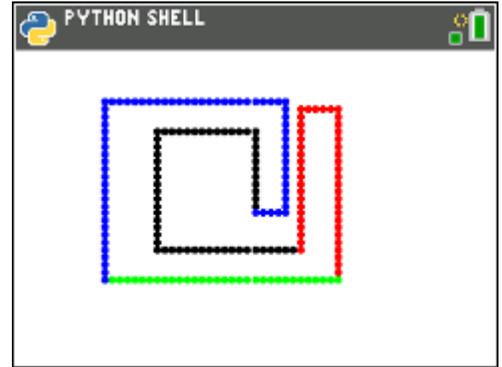




The most versatile tool in the TI-84 Plus CE Python library is the `wait_key()` function. In this activity you will make use of this function to make an interactive 'Etch-A-Sketch'® style drawing program.

- 0. Using the `wait_key()` function found in the `ti_system` module it is possible to make dynamic, *interactive* graphical applications (and games!). In this activity you will build a simple drawing program that begins by using the four arrow keys and can be extended to make use of other keys for changing color, point style, making 'stamps' and other features.



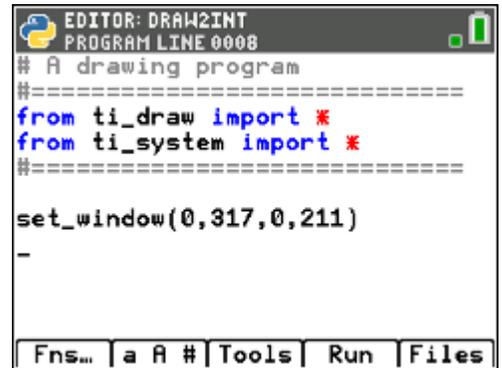
- 1. Begin a new program and add **from ti\_draw import \***

found in <Fns...> Modul <Add-Ons>

Also import the `ti_system` module to use the `wait_key()` function.

The first statement sets up a comfortable canvas coordinate system with (0,0) in the *lower left corner* and each pixel representing one unit:

**set\_window(0, 317, 0, 211)**



- 2. Start in the center of the screen. Assign 159 to the variable `x` and 105 to the variable `y`. This can be done in a single line:

**x, y = 159, 105**

We usually use the statement `while not escape():` to end a program, but in this project we need to *test* the value of `wait_key()` to see which key is pressed then act accordingly, so we will assign `wait_key()` to a variable named `k`. First assign 0 to the variable `k`:

**k = 0**

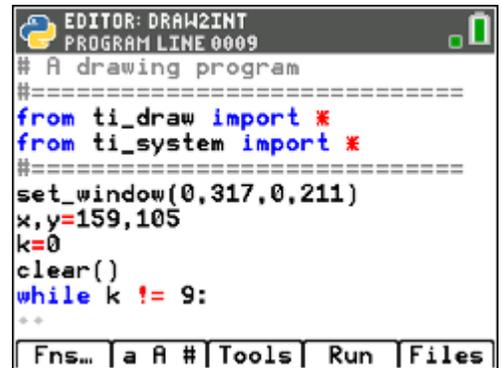
Before the `while` loop, use the `clear()` function found on [math] ti\_draw... Control to clear the screen:

**clear()**

Then write the `while` loop that terminates when `k` is 9.

**while k != 9:**

*9 is the value that wait\_key() returns when pressing the [clear] key.*





- In the loop body, plot the point (x, y) using the `ti_draw` function

**plot\_xy(x, y, 1)**

*Note: the third argument is the point 'style' which can be a value from 1 to 13. You can try other styles.*

Now assign the variable `k` the result of the `wait_key()` function:

**k = wait\_key()**

`wait_key()` returns a unique number for each key on the keypad.

- You can **<Run>** the program now to see a dot in the center of the screen, but it does not move yet.

Press **[clear]** to end the program then return to the **<Editor>**.

- Check the value of `k` and cause the point to be repositioned at a new location depending on the key pressed. The four arrow keys on the keypad are numbered 1: "right", 2: "left", 3: "up", and 4: "down" and each one will impact either the variable `x` or the variable `y`.

The first of four `if` statements is:

**if k == 1:  
    x += 5**

*If the 'rightarrow' key is pressed add 5 to the value of x.*

You could also write `x = x + 5` (or some other number).

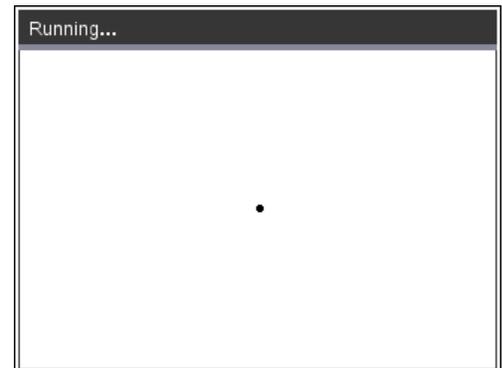
Write the other three `if` statements now.

- Did you write these three?

**if k == 2:  
    x -= 5  
if k == 3:  
    y += 5  
if k == 4:  
    y -= 5**

```

EDITOR: DRAW2INT
PROGRAM LINE 0014
=====
set_window(0,317,0,211)
x,y=159,105
k=0
clear()
while k != 9:
    plot_xy(x,y,1)
    k=wait_key()
  
```



```

EDITOR: DRAW2INT
PROGRAM LINE 0016
=====
x,y=159,105
k=0
clear()
while k != 9:
    plot_xy(x,y,1)
    k=wait_key()
    if k==1:
        x+=5
  
```

```

EDITOR: DRAW2INT
PROGRAM LINE 0012
=====
plot_xy(x,y,1)
k=wait_key()
if k==1:
    x+=5
if k==2:
    x-=5
if k==3:
    y+=5
if k==4:
    y-=5
  
```

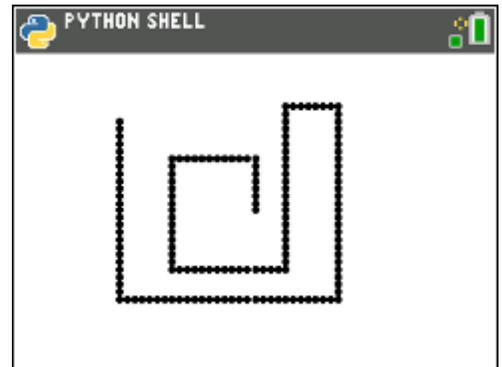


# 10 MOC: Python Modules

## TI-84 PLUS CE PYTHON

- 7. Test your program now. When you see the dot, use the four arrow keys to draw.

Again, press **[clear]** to end the program.



- 8. **Side note:** Almost all keys on the keypad return a value like the arrow keys. All values are numbers. To determine these numbers, you could write this short program:

**[2nd]** and **[alpha]** do not give values because they are *modifier* keys that change the behaviors of some of the other keys. Make notes about which keys produce which values.



- 9. You can use other keys to add features to your program. Many other features are possible. Just add an **if** statement to your program to incorporate the feature. Some possibilities:

- use three other keys and the **set\_color( , , )** function to change the drawing color to red, green, or blue.
- Use another key to **clear( )** the screen (but not the **[clear]** key – that’s already taken)
- Use another key to switch to erase mode (set the color to white, the background color). But you will need some other key to return to black
- Add a key to make a stamp (“s”?). The stamp can be something like a polygon or a circle.
- Change the increment (we originally used 5 in this activity) values in your code to make the point move more or less for each keypress as seen here where 1 was used to make it look like solid lines.

