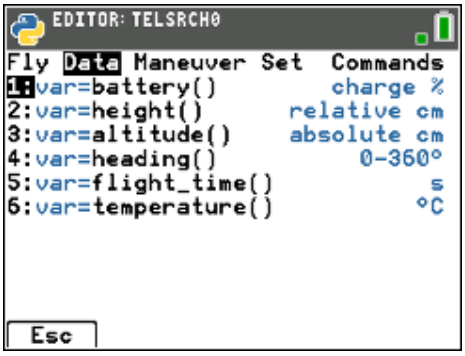
0. Tello contains sensors that can detect some information about its surroundings and position. This mini-project introduces some of these sensors and explores the pattern between the data collected.

1. This image shows the **[math] > tello drone… > Data** menu. You have been using the **.battery()** monitor in the previous projects. You will now use **.flight_time()**, and **.height()** data in this project to establish a relationship between height from takeoff and flying time as you let the program control Tello's height above the floor.



Note: there is a difference between 'height' and 'altitude'. Height comes from a 'time-of-flight' sensor (like a Ranger but using light rather than ultrasound). Altitude is calculated using the barometer, measures the change since takeoff and is not as accurate as height. Altitude could be 0 or negative. Minimum height value is 10cm when landed. Minimum height in flight is 30cm.

2. Start a new Python program, **import** the Tello module and check the **.battery()** as usual.
Create two empty lists for storing collected data:

times = [ ] for the flight times (seconds)
hts = [ ] for the heights (cm)

Get the square brackets from **[2nd] [stat]**, or **<a A #>**

3. Ready to take flight:

tello.takeoff()

and, if you are working in a room with an 8-foot ceiling, make Tello go as low as possible to have room to collect data at different heights:

tello.down(60)

Note: Tello will not go lower than 30cm off the ground.

4. Use a **for** loop to let the program automatically operate the Tello.

   for i in range(12):

5. Now construct the loop body using these two big ideas:
   - Collect data from Tello (time and height)
   - Add the data to the two lists you created

   Try it yourself before moving on to the next steps…

6. To collect and store the data:
   Use the appropriate data methods found on
   [menu] More Modules > Tello > Flight Data >
   Create a simple variable for both **tello.height()** and **tello.flight_time()**

   h = tello.height( )
   t = tello.flight_time( )

   then use the list function **.append( )** to add these values to each list:

   hts.append(h)
   times.append(t)

   .append( ) is found on the **<Fns…> List** menu

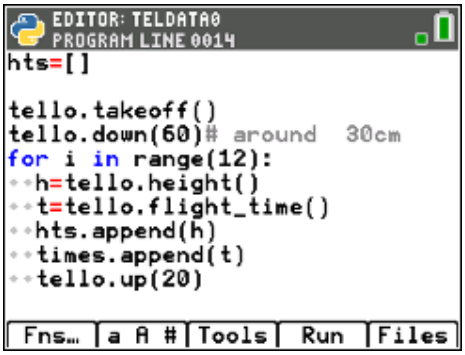
   Note: flight_time begins when Tello first takes off but continues until Tello is turned off, not when Tello lands. A second run of the program will show cumulative flight times beginning with the first takeoff. To reset flight_time to 0, turn Tello off and on again between flights. Or modify the program to account for this feature.

7. After the data has been collected and stored, make Tello go up 20cm.

   tello.up(20)

   Recall that **.up( )** is found on **[math] > tello drone… > Fly**

   This is the end of the **for** loop body, but the data from the last increase in height still needs to be collected and stored…



8. You can copy/paste the collect and store statements in the loop body but be sure to de-dent them so that they are *not* part of the loop body.

   h = tello.height( )
   t = tello.flight_time( )
   hts.append(h)
   times.append(t)

9. You are done with data collection. After the loop ends and all data is stored in two Python lists, land Tello and store the two Python lists into TI-84 lists for analysis elsewhere in the calculator. These statements are also not indented.

   store_list("HTS", hts)
   store_list("TIMES", times)
   tello.land()

   You will find **store_list(“” ,)** on
   [math] > ti_system…
   and you must add

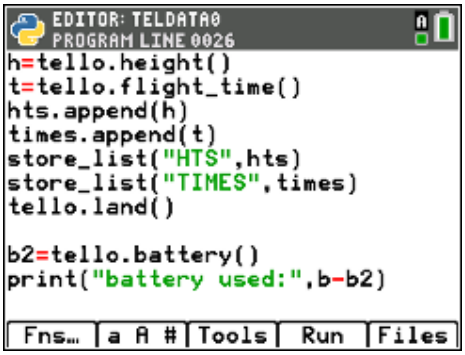
   from ti_system import *

   at the top of your program (not shown).

   *Note: the calculator lists must be “*UPPERCASE*” names and limited to 5 characters.*

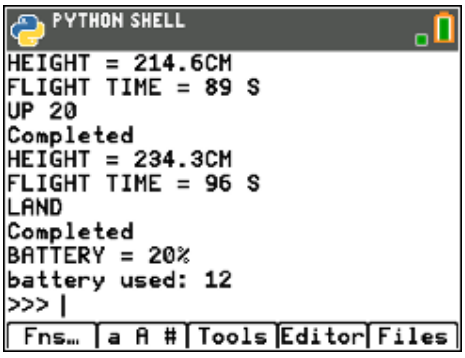
10. **Bonus:** Also check the battery level now and compare it to the starting battery level.

```
b2 = tello.battery( )
print( "battery used:" , b - b2 )
```



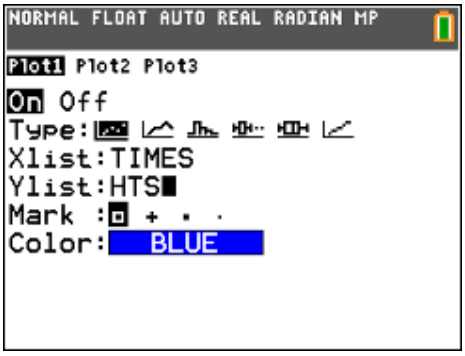
This information can help you modify your program later so that it will only fly if there's enough charge left in the battery. Tello will stop working when the battery charge goes below 10%. Our tests consumed about 20% battery, so we determined that it would be safe to fly only when the battery is above 30%.

11. **<Run>** the program. The calculator screen displays the data values as they are being collected.

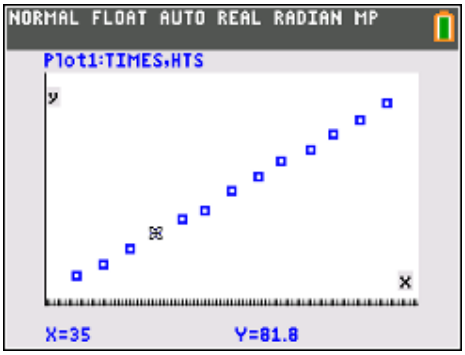


Our test flight took 96 seconds, reached a height of 234.3cm and consumed 12% of the battery's charge. Note that there is not enough charge left in the battery for another flight: when the charge goes down to 10% the flight is aborted.

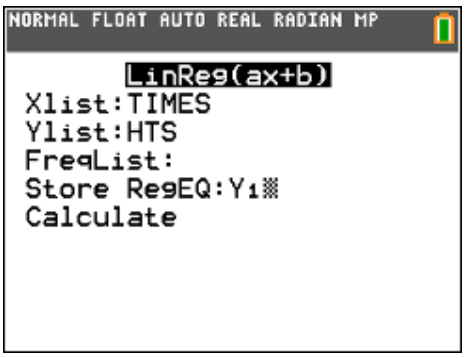
12. After collecting the data, quit the Python app, set up a **stat plot** (**[2nd] [y=]**) of the data. Use the list **TIMES** for the x-list and the list **HTS** for the y-list. Find these list names on **[2nd] [stat]** (the **list** menu)



13. Press **[zoom] Data** to see a plot similar to the one at the right (showing the **[trace]** feature.

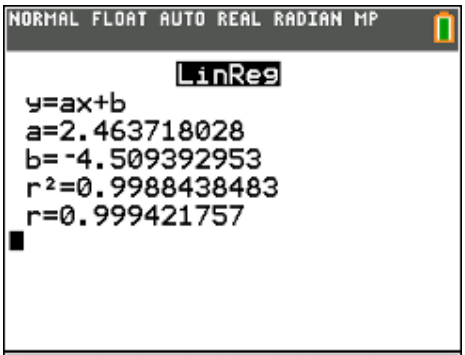


14. It looks sort of linear, yes? Make a linear function that best models the data (Linear Regression). Press **[stat] Calc** and choose **LinReg (ax+b)** and complete the dialog box appropriately.

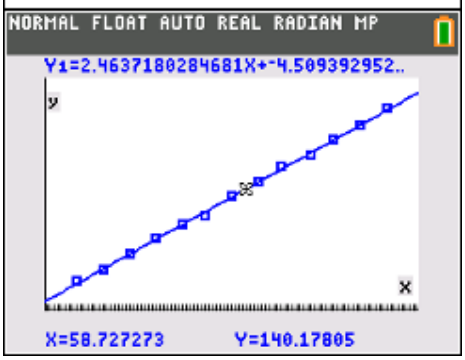


15. The result of the Linear regression is displayed.
More about these numbers in a moment. But first…

Note: if you do not see 'r²= ' and 'r= ',
press **[mode]** *and select* **STATDIAGNOSTICS: ON**
then repeat the Linear Regression.



16. Press **[zoom] Data**. The graph of the line is plotted over the stat plot showing a very good fit to the data.
Note: this screen shows the trace of the line showing the function at the top of the screen.



What do the numbers in the function represent?

Answers: the **slope** (2.46… in the image) is the rate of change in height over time: the average speed of the Tello was 2.46cm/second, or about one inch per second, since 2.54cm= 1 inch).
The **y-intercept** (-4.50…) is the approximate starting position, which was 0 in our experiment.

17. Challenge: What is the average change in height in each step of the flight?