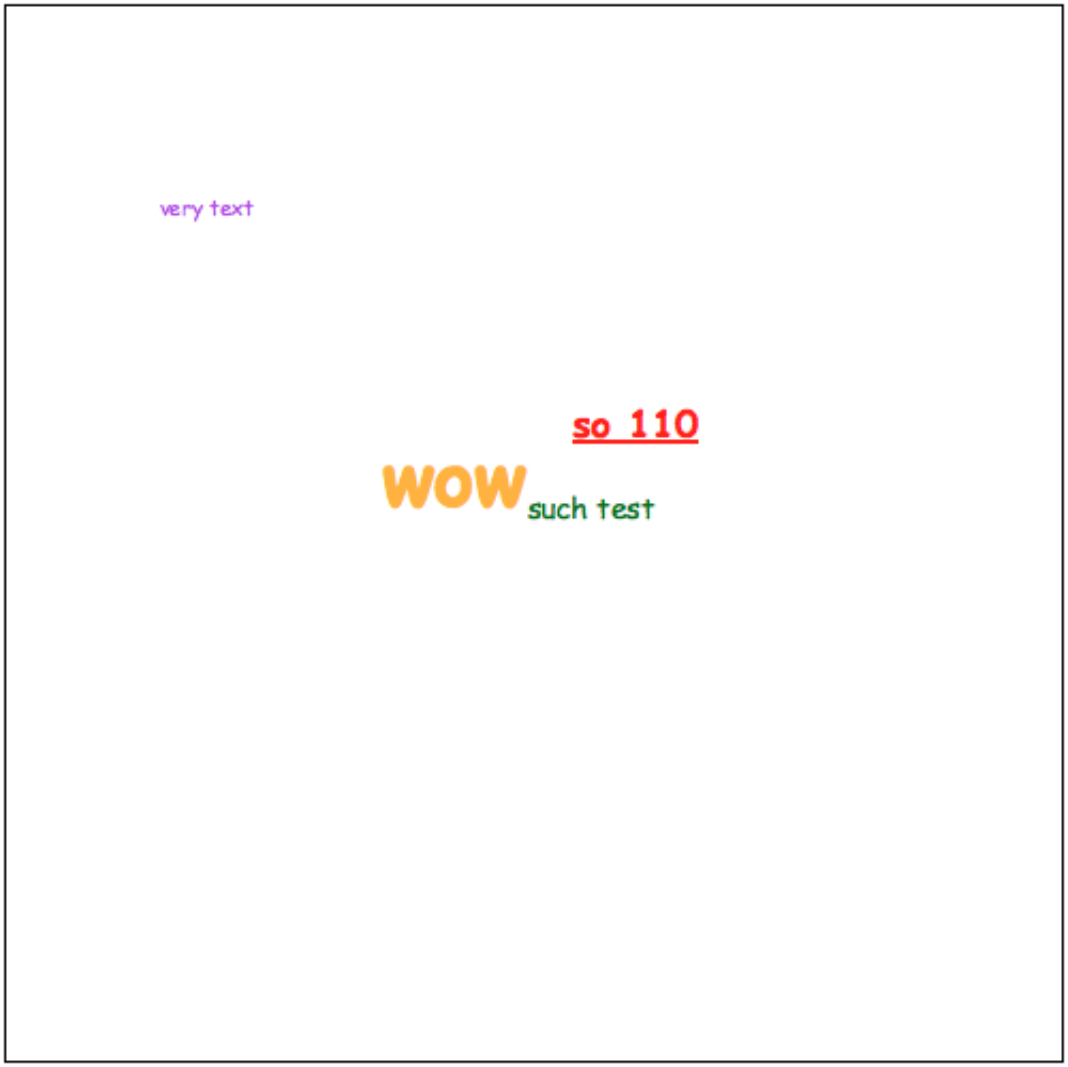
  The default value is "center", which will place the center point of all of the text on the point $(x_0, y_0)$.

  Other options include "n", "e", "s", or "w", to place the center of the northern, eastern, southern, or western edge of the text on the point $(x_0, y_0)$, respectively.

  Additionally, "ne", "se", "sw", or "nw" can be used to place the corresponding corner of the text on the point $(x_0, y_0)$.
- fill: Draws the text in the color specified. The default value is black.
- font: The value here must be a tuple consisting of a font name ("Times", "Helvetica", etc.), a font size in points (11, 12, 16, etc.), and optionally a string representing style ("bold", "italic", "underline", "bold italic", etc.). The text is then drawn in the font specified. The default may vary between machines.
- width: Specifies a maximum line length in pixels, at which point additional text will wrap to the line below. The default value disables all text wrapping.

Example:

```
c.create_text(100,100, text="very text",
              font=("Comic Sans MS", 10),
              fill="purple")
c.create_text(250,250, text="wow",
              anchor="se",
              font=("Comic Sans MS", 36, "bold"),
              fill= "orange")
c.create_text(250,250, text="such test",
              anchor="sw",
              font=("Comic Sans MS", 14),
              fill="dark green")
c.create_text(300,200, text="so 110",
              font=("Comic Sans MS", 18, "bold underline"),
              fill="red")
```


very text
so 110
wow
such test