



TI-84 PLUS CE PYTHON

Getting Started with Tello

Tello Search and Rescue

Intro 1

On Thanksgiving (USA) 2022, a cruise passenger fell overboard in the Gulf of Mexico. The US Coast Guard set out on a rescue mission to locate the passenger. Their search area was over 7,000 square miles. They located and rescued the passenger after he spent 20 hours treading water in the middle of the Gulf. How did they do that? In December 2022, two men started sailing a boat from New Jersey to Florida. The Coast Guard started a search after reports the men were overdue in their journey. Eventually, the agency searched a combined 21,164 square miles from northern Florida to New Jersey. The men were rescued after 10 days at sea. How did they do that?

See [“Using drones in Search and Rescue”](#)

Intro 2

In this mini-project, you will write a Python program to conduct a coordinated ‘search’ of an area using Tello. There are several search patterns (routes) that professionals use to conduct a systematic search of a region depending on the situation.

Get your TI-84 Plus CE Python and Tello system ready to fly...



1. Here is your first pattern, called the ‘**parallel track search**’. A drone starts in the lower left corner and flies the indicated path. At the end of the search, the drone can either land or return to its starting position.

Inputs to the program will be the length and width of the rectangular search area plus *one other important measurement*.

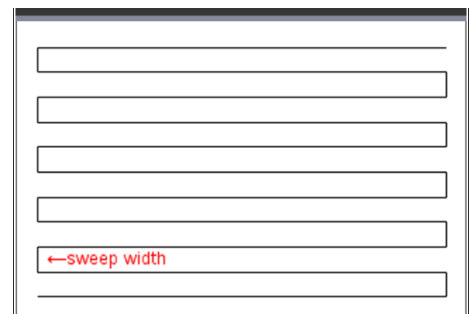
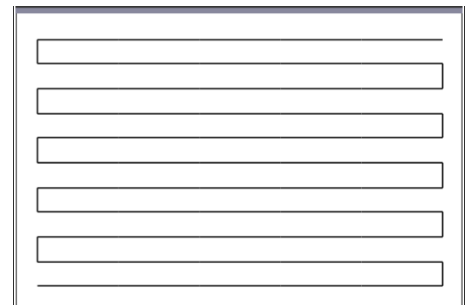
The program will determine the lengths of each of the flight segments and the number of steps needed to complete the search and then use a loop to conduct the flight.

Note: screenshot made using turtle graphics.

2. For the purpose of this project, **length** is the horizontal distance (the parallel tracks) and **width** is the vertical distance in the image shown here.

The *other important measurement* in this search pattern is called the “**sweep width**”, the distance between the (horizontal) flight segments.

Using **length**, **width** and **sweep_width** you can create an algorithm for Tello to complete this mission.





10 Minutes of Code: Python Modules

TELLO SEARCH AND RESCUE

TI-84 PLUS CE PYTHON

- Begin a new Python program (ours is named **telsrch0.py**). Import the Tello module and perform a battery check.

Use **[math] > tello drone...** for all the Tello functions.

Create three variables to define the search pattern:

```
w = 80
lnt = 100
sw = 20 # minimum distance*
```

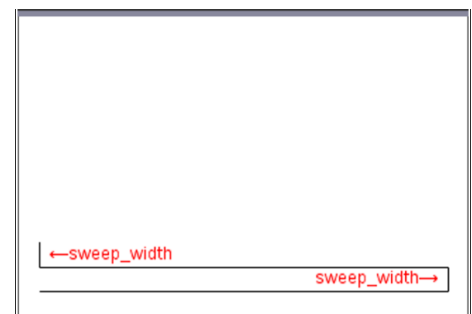
These three values are all that is needed to conduct the search.

** Note: Tello flight distances are limited to be between 20 and 500 centimeters.*

```
EDITOR: TELSRCH0
PROGRAM LINE 0007
from tello import *
b=tello.battery()
# search area
w=80
lnt=100
sw=20 # minimum distance
```

- One **'step'** of the search consists of two horizontal segments and two short **sweep_width** segments as seen here. At the end of this path, Tello will be pointing in the same direction as it started (facing to the right in the image) so that this pattern can be repeated to complete the entire search.

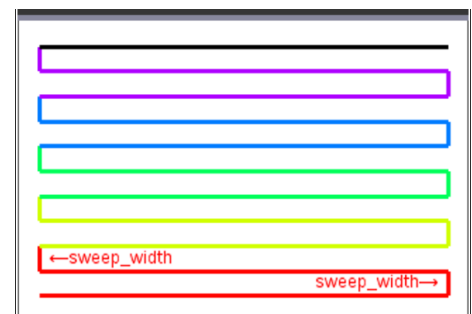
We need to calculate the number of these **steps** needed to complete the search. See the next step...



- In the image shown here, each flight segment is shown in a different color. The last (top) segment is added separately. How many steps are needed? **steps** depends only on the **sweep_width** and the **width** of the search area. Notice that there are **two sweep_widths** in each segment, so the number of steps needed is:

$$s = w // (2 * sw)$$

We use integer division (//) because this value will be used in a for loop. Parentheses are required to control the order of operations.



- In your program, calculate the number of steps, launch Tello, and start a **for** loop to conduct the search:

```
s = w // (2 * sw)
tello.takeoff()
for i in range(s):
    ♦ ♦
```

*Note that **steps** must be an integer.*

```
EDITOR: TELSRCH0
PROGRAM LINE 0011
from tello import *
b=tello.battery()
# search area
w=80
lnt=100
sw=20 # minimum distance

s=w//(2*sw)
tello.takeoff()
for i in range(s):
    ♦ ♦
```



10 Minutes of Code: Python Modules

TELLO SEARCH AND RESCUE

TI-84 PLUS CE PYTHON

7. The loop body (indented) will contain two long flights (the **length** of the area) and two short ones, the **sweep_width**. Recall from the last mini-project (Tour the Room) that there are two methods you can use: turn at each corner or just fly in the proper direction. Since we used the turn method in the last project, let's use only the proper fly routines here. Start by flying forward the length of the area:

◆◆ **tello.forward(lnth)**

Try completing the loop yourself before looking at the code in the next step...

8. The complete loop is:

for i in range(s):

◆◆ **tello.forward(lnth)**

◆◆ **tello.fly_left(sw)**

◆◆ **tello.backward(lnth)**

◆◆ **tello.fly_left(sw)**

9. This loop (repeated **s** times) will leave Tello in the upper *left* corner. You have two options: quit searching here or complete the top segment to go to the upper right corner (the bold black segment pictured here). In either case, you must make sure Tello returns to the landing zone where it started because it might not be safe to land anywhere else.

We'll take on the challenge of completing the entire pattern and then returning home...

10. After the loop, add another
tello.forward(lnth)
to complete the pattern.

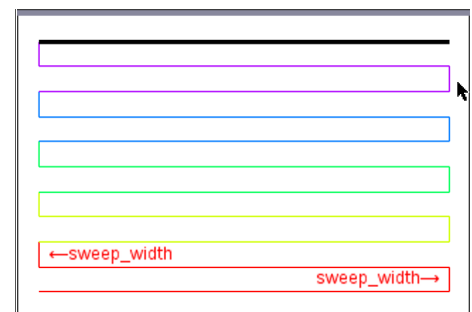
*This statement (and the ones that will follow) is not indented because it is not part of the **for** loop.*

```
EDITOR: TELSRCH0
PROGRAM LINE 0012
b=tello.battery()
# search area
w=80
lnth=100
sw=20 # minimum distance

s=w//(2*sw)
tello.takeoff()
for i in range(s):
    tello.forward(lnth)
    _
Fns... a A # Tools Run Files
```

```
EDITOR: TELSRCH0
PROGRAM LINE 0014
w=80
lnth=100
sw=20 # minimum distance

s=w//(2*sw)
tello.takeoff()
for i in range(s):
    tello.forward(lnth)
    tello.fly_left(sw)
    tello.backward(lnth)
    tello.fly_left(sw)
    _
Fns... a A # Tools Run Files
```



```
EDITOR: TELSRCH0
PROGRAM LINE 0016
sw=20 # minimum distance

s=w//(2*sw)
for i in range(s):
    tello.forward(lnth)
    tello.fly_left(sw)
    tello.backward(lnth)
    tello.fly_left(sw)
    _
tello.forward(lnth)
_
Fns... a A # Tools Run Files
```

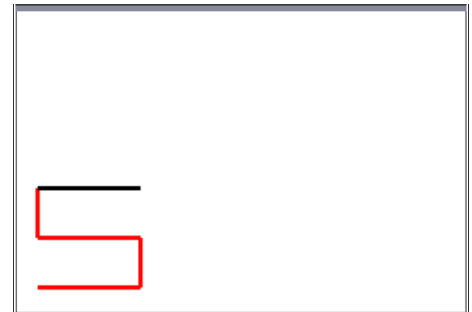
TI-84 PLUS CE PYTHON

11. To get Tello back to the landing zone use:

```
tello.fly_right(w)
tello.backward(lnth)
tello.land()
```

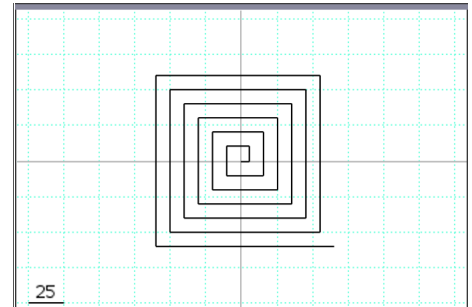
```
EDITOR: TELSRCH0
PROGRAM LINE 0019
for i in range(s):
    tello.forward(lnth)
    tello.fly_left(sw)
    tello.backward(lnth)
    tello.fly_left(sw)
tello.forward(lnth)
tello.fly_right(w)
tello.backward(lnth)
tello.land()
```

12. This flight can be tricky. Start with a small search area. Since the **sweep width** must be at least 20cm, try **lngh = 50, w = 60, sw = 30** to test your code. Use a large, safe, open area (a square meter or more) for testing, Start Tello in one corner of the region facing in the proper direction, and give it a try. You should get the pattern seen here and Tello should return to its takeoff site.



13. There are other search and rescue patterns that can be found online.

Another interesting pattern is the “expanding box” pictured here using Turtle Graphics. Can you make Tello fly this ‘square spiral” pattern? How about other spirals?



Special thanks to the **United States Coast Guard** for their courage, talent, and perseverance.

14. **Extensions (optional):** For additional activities to use Tello, check out the downloads here:

- **Tello’s Data:** 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.
- **Fly the Cube:** Tello flies forward, backward, leftward, rightward, upward, & downward. It can fly in THREE DIMENSIONS (3D, also known as space). Using a special Tello “Mission Pad”, which assigns Tello a 3D coordinate system so that you can tell Tello to directly **goto()** a particular point in space by giving the point’s three coordinates as a location. **Note: Tello EDU model is required for this activity.**