

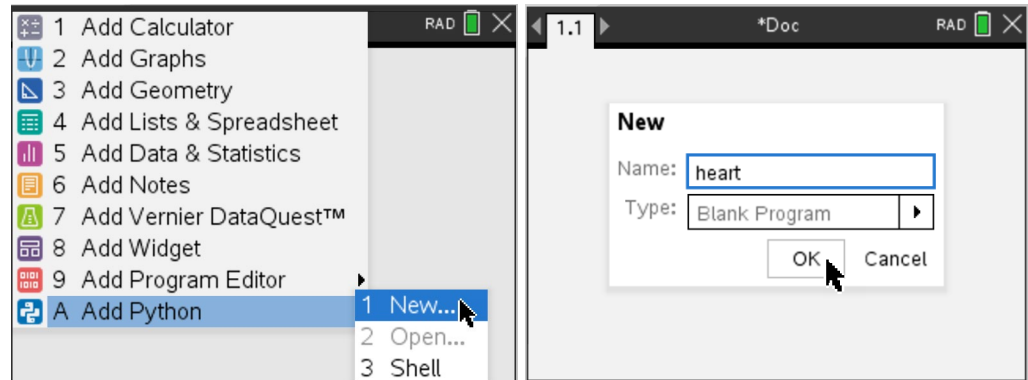
Getting Started

First, you will need to open a new document, add a Python editor and name the program.

***Note- the screenshots are shown for the TI-Nspire™ CX II handheld. If you are using a TI-84 Plus CE Python calculator, some modification will be necessary, but menus will be similar.*

Doc > File > New Document > Add Python > New

Pick a name for your program. Click on OK. I named my program, "heart".



Here we add a Python Editor.

Name the program

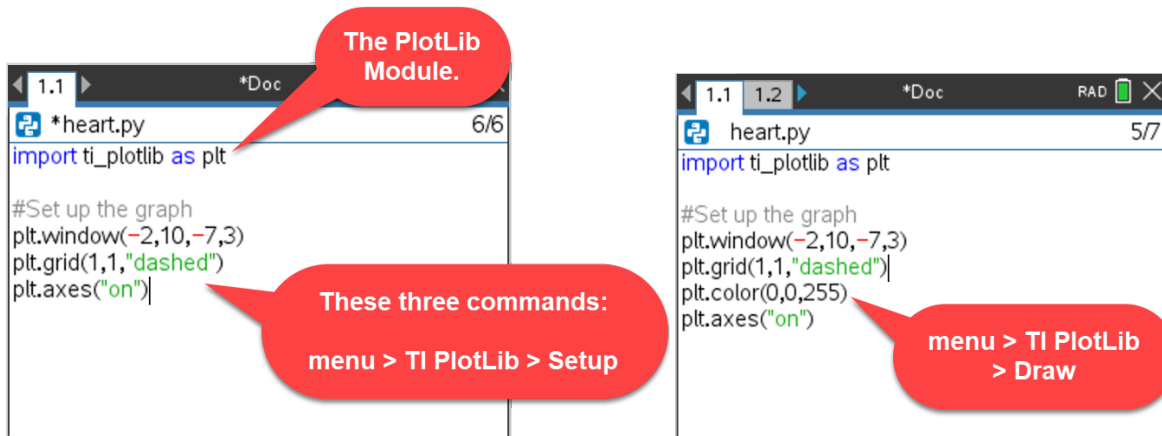
Building the Foundation:

First, we must lay the foundation for the activity. We will add in all the basic graphing elements. In Python, you need to define your window and put in your axes. To help see where the points are located, we will also add a grid to our graph. Since we don't want our graph to look like it came from a 1950's TV, we will also add some color.

We are going to use Python's **PlotLib** module. Python is designed to use as few resources as possible. Each module can add functions, classes and variables. Basically, each one adds more commands to be used. Think of Python as a giant bookshelf that only has a few books on it. Adding a module is like adding a new book and thus new information to the bookshelf. The PlotLib will add commands to let us set up and draw our graph.

Add the PlotLib module: menu > **TI PlotLib** > **import ti_plotlib as plt**

Now that we have access to the commands we need, add the grid, add the axes and pick a color.



Setting up the graph

Adding a little bit of color

Time to Play (Experiment and Learn)

Let's run the program and see what happens. To run a Python program, press **ctrl** then **R** (**ctrl + R**).

Does the order of the commands make a difference? Rearrange the commands and run the program again. What happens? Try more rearrangements. Why was the original order of the commands selected?

Can you make different colors? The color is based on a mixture of red, green, and blue. Each of the colors can vary in value from 0 to 255. What colors can you make? Can you make red? Can you make orange? Write down some colors you found and the numbers that go with them.

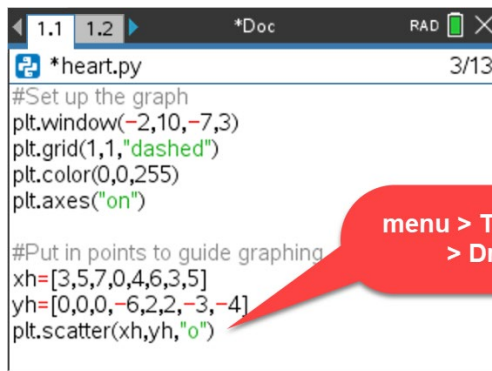
Preparing for the Picture (and Some Math):

We are going to shoot an arrow through a heart. So, we need to draw a heart, and if you are going to draw a picture you need dots. How else are you supposed to draw a picture if you don't have dots to connect? (My best artwork always involved books where you connect the dots.). We are going to graph our dots by graphing a scatterplot. First, we will type in two lists:

```
xh = [3,5,7,0,4,6,3,5]
yh = [0,0,0,-6,2,2,-3,-4]
```

I chose xh to be the name for the x-coordinates of the heart and yh to be the y-coordinates of the heart. You can use any names you want. Have fun. Personally, I went for short names.

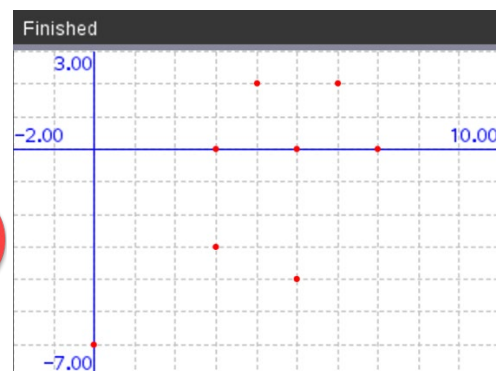
Using the lists, we can now draw the scatterplot.



```
*heart.py 3/13
#Set up the graph
plt.window(-2,10,-7,3)
plt.grid(1,1,"dashed")
plt.color(0,0,255)
plt.axes("on")

#Put in points to guide graphing
xh=[3,5,7,0,4,6,3,5]
yh=[0,0,0,-6,2,2,-3,-4]
plt.scatter(xh,yh,"o")
```

Code to add dots



Can you make your dots red?

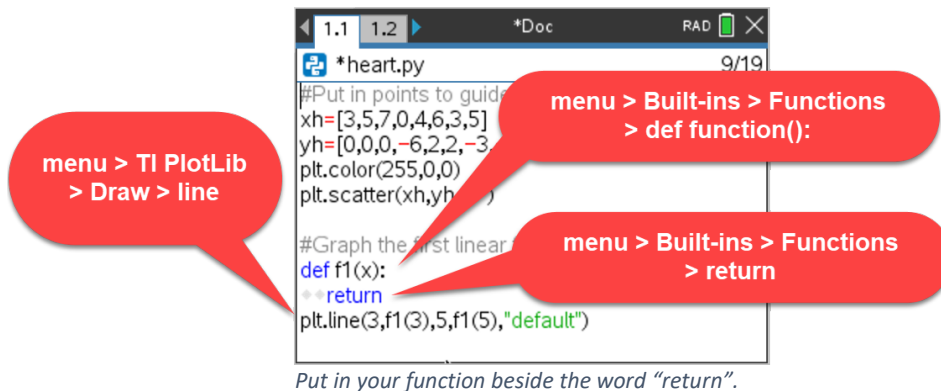
Time to Play (Experiment and Learn)

Put in the code and run the program. (**ctrl + R**).

Can you make the dots red? What command did you use and where did you put it?

Drawing the First Linear Function from (3,0) to (5,-4):

The first function is going to draw a line from the point (3,0) to the point (5,-4). This will be left side of the bottom of the heart



Note about the command `plt.line`: The 3 and `f1(3)` are the x,y-coordinates of the first point of the line segment being plotted, and the 5 and `f1(5)` are the x,y-coordinate of the last point.

Time to Play (Experiment and Learn)

What linear function passes through the points (3, 0) and (5, -4)?

Check your work. Write your function beside the "return" command and run the program.

(Note about how to write your function: Multiplication is represented by *. So, a linear function would be written $5*x+7$. By the way, $5*x+7$ is the wrong function.)

What happened? If your function did not connect the dots, make some changes and try again. Coding is always a process of testing and correcting code.

Draw the Second Linear Function from (5,-4) to (7,0):

The second linear function is going to draw a line from the point (5,-4) to the point (7,0). This will be right side of the bottom of the heart.

```
*heart.py
#Put in points to guide
xh=[3,5,7,0,4,6,3,5]
yh=[0,0,0,-6,2,2,-3,0]
plt.color(255,0,0)
plt.scatter(xh,yh)

#Graph the first linear
def f1(x):
    return
plt.line(3,f1(3),5,f1(5),"default")
```

The second function code will be similar.

Graph after adding the two linear functions

Time to Play (Experiment and Learn)

What linear function passes through the points (5, -4) and (7, 0)?

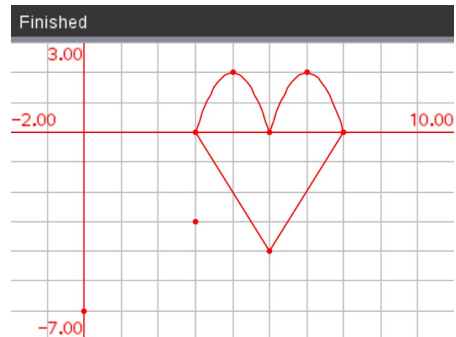
What changes need to be made to the code for the first linear function to make it draw the second linear function?

Check your work. Put in all of your code and run the program.

What happened? If your function did not connect the dots, make some changes and try again. The only mistake is when you stop trying.

Draw the Top of the Heart: Two Quadratic Functions (through (3,0), (4,2), (5,0) and (5,0), (6,2), (7,0)):

Again, we are going to use the function structure and the plot line command. There is not a predefined command to graph curved functions in Python, but they are still easy to graph. We are going to use a “for loop” to draw a lot of very short little lines to produce our quadratic functions.



Here is the code for the first quadratic function (through (3,0), (4,2), (5,0)):

```
#graph the first quadratic function
def f3(x):
    ♦♦ return
    for i in range(30,50):
        ♦♦ plt.line(i/10,f3(i/10),(i+1)/10,f3((i+1)/10),"default")
```

menu > Built-ins
> Control

Your quadratic function goes here.

Notes about the Quadratic code: It may seem strange at first for the range to run from 30 to 50 in the “for loop”, but Python does not allow steps that are not integers. So, if you want to take a small step, you have to be creative. We want to run a graph from 3 to 5 with steps of a tenth. So, we go from 30 to 50 with steps of 1 and then divide each i by 10.

Time to Play (Experiment and Learn)

What quadratic function passes through the points (3,0), (4,2), (5,0)?

Check your work. Put in the code with your function beside the “return” command and run the program.
(Note about how to write your function: Multiplication is represented by * and powers are represented using **. So, a quadratic function would be written $2*(x-7)**2+3$ or $2*x**2+3*x+7$. By the way, these are not the correct functions.)

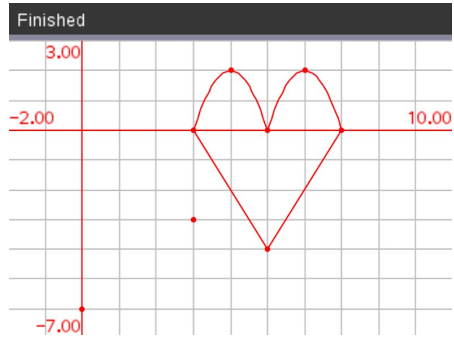
What happened? If your function did not connect the dots, make some changes and try again. Making mistakes is just a part of learning.

Here again is the code for the first quadratic function (through (3,0), (4,2), (5,0)). We are going to update this code to draw the second quadratic function:

```
#graph the first quadratic function
def f3(x):
    return
for i in range(30,50):
    plt.line(i/10,f3(i/10),(i+1)/10,f3((i+1)/10),"default")
```

menu > Built-ins
> Control

Your quadratic function goes here.



What quadratic function passes through the points (5,0), (6,2), (7,0)?

What changes need to be made to the code for the first quadratic function to make it draw the second one?

Check your work. Put in your code, write your function beside the “return” command and run the program.

What happened? If your function did not connect the dots, make some changes and try again. It will be rewarding to get the correct answer!

Firing the Arrow Through the Heart:

First, we need an arrow (a linear function). You are an old pro at graphing linear functions at this point. So, here we go.

Time to Play (Experiment and Learn)

What linear function passes through the points (0, -6) and (3, -3)?

How do you code this line so that it will plot in the correct place?

Check your work. Add in all your code and run the program.

What happened? If your function did not connect the dots, make some changes and try again. Always believe that you will be able to make the code work!

This part of the code is fun. We are going to make the arrow move. Here is the code:

```
#make the arrow move
for i in range(30,130,1):
    plt.color(255,0,0)
    plt.pen("thin","solid")
    plt.line(i/10,f5(i/10),(i+1)/10,f5((i+1)/10),"default")
    plt.a=i-30
    plt.color(255,255,255)
    plt.pen("medium","solid")
    plt.line(a/10,f5(a/10),(a+1)/10,f5((a+1)/10),"default")
    plt.sleep(0.01)
```

Code to fire the arrow

Time to Play (Experiment and Learn)

Run the code. If the arrow is moving too fast, you can slow it down.

Go back to the top of the program and add the time module: **menu > More Modules > Time > from time import***

You slow the loop down by adding a sleep command. **menu > More Modules > Time > sleep(seconds)**

Seconds is a number. A very small number works best (less than one). Where in the program do you add the command? Experiment with command's placement and the number of seconds until you are happy with the results.

Have a Happy Valentine's Day!