*class Drone:*

*def \_\_init\_\_(self, model, max\_speed, battery\_capacity):*

*self.model = model*

*self.max\_speed = max\_speed*

*self.battery\_capacity = battery\_capacity*

*self.current\_speed = 0*

*self.battery\_level = 100*

*def take\_off(self):*

*print(f"{self.model} taking off!")*

*self.current\_speed = 10*

*def land(self):*

*print(f"{self.model} landing.")*

*self.current\_speed = 0*

*def increase\_speed(self, speed\_increase):*

*if self.current\_speed + speed\_increase <= self.max\_speed:*

*self.current\_speed += speed\_increase*

*print(f"{self.model} speed increased to {self.current\_speed} m/s.")*

*else:*

*print(f"{self.model} cannot exceed maximum speed of {self.max\_speed} m/s.")*

*def decrease\_speed(self, speed\_decrease):*

*if self.current\_speed - speed\_decrease >= 0:*

*self.current\_speed -= speed\_decrease*

*print(f"{self.model} speed decreased to {self.current\_speed} m/s.")*

*else:*

*print(f"{self.model} cannot have negative speed.")*

*def check\_battery(self):*

*print(f"{self.model} battery level: {self.battery\_level}%.")*

*def fly(self, distance):*

*energy\_consumed = distance \* 0.1 # Assume a simple energy consumption model*

*if self.battery\_level - energy\_consumed >= 0:*

*print(f"{self.model} flying for {distance} meters.")*

*self.battery\_level -= energy\_consumed*

*else:*

*print(f"{self.model} does not have enough battery to fly {distance} meters.")*

***Code snippet for modelling drone as OOP object***

*# Example usage:*

*my\_drone = Drone(model="DJI Phantom", max\_speed=20, battery\_capacity=5000)*

*my\_drone.take\_off()*

*my\_drone.increase\_speed(15)*

*my\_drone.fly(100)*

*my\_drone.land()*

*my\_drone.check\_battery()*

*Code snippet for example usage for drone as OOP object*

*from pymavlink import mavutil*

*import time*

*class DroneInterface:*

*def \_\_init\_\_(self, connection\_string):*

*self.connection\_string = connection\_string*

*self.vehicle = mavutil.mavlink\_connection(connection\_string)*

*def arm\_and\_takeoff(self, altitude):*

*print("Arming motors")*

*self.vehicle.arducopter\_arm()*

*self.vehicle.simple\_takeoff(altitude)*

*def send\_velocity\_command(self, vx, vy, vz):*

*msg = self.vehicle.message\_factory.set\_position\_target\_local\_ned\_encode(*

*0, # time\_boot\_ms*

*0, # target\_system*

*0, # target\_component*

*3, # coordinate\_frame (MAV\_FRAME\_BODY\_NED)*

*0b0000111111000111, # type\_mask (enable all, ignore others)*

*0, 0, 0, # x, y, z positions (not used)*

*vx, vy, vz, # m/s*

*0, 0, 0, # x, y, z acceleration (not used)*

*0, 0) # yaw, yaw\_rate (not used)*

*self.vehicle.send\_mavlink(msg)*

*def land(self):*

*print("Landing")*

*self.vehicle.mode = "LAND"*

*def close\_connection(self):*

*print("Closing connection")*

*self.vehicle.close()*

*# Example usage:*

*if \_\_name\_\_ == "\_\_main\_\_":*

*drone = DroneInterface(connection\_string='/dev/ttyUSB0') # Replace with your connection string*

*try:*

*drone. Arm\_and\_takeoff(10) # Take off to 10 meters altitude*

*time. Sleep(5) # Fly for 5 seconds*

*# Send velocity command to move forward with a speed of 2 m/s*

*drone.send\_velocity\_command(2, 0, 0)*

*time.sleep(5) # Fly forward for 5 seconds*

*drone.land() # Land the drone*

*time.sleep(5) # Allow time for landing*

*finally:*

*drone.close\_connection()*

*Code snippet for creating a simple Drone Interface*

*import time*

*from pymavlink import mavutil*

*# Connection settings*

*connection\_string = 'udp:127.0.0.1:14550' # Example for connecting to a simulator*

*# Create a connection to the vehicle*

*vehicle = mavutil.mavlink\_connection(connection\_string)*

*# Wait for the heartbeat message to find the system ID*

*while not vehicle.wait\_heartbeat():*

*pass*

*print(f"Heartbeat from system {vehicle.target\_system}, component {vehicle.target\_component}")*

*# Request attitude information*

*vehicle.mav.request\_data\_stream\_send(*

*vehicle.target\_system, # target\_system*

*vehicle.target\_component, # target\_component*

*mavutil.mavlink.MAV\_DATA\_STREAM\_ALL, # req\_stream\_id*

*10, # req\_message\_rate*

*1 # start\_stop*

*)*

*try:*

*while True:*

*msg = vehicle.recv\_match(type='ATTITUDE', blocking=True)*

*if msg:*

*print(f"Roll: {msg.roll}, Pitch: {msg.pitch}, Yaw: {msg.yaw}")*

*time.sleep(1)*

*except KeyboardInterrupt:*

*print("Script interrupted by user")*

*vehicle.close()*

*Python implementation of the MAVLink protocol*

*from pymavlink import mavutil*

*# Connect to the drone (replace 'udp:127.0.0.1:14550' with the appropriate connection string)*

*connection\_string = 'udp:127.0.0.1:14550'*

*vehicle = mavutil.mavlink\_connection(connection\_string)*

*# Send a takeoff command*

*msg = vehicle.message\_factory.command\_long\_encode(*

*1, 1, # System and Component IDs*

*mavutil.mavlink.MAV\_CMD\_NAV\_TAKEOFF, # Command*

*0, 0, 0, 0, 0, 0, 0, 0, 0 # Parameters*

*)*

*vehicle.send\_mavlink(msg)*

*MAVLink command for drone take off operation*

*from dronekit import connect, VehicleMode*

*# Connect to the vehicle (update '127.0.0.1:14550' with your vehicle's address)*

*vehicle = connect('127.0.0.1:14550', wait\_ready=True)*

*# Display basic vehicle information*

*print("Connected to vehicle")*

*print("Vehicle Mode: %s" % vehicle.mode.name)*

*print("Global Location: %s" % vehicle.location.global\_frame)*

*# Close the vehicle connection*

*vehicle.close()*

*DroneKit-Python to connect to a vehicle*