November 8
2018

Python Pilots

Taehoon Kim, Charidy Paige, William Pepper, Siyu Xiang

Milestone 1
Presentation

Agenda

- Present to your class Milestone 1.0 of your project.
- You must present working code and explain what it does and how it fulfils the user stories.
- Show and explain how the code was tested.
- Explain what remains to be done to accomplish Milestone 2.0.

Overview

Our group intends to build our own application to control a Tello drone from a PC instead of using off-the-self mobile applications.

python™

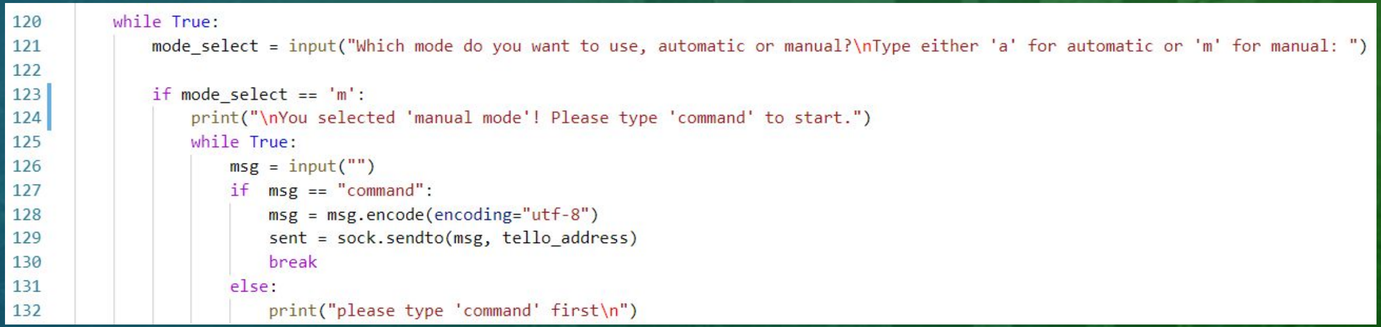
SQLite

Requirements: User Stories and Tasks

Choose Flight Mode

a. Description: As a user, I should be able to choose the flight mode (manual control or automatic flight plan) so that I can fly the drone in two different ways.

b. Tasks:
   i. Create a command to receive user input for flight mode (manual or automatic)



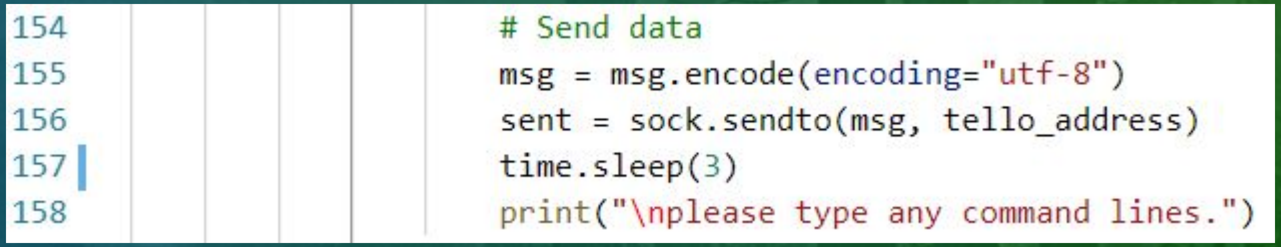
```
    while True:
        mode_select = input("Which mode do you want to use, automatic or manual?\nType either 'a' for automatic or 'm' for manual: ")

        if mode_select == 'm':
            print("\nYou selected 'manual mode'! Please type 'command' to start.")
            while True:
                msg = input("")
                if  msg == "command":
                    msg = msg.encode(encoding="utf-8")
                    sent = sock.sendto(msg, tello_address)
                    break
                else:
                    print("please type 'command' first\n")
```

Requirements: User Stories and Tasks

Fly Drone in Manual Mode

a. Description: As a user, I should be able to control the drone's movement in real time so that I can improvise the flight path.

b. Tasks:
   i. Create commands to receive user input for flight direction, distance, and speed.
   ii. Create commands to send user input to drone for execution



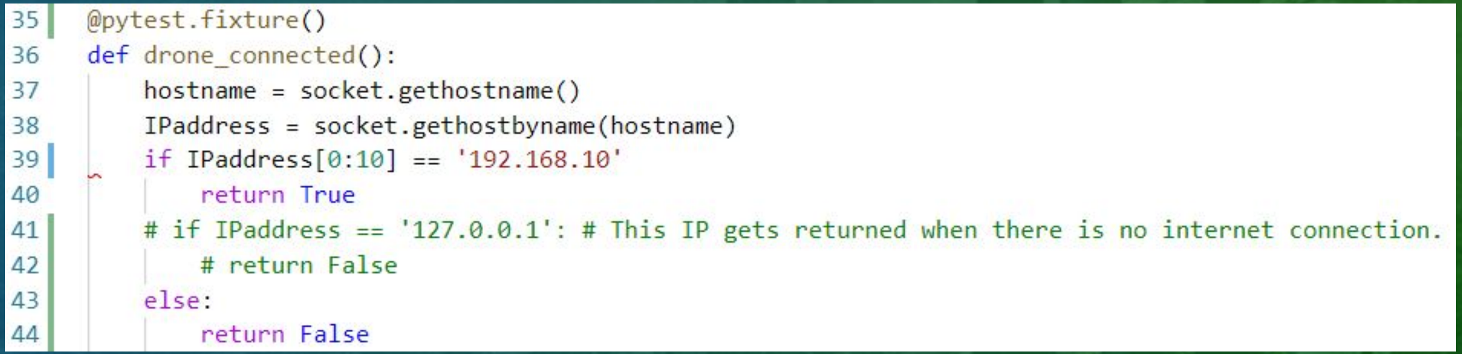
```
                # Send data
                msg = msg.encode(encoding="utf-8")
                sent = sock.sendto(msg, tello_address)
                time.sleep(3)
                print("\nplease type any command lines.")
```

Requirements: User Stories and Tasks

Display Drone Status

a. Description: As a user, I should be able to see the connection status of the drone. For example, if the drone is either connected or disconnected from the computer.

b. Tasks:
   i. Create a command to request the drone's status.
   ii. Create a return message which prints the connection status (drone is connected or disconnected).



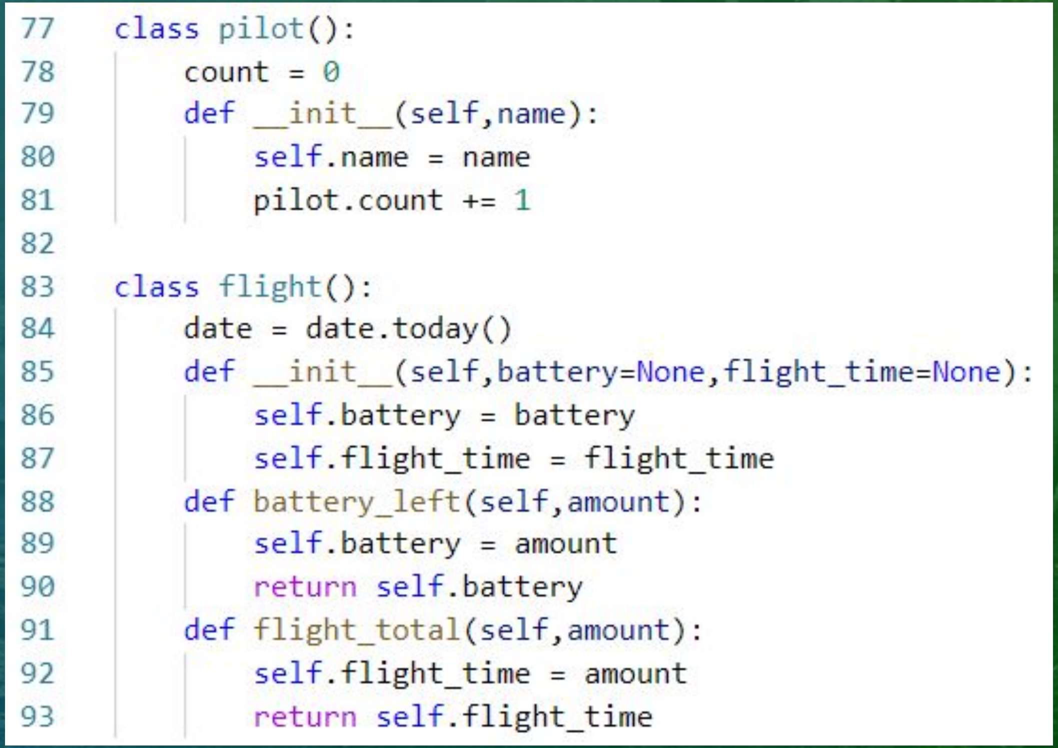
```
@pytest.fixture()
def drone_connected():
    hostname = socket.gethostname()
    IPaddress = socket.gethostbyname(hostname)
    if IPaddress[0:10] == '192.168.10'
        return True
    # if IPaddress == '127.0.0.1': # This IP gets returned when there is no internet connection.
        # return False
    else:
        return False
```

Requirements: User Stories and Tasks

Take Off

a. As a user, I should be able to launch the drone from the ground into the air, so that I can begin a flight route.

b. Tasks:
   i. Create a command to connect with the drone
   ii. Create a Flight class
   iii. Create a Flight object in the datastore



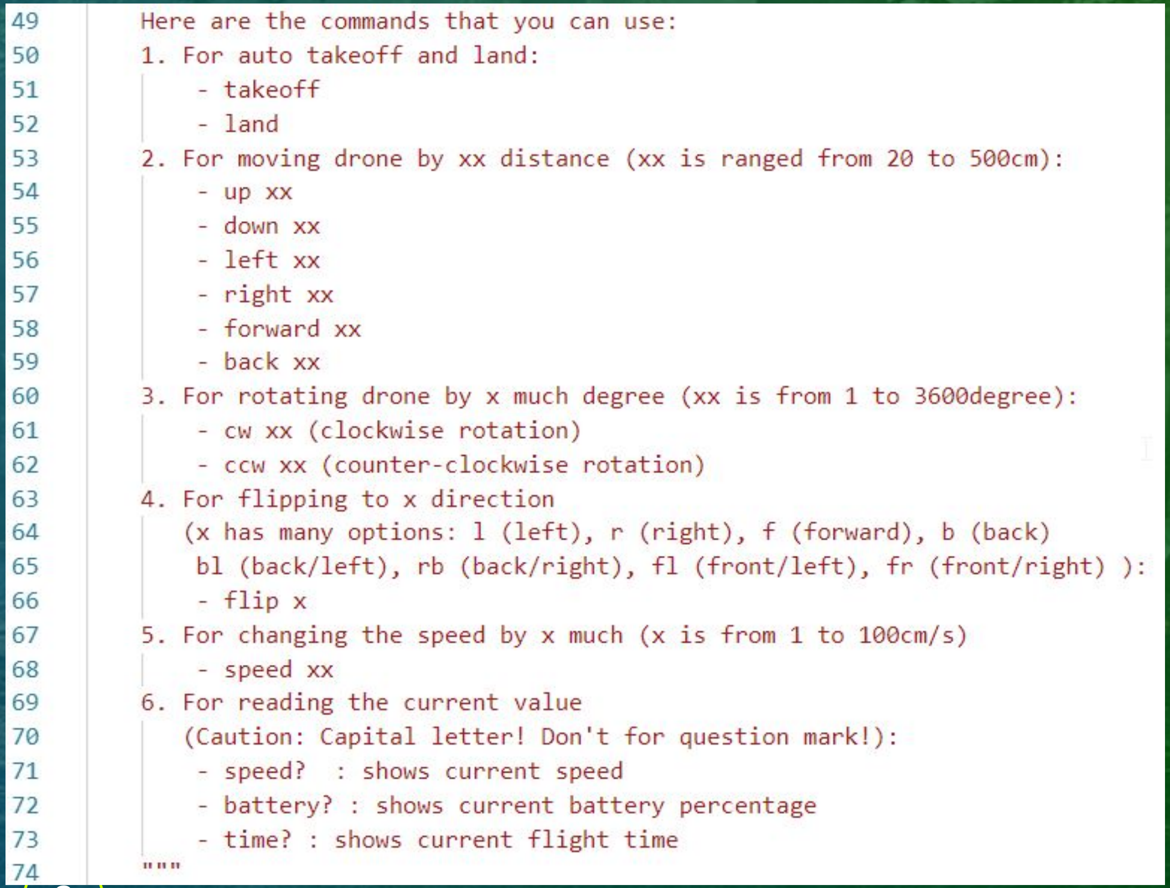
```
class pilot():
    count = 0
    def __init__(self,name):
        self.name = name
        pilot.count += 1

class flight():
    date = date.today()
    def __init__(self,battery=None,flight_time=None):
        self.battery = battery
        self.flight_time = flight_time
    def battery_left(self,amount):
        self.battery = amount
        return self.battery
    def flight_total(self,amount):
        self.flight_time = amount
        return self.flight_time
```

Requirements: User Stories and Tasks

Land

a. As a user, I should be able to land the drone safely on the ground so that I can complete the flight.

b. Tasks:
   i. Program the code for landing
      1. Create a command to initiate landing
      2. Stop drone after successful landing



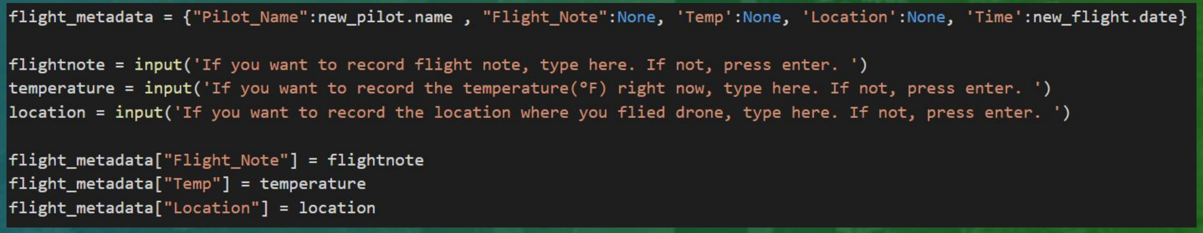
```
    Here are the commands that you can use:
    1. For auto takeoff and land:
        - takeoff
        - land
    2. For moving drone by xx distance (xx is ranged from 20 to 500cm):
        - up xx
        - down xx
        - left xx
        - right xx
        - forward xx
        - back xx
    3. For rotating drone by x much degree (xx is from 1 to 3600degree):
        - cw xx (clockwise rotation)
        - ccw xx (counter-clockwise rotation)
    4. For flipping to x direction
       (x has many options: l (left), r (right), f (forward), b (back)
        bl (back/left), rb (back/right), fl (front/left), fr (front/right) ):
        - flip x
    5. For changing the speed by x much (x is from 1 to 100cm/s)
        - speed xx
    6. For reading the current value
       (Caution: Capital letter! Don't for question mark!):
        - speed?  : shows current speed
        - battery? : shows current battery percentage
        - time? : shows current flight time
    """
```

Requirements: User Stories and Tasks

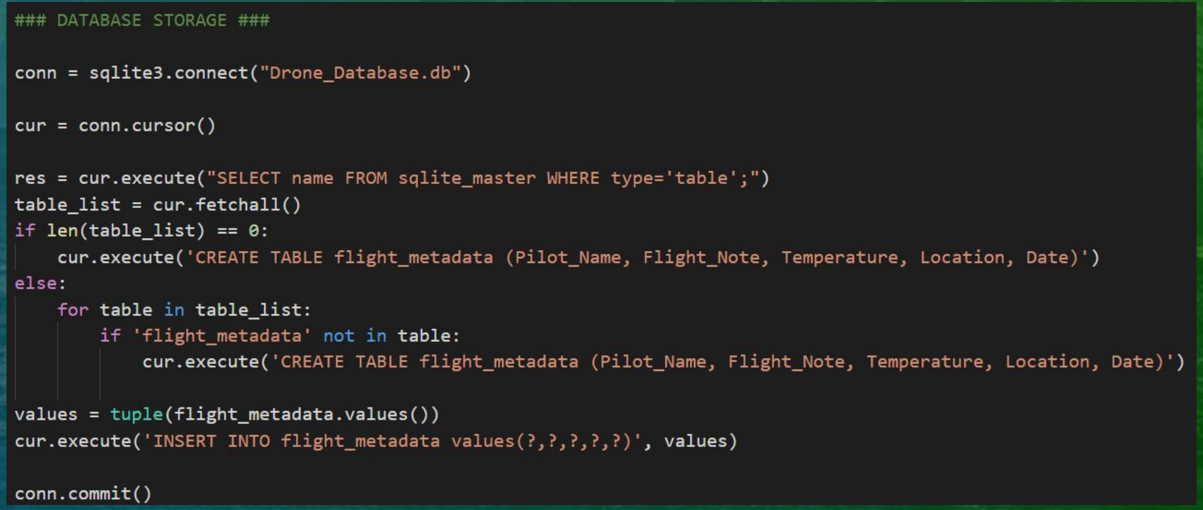
Record Flight Metadata

a. Description: As a user, I should be able to write flight notes after a flight so that I can have a record for flights for reference (e.g. debugging)

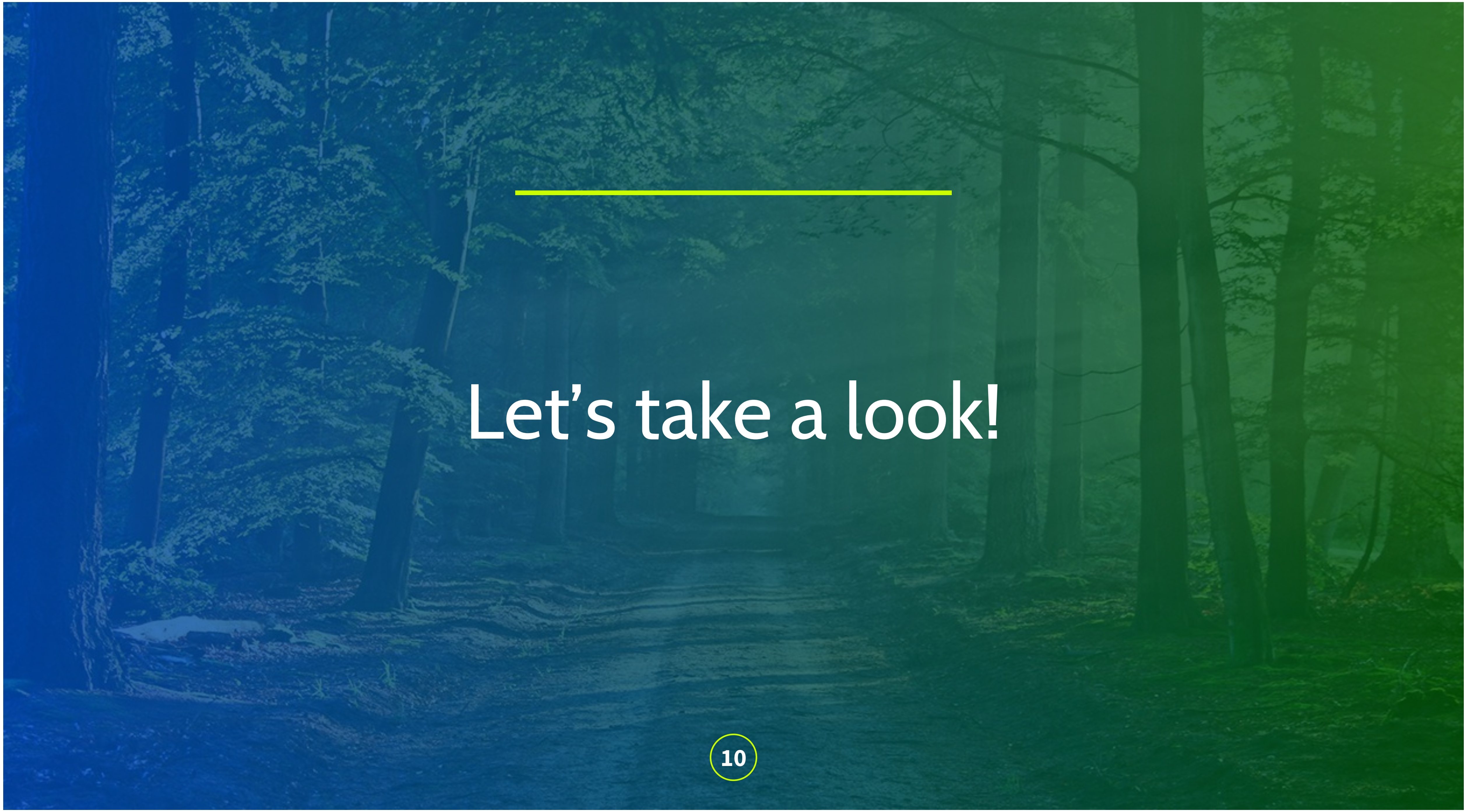
b. Tasks:
   i. Create a command to accept user input for flight notes
   ii. Create a command to accept user input for temperature during flight
   iii. Create a command to accept user input for pilot name after flight
   iv. Save flight notes, temperature, and pilot name as a part of the Flight object



```python
flight_metadata = {"Pilot_Name":new_pilot.name , "Flight_Note":None, 'Temp':None, 'Location':None, 'Time':new_flight.date}

flightnote = input('If you want to record flight note, type here. If not, press enter. ')
temperature = input('If you want to record the temperature(°F) right now, type here. If not, press enter. ')
location = input('If you want to record the location where you flied drone, type here. If not, press enter. ')

flight_metadata["Flight_Note"] = flightnote
flight_metadata["Temp"] = temperature
flight_metadata["Location"] = location
```

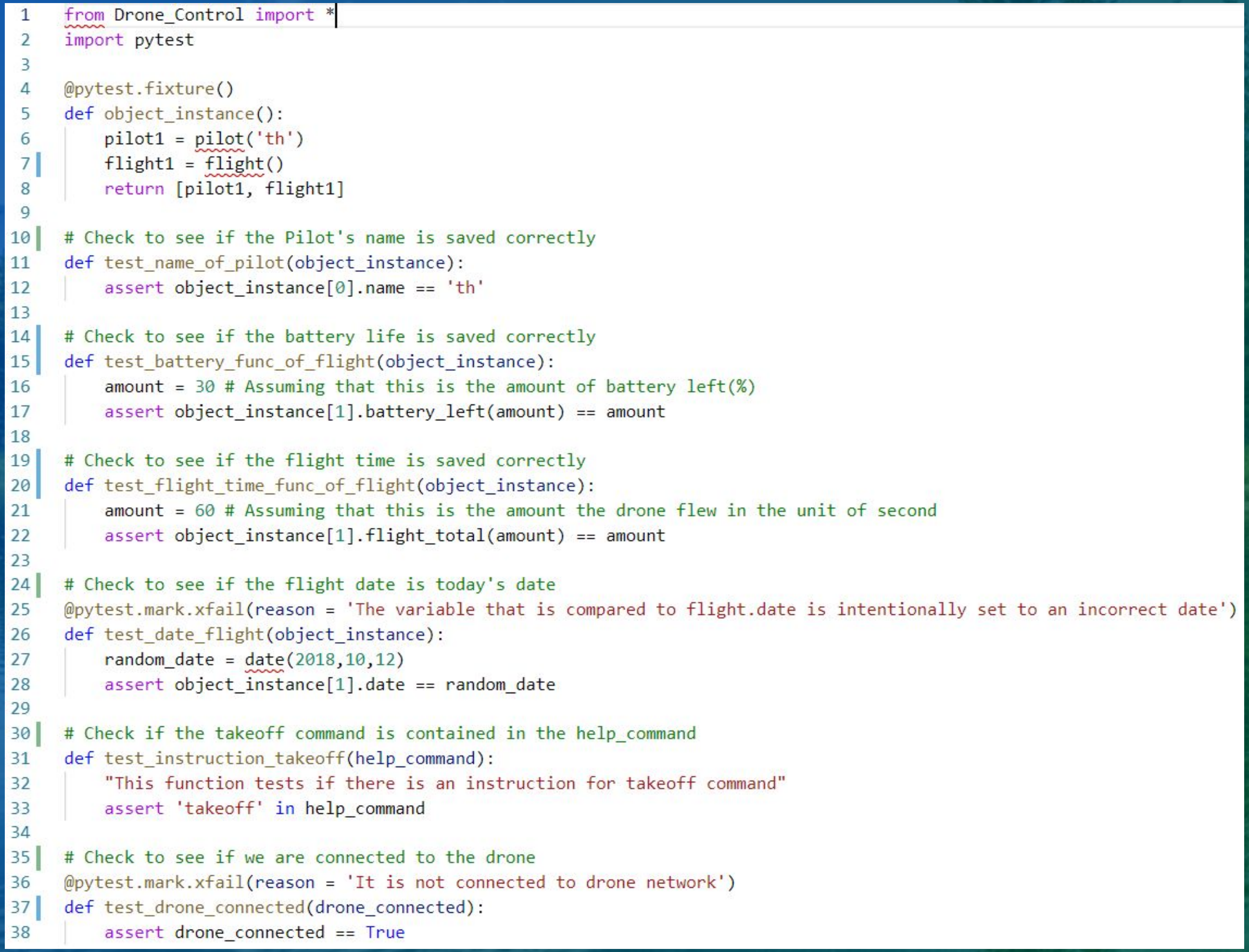


```python
### DATABASE STORAGE ###

conn = sqlite3.connect("Drone_Database.db")

cur = conn.cursor()

res = cur.execute("SELECT name FROM sqlite_master WHERE type='table';")
table_list = cur.fetchall()
if len(table_list) == 0:
    cur.execute('CREATE TABLE flight_metadata (Pilot_Name, Flight_Note, Temperature, Location, Date)')
else:
    for table in table_list:
        if 'flight_metadata' not in table:
            cur.execute('CREATE TABLE flight_metadata (Pilot_Name, Flight_Note, Temperature, Location, Date)')

values = tuple(flight_metadata.values())
cur.execute('INSERT INTO flight_metadata values(?,?,?,?,?)', values)

conn.commit()
```



Let’s take a look!

Tests



```python
from Drone_Control import *
import pytest

@pytest.fixture()
def object_instance():
    pilot1 = pilot('th')
    flight1 = flight()
    return [pilot1, flight1]

# Check to see if the Pilot's name is saved correctly
def test_name_of_pilot(object_instance):
    assert object_instance[0].name == 'th'

# Check to see if the battery life is saved correctly
def test_battery_func_of_flight(object_instance):
    amount = 30 # Assuming that this is the amount of battery left(%)
    assert object_instance[1].battery_left(amount) == amount

# Check to see if the flight time is saved correctly
def test_flight_time_func_of_flight(object_instance):
    amount = 60 # Assuming that this is the amount the drone flew in the unit of second
    assert object_instance[1].flight_total(amount) == amount

# Check to see if the flight date is today's date
@pytest.mark.xfail(reason = 'The variable that is compared to flight.date is intentionally set to an incorrect date')
def test_date_flight(object_instance):
    random_date = date(2018,10,12)
    assert object_instance[1].date == random_date

# Check if the takeoff command is contained in the help_command
def test_instruction_takeoff(help_command):
    "This function tests if there is an instruction for takeoff command"
    assert 'takeoff' in help_command

# Check to see if we are connected to the drone
@pytest.mark.xfail(reason = 'It is not connected to drone network')
def test_drone_connected(drone_connected):
    assert drone_connected == True
```

Milestone 2 Prep

Next steps

- Automatic mode and flight plan
- Enhance GUI
- Connect GUI to program
- Enhancements to accommodate multiple network connections
- Experiment with remote server (persistent database)
- Further testing (e.g. database)

Requirements User Stories and Tasks

Fly Drone in Automatic Mode

a. Description: As a user, I should be able to choose a pre programmed flight route and the drone fly with that one command.

b. Tasks:
   i. Create commands to receive user input for flight direction, distance, and speed.
   ii. Create commands to send user input to drone for execution

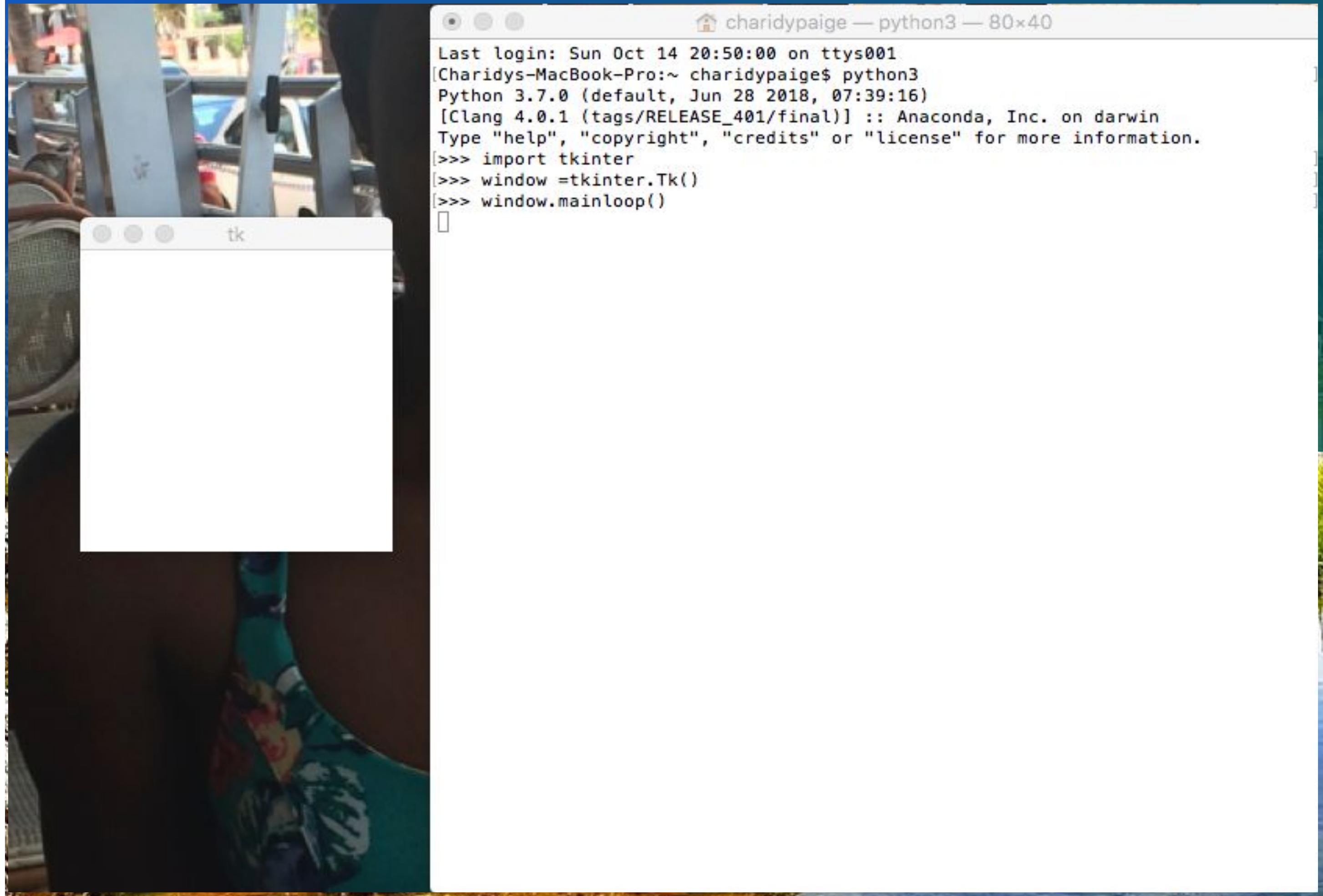
Requirements User Stories and Tasks

Define Flight Plan

a. Description: As a user, I should be able to design a flight plan in the GUI and have the drone execute the flight plan so that I can fly the drone automatically.

b. Tasks:
   i. Create a Flight Plan class
   ii. Accept user input for Flight Plan
   iii. Create Flight Plan object
   iv. Execute Flight Plan object
   v. Convert user input into executable instructions
   vi. Example test case: Execute and test a flight plan in a square

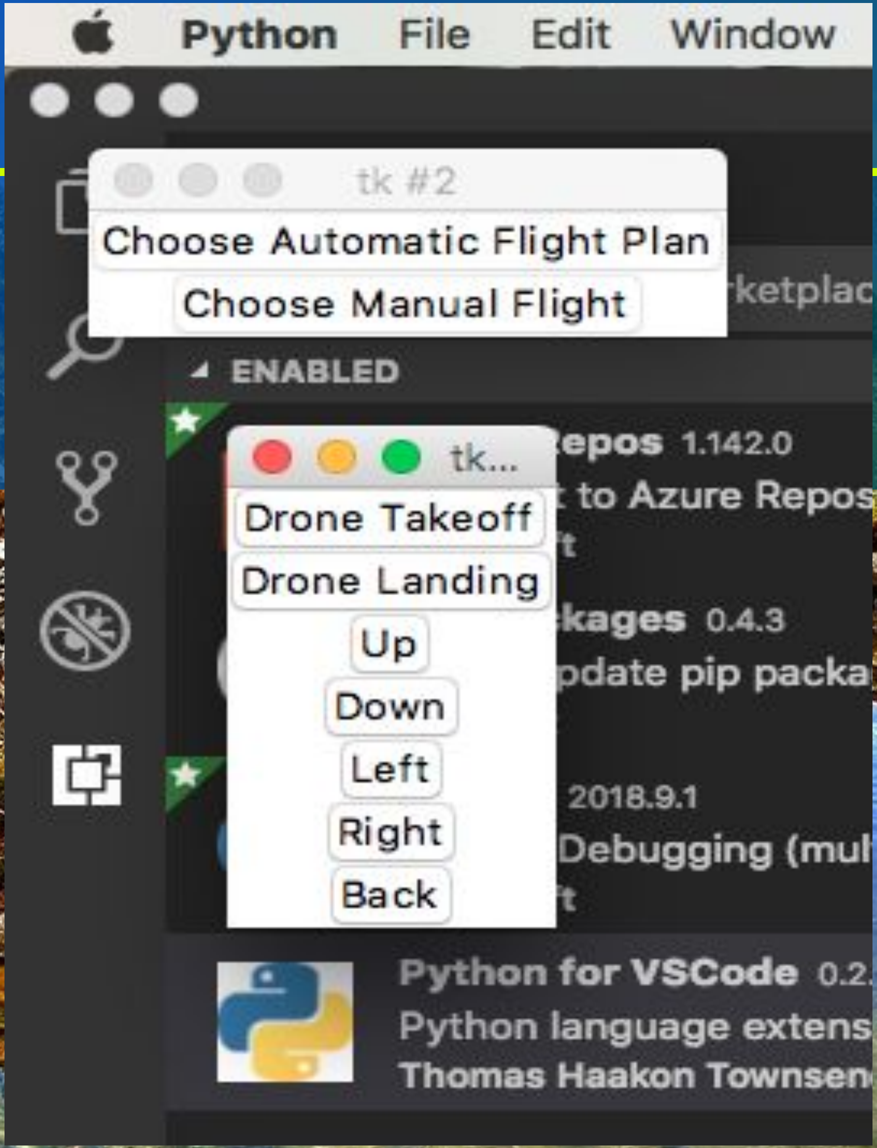
GUI- Tkinter



```
Last login: Sun Oct 14 20:50:00 on ttys001
Charidys-MacBook-Pro:~ charidypaige$ python3
Python 3.7.0 (default, Jun 28 2018, 07:39:16)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import tkinter
>>> window =tkinter.Tk()
>>> window.mainloop()
```

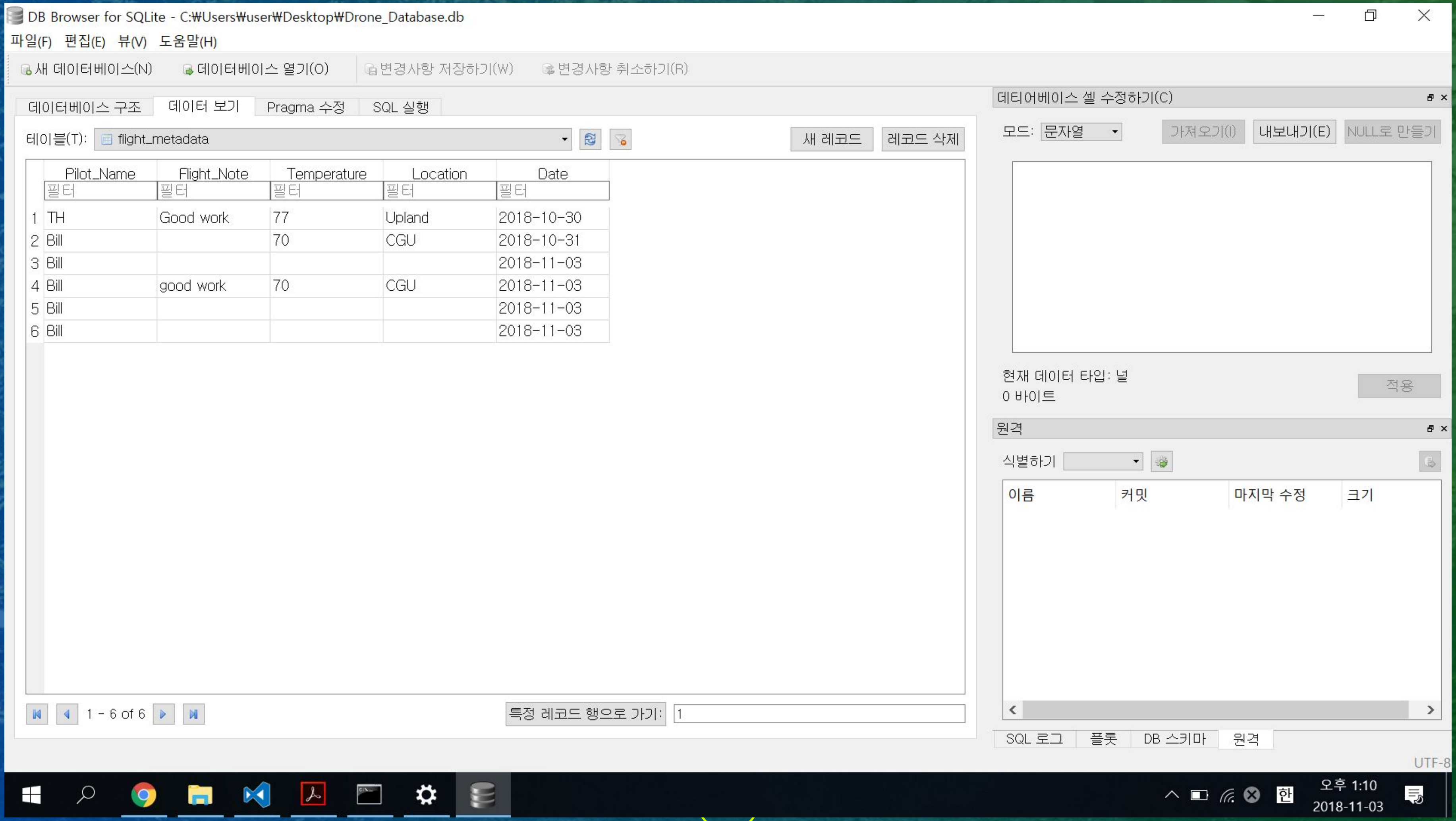
- Tkinter is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI.
- Not the only programming toolkit for Python, but the most popular.

GUI- Tkinter



- Current GUI
- Make more visually appealing.
- Add more sophisticated looking buttons
- Add color

Database - SQLite3



테이블(T): flight_metadata

	Pilot_Name	Flight_Note	Temperature	Location	Date
	필터	필터	필터	필터	필터
1	TH	Good work	77	Upland	2018-10-30
2	Bill		70	CGU	2018-10-31
3	Bill				2018-11-03
4	Bill	good work	70	CGU	2018-11-03
5	Bill				2018-11-03
6	Bill				2018-11-03

Challenges & Lessons Learned

- Agile methodology
  - Breaking user stories into concrete tasks
  - Planning Iterations and Milestones
- Limitations of third party systems
- Understanding SDK documentation
- User Datagram Protocol (UDP) as a method to transmit data
- How to use Tkinter and SQLite3

Link to our Github repository

https://github.com/thkim91/IST_303-GroupProject--Group2.git

THANK YOU!

Questions?

Taehoon Kim, Charidy Paige, William Pepper, Siyu Xiang