PYTHON

```
from dronekit import connect, VehicleMode, LocationGlobalRelative
import time
import math

# Connect to the Vehicle (replace with your connection string)
connection_string = '127.0.0.1:14550'  # For SITL or telemetry port
vehicle = connect(connection_string, wait_ready=True)

def arm_and_takeoff(target_altitude):
    print("Arming motors")
    while not vehicle.is_armable:
        print(" Waiting for vehicle to initialise...")
        time.sleep(1)
    vehicle.mode = VehicleMode("GUIDED")
    vehicle.armed = True

    while not vehicle.armed:
        print(" Waiting for arming...")
        time.sleep(1)

    print("Taking off!")
    vehicle.simple_takeoff(target_altitude)

    # Wait until the vehicle reaches a safe height
    while True:
        print(f" Altitude: {vehicle.location.global_relative_frame.alt:.1f}m")
        if vehicle.location.global_relative_frame.alt >= target_altitude * 0.95:
            print("Reached target altitude")
            break
        time.sleep(1)
```

```
def send_ned_velocity(velocity_x, velocity_y, velocity_z, duration):
    """
    Move vehicle in direction based on specified velocity vectors.
    velocity_x: velocity in m/s (north +ve, south -ve)
    velocity_y: velocity in m/s (east +ve, west -ve)
    velocity_z: velocity in m/s (down +ve, up -ve)
    duration: time to send this velocity command in seconds
    """
    msg = vehicle.message_factory.set_position_target_local_ned_encode(
        0,      # time_boot_ms (not used)
        0, 0,    # target system, target component
        0b0000111111000111,  # type_mask (only velocities enabled)
        0, 0, 0,            # x, y, z positions (not used)
        velocity_x, velocity_y, velocity_z,  # velocities in m/s
        0, 0, 0,            # accelerations (not supported)
        0, 0)               # yaw, yaw_rate (not supported)
    for _ in range(duration):
        vehicle.send_mavlink(msg)
        time.sleep(1)


def condition_yaw(heading, relative=False):
    """
    Rotate drone to a specific heading (in degrees).
    heading: 0-360 degrees
    relative: True for relative rotation, False for absolute heading
    """
    is_relative = 1 if relative else 0
    msg = vehicle.message_factory.command_long_encode(
        0, 0,    # target system, target component
        115,     # command MAV_CMD_CONDITION_YAW
```

```
        0,      # confirmation
        heading, # param 1: target angle
        0,      # param 2: speed deg/s (0 = default)
        1,      # param 3: direction -1 ccw, 1 cw
        is_relative, # param 4: relative offset 1, absolute 0
        0, 0, 0) # param 5-7 not used
    vehicle.send_mavlink(msg)

# Example usage:
try:
    arm_and_takeoff(10)  # Take off to 10 meters altitude

    print("Moving forward")
    send_ned_velocity(5, 0, 0, 3)  # Move north at 5 m/s for 3 seconds

    print("Moving right")
    send_ned_velocity(0, 5, 0, 3)  # Move east at 5 m/s for 3 seconds

    print("Moving backward")
    send_ned_velocity(-5, 0, 0, 3)  # Move south at 5 m/s for 3 seconds

    print("Moving left")
    send_ned_velocity(0, -5, 0, 3)  # Move west at 5 m/s for 3 seconds

    print("Rotating clockwise 90 degrees")
    condition_yaw(90, relative=True)
    time.sleep(5)

    print("Landing")
    vehicle.mode = VehicleMode("LAND")
```

```
    # Close vehicle object before exiting script
    vehicle.close()

except Exception as e:
    print(f"An error occurred: {e}")
    vehicle.close()
```

This code covers basic autonomous commands: takeoff, move in cardinal directions, rotate, and land. You can extend it with more complex logic, obstacle avoidance, and payload management.