Introduction to Robotics Using Python

Learning Objectives

- Understand robot motion, sensors, and control.
- Simulate robot movement in 2D.
- Write Python code to simulate robot navigation.

Prerequisites

- Basic Python
- Install: pip install matplotlib numpy pygame

Lesson 1: Robot Kinematics

Concepts:

- Differential drive
- Coordinate systems

Code Example:

Use matplotlib to plot robot path using forward and angular velocity.

```
import matplotlib.pyplot as plt
import numpy as np

x, y, theta = 0, 0, 0
v = 1.0
w = np.pi / 8
dt = 0.1
x_path, y_path = [x], [y]

for _ in range(100):
    x += v * np.cos(theta) * dt
    y += v * np.sin(theta) * dt
    theta += w * dt
    x_path.append(x)
    y_path.append(y)

plt.plot(x_path, y_path, label="Robot Path")
plt.xlabel("X position (m)")
plt.ylabel("Y position (m)")
plt.title("Simulated Robot Motion")
plt.axis("equal")
plt.grid(True)
plt.legend()
```

```
plt.show()
```

Lesson 2: Real-Time Robot in 2D (Pygame)

Use arrow keys to move a robot on screen using differential drive logic.

```
import pygame
import math

pygame.init()
screen = pygame.display.set_mode((600, 600))
pygame.display.set_caption("Differential Drive Robot")
clock = pygame.time.Clock()

x, y = 300, 300
angle = 0
speed = 0
angular_speed = 0

running = True
while running:
    screen.fill((255, 255, 255))
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    keys = pygame.key.get_pressed()
    if keys[pygame.K_UP]:
        speed = 2
    elif keys[pygame.K_DOWN]:
        speed = -2
    else:
        speed = 0

    if keys[pygame.K_LEFT]:
        angular_speed = -0.05
    elif keys[pygame.K_RIGHT]:
        angular_speed = 0.05
    else:
        angular_speed = 0

    angle += angular_speed
    x += speed * math.cos(angle)
    y += speed * math.sin(angle)

    pygame.draw.circle(screen, (0, 0, 255), (int(x), int(y)), 10)
    pygame.draw.line(screen, (255, 0, 0), (x, y),
                     (x + 20 * math.cos(angle), y + 20 * math.sin(angle)), 2)
```

```
    pygame.display.flip()
    clock.tick(30)

pygame.quit()
```

Lesson 3: Adding Obstacles

Use pygame.Rect to simulate obstacles.

Check for collisions with robot.

```
# Add inside the Pygame loop
obstacle = pygame.Rect(200, 200, 50, 50)
pygame.draw.rect(screen, (0, 0, 0), obstacle)

robot_rect = pygame.Rect(x - 10, y - 10, 20, 20)
if robot_rect.colliderect(obstacle):
    print("Collision Detected!")
    speed = 0
```

Lesson 4: Line Following Robot

Simulate sensor by detecting black pixels under robot head.

Change angular velocity based on detection.

```
# Add inside Pygame loop
pygame.draw.line(screen, (0, 0, 0), (0, 300), (600, 300), 5)

sensor_x = int(x + 10 * math.cos(angle))
sensor_y = int(y + 10 * math.sin(angle))

try:
    color = screen.get_at((sensor_x, sensor_y))
    if color == (0, 0, 0, 255):
        speed = 2
    else:
        angular_speed = 0.05
except:
    pass
```

Extensions

- Simulate multiple robots

- Add ray-cast sensor

- Use PID control for smoother behavior.

Introduction to Robotics Using Python

Conclusion

This tutorial introduces key robotics concepts through simulation and interactive control using Python.