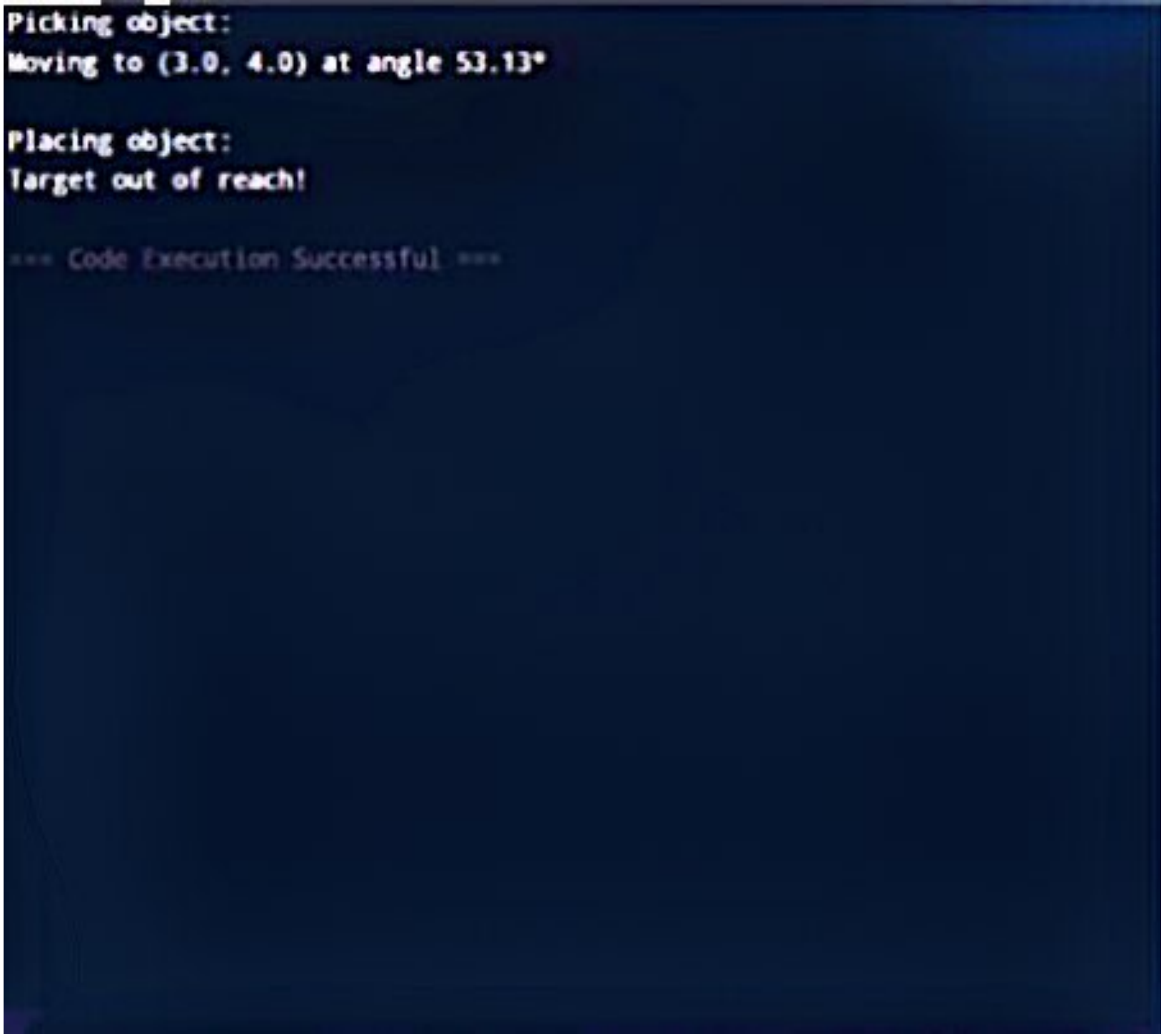
```python
import math
arm_length = 5
base_position = (0, 0)
object_position = (3, 4)
place_position = (6, 0)
def reach_target(target):
    dx = target[0] - base_position[0]
    dy = target[1] - base_position[1]
    distance = math.hypot(dx, dy)

    if distance > arm_length:
        return "Target out of reach!"

    angle = math.atan2(dy, dx)
    end_effector_x = base_position[0] + arm_length * math.cos(angle)
    end_effector_y = base_position[1] + arm_length * math.sin(angle)

    return f"Moving to ({round(end_effector_x, 2)}, {round
        (end_effector_y, 2)}) at angle {round(math.degrees(angle), 2
        )}°"

print("Picking object:")
print(reach_target(object_position))

print("\nPlacing object:")
print(reach target(place position))
```



main.py

Share Run

```python
import numpy as np
import random

# Mock sensor fusion
def fuse_sensors(lidar_data, radar_data, camera_data):
    return np.mean([lidar_data, radar_data, camera_data], axis=0)

# Simulated decision-making
def ai_decision(environment_data):
    if environment_data['pedestrian_detected']:
        return "Stop"
    elif environment_data['obstacle_distance'] < 5:
        return "Brake"
    elif environment_data['lane_clear']:
        return "Proceed"
    return "Idle"

# Mock environment
def simulate_environment():
    return {
        'pedestrian_detected': random.choice([True, False]),
        'obstacle_distance': random.uniform(0, 20),
        'lane_clear': random.choice([True, False])
    }

# Run simulation
```

Output

Clear

```
Sensor Data: {'pedestrian_detected': True, 'obstacle_distance': 17
    .100130232783638, 'lane_clear': True} => Decision: Stop
Sensor Data: {'pedestrian_detected': True, 'obstacle_distance': 14
    .334958232461403, 'lane_clear': False} => Decision: Stop
Sensor Data: {'pedestrian_detected': True, 'obstacle_distance': 13
    .66871634171569, 'lane_clear': True} => Decision: Stop
Sensor Data: {'pedestrian_detected': False, 'obstacle_distance': 5
    .764521778717753, 'lane_clear': False} => Decision: Idle
Sensor Data: {'pedestrian_detected': False, 'obstacle_distance': 11
    .217966272914719, 'lane_clear': True} => Decision: Proceed
Sensor Data: {'pedestrian_detected': False, 'obstacle_distance': 1
    .371133604226027, 'lane_clear': True} => Decision: Brake
Sensor Data: {'pedestrian_detected': True, 'obstacle_distance': 8
    .584929792628515, 'lane_clear': True} => Decision: Stop
Sensor Data: {'pedestrian_detected': True, 'obstacle_distance': 18
    .16610525918679, 'lane_clear': True} => Decision: Stop
Sensor Data: {'pedestrian_detected': False, 'obstacle_distance': 5
    .964051010569797, 'lane_clear': True} => Decision: Proceed
Sensor Data: {'pedestrian_detected': True, 'obstacle_distance': 15
    .640089546639977, 'lane_clear': False} => Decision: Stop

=== Code Execution Successful ===
```

main.py

Share Run

```python
import random

# Simulate obstacle detection and response
obstacle = random.choice([True, False])
if obstacle:
    print("Obstacle detected! Changing path...")
else:
    print("Path is clear. Moving forward...")

# Simulate robotic task
print("Picking object...")
print("Placing object... Done.")

# Simulate sensor data
print("Sensor: Distance =", random.randint(10, 100), "cm")
```

Output

Clear

```
Path is clear. Moving forward...
Picking object...
Placing object... Done.
Sensor: Distance = 52 cm

=== Code Execution Successful ===
```