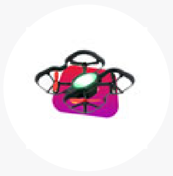
1.3 Flight Movements: Roll and Pitch

Now that we can stay of the ground and in the same place, let's learn how to move forward, backward, and side to side!

This lesson works with:


CoDrone EDU

CoDrone EDU
(JROTC ed.)

Python

beginner | CDE | CoDrone EDU | flight movements | Python

Grade level:
6 - 12+

Approx. time required:
30 - 45 mins

Step 1

Goals and Steps to Success

By the end of this lesson, you should be able to:

- Describe how roll and pitch change the drone's movement in the air.
- Explain the difference between parameters and arguments when using functions.
- Use the roll and pitch throttle commands to make the drone navigate a path of your choosing.
- Incorporate your knowledge to build a full flight path for the drone including takeoff and landing.

Step 2

Create a new Python file

Before we learn how drones move, we need to have a program created for testing movement. To do this, follow the steps below:

1. Open Pycharm
2. Click on the CoDrone EDU Directory
3. Right Click and Choose New -> Directory
4. In the name box that appears, enter 1b_flight_directions
5. You should now have two folders under your CoDrone EDU directory, 1a_flight_events and 1b_flight_directions
6. Right click on the 1b_flight_directions folder you just created and click New -> Python File
7. Name the file roll
8. Enter the commands you learned from the previous lesson to pair with the drone, take off, and land. (Hint: You can go back to the previous lesson's folder to look at what you wrote if you can't remember)

You are now ready to start testing some of the movements that your drone can perform!

Step 3

Get a Move On! How Air Vehicles Move

Air vehicles move a little differently than land vehicles. Since land vehicles operate in two dimensions (we can call these X and Y), they only move forward and backward or left and right. Air vehicles, however, move in three dimensional space (X,Y, and Z). This means that they not only move in the same directions as a land vehicle, but they also move up and down and can spin. Because of this we have different terms to describe how air vehicles move through the sky as you will see below

Movement Commands

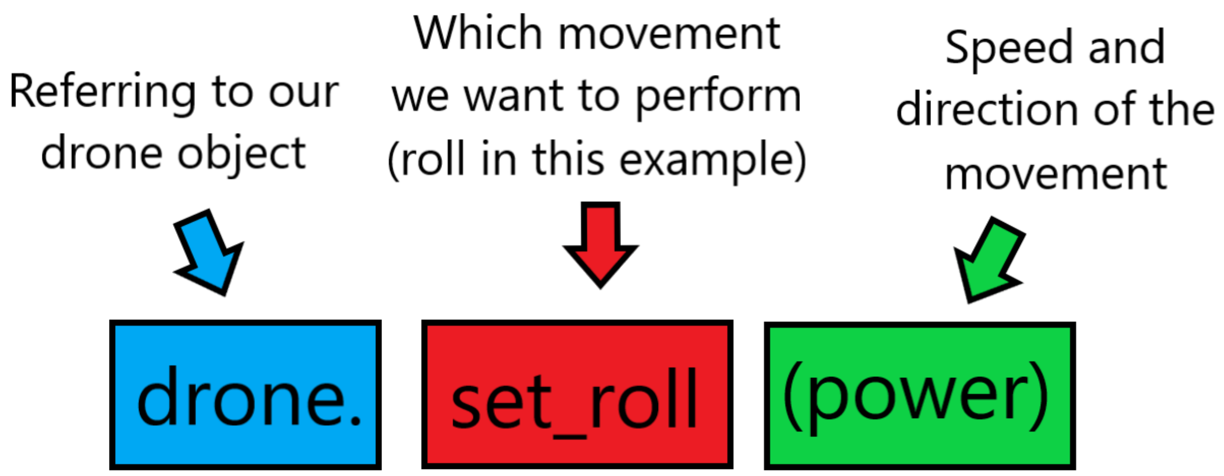
Key Terms

Parameter: a placeholder for a value that a function will use. These let the user know that values are required for the function to behave properly.

Argument: the actual value that a user puts into a function when it is called. For example, in the code drone.set_roll(25) the value 25 is the argument for this function.

There are three components to each movement of the drone that you will specify in order to make it behave properly:

1. The movement you want the drone to make (Roll, Pitch, Yaw, Throttle)
2. How much power to apply to the movement (How fast you want the movement to be)
3. How long you want the movement to execute or last


Referring to our drone object
Which movement we want to perform (roll in this example)
Speed and direction of the movement
drone.
set_roll
(power)

As you see in the image above, each of the movement commands follows the same format. You specify the drone object that you want to move, tell it which of the movement commands to do, and then specify what direction and speed you want the movement to occur. When setting the speed and direction, you put into the commands an **argument** value ranging from -100 to 100. The values 0-100 are the percentage of the drone's power you want to use. 0 would be no power while 100 would be full power (100%). Whether the **argument** value is positive or negative determines the direction the drone will take. For example, -50 and 50 will move at the same speed but in opposite directions.

BE CAREFUL! Setting the drone's power to 100% will make it move very quickly and result in collisions with objects. We recommend that you start at lower power settings and work your way up to higher settings once you understand how quickly the drone moves.

Calling the function takes care of steps 1 and 2, but not step 3. You will notice that there is no **parameter** in the movement commands that tells the drone how long to move. This is because the movement commands are all SET commands. What this means is that we are changing the value of a variable (the movement) to a new value. We will learn the difference between setter and getter commands and what variables are in later lessons. Since we are only setting or changing the speed and direction of the drone with the movement commands, we have a separate function called move that tells the drone the duration or time that it should move:

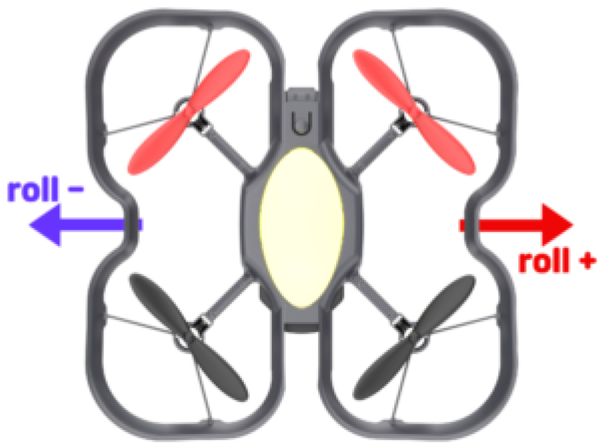
- drone.move(1) # has the drone move for 1 second
- drone.move(2.5) # has the drone move for two and a half seconds

By combining the movement and move commands, we can make the drone move in a specific direction for an amount of time. So, each time we program one of the movement commands, don't forget to pair it with a move() command as well!

Step 4

Roll

This function controls the CoDrone EDU's horizontal, or side to side, movement. Positive roll will make the CoDrone EDU move to the right, and negative roll will make the drone move to the left. This will look like the drone is sliding to the left or the right. Take a moment and consider how the drone's propellers generate thrust. What do you think is happening when the drone rolls right or left?



The general format of this command can be seen here:

```
drone.set_roll(power) # power can be between -100% and 100%
```

Try it out yourself! The code below will move the drone to the right, so make sure you have enough room.

```
from codrone_edu.drone import *
drone = Drone()
drone.pair()

drone.takeoff()
drone.set_roll(30) # set roll value to +30
drone.move(1)      # move with the set parameters for 1 second
drone.land()

drone.close()
```

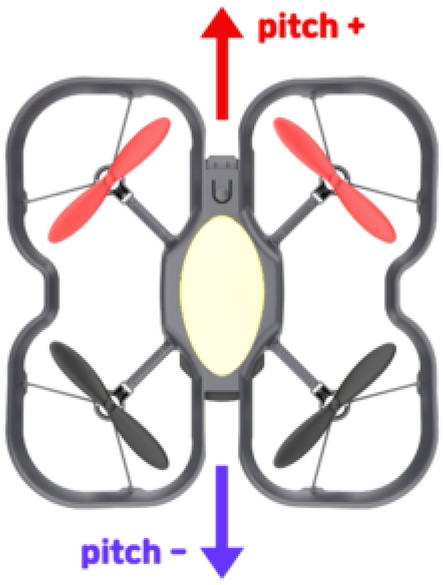
What did you observe? The drone should have moved to the right for 1 second.

Practice: Modify the code to make your drone fly LEFT for 1.5 seconds.

Step 5

Pitch

This is the CoDrone EDU's forward and backward tilt. Positive pitch will make the CoDrone EDU tilt and move forward, and negative pitch will make the CoDrone EDU tilt and move backwards.



```
drone.set_pitch(power)  # power can be between -100 and 100
```

To test out this code, make a copy of your roll.py file and call it pitch.py. Now change set_roll to set_pitch. You can leave all other values the same. So your code should look like:

```
from codrone_edu.drone import *
drone = Drone()
drone.pair()

drone.takeoff()
drone.set_pitch(30) # set pitch value to +30
drone.move(1)       # move with the set parameters for 1 second
```

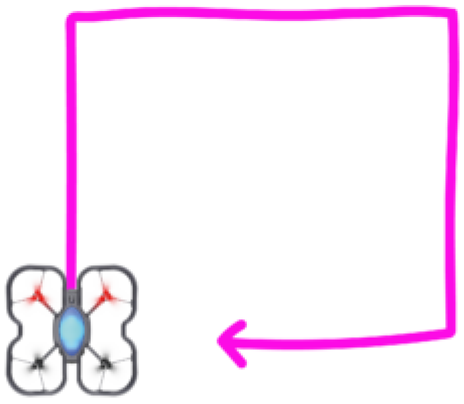
What did you observe? The drone should have moved forward for 1 second.

Practice: Modify the coded to make your drone fly BACKWARD for 1.5 seconds.

Step 6

Squaring Off

Now that we know the basics of our pitch and roll movements we can make simple shapes like squares! When setting flight movements its important to reset them back to 0 when you want a certain movement to stop. For example let's take a look at this code snippet below. First, our drone will go forward for 1 second. However, since we never reset our pitch to 0 our drone will combine both the pitch and roll to make a diagonal movement for 1 second on the second move command.



```
drone.set_pitch(30)
drone.move(1)
drone.set_roll(30)
drone.move(1)
```

In order to correctly go forward then go right to make the first half of a square we must reset the pitch to 0 like so

```
drone.set_pitch(30)
drone.move(1)
drone.set_pitch(0)
drone.set_roll(30)
drone.move(1)
```

If you feel confident try to finish the rest of the square! If you're unable to finish it feel free to checkout the finished square code below

```
from codrone_edu.drone import *
drone = Drone()
drone.pair()
drone.takeoff()

drone.set_pitch(30)  # setting forward pitch

drone.move(1)  # moving forward!

drone.set_pitch(0)  # resetting pitch to 0

drone.set_roll(30)  # setting roll to the right

drone.move(1)  # moving right!

drone.set_roll(0)  # resetting roll to 0

drone.set_pitch(-30)  # setting backwards pitch

drone.move(1)  # moving backwards!

drone.set_pitch(0)  # resetting pitch to 0

drone.set_roll(-30)  # setting roll to the left

drone.move(1)  # moving left!

drone.land()

drone.close()
```

Step 7

Challenges

Challenge: Flight Practice!

For this challenge, find two pieces of paper and some scotch tape. If you don't have scotch tape that's okay! Find a paper weight or use something heavier like a text book. For this lesson your two pieces of paper will mark a take off zone or a landing zone. Once you determine where each zone is use your scotch tape to keep your landing zone in place. The farther apart these zones are the greater the challenge will be! With our new knowledge of Pitch and Roll we will write a program to get some practice with these new flight movements!

1. Write a program where your drone flies from the take off zone to the landing zone. For this exercise have your drone start by facing towards the landing zone

2. Write a program where your drone flies from the take off zone to the landing zone. For this exercise face your drone so that it is not facing directly towards or away from the landing zone. Make sure your mark on your paper where your drone is facing. This way it can start from the same position every time. (hint: You should have to use roll this time!)

3. Write a program where your drone flies from the takeoff zone to the landing zone. For this exercise place an obstacle directly in between your take off and landing zones. Your drone can start by facing any direction. (hint: You should use both Pitch and Roll to avoid crashing into the obstacle!)

Challenge: Square Up!

For this challenge, try to program your CoDrone EDU to fly in the shape of a square using your new knowledge of Pitch and Roll. As an additional rule, make sure that the CoDrone EDU is facing the same direction for all parts of the square!

Step 8

Lesson complete

In this lesson, you learned how to send roll and pitch commands. In the next lesson, you will learn how to fly your drone up and down and turn!

Lesson Complete!