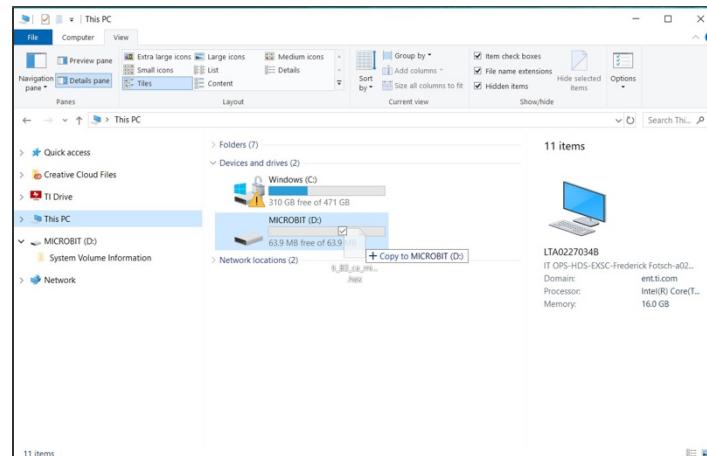


# Getting Started Guide

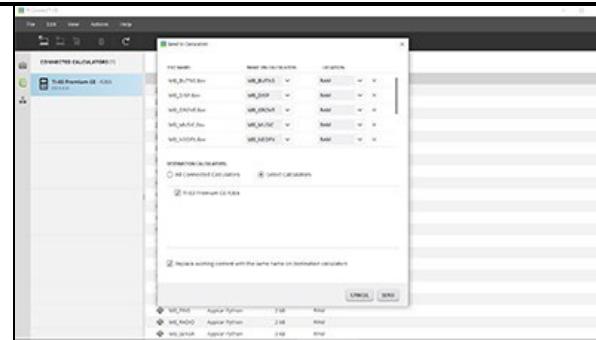
## Installation

1. Use a USB cable to connect the micro:bit to a computer. The micro:bit will appear as a drive on your computer. Drag and drop the `TI_Runtime_for_microbit.hex` file to the micro:bit drive. The file will copy to the micro:bit and provide enhanced functionality between the TI-84 Plus CE-T PYTHON EDITION and the micro:bit.

The runtime installation is an onetime process. If the micro:bit is connected back to the PC and programmed in a different language other than python, such as MakeCode, JavaScript, C++, or Scratch, the `ti_ce_runtime.hex` will need to be installed again. See additional information at the end of the document.



2. Use TI-Connect to transfer all of the micro:bit modules from your computer to the calculator.



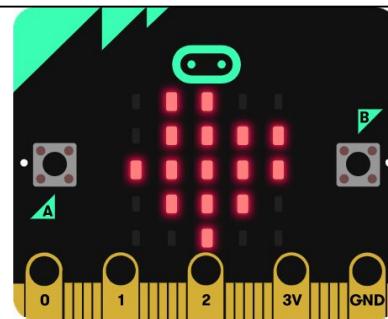
3. Connect the micro:bit to the TI-84 Plus CE-T PYTHON EDITION calculator using the unit-to-unit cable and a mini USB female to micro USB male adapter.

Connect the adapter to the B-end of the cable, and plug the A-end into the calculator's USB port.



# Getting Started Guide

4. Press the reset button on the micro:bit. If the runtime has loaded successfully, the message TI-84 Plus CE-T PYTHON EDITION will scroll across the micro:bit LED display, followed by the TI logo.



5. Open the python application. The micro:bit modules do not appear in the file manager; they are stored in archive memory.

FILE MANAGER		NORMAL FLOAT AUTO REAL RADIAN MP	
(no programs loaded)		RAM FREE 130171 ARC FREE 1260K MB_BUTNS 1643 MB_DISP 4193 MB_GROVE 3705 MB_MUSIC 2512 MB_NEOPX 1770 MB_PINS 1725 MB_RADIO 1934 MB_SENSR 4284	
<b>Run</b>	<b>Edit</b>	<b>New</b>	<b>Shell</b> <b>Manage</b>

## Creating Your First micro:bit Program

1. Select [New] to create a script with an appropriate name.

FILE MANAGER		EDITOR: HEART	
NEW PROGRAM Name:HEARTA		PROGRAM LINE 0001 -	
Allowed - Up to 8 characters - First character:A-Z - Remaining characters:A-Z 0-9 -		Fns... a A # Tools Run Files	
Optional <b>Esc</b> <b>Types</b> <b>Ok</b>			

2. Select [2<sup>nd</sup>][catalog] and choose from **PROGAM import \***

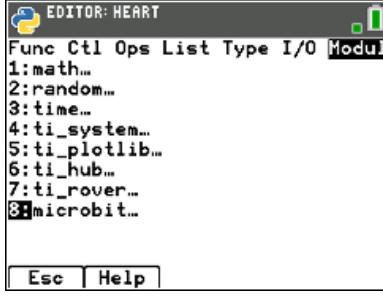
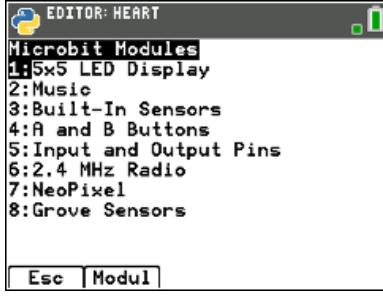
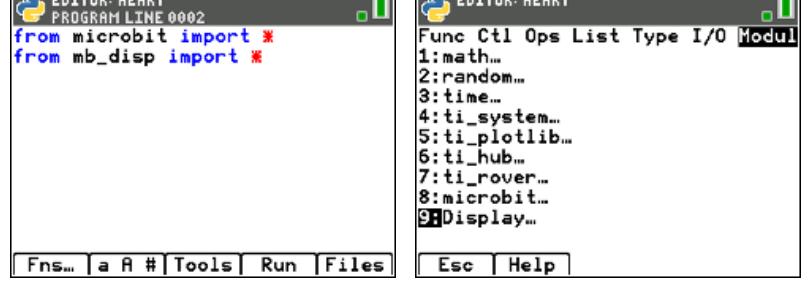
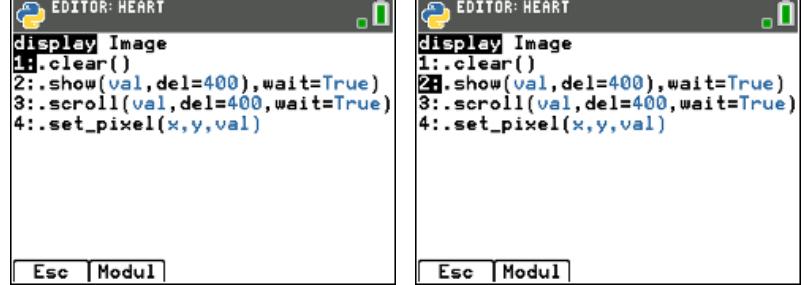
EDITOR: BOXTST		EDITOR: HEART	
CATALOG from PROGRAM import *		PROGRAM LINE 0001 from import *	
from math import * from random import * from time import * from ti_system import * from ti_hub import * global grid(xscl,yscl,"style") grid(xscl,yscl,"style", (r,g,b)) hex(integer) if ..		Fns... a A # Tools Run Files	
<b>Esc</b>			

3. Position cursor after **from** and enter **microbit**. Select [a A #] and select then paste **microbit** into the **from \_ import \*** statement.

Alternatively, use the alpha keys.

EDITOR: HEART		EDITOR: HEART	
PROGRAM LINE 0001 microbit_		PROGRAM LINE 0002 from microbit import *	
<pre># " ! : ; . ! ? - \ " a b c d e f g h i j k l m n o p q r s t u v w x y z  () [] {} * ** % //  = == != &lt; &lt;= &gt; &gt;= and or not True False &lt;&lt; &gt;&gt; &amp;   ^ ~</pre>		Fns... a A # Tools Run Files	
<b>Esc</b> <b>a A #</b> <b>Select</b> <b>Paste</b>			

# Getting Started Guide

<p>4. Select [enter] to go to the next line in the script after the import statement. This action will import the module into the editor and add the microbit menu option to the bottom of the module list.</p> <p>Select[Fns...] and select <b>Modul</b> tab and press [enter]. The menu will display all of the microbit modules.</p>	
<p>5. Select the <b>5x5 LED Display</b> module needed for the new script. The import for the selected module is added to the script. If additional modules are required, return to this microbit menu to add them.</p> <p>Choose only modules that are essential to the script to conserve RAM. You may choose other modules to include except the TI-Innovator Hub.</p>	
<p>6. The module named mb_disp is added to the script. Select the [Fns] key and select <b>Modul</b>. Notice the new <b>Display...</b> menu item is now at the bottom of the list. All of the other microbit modules work in this way.</p>	
<p>7. Select <b>Display...</b> and then <code>display.show(val,del=400,wait=True)</code> Ensure the cursor is positioned to the left of the comma in the command. <code>display.show(_ ,del=400,wait=True)</code></p>	
<p>8. Select the [Fns] key and select <b>Modul</b>. Select the <b>Display...</b> and then <b>Image</b>. Select <b>HEART</b> to paste "Image.HEART" into the script.</p>	



# Getting Started Guide

9. Select [Fns...], and **I/O** then print(). Type in "I Heart Python".

The print() statement illustrates how microbit statements are used along with standard or TI module python statements within the same script.

```
EDITOR: HEART
PROGRAM LINE 0004
from microbit import *
from mb_disp import *

display.show("Image.HEART",_delay
=400,wait=True)
```

```
EDITOR: HEART
PROGRAM LINE 0006
from microbit import *
from mb_disp import *

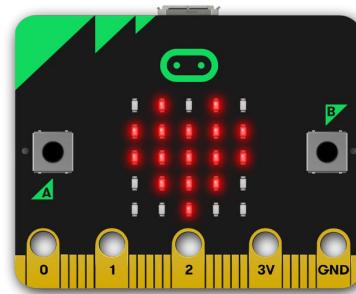
display.show("Image.HEART",_delay
=400,wait=True)
print("I Heart Python")_
```

10. Be sure the micro:bit is connected to the calculator using the unit-to-unit cable and mini->micro adapter and then select [Run].

Congratulations! You have programmed the micro:bit with the TI 84 Plus CE-T PYTHON EDITION!

```
PYTHON SHELL
>>> # Shell Reinitialized
>>> # Running HEART
>>> from HEART import *
>>> |
```

```
PYTHON SHELL
>>> # Shell Reinitialized
>>> # Running HEART
>>> from HEART import *
I Heart Python
>>> |
```



## Modules and Methods

### Microbit Modules

- 5x5 LED Display - imports **mb\_music**
- Music - imports **mb\_music**
- Built-In Sensors - imports **mb\_sensr**
- A and B Buttons - imports **mb\_butns**
- Input and Output Pins - imports **mb\_pins**
- 2.4 MHz Radio - imports **mb\_radio**
- NeoPixel - imports **mb\_neopx**
- Grove Sensors - imports **mb\_grove**

```
EDITOR: HEART
PROGRAM LINE 0002
from microbit import *
```

```
EDITOR: HEART
Microbit Modules
1:5x5 LED Display
2:Music
3:Built-In Sensors
4:A and B Buttons
5:Input and Output Pins
6:2.4 MHz Radio
7:NeoPixel
8:Grove Sensors

Esc Modul
```

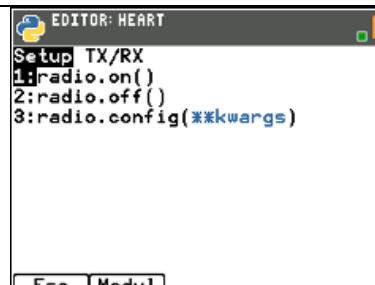
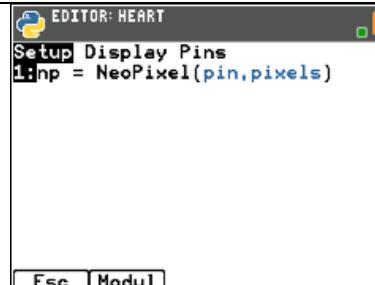


# Getting Started Guide

<p><b>5x5 LED Display Module</b></p> <p><b>display</b></p> <pre>display.clear() display.show(value,delay=400,wait=True) display.scroll(value,delay=400,wait=True) display.setpixel(x,y,value)</pre> <p><b>Image</b></p> <pre>var=Image(string) Image.NAME</pre>	<pre>EDITOR: HEART display Image 1:.clear() 2:.show(val,delay=400) 3:.scroll(val,delay=400,wait=True) 4:.set_pixel(x,y,val)</pre>	<pre>EDITOR: HEART display Image 1:var = Image(string) 2:HEART 3:HEART_SMALL 4:HAPPY 5:SMILE 6:SAD 7:CONFUSED 8:ANGRY 9:ASLEEP 0:SURPRISED</pre>
<p><b>Music Module</b></p> <pre>music.play(value) music.pitch(frequency, duration, wait = 400) music.set_tempo(ticks,BPM)</pre>	<pre>EDITOR: HEART Play Songs 1:music.play(song,wait=True) 2:music.pitch(f,t,wait=True) 3:music.set_tempo(ticks,BPM)</pre>	<pre>EDITOR: HEART Play Songs 1:DADADADUM 2:ENTERTAINER 3:PRELUDE 4:ODE 5:NYAN 6:RINGTONE 7:FUNK 8:BLUES 9:BIRTHDAY 0:WEDDING</pre>
<p><b>Built-In Sensors Module</b></p> <pre>var = accelerometer.get_x() var = accelerometer.get_y() var = accelerometer.get_z() var,var,var = accelerometer.get_values() var = accelerometer.magnitude() var = compass.heading() var = compass.get_field_strength() var = compass.is_calibrated() compass.clear_calibration() compass.calibrate() <i>Follow instruction on micro:bit screen to complete calibration.</i> var = temperature() <i>sensor on front of board</i> var = display.read_light_level() <i>sensor on back of board</i></pre>	<pre>EDITOR: HEART Compass Accel Temp-Light 1:var= .heading() 2:var= .is_calibrated() 3:var= .get_field_strength() 4:.calibrate() 5:.clear_calibration()</pre>	<pre>EDITOR: HEART Compass Accel Temp-Light 1:var= .get_x() 2:var= .get_y() 3:var= .get_z() 4:var,var,var= .get_values() 5:var= .magnitude()</pre>
	<pre>EDITOR: HEART Compass Accel Temp-Light 1:var=temperature() 2:var=display.read_light_level()</pre>	
<p><b>A and B Buttons Module</b></p> <pre>var = button_a.is_pressed() var = button_a.was_pressed() var = button_a.get_presses() var = button_b.is_pressed() var = button_b.was_pressed() var = button_b.get_presses()</pre>	<pre>EDITOR: HEART Button A Button B 1:button_a.is_pressed() 2:button_a.was_pressed() 3:button_a.get_presses()</pre>	<pre>EDITOR: HEART Button A Button B 1:button_b.is_pressed() 2:button_b.was_pressed() 3:button_b.get_presses()</pre>



# Getting Started Guide

<b>Input and Output Pins Module</b> <pre>var = pin0.read_digital() var = pin0.read_analog() pin0.write_digital(value) pin0.write_analog(value) var = pin1.read_digital() var = pin1.read_analog() pin1.write_digital(value) pin1.write_analog(value) var = pin2.read_digital() var = pin2.read_analog() pin2.write_digital(value) pin2.write_analog(value)</pre>	 	
<b>2.4 MHz Radio Module</b> <pre>radio.on() radio.off() radio.config(channel=7,power=6,group=0) <i>Two radios must share channel and group to communicate.</i> radio.sent(message) var = radio.receive()</pre>		
<b>NeoPixel Module</b> <pre>np = NeoPixel(pin = pin0, pixels = 16) <i>This constructor is optional to change pin and/or number of pixels on device. Maximum is 16.</i> Current is regulated to not exceed 90mA. np[index] = (red,green,blue) np.show() np.clear()</pre>	 	

# Getting Started Guide

## Grove Sensors Module

```
var = grove.read_sht35()
var = grove.read_temperature(pin)
var = grove.read_lightlevel(pin)
var = grove.read_temperature(pin)
var = grove.read_moisture(pin)
var = grove.read_pressure(pin)
var = grove.read_ranger_time(pin)
var = grove.read_ranger_cm(pin)
grove.set_power(pin,pwr)
grove.set_relay(pin,state)
pin = pin0, pin1, pin2, pin8, pin16
```

## EDITOR: HEART

```
Input Output Pins
1:var(t),var(h)=.read_sht()
2:var=.read_temperature(pin)
3:var=.read_lightlevel(pin)
4:var=.read_moisture(pin)
5:var=.read_pressure(pin)
6:var=.read_ranger_time(pin)
7:var=.read_ranger_cm(pin)
```

Esc Modul

## EDITOR: HEART

```
Input Output Pins
1:grove.power(pin,value)
2:grove.relay(pin,value)
```

Esc Modul

## EDITOR: HEART

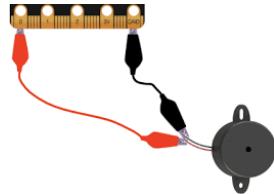
```
Input Output Pins
1:pin0
2:pin1
3:pin2
4:pin8
5:pin16
```

Esc Modul

## Example Test Programs

**DISPTEST.8XV:** Demonstrates all of the 5x5 matrix commands.

**MUSCTEST.8XV:** Demonstrates all of the music and tone commands. A speaker must be connected to pin0 and ground.



**SNSRTEST.8XV:** Demonstrates the compass, accelerometer, temperature, and light sensors. Follow the directions on the microbit to calibrate the compass.

**BUTNTTEST.8XV:** Demonstrates all of the A and B button commands.

**PINSTEST.8XV:** Demonstrates the analog and digital inputs. Use an alligator clip to connect 3V or Gnd alternately to the pins 0,1, and 2. Connect an LED to Gnd and alternately to pins 0,1,2 for analog and digital outputs.

**RADITEST.8XV:** Demonstrates the radio commands. Two or more calculators with micro:bits attached must run this program concurrently for the demonstration.

**NPTEST.8XV:** Demonstrates the NeoPixel commands. To use TI-RGB array with this test, make the following connections:



Microbit

3V

GND

Pin0 (can be changed in the constructor)  
pin1.read\_analog()

TI-RGB Array

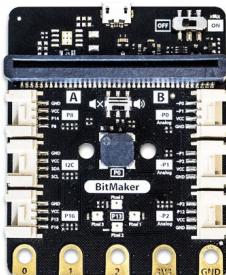
5V (Red)

(BLACK)  
BB2(YELLOW)  
BB5(BLUE)

# Getting Started Guide

**GROVTEST.8XV:** Demonstration requires an expansion board with Grove receptacles and the corresponding Grove sensors. The SHT35 Temperature and Humidity sensor must be plugged into an I2C port. Check the program for other sensor locations.

## Resources



Expansion board with Grove ports

<https://www.seeedstudio.com/BitMaker-p-4353.html>



Speaker with PCB mount pins

<https://www.jameco.com/webapp/wcs/stores/servlet/ProductDisplay?langId=-1&storeId=10001&catalogId=10001&productId=135722>



USB Mini to USB Micro adapter

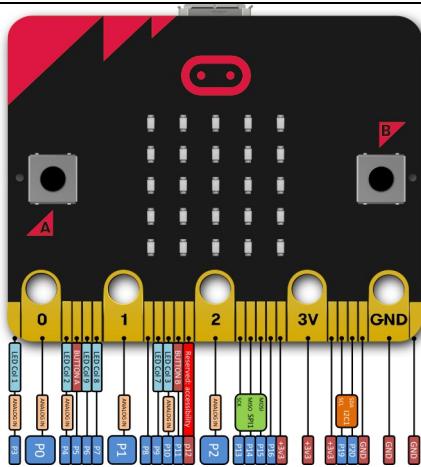
[https://www.sfcable.com/usb-micro-male-to-mini-5pin-female-adapter.html?gclid=CjwKCAjw1v\\_0BRAkEiwALFkj5pILcMI-JI\\_cWTZPnnjpWpM2Tinpp59tJSMTVlpMszd2ZwiErYtgiRoCRXwQAvD\\_BwE](https://www.sfcable.com/usb-micro-male-to-mini-5pin-female-adapter.html?gclid=CjwKCAjw1v_0BRAkEiwALFkj5pILcMI-JI_cWTZPnnjpWpM2Tinpp59tJSMTVlpMszd2ZwiErYtgiRoCRXwQAvD_BwE)



Alligator Clips

[https://www.amazon.com/WGGE-WG-026-Pieces-Colors-Alligator/dp/B06XX25HFX/ref=sr\\_1\\_1\\_sspa?dchild=1&keywords=alligator+clip&qid=1590053249&sr=8-1-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEzSkNYOVUzQ1QyUjM5JmVuY3J5cHRIZElkPUExMDI2NDI3QlhTVTFCQjZKNIFXJmVuY3J5cHRIZEFkSWQ9QTAxOTMyMzVPSVAyNINPUEs2OUgmd2IkZ2V0TmFtZT1zcF9hdGYmYWN0aW9uPWNsaWNrUmVkaXJIY3QmZG9Ob3RMb2dDbGljaz10cnVI](https://www.amazon.com/WGGE-WG-026-Pieces-Colors-Alligator/dp/B06XX25HFX/ref=sr_1_1_sspa?dchild=1&keywords=alligator+clip&qid=1590053249&sr=8-1-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEzSkNYOVUzQ1QyUjM5JmVuY3J5cHRIZElkPUExMDI2NDI3QlhTVTFCQjZKNIFXJmVuY3J5cHRIZEFkSWQ9QTAxOTMyMzVPSVAyNINPUEs2OUgmd2IkZ2V0TmFtZT1zcF9hdGYmYWN0aW9uPWNsaWNrUmVkaXJIY3QmZG9Ob3RMb2dDbGljaz10cnVI)

# Getting Started Guide



micro:bit Application Programming Interface (API)

The TI microbit module is aligned to the standard micro:bit API. Please use this reference as you include micro:bit statements into your TI-python programs. <https://microbit-micropython.readthedocs.io/en/v1.0.1/>

micro:bit Let' Code

If the micro:bit is programed with another language such as MakeCode blocks, JavaScript, or C++, a different runtime .hex file is loaded onto the micro:bit. To restore the calculator's communication functionality, the TI-84 Plus CE-T PYTHON EDITION runtime must be reinstalled, as directed in the first step of this document.

Switching runtimes does not harm the micro:bit. This step is essential for micropython to be loaded on the micro:bit and enable python programming. <https://microbit.org/code/>

Micro:bit runtime

<https://lancaster-university.github.io/microbit-docs/>

Micro:bt Micropython

<https://tech.microbit.org/software/micropython/>

## Example Programs

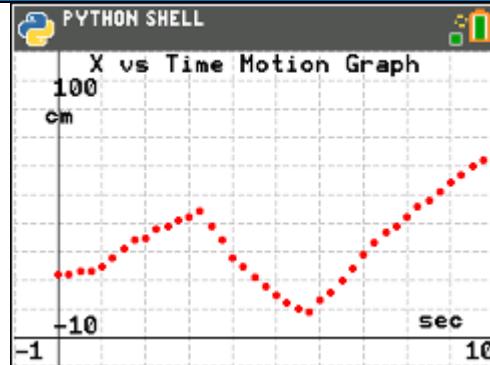
### Program 1

```
import ti_pltlib as plt
from ti_system import *
from microbit import *
from mb_grove import *

plt.cls()
plt.window(-1,10,-10,100)
plt.grid(1,10,"dash")
plt.axes("on")
plt.labels("sec "," cm",11,3)
plt.title("X vs Time Motion Graph")
plt.color(255,0,0)

initilize=grove.read_ranger_cm(pin0)

for i in range(40):
    echo=grove.read_ranger_time(pin0)
    d=echo/2*34000
    t=i*.25
    sleep(250)
    plt.plot(t,d,"o")
plt.show_plot()
```



# Getting Started Guide

## Program 2

```
import ti_pltlib as plt
from ti_system import *
from microbit import *
from mb_sensr import *

plt.window(-1800,1800,-1200,1200)
x = 318/2
y = 30+212/2
r = 981*318/(plt.xmax-plt.xmin)
plt.color(240,240,240)
plt.cls()
plt.gr.fillCircle(x,y,r)
plt.grid(200,200,"solid")
plt.color(0,0,0)
plt.gr.drawArc(x-r,y-r,2*r,2*r,0,3600)
plt.axes("on")
plt.labels("x (mg)", "y (mg)",8,1)
plt.gr.drawString("981 mg",x+15,y-15)
plt.color(255,0,0)

while not escape():
    accx = accelerometer.get_x()
    accy = accelerometer.get_y()
    plt.plot(accx,accy,"o")
    plt.show_plot()
```

