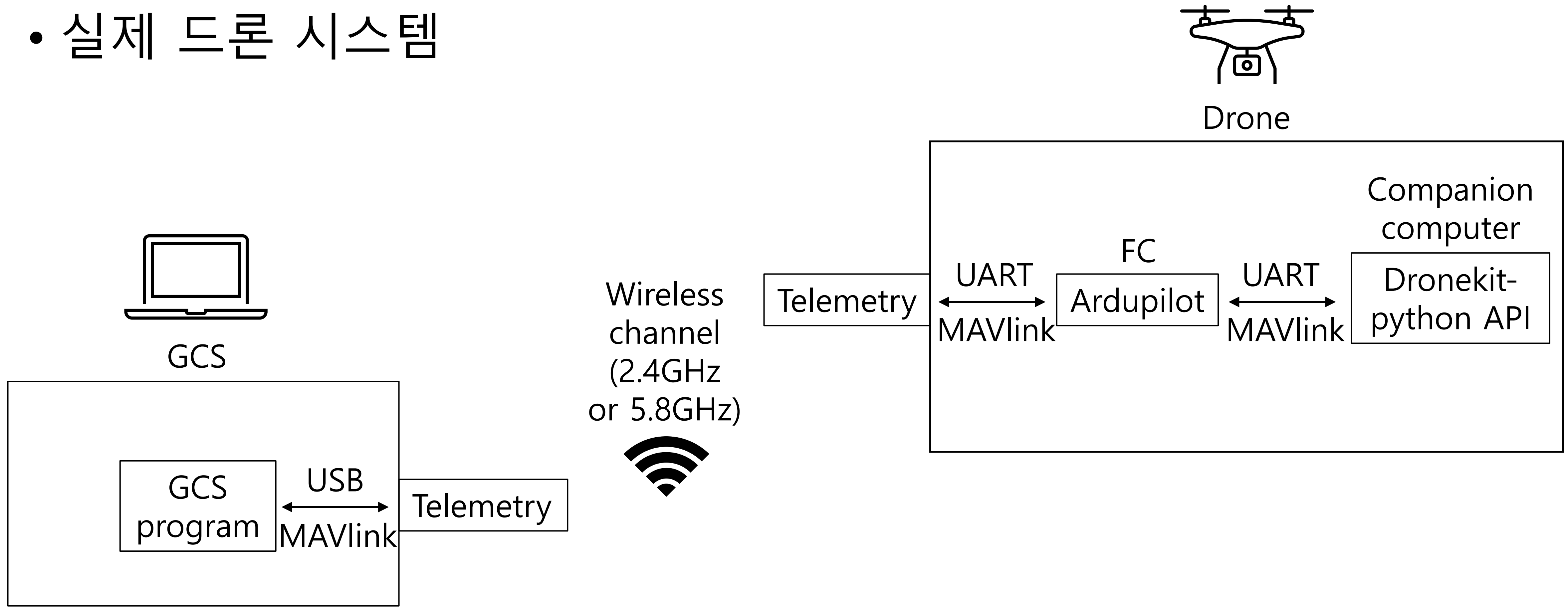
드론 세미나 3회차

HandS 2022년 1학기 드론 세미나

목차

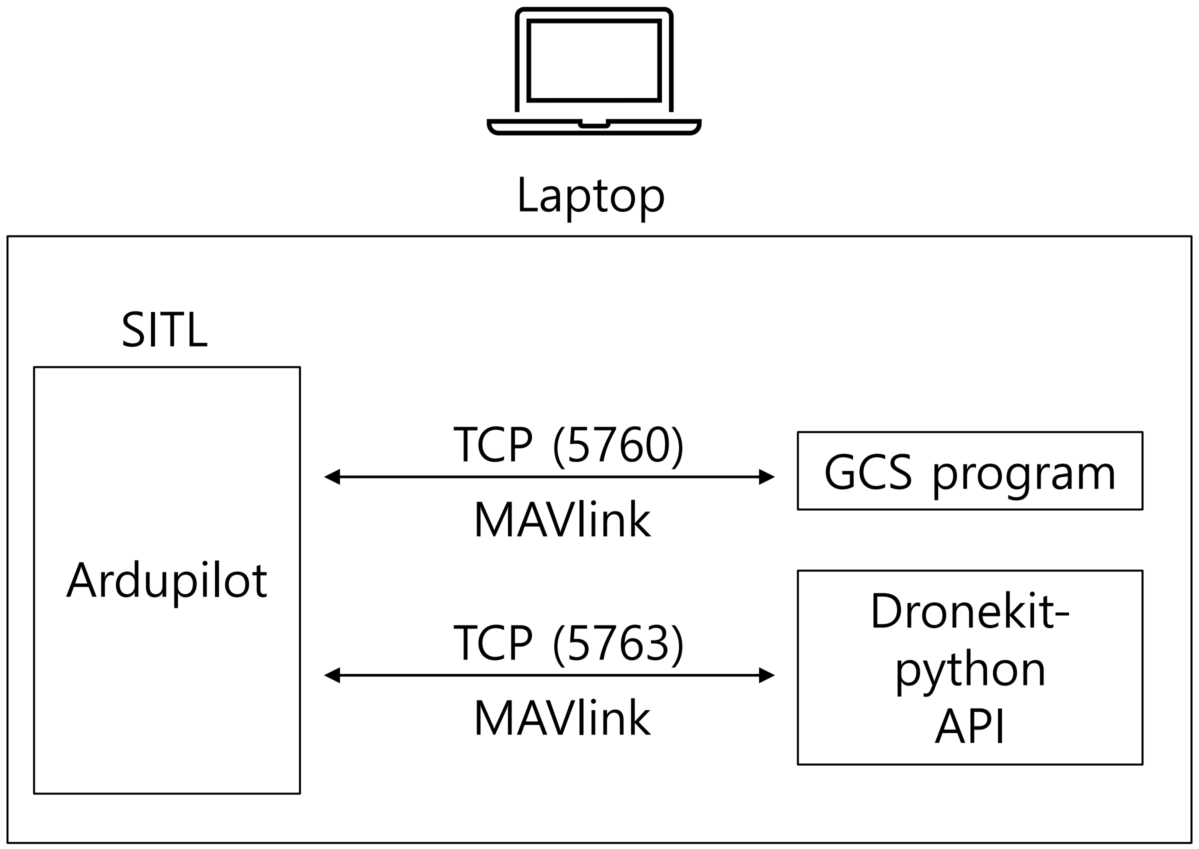
드론 시스템 개요

- 실제 드론 시스템



드론 시스템 개요

- SITL 시스템



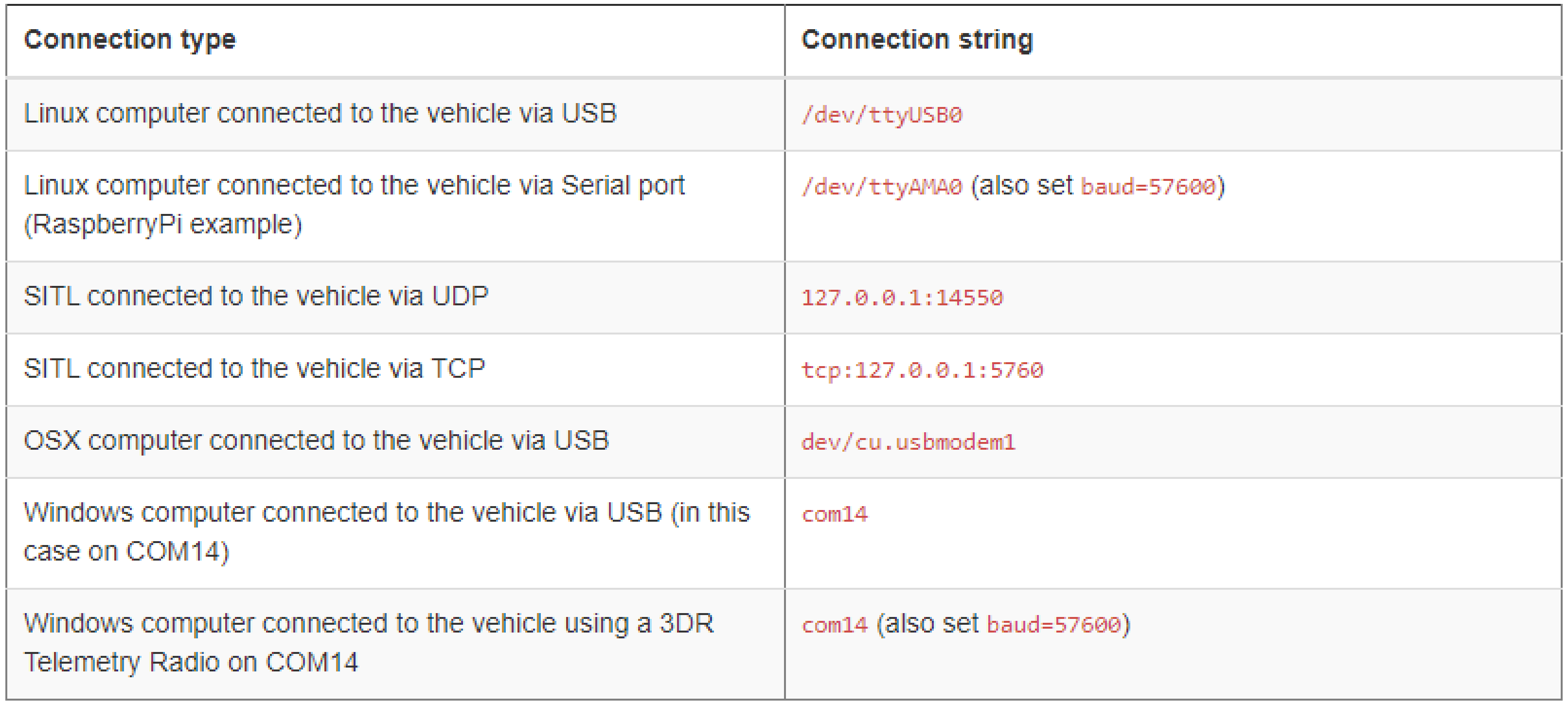
https://dronekit-python.readthedocs.io/en/latest/guide/connecting_vehicle.html
https://dronekit-python.readthedocs.io/en/latest/automodule.html#dronekit.connect

Connecting to a Vehicle

- dronekit.connect() 함수로 FC에서 구동 중인 Ardupilot과 연결하여 dronekit.Vehicle class의 instance로 불러옴

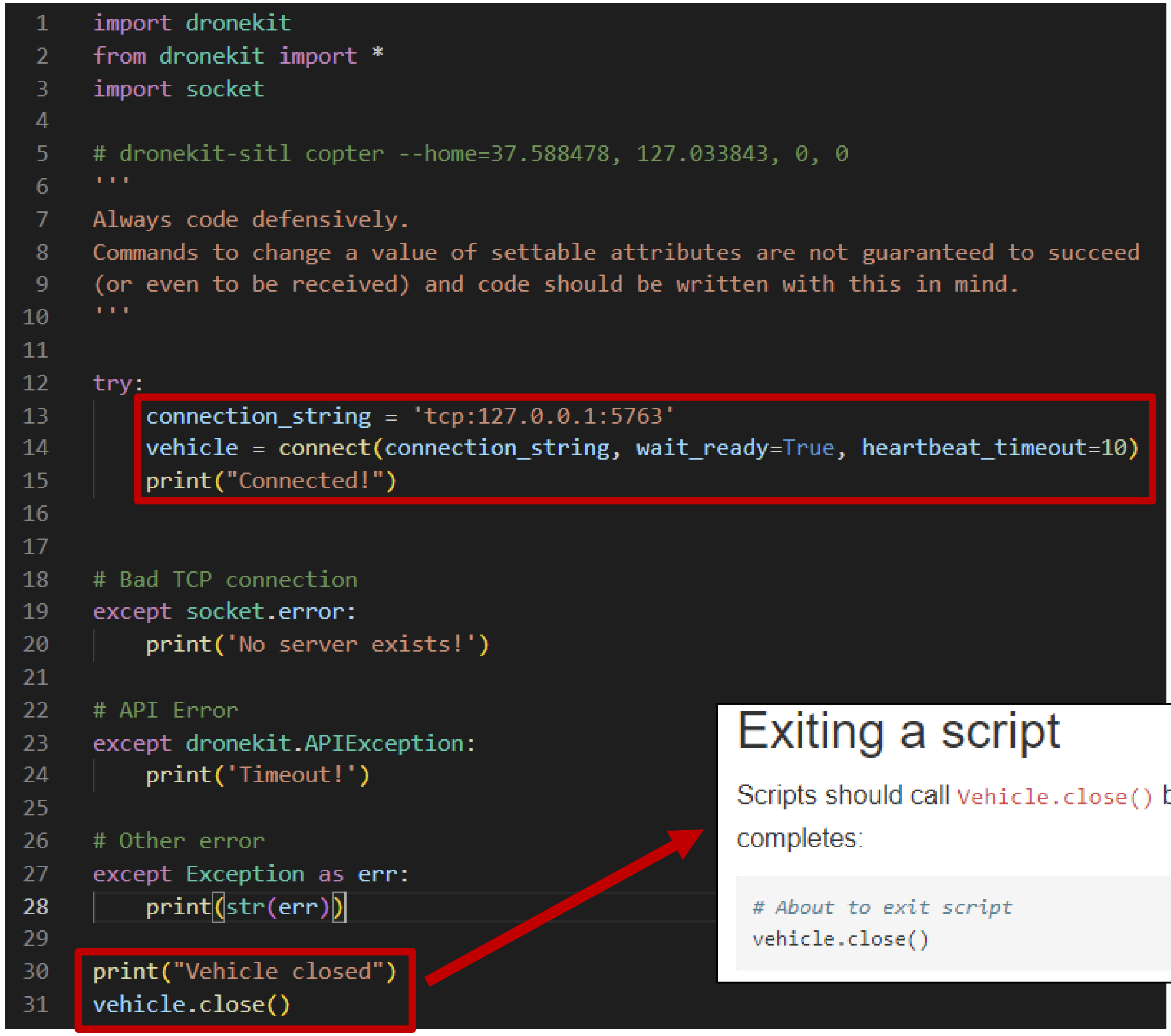
- 주요 parameters
  - ip(connection_string): 연결하려는 Ardupilot의 target address
  - wait_ready: True면 드론의 default 속성에 대한 정보를 받아올 때까지 다음 명령어로 넘어가지 않고 기다림 (default = None)
  - baud: dronekit-python과 드론 사이의 통신 속도 (default = 115200)
  - heartbeat_timeout: timeout value in seconds (default = 30)

Connecting to a Vehicle

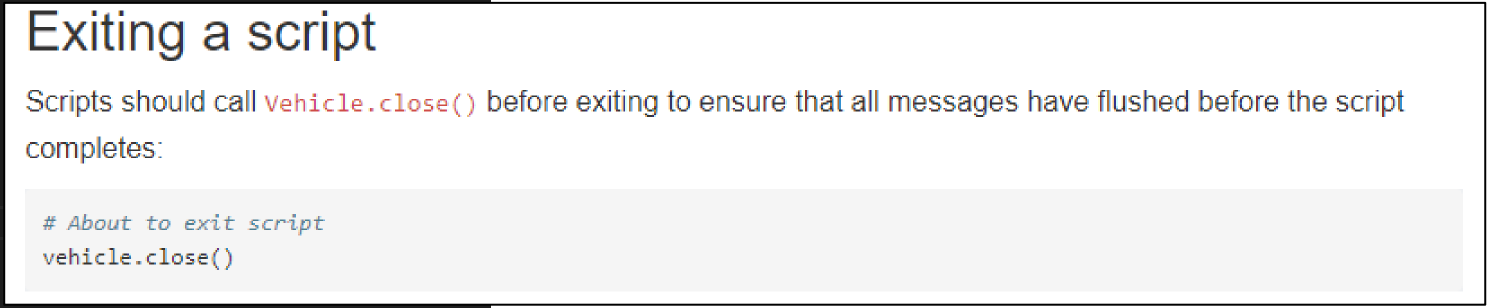


Connection type	Connection string
Linux computer connected to the vehicle via USB	`/dev/ttyUSB0`
Linux computer connected to the vehicle via Serial port (RaspberryPi example)	`/dev/ttyAMA0` (also set `baud=57600`)
SITL connected to the vehicle via UDP	`127.0.0.1:14550`
SITL connected to the vehicle via TCP	`tcp:127.0.0.1:5760`
OSX computer connected to the vehicle via USB	`dev/cu.usbmodem1`
Windows computer connected to the vehicle via USB (in this case on COM14)	`com14`
Windows computer connected to the vehicle using a 3DR Telemetry Radio on COM14	`com14` (also set `baud=57600`)

Connecting to a Vehicle



```python
import dronekit
from dronekit import *
import socket

# dronekit-sitl copter --home=37.588478, 127.033843, 0, 0
'''
Always code defensively.
Commands to change a value of settable attributes are not guaranteed to succeed
(or even to be received) and code should be written with this in mind.
'''

try:
    connection_string = 'tcp:127.0.0.1:5763'
    vehicle = connect(connection_string, wait_ready=True, heartbeat_timeout=10)
    print("Connected!")


# Bad TCP connection
except socket.error:
    print('No server exists!')

# API Error
except dronekit.APIException:
    print('Timeout!')

# Other error
except Exception as err:
    print(str(err))

print("Vehicle closed")
vehicle.close()
```



Exiting a script

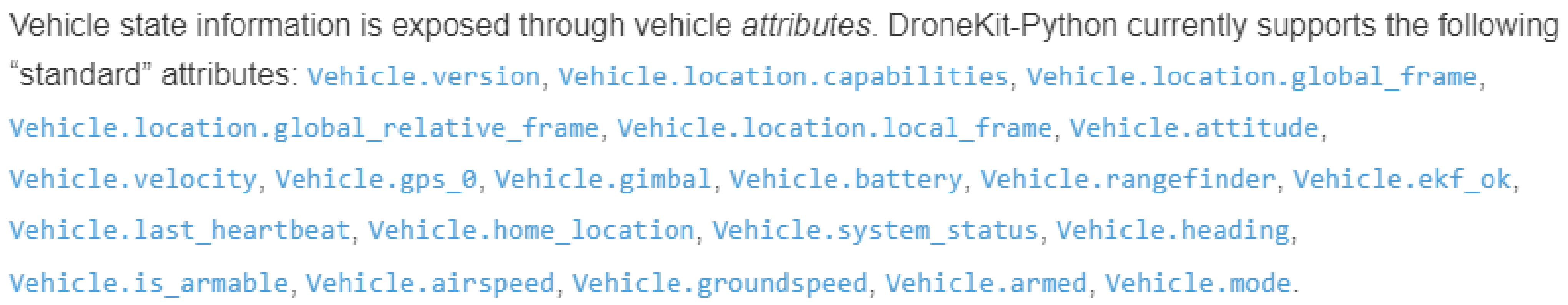
Scripts should call `Vehicle.close()` before exiting to ensure that all messages have flushed before the script completes:

```python
# About to exit script
vehicle.close()
```

https://dronekit-python.readthedocs.io/en/latest/guide/vehicle_state_and_parameters.html

Vehicle State and Settings

- dronekit.Vehicle class의 속성으로 펌웨어 및 드론의 상태에 관한 여러 정보를 불러올 수 있음



Vehicle state information is exposed through vehicle *attributes*. DroneKit-Python currently supports the following "standard" attributes: `Vehicle.version`, `Vehicle.location.capabilities`, `Vehicle.location.global_frame`, `Vehicle.location.global_relative_frame`, `Vehicle.location.local_frame`, `Vehicle.attitude`, `Vehicle.velocity`, `Vehicle.gps_0`, `Vehicle.gimbal`, `Vehicle.battery`, `Vehicle.rangefinder`, `Vehicle.ekf_ok`, `Vehicle.last_heartbeat`, `Vehicle.home_location`, `Vehicle.system_status`, `Vehicle.heading`, `Vehicle.is_armable`, `Vehicle.airspeed`, `Vehicle.groundspeed`, `Vehicle.armed`, `Vehicle.mode`.

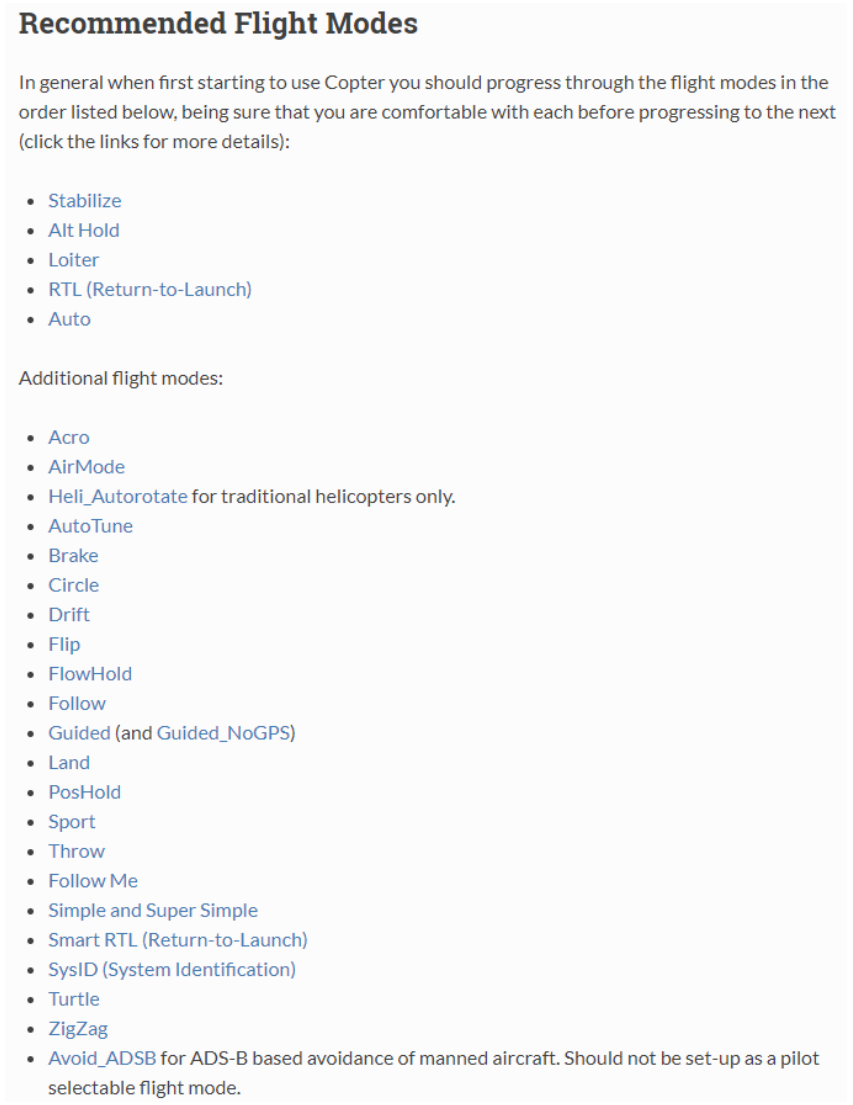
- home_location, gimbal, airspeed, groundspeed, mode, armed 만 설정(쓰기) 가능하고 나머지는 읽기만 됨

Vehicle State and Settings

- 주요 attributes
    - home_location: 보통 드론이 켜지고 GPS 신호를 처음 포착한 위치 (launch site)
    - global_frame: 위도(°), 경도(°), 고도(m)/ altitude는 해발고도
    - global_relative_frame: 위도(°), 경도(°), 고도(m)/ altitude는 home_location의 altitude를 기준으로 잡음
    - attitude: 드론의 pitch, yaw, roll 값(radian)을 불러옴
    - velocity: x, y, z 방향으로의 속도(m/s)
    - battery: 배터리 전압(V), 전류(10*mA), 잔량(%) 정보
    - airspeed: 수직방향 속력(m/s)
    - groundspeed: 수평방향 속력(m/s)
    - is_armable: 드론이 모터를 회전시킬 준비가 됐는지에 대한 Boolean 변수
    - armed: 모터가 돌아가는지를 나타내는 Boolean 변수
    - mode: https://ardupilot.org/copter/docs/flight-modes.html

https://dronekit-python.readthedocs.io/en/latest/guide/taking_off.html

Vehicle State and Settings



Recommended Flight Modes

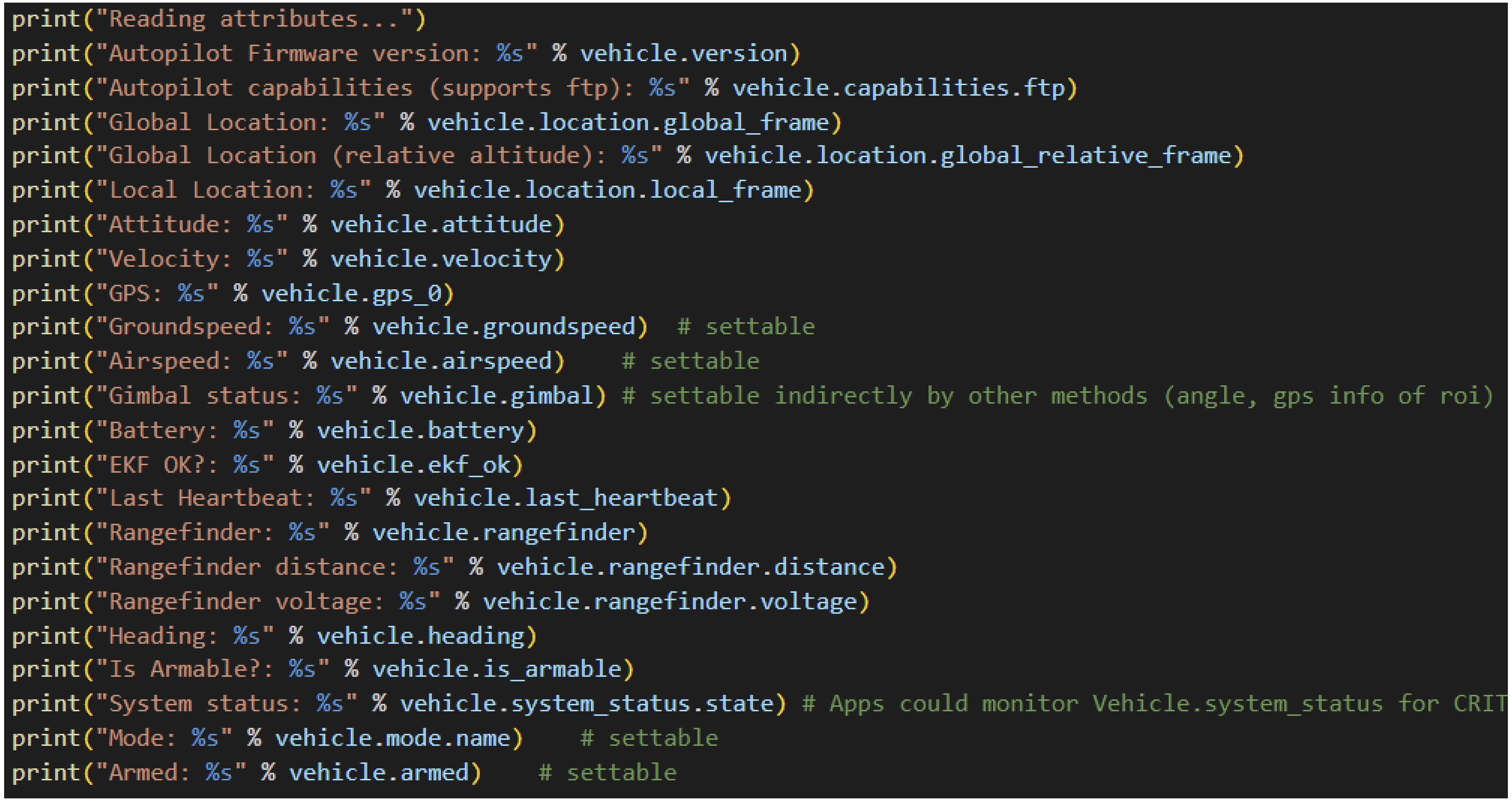
In general when first starting to use Copter you should progress through the flight modes in the order listed below, being sure that you are comfortable with each before progressing to the next (click the links for more details):

- Stabilize
- Alt Hold
- Loiter
- RTL (Return-to-Launch)
- Auto

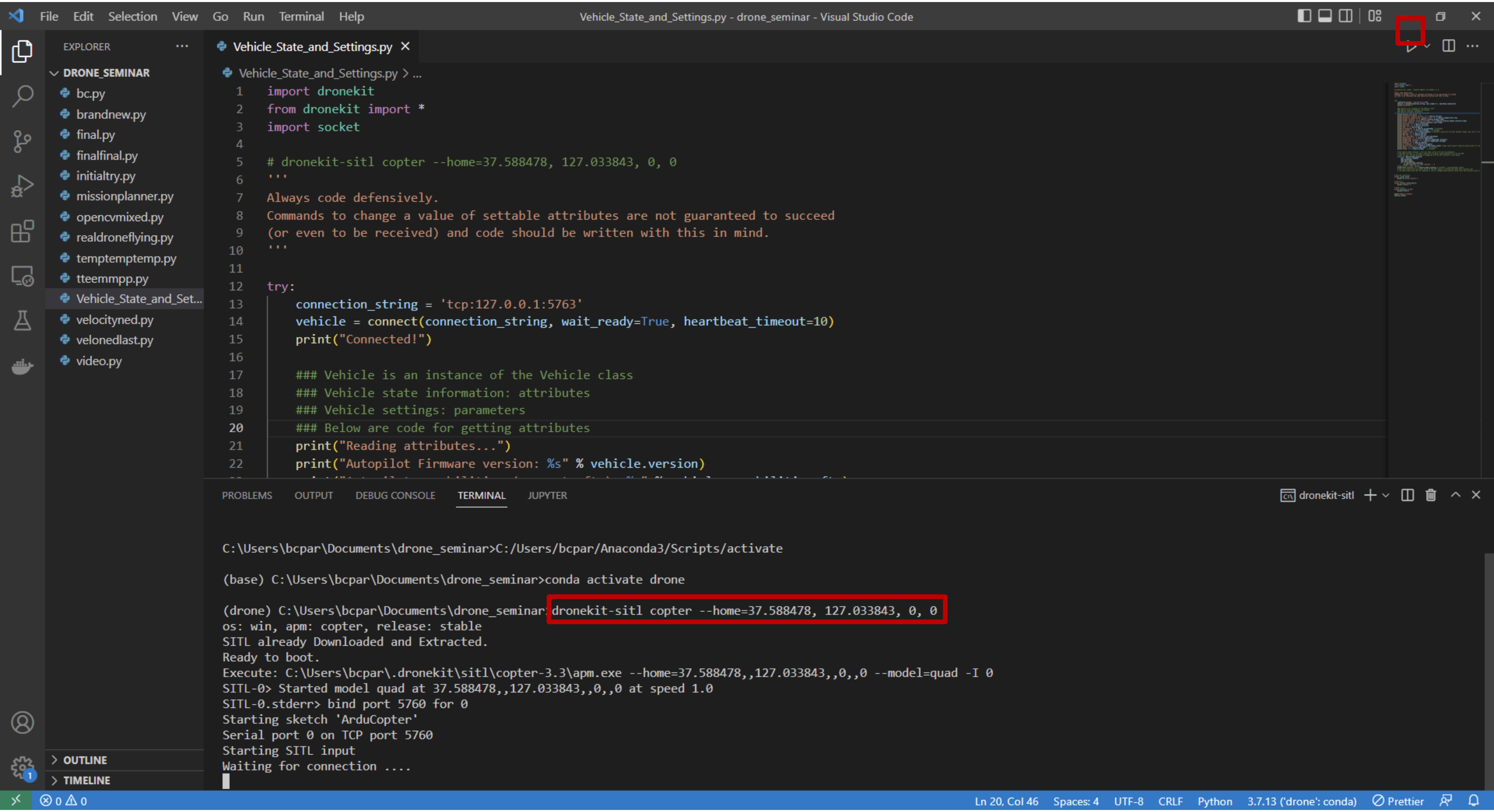
Additional flight modes:

- Acro
- AirMode
- Heli_Autorotate for traditional helicopters only.
- AutoTune
- Brake
- Circle
- Drift
- Flip
- FlowHold
- Follow
- Guided (and Guided_NoGPS)
- Land
- PosHold
- Sport
- Throw
- Follow Me
- Simple and Super Simple
- Smart RTL (Return-to-Launch)
- SysID (System Identification)
- Turtle
- ZigZag
- Avoid_ADSB for ADS-B based avoidance of manned aircraft. Should not be set-up as a pilot selectable flight mode.

Vehicle State and Settings

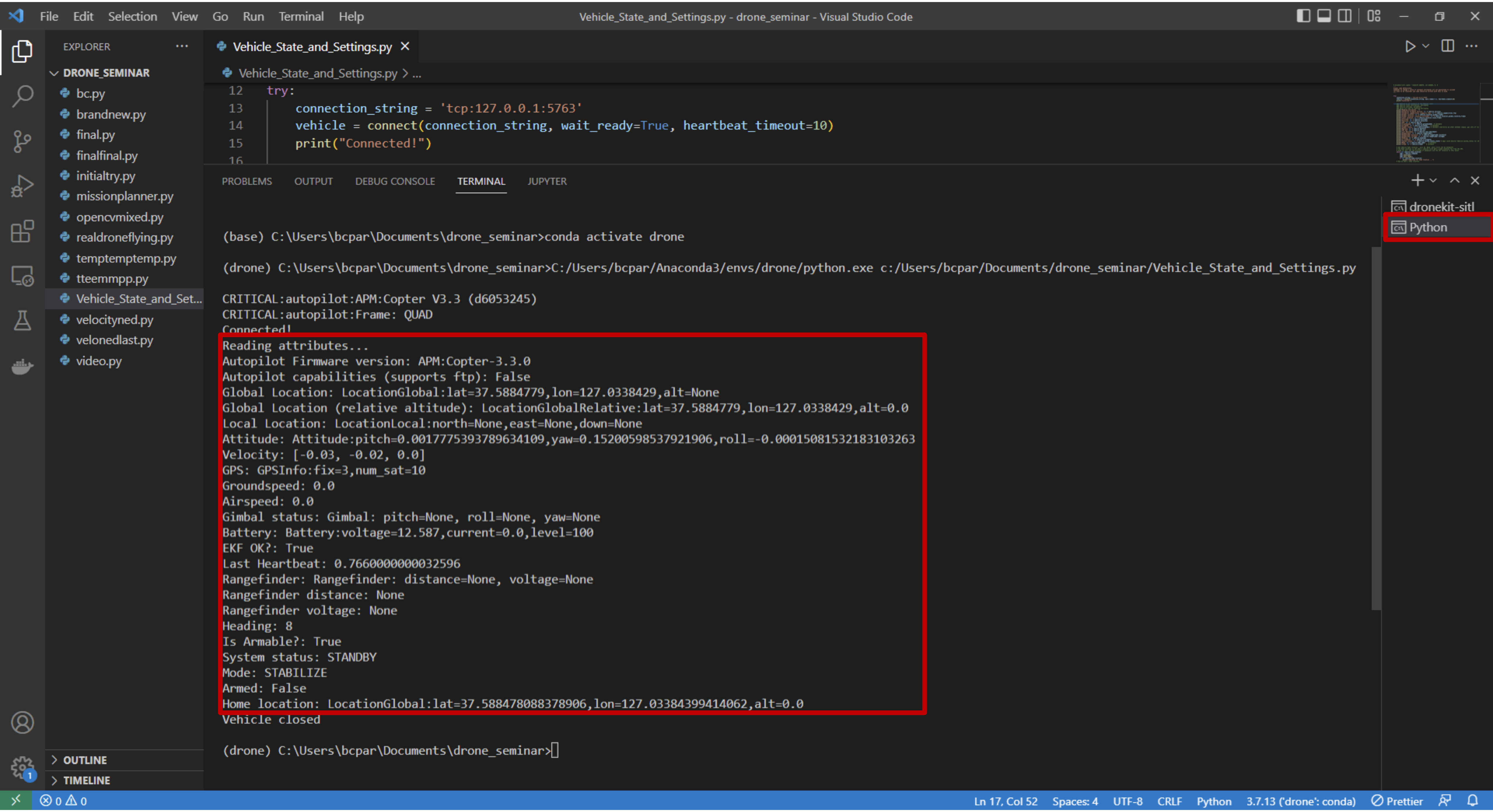


```
print("Reading attributes...")
print("Autopilot Firmware version: %s" % vehicle.version)
print("Autopilot capabilities (supports ftp): %s" % vehicle.capabilities.ftp)
print("Global Location: %s" % vehicle.location.global_frame)
print("Global Location (relative altitude): %s" % vehicle.location.global_relative_frame)
print("Local Location: %s" % vehicle.location.local_frame)
print("Attitude: %s" % vehicle.attitude)
print("Velocity: %s" % vehicle.velocity)
print("GPS: %s" % vehicle.gps_0)
print("Groundspeed: %s" % vehicle.groundspeed)  # settable
print("Airspeed: %s" % vehicle.airspeed)    # settable
print("Gimbal status: %s" % vehicle.gimbal) # settable indirectly by other methods (angle, gps info of roi)
print("Battery: %s" % vehicle.battery)
print("EKF OK?: %s" % vehicle.ekf_ok)
print("Last Heartbeat: %s" % vehicle.last_heartbeat)
print("Rangefinder: %s" % vehicle.rangefinder)
print("Rangefinder distance: %s" % vehicle.rangefinder.distance)
print("Rangefinder voltage: %s" % vehicle.rangefinder.voltage)
print("Heading: %s" % vehicle.heading)
print("Is Armable?: %s" % vehicle.is_armable)
print("System status: %s" % vehicle.system_status.state) # Apps could monitor Vehicle.system_status for CRIT
print("Mode: %s" % vehicle.mode.name)    # settable
print("Armed: %s" % vehicle.armed)    # settable
```



```python
import dronekit
from dronekit import *
import socket

# dronekit-sitl copter --home=37.588478, 127.033843, 0, 0
'''
Always code defensively.
Commands to change a value of settable attributes are not guaranteed to succeed
(or even to be received) and code should be written with this in mind.
'''

try:
    connection_string = 'tcp:127.0.0.1:5763'
    vehicle = connect(connection_string, wait_ready=True, heartbeat_timeout=10)
    print("Connected!")

    ### Vehicle is an instance of the Vehicle class
    ### Vehicle state information: attributes
    ### Vehicle settings: parameters
    ### Below are code for getting attributes
    print("Reading attributes...")
    print("Autopilot Firmware version: %s" % vehicle.version)
```

```
C:\Users\bcpar\Documents\drone_seminar>C:/Users/bcpar/Anaconda3/Scripts/activate

(base) C:\Users\bcpar\Documents\drone_seminar>conda activate drone

(drone) C:\Users\bcpar\Documents\drone_seminar>dronekit-sitl copter --home=37.588478, 127.033843, 0, 0
os: win, apm: copter, release: stable
SITL already Downloaded and Extracted.
Ready to boot.
Execute: C:\Users\bcpar\.dronekit\sitl\copter-3.3\apm.exe --home=37.588478,,127.033843,,0,,0 --model=quad -I 0
SITL-0> Started model quad at 37.588478,,127.033843,,0,,0 at speed 1.0
SITL-0.stderr> bind port 5760 for 0
Starting sketch 'ArduCopter'
Serial port 0 on TCP port 5760
Starting SITL input
Waiting for connection ....
```

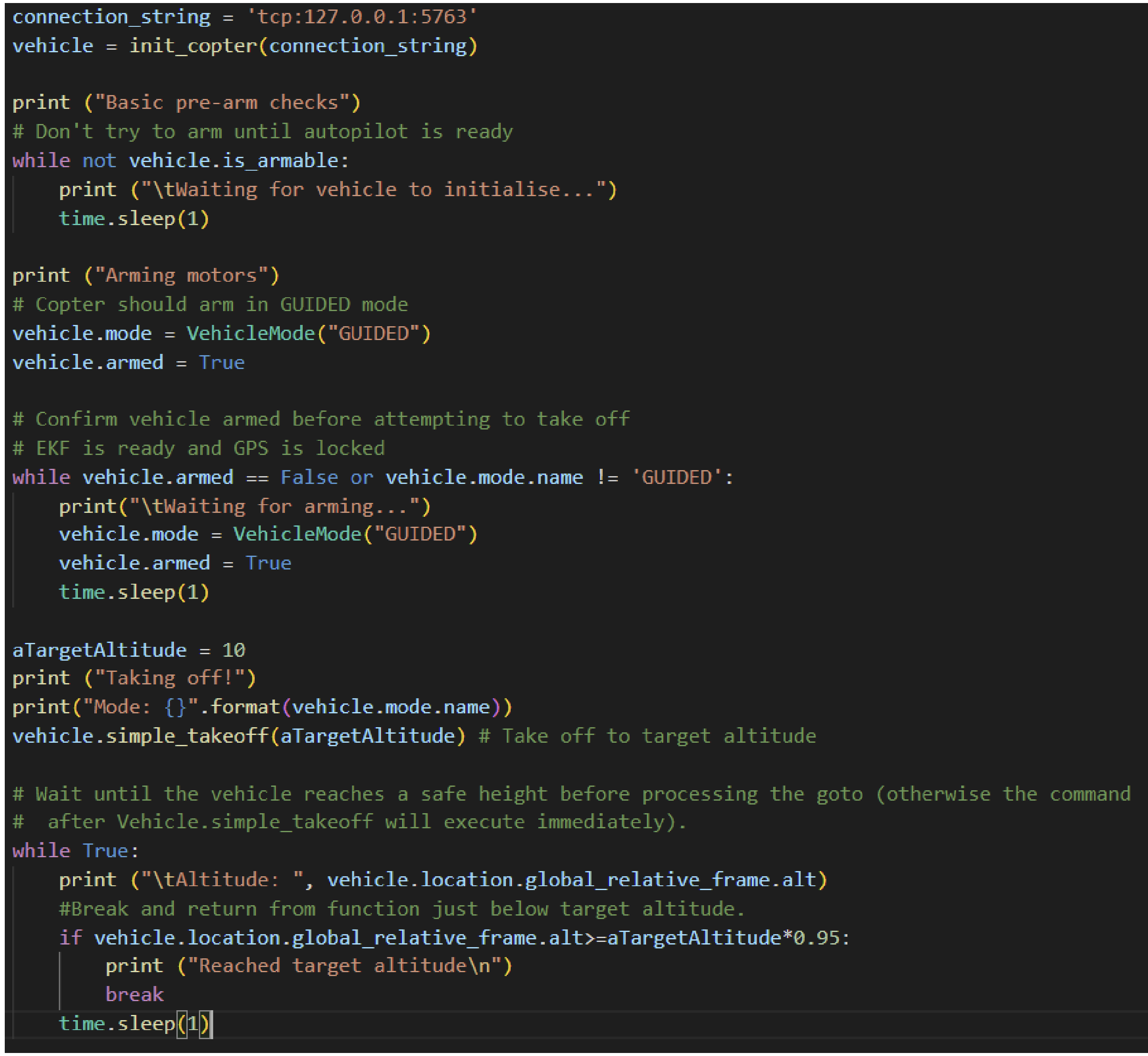


```python
try:
    connection_string = 'tcp:127.0.0.1:5763'
    vehicle = connect(connection_string, wait_ready=True, heartbeat_timeout=10)
    print("Connected!")
```

```
(base) C:\Users\bcpar\Documents\drone_seminar>conda activate drone

(drone) C:\Users\bcpar\Documents\drone_seminar>C:/Users/bcpar/Anaconda3/envs/drone/python.exe c:/Users/bcpar/Documents/drone_seminar/Vehicle_State_and_Settings.py

CRITICAL:autopilot:APM:Copter V3.3 (d6053245)
CRITICAL:autopilot:Frame: QUAD
Connected!
Reading attributes...
Autopilot Firmware version: APM:Copter-3.3.0
Autopilot capabilities (supports ftp): False
Global Location: LocationGlobal:lat=37.5884779,lon=127.0338429,alt=None
Global Location (relative altitude): LocationGlobalRelative:lat=37.5884779,lon=127.0338429,alt=0.0
Local Location: LocationLocal:north=None,east=None,down=None
Attitude: Attitude:pitch=0.0017775393789634109,yaw=0.15200598537921906,roll=-0.00015081532183103263
Velocity: [-0.03, -0.02, 0.0]
GPS: GPSInfo:fix=3,num_sat=10
Groundspeed: 0.0
Airspeed: 0.0
Gimbal status: Gimbal: pitch=None, roll=None, yaw=None
Battery: Battery:voltage=12.587,current=0.0,level=100
EKF OK?: True
Last Heartbeat: 0.7660000000032596
Rangefinder: Rangefinder: distance=None, voltage=None
Rangefinder distance: None
Rangefinder voltage: None
Heading: 8
Is Armable?: True
System status: STANDBY
Mode: STABILIZE
Armed: False
Home location: LocationGlobal:lat=37.588478088378906,lon=127.03384399414062,alt=0.0
Vehicle closed

(drone) C:\Users\bcpar\Documents\drone_seminar>
```

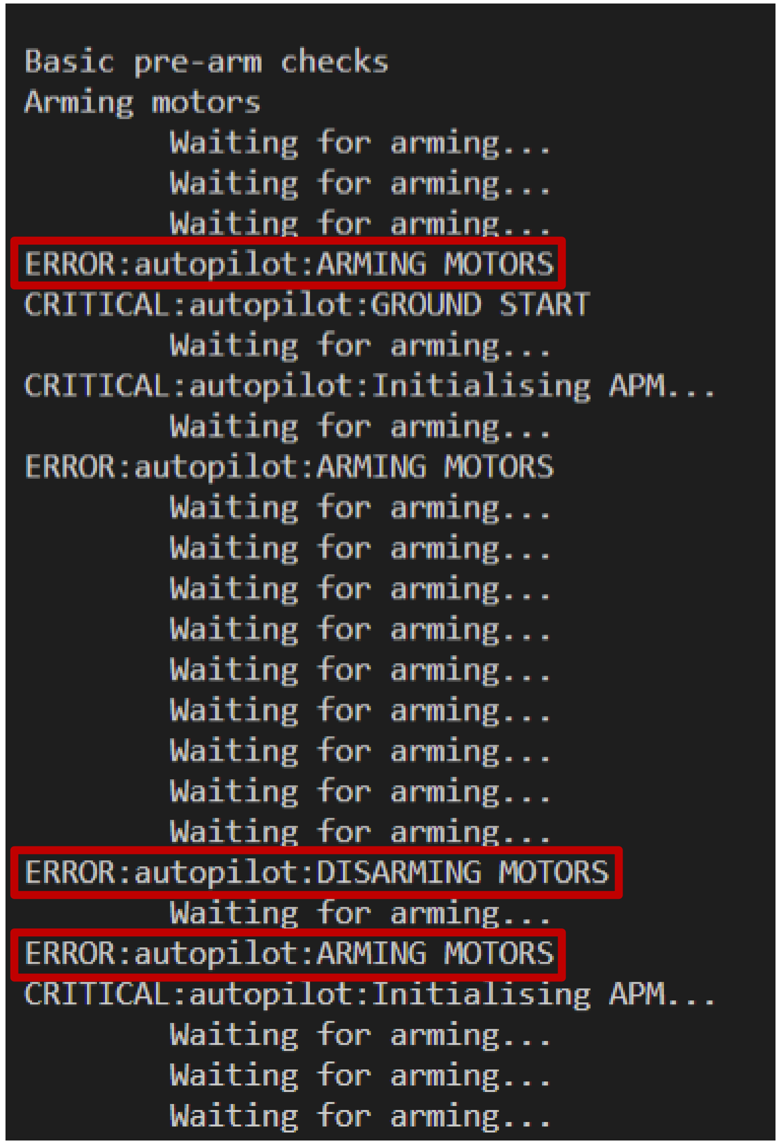
Taking off

- 이륙 순서
  - is_armable로 드론이 이륙할 준비가 됐는지 체크
  - 비행모드를 GUIDED로 바꿈
  - armed=True로 모터 작동
  - 이륙
  - 원하는 고도에 다다를 때까지 다음 명령어 실행 대기

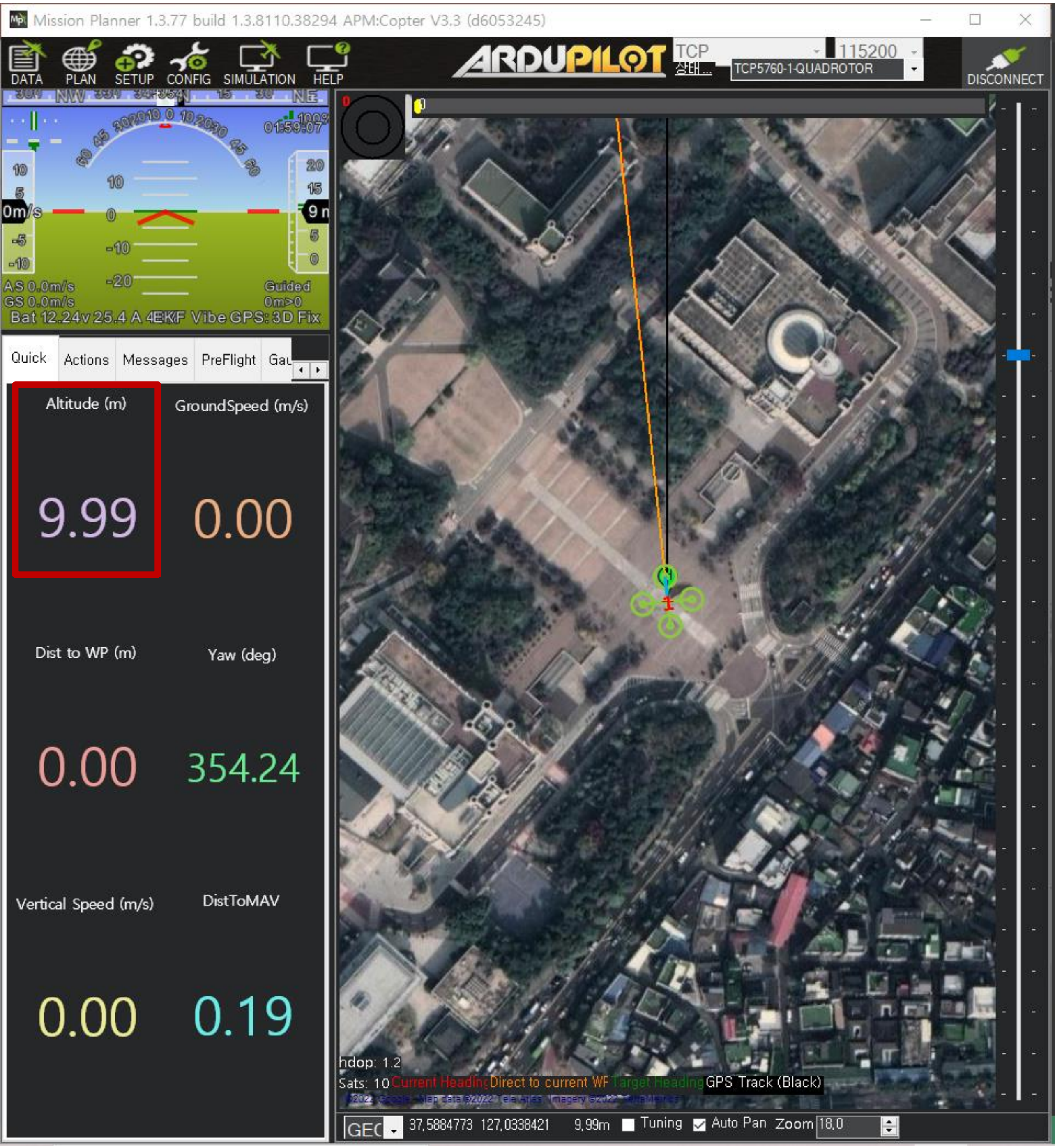


```python
connection_string = 'tcp:127.0.0.1:5763'
vehicle = init_copter(connection_string)

print ("Basic pre-arm checks")
# Don't try to arm until autopilot is ready
while not vehicle.is_armable:
    print ("\tWaiting for vehicle to initialise...")
    time.sleep(1)

print ("Arming motors")
# Copter should arm in GUIDED mode
vehicle.mode = VehicleMode("GUIDED")
vehicle.armed = True

# Confirm vehicle armed before attempting to take off
# EKF is ready and GPS is locked
while vehicle.armed == False or vehicle.mode.name != 'GUIDED':
    print("\tWaiting for arming...")
    vehicle.mode = VehicleMode("GUIDED")
    vehicle.armed = True
    time.sleep(1)

aTargetAltitude = 10
print ("Taking off!")
print("Mode: {}".format(vehicle.mode.name))
vehicle.simple_takeoff(aTargetAltitude) # Take off to target altitude

# Wait until the vehicle reaches a safe height before processing the goto (otherwise the command
#  after Vehicle.simple_takeoff will execute immediately).
while True:
    print ("\tAltitude: ", vehicle.location.global_relative_frame.alt)
    #Break and return from function just below target altitude.
    if vehicle.location.global_relative_frame.alt>=aTargetAltitude*0.95:
        print ("Reached target altitude\n")
        break
    time.sleep(1)
```

Taking off



```
Basic pre-arm checks
Arming motors
        Waiting for arming...
        Waiting for arming...
        Waiting for arming...
ERROR:autopilot:ARMING MOTORS
CRITICAL:autopilot:GROUND START
        Waiting for arming...
CRITICAL:autopilot:Initialising APM...
        Waiting for arming...
ERROR:autopilot:ARMING MOTORS
        Waiting for arming...
        Waiting for arming...
        Waiting for arming...
        Waiting for arming...
        Waiting for arming...
        Waiting for arming...
        Waiting for arming...
        Waiting for arming...
        Waiting for arming...
ERROR:autopilot:DISARMING MOTORS
        Waiting for arming...
ERROR:autopilot:ARMING MOTORS
CRITICAL:autopilot:Initialising APM...
        Waiting for arming...
        Waiting for arming...
        Waiting for arming...
```



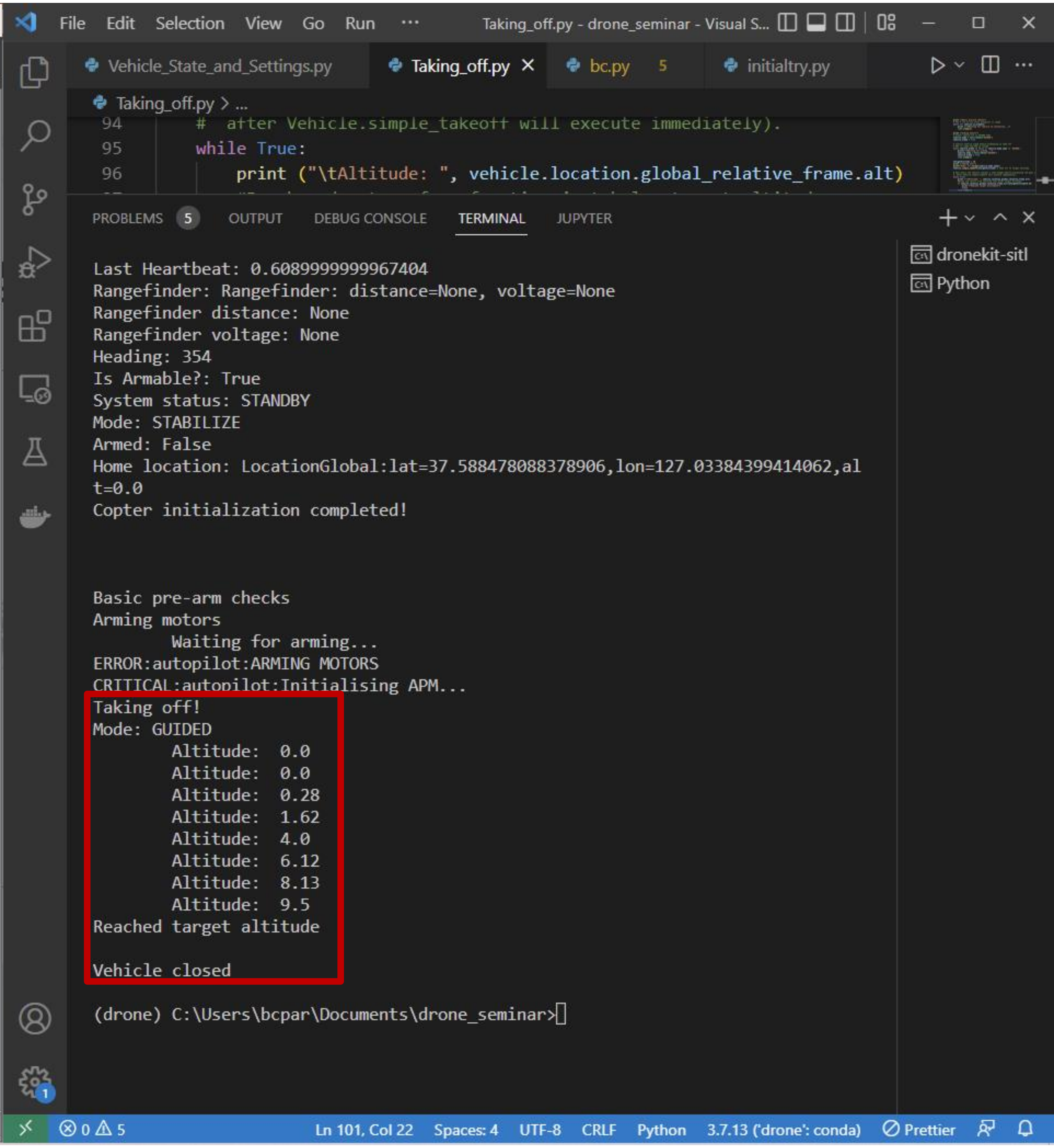
Mission Planner 1.3.77 build 1.3.8110.38294 APM:Copter V3.3 (d6053245)

AS 0.0m/s
GS 0.0m/s
Guided
Bat 12.24v 25.4 A 4EKF Vibe GPS: 3D Fix

Altitude (m)	GroundSpeed (m/s)
9.99	0.00
Dist to WP (m)	Yaw (deg)
0.00	354.24
Vertical Speed (m/s)	DistToMAV
0.00	0.19

hdop: 1.2
Sats: 10

37.5884773 127.0338421 9.99m Tuning ✓ Auto Pan Zoom 18.0



Taking_off.py - drone_seminar - Visual Studio Code

```
    #  after Vehicle.simple_takeoff will execute immediately).
    while True:
        print ("\tAltitude: ", vehicle.location.global_relative_frame.alt)
```

```
Last Heartbeat: 0.6089999999967404
Rangefinder: Rangefinder: distance=None, voltage=None
Rangefinder distance: None
Rangefinder voltage: None
Heading: 354
Is Armable?: True
System status: STANDBY
Mode: STABILIZE
Armed: False
Home location: LocationGlobal:lat=37.588478088378906,lon=127.03384399414062,al
t=0.0
Copter initialization completed!



Basic pre-arm checks
Arming motors
         Waiting for arming...
ERROR:autopilot:ARMING MOTORS
CRITICAL:autopilot:Initialising APM...
Taking off!
Mode: GUIDED
         Altitude:  0.0
         Altitude:  0.0
         Altitude:  0.28
         Altitude:  1.62
         Altitude:  4.0
         Altitude:  6.12
         Altitude:  8.13
         Altitude:  9.5
Reached target altitude

Vehicle closed

(drone) C:\Users\bcpar\Documents\drone_seminar>
```

Ln 101, Col 22 Spaces: 4 UTF-8 CRLF Python 3.7.13 ('drone': conda) Prettier