

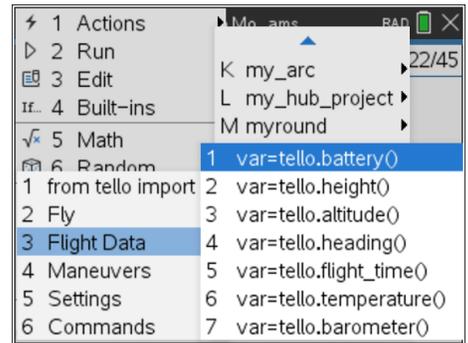
Getting Started with Tello

Tello's Data

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



1. This image shows the **Tello > Flight 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 time-of-flight as you let the program control Tello's height above the floor.

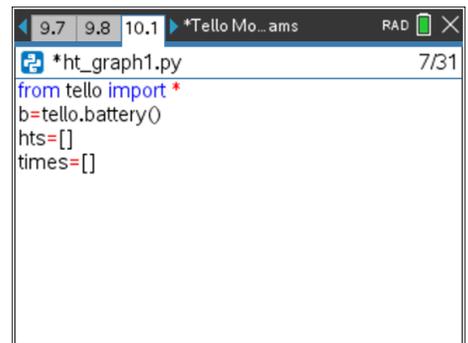


Note: there is a difference between 'height' and 'altitude'. Height comes from the time-of-flight sensor (like a Ranger but using light rather than ultrasound). Altitude is calculated using the barometer, measures the change since takeoff and is not as accurate as height. Altitude could be 0 or negative.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

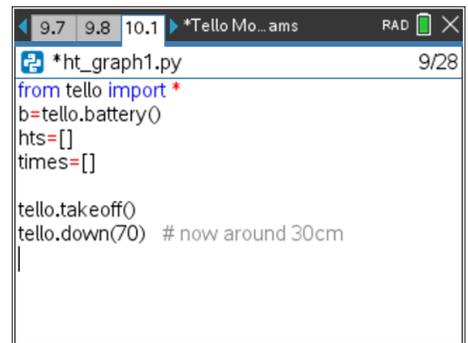


3. Ready to take flight:

```
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(70)
```



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



TI-NSPIRE CX II

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

for i in range(12):

- 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...

- 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.time()

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

hts.append(h)

times.append(t)

.append() is found on **[menu] > Built-ins > Lists**

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.

```

9.7 9.8 10.1 *Tello Mo...ams RAD 9/28
*ht_graph1.py
from tello import *
b=tello.battery()
hts=[]
times=[]

tello.takeoff()
tello.down(70) # now around 30cm
for i in range(12):
  |
  
```

```

9.7 9.8 10.1 *Tello Mo...ams RAD 9/28
*ht_graph1.py
from tello import *
b=tello.battery()
hts=[]
times=[]

tello.takeoff()
tello.down(70) # now around 30cm
for i in range(12):
  |
  
```

```

9.7 9.8 10.1 *Tello Mo...ams RAD 3/24
*ht_graph1.py
hts=[]
times=[]

tello.takeoff()
tello.down(70) # now around 30cm
for i in range(12):
  h=tello.height()
  t=tello.flight_time()
  hts.append(h)
  times.append(t)
  
```

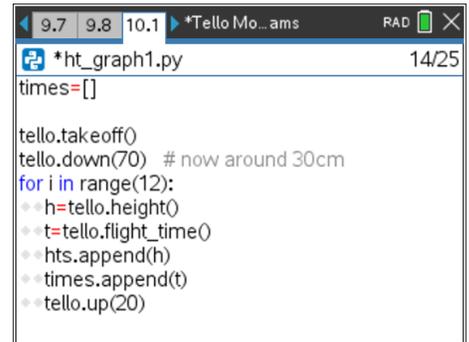
TI-NSPIRE CX II

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

tello.up(20)

Recall that `.up()` is found on **[menu] > More Modules > Tello > 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...



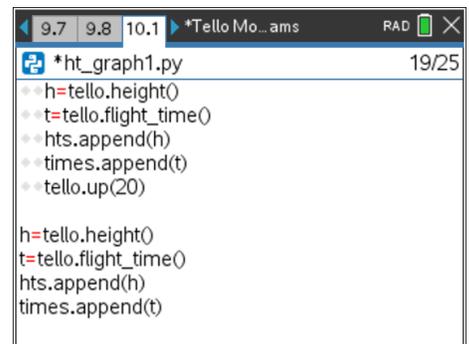
```

9.7 9.8 10.1 *Tello Mo...ams RAD 14/25
*ht_graph1.py
times=[]

tello.takeoff()
tello.down(70) # now around 30cm
for i in range(12):
    h=tello.height()
    t=tello.flight_time()
    hts.append(h)
    times.append(t)
    tello.up(20)
    
```

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

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



```

9.7 9.8 10.1 *Tello Mo...ams RAD 19/25
*ht_graph1.py
h=tello.height()
t=tello.flight_time()
hts.append(h)
times.append(t)
tello.up(20)

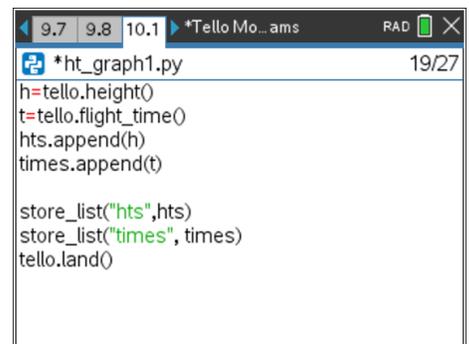
h=tello.height()
t=tello.flight_time()
hts.append(h)
times.append(t)
    
```

9. You are done with data collection. After the loop ends and all data is stored on Python lists, land Tello and store the two Python lists into TI-Nspire 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

[menu] > More Modules > Tello > Commands



```

9.7 9.8 10.1 *Tello Mo...ams RAD 19/27
*ht_graph1.py
h=tello.height()
t=tello.flight_time()
hts.append(h)
times.append(t)

store_list("hts",hts)
store_list("times", times)
tello.land()
    
```

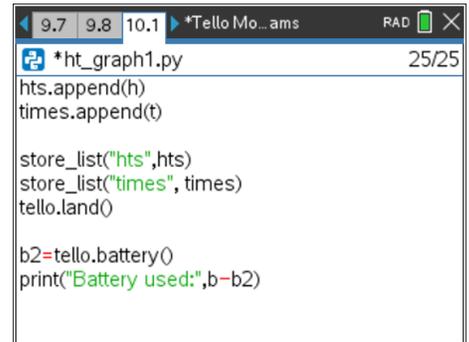
TI-NSPIRE CX II

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

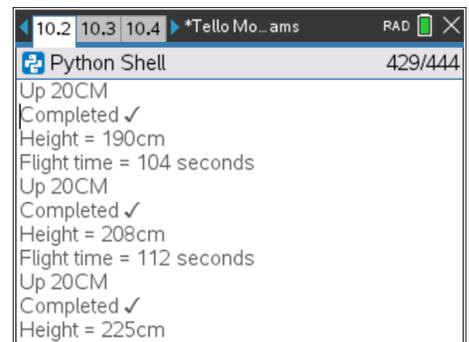
```
b2 = tello.battery( )  
print( "Battery used:" , b - b2 )
```

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 (ctrl-R). The calculator screen displays the data values as they are being collected.

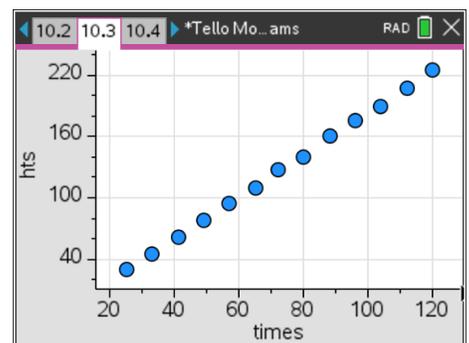


```
*Tello Mo...ams 25/25  
*ht_graph1.py  
hts.append(h)  
times.append(t)  
  
store_list("hts",hts)  
store_list("times", times)  
tello.land()  
  
b2=tello.battery()  
print("Battery used:",b-b2)
```



```
*Tello Mo...ams 429/444  
Python Shell  
Up 20CM  
Completed ✓  
Height = 190cm  
Flight time = 104 seconds  
Up 20CM  
Completed ✓  
Height = 208cm  
Flight time = 112 seconds  
Up 20CM  
Completed ✓  
Height = 225cm
```

12. After collecting the data, add a TI-Nspire **Data and Statistics** app to your document (press **[ctrl]-[doc]** and select the app). Click the bottom box and select the list **times** for the independent variable and click the left side box to select the list **hts** for the dependent variable. You should see a graph similar to the one at the right.



TI-NSPIRE CX II

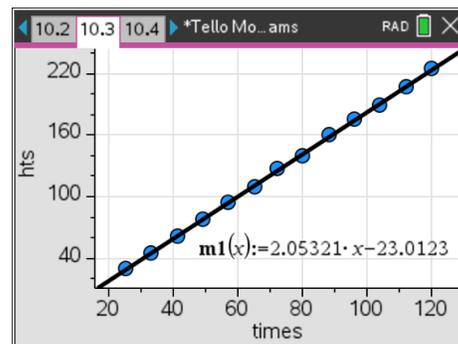
13. It looks sort of linear, yes? Add a moveable line to the app using
[menu] > Analyze > Add moveable line

Move* the line around until you feel that it models the collected data as seen here. What does the equation of the line tell you?

**Note: There are two ways to move the line: grab it near the center of the screen to translate the line. Grab it near the ends to rotate the line.*

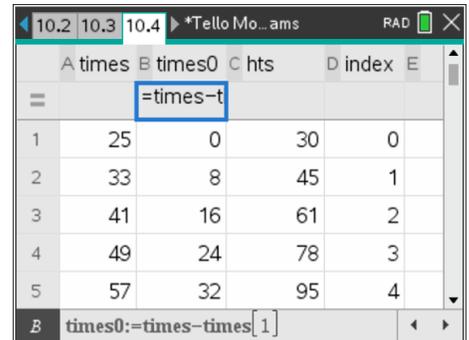
Answer:

- The slope (about 2) is the *average speed* of Tello during the flight. It seems small, but consider all the time Tello spends at each level.
- The y-intercept (about -23) is misleading: it's the 'starting time' but it includes the prep, takeoff and descent to a lower level. The first data point is collected after about 25 seconds. We'll make a correction to the times data soon. But first...



TI-NSPIRE CX II

14. Add another list to the problem representing the **index** number of the data [0,1, 2, 3, ...]. You can do this in the **Lists and Spreadsheet** app as seen here. See the column named **index**. Just enter the numbers manually to pair with the heights and start with 0.
- You may also display the other collected lists in this spreadsheet for reference. Just type their names in the top cells.
- Consider adding a **times0** list to the spreadsheet which will always begin with time = 0 for the first height, regardless of the actual times.
- The formula for **times0** is
- $$\mathbf{times0 := times - times[1]}$$
- as seen at the bottom of the spreadsheet. Notice the pattern in **times0**

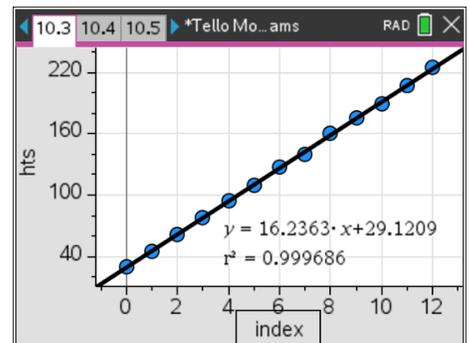


	A times	B times0	C hts	D index
=		=times-t		
1	25	0	30	0
2	33	8	45	1
3	41	16	61	2
4	49	24	78	3
5	57	32	95	4

times0 := times - times[1]

This spreadsheet will be properly updated every time you run the Python program.

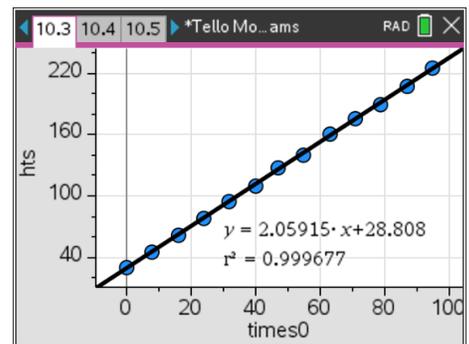
15. After completing the previous step, go back to the data plot and change the **times** list on the bottom to **index**.
- Adjust your moveable line to match the **hts** v. **index** plot.
- Interpret the new values in your moveable line equation.
- Note: we're now showing a linear regression line in this image using [menu] > Analyze > Regression > Show Linear (mx + b).)*



Answer:

- the *slope* (16.2363) means that, at each step, the height changes by about 16cm (not '20cm' as the code requested!)
- the *y-intercept* (29.1209) is the approximate height of the first data collection, around 30cm.

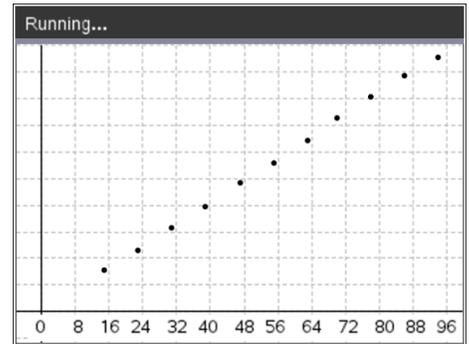
If you created the **times0** list, change the independent variable on the graph as well. Observe that the plot does not change much but the equations are very different in a special way. Can you see why?



Answer: The *scale* on the *x-axis* is different.

TI-NSPIRE CX II

16. **Challenge 1:** Incorporate `tiplotlib` or `ti_draw` in your Python program to make a dynamic plot of the data (`t` , `h`) as it is being collected.



17. **Challenge 2:** The flight path (time, height) is better modeled by a 'step function' as seen here. Tello starts 30cm above the ground. Every 8 seconds, Tello flies up about 16cm. What function creates this graph?

Hint: start with the function $f1(x) = \text{int}(x)$

