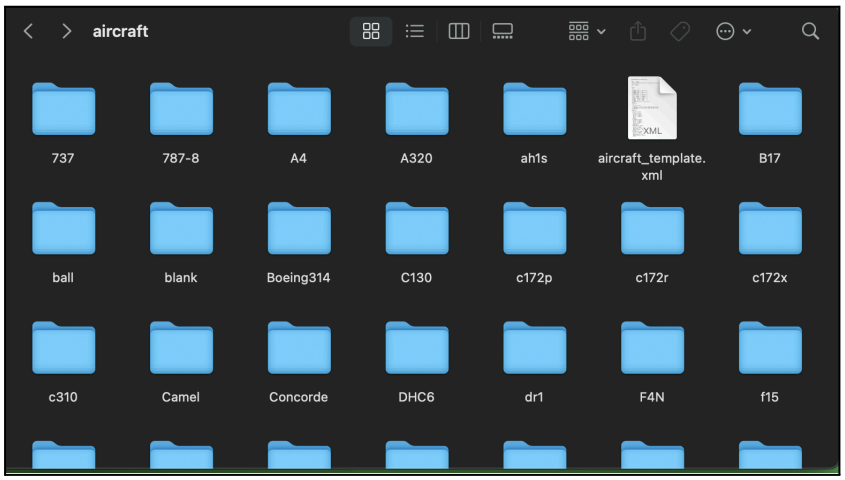
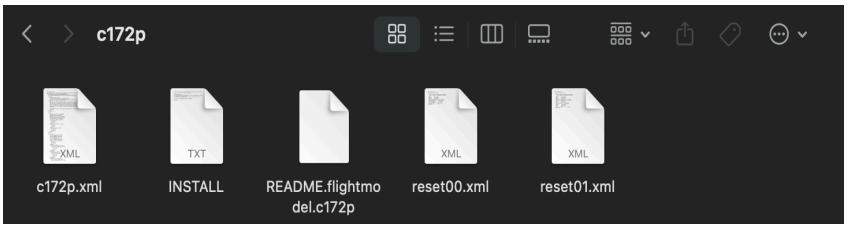
Phase 3 documentation :

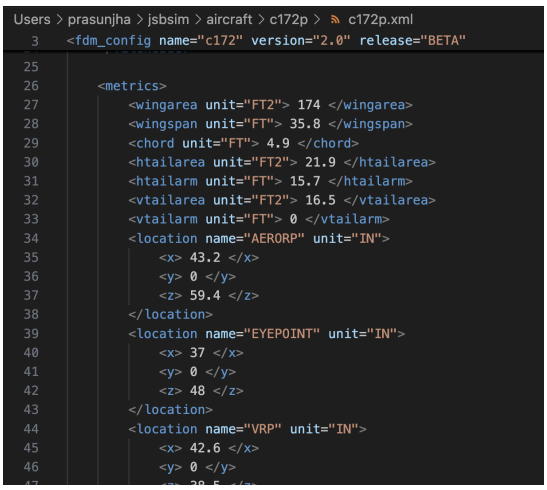
Aim: Understand how an aircraft is defined and controlled in JSBSim using Python.

- Explanation of how JSBSim represents an aircraft:

Inside the jsbsim folder, we have an aircraft folder. This folder contains multiple aircraft. Inside each aircraft folder, let's say c172p, are multiple XML files.. JSBSim represents an aircraft through XML files that define its physical, aerodynamic, and control characteristics. These include files for aerodynamics, engine performance, mass and balance, flight control systems, and landing gear. All simulation data—such as control inputs, position, orientation, and velocities—is managed within a structured property tree, which allows users to set inputs (like throttle or elevator) and retrieve outputs (like altitude or pitch). This architecture allows JSBSim to simulate the real-time response of an aircraft to various control commands and environmental conditions.







Eg of XML files containing the physics and plane dynamics

- Python Script :

```
import jsbsim
import time

# Create a JSBSim instance
sim = jsbsim.FGFDMExec(None)
sim.set_debug_level(0)  # Optional: 0 = no debug, 1 = verbose

# Set the aircraft and initial condition
sim.load_model("c172p")  # Load Cessna 172P
sim.set_property_value("ic/alt-ft", 1000)  # Initial altitude
sim.set_property_value("ic/u-fps", 100)    # Forward speed
sim.set_property_value("ic/psi-deg", 0)    # Heading
sim.set_property_value("ic/theta-deg", 0)  # Pitch
sim.set_property_value("ic/phi-deg", 0)    # Roll
sim.run_ic()  # Initialize the aircraft with the initial conditions

# Set controls
sim.set_property_value("fcs/throttle-cmd-norm", 0.8)  # 80% throttle
sim.set_property_value("fcs/elevator-cmd-norm", 0.0)  # Neutral
elevator
sim.set_property_value("fcs/aileron-cmd-norm", 0.0)   # Neutral
aileron
sim.set_property_value("fcs/rudder-cmd-norm", 0.0)    # Neutral rudder


dt = 0.01
for i in range(1000):
   sim.run()
   altitude = sim.get_property_value("position/h-sl-ft")
   airspeed = sim.get_property_value("velocities/vc-kts")
   pitch = sim.get_property_value("attitude/pitch-deg")
   print(f"Time: {i*dt:.2f}s | Altitude: {altitude:.2f} ft | Airspeed:
{airspeed:.2f} kts | Pitch: {pitch:.2f} deg")
   time.sleep(dt)

print("Simulation complete.")
```

Output:

```
Time: 9.87s | Altitude: 4.43 ft | Airspeed: 51.17 kts | Pitch: 0.00 deg
Time: 9.88s | Altitude: 4.43 ft | Airspeed: 51.16 kts | Pitch: 0.00 deg
Time: 9.89s | Altitude: 4.43 ft | Airspeed: 51.16 kts | Pitch: 0.00 deg
Time: 9.90s | Altitude: 4.43 ft | Airspeed: 51.15 kts | Pitch: 0.00 deg
Time: 9.91s | Altitude: 4.43 ft | Airspeed: 51.14 kts | Pitch: 0.00 deg
Time: 9.92s | Altitude: 4.43 ft | Airspeed: 51.13 kts | Pitch: 0.00 deg
Time: 9.93s | Altitude: 4.43 ft | Airspeed: 51.13 kts | Pitch: 0.00 deg
Time: 9.94s | Altitude: 4.43 ft | Airspeed: 51.12 kts | Pitch: 0.00 deg
Time: 9.95s | Altitude: 4.43 ft | Airspeed: 51.11 kts | Pitch: 0.00 deg
Time: 9.96s | Altitude: 4.43 ft | Airspeed: 51.11 kts | Pitch: 0.00 deg
Time: 9.97s | Altitude: 4.43 ft | Airspeed: 51.10 kts | Pitch: 0.00 deg
Time: 9.98s | Altitude: 4.43 ft | Airspeed: 51.09 kts | Pitch: 0.00 deg
Time: 9.99s | Altitude: 4.43 ft | Airspeed: 51.08 kts | Pitch: 0.00 deg
Simulation complete.
```