```
import math
from ufosim3_2_9q import UfoSim
from ufo_autopilot import distance
from ufo_autopilot import angle_q1
from ufo_autopilot import angle
from ufo_autopilot import flight_distance
from ufo_autopilot import fly_to
from ufo_autopilot import format_flight_data
from ufo_autopilot import takeoff
from ufo_autopilot import cruise
from ufo_autopilot import landing
from ufo_autopilot import fac

# Global variables
test = 0
result = ""

# Function for the test evaluation
def assertEqual(is_value, should_value, msg=""):
    global test
    global result
    test = test + 1
    if is_value != should_value:
        result = result + "\n" + msg + " IS " + str(is_value) + " SHOULD BE " +
str(should_value)

def assertAlmostEqual(is_value, should_value, delta, msg=""):
    global test
    global result
    test = test + 1
    if abs(is_value - should_value) > delta:
        result = result + "\n" + msg + " IS " + str(is_value) + " SHOULD BE " +
str(should_value)

def assertNotEqual(is_value, should_value, msg=""):
    global test
    global result
    test = test + 1
    if is_value == should_value:
        result = result + "\n" + msg + " IS " + str(is_value) + " SHOULD BE
different from " + str(should_value)

def assertNotEqualNoRes(is_value, should_value, msg=""):
    global test
    global result
    test = test + 1
    if is_value != should_value:
        result = result + "\n" + msg

# Tests
assertAlmostEqual(distance(1.0, 1.0, 2.0, 1.0), 1.0, 1e-3, "distance(1.0, 1.0,
2.0, 1.0)")
assertAlmostEqual(distance(1.0, 1.0, 0.0, 2.0), math.sqrt(2.0), 1e-3,
"distance(1.0, 1.0, 0.0, 2.0)")
assertAlmostEqual(angle_q1(0.0, 0.0, 1.0, 0.0), 0.0, 1e-3, "angle_q1(0.0, 0.0,
1.0, 0.0)")
assertAlmostEqual(angle_q1(0.5, 1.5, 3.5, 2.5), math.atan(1/3) / math.pi *
180.0, 1e-3, "angle_q1(0.5, 1.5, 3.5, 2.5)")
assertAlmostEqual(angle_q1(1.0, 1.0, 2.0, 2.0), 45.0, 1e-3, "angle_q1(1.0, 1.0,
2.0, 2.0)")
assertAlmostEqual(angle_q1(1e-4, 0.0, 1.0, 1.0), 45.002, 1e-3, "angle_q1(1e-4,
0.0, 1.0, 1.0)")
```

```
assertAlmostEqual(angle_q1(0.5, 1.5, 1.5, 4.5), math.atan(3/1) / math.pi *
180.0, 1e-3, "angle_q1(0.5, 1.5, 1.5, 4.5)")
assertAlmostEqual(angle_q1(1.0, 1.0, 1.0, 2.0), 90.0, 1e-3, "angle_q1(1.0, 1.0,
1.0, 2.0)")
assertNotEqual(angle_q1(1.0, 1.0, 1.0, 1.0), None, "angle_q1(1.0, 1.0, 1.0,
1.0)")
assertAlmostEqual(angle(0.0, 0.0, 1.0, 0.0), 0.0, 1e-3, "angle(0.0, 0.0, 1.0,
0.0)")
assertAlmostEqual(angle(0.5, 1.5, 3.5, 2.5), math.atan(1/3) / math.pi * 180.0,
1e-3, "angle(0.5, 1.5, 3.5, 2.5)")
assertAlmostEqual(angle(1.0, 1.0, 2.0, 2.0), 45.0, 1e-3, "angle(1.0, 1.0, 2.0,
2.0)")
assertAlmostEqual(angle(0.5, 1.5, 1.5, 4.5), math.atan(3/1) / math.pi * 180.0,
1e-3, "angle(0.5, 1.5, 1.5, 4.5)")
assertAlmostEqual(angle(1.0, 1.0, 1.0, 2.0), 90.0, 1e-3, "angle(1.0, 1.0, 1.0,
2.0)")
assertAlmostEqual(angle(1.0, 1.0, 0.0, 2.0), 135.0, 1e-3, "angle(1.0, 1.0, 0.0,
2.0)")
assertAlmostEqual(angle(1.0, 1.0, -2.0, 1.0), 180.0, 1e-3, "angle(1.0, 1.0,
-2.0, 1.0)")
assertAlmostEqual(angle(0.0, 1.0, -3.0, -1.0), 213.690, 1e-3, "angle(0.0, 1.0,
-3.0, -1.0)")
assertAlmostEqual(angle(1.0, 1.0, 2.0, 0.0), 315.0, 1e-3, "angle(1.0, 1.0, 2.0,
0.0)")
assertAlmostEqual(angle(0.0, 0.0, 30.0, -10.0), 341.565, 1e-3, "angle(0.0, 0.0,
30.0, -10.0)")
assertAlmostEqual(angle(55.0, 20.0, -115.0, 95.0), 156.194, 1e-3, "angle(55.0,
20.0, -115.0, 95.0)")
assertAlmostEqual(angle(1e-4, 0.0, 1.0, 1.0), 45.002, 1e-3, "angle(1e-4, 0.0,
1.0, 1.0)")
assertNotEqual(angle(-10.0, 1e-15, 0.0, 0.0), 360.0, "angle(-10.0, 1e-15, 0.0,
0.0)")
assertNotEqual(angle(1.0, 1.0, 1.0, 1.0), None, "angle(1.0, 1.0, 1.0, 1.0)")
assertAlmostEqual(flight_distance(1.0, 1.0, 2.0, 1.0, 10.0), 21.0, 1e-3,
"flight_distance(1.0, 1.0, 2.0, 1.0, 10.0)")
assertAlmostEqual(flight_distance(1.0, 1.0, 0.0, 2.0, 10.0),
math.sqrt(2.0)+20.0, 1e-3, "flight_distance(1.0, 1.0, 0.0, 2.0, 10.0)")

sim = UfoSim()
format_flight_data(sim)

sim = UfoSim()
sim.start(10, 0, [])
fly_to(sim, 20.0, 20.0, 10.0)
sim.terminate()
assertAlmostEqual(sim.get_x(), 20.0, 1.0, "after fly_to(sim, 20.0, 20.0, 10.0):
sim.get_x()")
assertAlmostEqual(sim.get_y(), 20.0, 1.0, "after fly_to(sim, 20.0, 20.0, 10.0):
sim.get_y()")
assertAlmostEqual(sim.get_z(), 0.0, 0.1, "after fly_to(sim, 20.0, 20.0, 10.0):
sim.get_z()")

sim = UfoSim()
sim.start(10, 0, [])
takeoff(sim, 10.0)
cruise(sim, 20.0, 20.0)
landing(sim)
sim.terminate()
assertAlmostEqual(sim.get_x(), 20.0, 1.0, "after takeoff(sim, 10.0), cruise(sim,
20.0, 20.0), landing(sim): sim.get_x()")
assertAlmostEqual(sim.get_y(), 20.0, 1.0, "after takeoff(sim, 10.0), cruise(sim,
20.0, 20.0), landing(sim): sim.get_y()")
```

```
assertAlmostEqual(sim.get_z(), 0.0, 0.1, "after takeoff(sim, 10.0), cruise(sim,
20.0, 20.0), landing(sim): sim.get_z()")

sim = UfoSim()
sim.start(10, 0, [])
fly_to(sim, 0.0, 0.0, 10.0)
sim.terminate()

sim = UfoSim()
sim.start(10, 0, [])
fly_to(sim, 10.0, 0.0, 10.0)
sim.terminate()
assertAlmostEqual(sim.get_x(), 10.0, 1.0, "after fly_to(sim, 10.0, 0.0, 10.0):
sim.get_x()")
assertAlmostEqual(sim.get_y(), 0.0, 1.0, "after fly_to(sim, 10.0, 0.0, 10.0):
sim.get_y()")
assertAlmostEqual(sim.get_z(), 0.0, 0.1, "after fly_to(sim, 10.0, 0.0, 10.0):
sim.get_z()")

assertEqual(fac(5,3), 60, "fac(5,3)")
assertEqual(fac(4,3), 12, "fac(4,3)")
assertEqual(fac(3,3), 3, "fac(3,3)")
assertEqual(fac(2,3), 1, "fac(2,3)")
assertEqual(fac(2,5), 1, "fac(2,5)")
assertEqual(fac(4), 24, "fac(4)")
assertEqual(fac(), 1, "fac()")
assertEqual(fac(3,-2), 0, "fac(3,-2)")
assertEqual(fac(-2,-4), -24, "fac(-2,-4)")

# Print the test results
print()
print(test, "tests executed. ", end="")
if result == "":
    print("Passed.")
else:
    print("Failed:", end="")
    print(result)
```