Tello Drones Workshop

Roberto Tron

Wednesday 17th April, 2019

Problem 1: First programming steps

1.1 Opening the editor

Open the IDLE editor by double-clicking on the icon on the desktop. We will use this simple program to write and execute our code.

Info: *Programs that allow you to write, organize, and run code, all without leaving that program, are called Integrated Development Environments (IDE).*

1.2 Python

Pythons are serpents, but Python (with the capital and no "s") is a programming language. Nowadays, Python is among the most popular programming languages; this is probably due to the low difficulty in getting started, and the fact that its functionality can be extended with a large number of *modules*.

Python is an *intepreted language*. In practice, this means that you can program in one of two ways:

1) Write commands at a *prompt*. This is what you see after opening IDLE. After typing a command, it gets immediately executed (giving an error if it is not correct). Afterward, it the prompt waits for another command, and the cycle repeats until you exit.

2) Write commands in a file, which is called a *script*. You can then execute all the commands one after the other by *running* the script. This is equivalent to typing each command at the prompt, although the execution of the script stops if it encounters an error.

1.3 Hello world

At the IDLE prompt, type

```
print('Hello world!')
```

The `''` characters delimit a *string*, which is simply a collection of characters. `print` is a *function*; functions take in *arguments* (in this case, a string) to perform some action (in this case, showing it on the screen). Sometimes, it also *returns* a result (more on this later).

Try this: *Select the menu* File, New *to open a new file. Write a sequence of print statements. Save and run the script.*

1.4 Variables

Variables are like named labels that you put on specific *objects* (items) in memory. For instance, the code

```
a='Hello world!'
```

gives the label `a` to the string `'Hello world'` in memory.

Info: *Sometimes, you might also hear that the variable* `a` contains *the string.*

You can then use `a` instead of using the object directly. For instance,

```
print(a)
```

produces the same result as in the previous section.

1.5 Modules and extending functionality with imports

You can extend functionality by *importing modules*. For instance, the Python language by itself cannot make the drone fly. However, you can import functionality as in this example:

```
from djitellopy import Tello
```

In this case, we import `Tello` (which is a *class*, more on this later) from the module `djitellopy`.

You can then used the functions or classes that you imported:

```
tello_object=Tello()
```

Info: *Names in Python are case-sensitive, so* `tello` *and* `Tello` *are recognized as different things.*

1.6 Objects

Objects are essentially pieces of memory with a prescribed organization. Objects can contain variables (which are labels for other objects) or functions; *classes* define what variables and functions go inside an object. For instance, in the previous example

```
tello_object=Tello()
```

you created an object `tello_object` of class `Tello`.

You can access variables or functions inside an object as in the example below.

Try this: *Before running the example, you will need to turn on the drone and connect to its WiFi network. It is best to use run the example as a script.*

```
from djitellopy import Tello
tello_object=Tello()

tello_object.connect()
tello_object.end()
```

As shown in the example, you need to use the name of the object `tello_object` followed by dot `.`, followed by the function or variable name (in this case, `connect()` first, and then `end()`).

Info: *To be precise,* `tello_object` *is a variable pointing to the object, but in general this subtle distinction is omitted.*

Info: *When using the drone, you need to always call the* `end()` *function, otherwise you might have trouble to connect to the drone again (this can be fixed, but it requires a little bit of time).*

Problem 2: The state of the drone

You can see the state of the drone (roll/pitch/yaw, accelerometer readings, and other information) by using the function `print_state()` in the drone.

```
from djitellopy import Tello
tello=Tello()

tello.connect()
tello.print_state()
tello.end()
```

2.1 More programming: repeating things (loops)

Sometimes, you would like to repeat a command more than once. For instance, you would like to show the state multiple times for each execution. This can be done with a *for loop*. The easiest way to write for loops in Python is as in this example:

```
for count in range(0,4):
    #commands in the loop to be repeated
    #all commands to be repeated should be indented with spaces in front
    #the number of spaces in front is usually 4
```

Info: *Do not forget the semicolon in the first line, otherwise you will get an error.*

In this case, the commands inside the loop are repeated four times; each time, the variable `count` will have one of the values `0`, `1`, `2`, `3`. How many times the loop gets repeated and what values the variable will cycle through depend on the arguments to the function `range` (which are `0` and `4`)

Try this: *What do you think the following commands will do?*

```
for count in range(0,5):
    print(count)
```

2.2 Waiting a given amount of time

By default, for loops get executed as fast as Python can run the commands. However, you can insert pauses by using the function `sleep(sec)` from the `time` module; `sec` specifies for how many seconds the execution should wait.

Try this: *This example is similar to the previous, but with two-seconds pauses after each repetition of the loop.*

```
from time import sleep
for count in range(0,5):
    print(count)
    sleep(2)
```

2.3 Reading the state of the drone multiple times

Try this: *We can put together all the material in this subsection*

```
from djitellopy import Tello
from time import sleep

tello=Tello()

tello.connect()
for count in range(0,5):
    tello.print_state()
    sleep(1.5)
tello.end()
```

Problem 3: Flying the drone

3.1 Takeoff and landing

You can have the drone takeoff and land by calling the functions `takeoff()` and `land()` in the `tello` object.

```
from djitellopy import Tello
from time import sleep

tello=Tello()

tello.connect()
tello.takeoff()
sleep(1.5)
tello.land()
tello.end()
```

3.2 Moving (translation)

You can move in any direction (up,down,left,right,forward,back) by using the corresponding functions in the `tello` object.

Info: *The argument to the function must be a distance in cm higher than 20 and no lower than 100*

```
from djitellopy import Tello

tello = Tello()

tello.connect()
tello.takeoff()
tello.move_up(30)
tello.move_down(30)
tello.move_forward(30)
```

```
tello.move_back(30)
tello.move_left(30)
tello.move_right(30)
tello.land()
```

3.3 Rotating

You can change the yaw of the drone with the functions `rotate_clockwise(deg)` and `rotate_counter_clockwise(deg)`; the argument `deg` must be a number of degrees between 1 and 360.

```
from djitellopy import Tello

tello = Tello()

tello.connect()
tello.takeoff()
tello.rotate_clockwise(180)
tello.rotate_counter_clockwise(180)
tello.land()
```

3.4 Putting things together

Info: *For the questions below, it is easier to first write the scripts by using* `print` *to show the commands on screen instead actually executing them (e.g.,* `print('tello.move_up(50)')`*). After you are satisfied with the sequencing, and when there are no errors, then you can substitute the* `print`*.*

Try this: *Can you write a script such that the drone traces an imaginary square?*

Try this: *Can you write a script such that the drone traces an imaginary square four times, with pauses of 2 seconds in between?*

Try this: *Can you write a script that traces the edges of the side faces of an imaginary cube?*

Problem 4: Working with images

4.1 More on importing modules

You can also import entire modules, and then decide what functions or classes to use from them later. For instance, the module `cv2` contains many functions to handle images. For instance:

```
import cv2
img=cv2.imread('BU_logo.png')
```

As shown in this example, functions inside the module (e.g., `imread`) can then be called by using the name of the module `cv2` and a dot `.` before their name.

4.2 Loading and showing images

The following expanded example loads an image with `imread`, shows it on screen with `imshow`, waits for a key to be pressed (**Note: you need to press the key while the image**

window, not the prompt, is in focus), then closes all the image windows.

Info: *Calling* `imshow` *does not show the image window immediately. It is necessary to use* `waitKey` *to make it appear.*

Info: *If you do not call* `destroyAllWindows`*, the window will remain in a "Not responding" state, so please always remember to call it before the end of the script.*

```
import cv2
img=cv2.imread('BU_logo.png')
cv2.imshow('Image title',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

4.3 Taking images from the drone and saving them

To be able to get images from the camera onboard the drone, you need to use the function `stream_on()` on the `tello` object; this function should be called only once right after the call to the function `connect()`. The actual images can then be obtained by calling the function `tello.get_frame()`. You can save the images on disk with the function `imwrite('filename.png',img)` from the `cv2` module (change `filename` to the name of the file that you prefer, but do not forget the extension `.png`).

The following example summarizes all these commands:

```
from djitellopy import Tello
import cv2
tello.connect()
tello.stream_on()
img = tello.get_frame()
cv2.imwrite('filename.png',img)
tello.end()
```

Note: *When running this script, please wait for it to end completely (it might take a few seconds). You should see* `update_frame: terminating` *in the prompt window. If you close the prompt before this, you might have trouble getting images.*

Info: *The image should get saved in the* Documents/djitello *directory.*

Try this: *Take a selfie with the drone.*

Try this: *Can you write a script to make the drone fly and then take a sequence of pictures in four or more directions?*

4.4 Debugging image acquisition problems

If there are problems in the system (e.g., a previous script crashed before the call to `tello.end()`), the variable `img` might be empty. The following example adds a `if` command to check for this, and print a warning if it happens.

```
from djitellopy import Tello
import cv2
tello.connect()
tello.stream_on()
img = tello.get_frame()
```

```
if not img:
    print('No frame')
else:
    cv2.imwrite('filename.png',img)
tello.end()
```