## Robotics and Python. Obstacle avoidance and path finding.

Sila, Eaaa. DMU. November 2024.

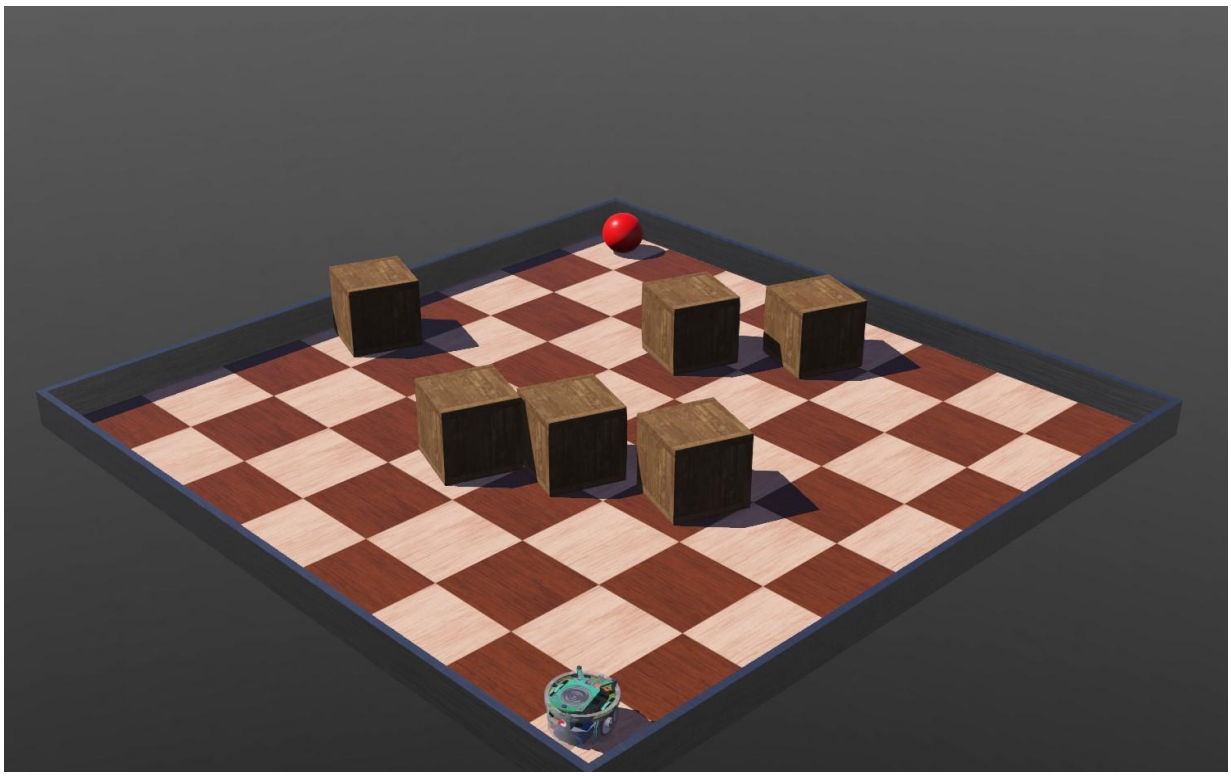
### Exercise 0.

To add a compass to an e-puck robot in Webots, follow these steps:

1. Open your Webots world containing the e-puck robot.
2. In the scene tree, locate and expand the e-puck node.
3. Find the "turret slot" under the e-puck node.
4. Right-click on the "turret slot" and select "Add new node."
5. In the node selection window, search for "Compass" and select it.
6. Click "Add" to add the Compass node to your e-puck robot.
7. Save your Webots world.

But don't worry. Both Compass and GPS have already been added to the e-puck robot.

In this exercise you must make changes to the control script, so that the e-puck robot gets to the target, the red ball.



In all of the exercises below the robot start in one corner and must move to the opposite corner.

## Exercise 1.

Remove all of the boxes.
Make changes to the control script, so that the robot goes to the red ball as fast as possible.

I.e. Start with a template algorithm like this.

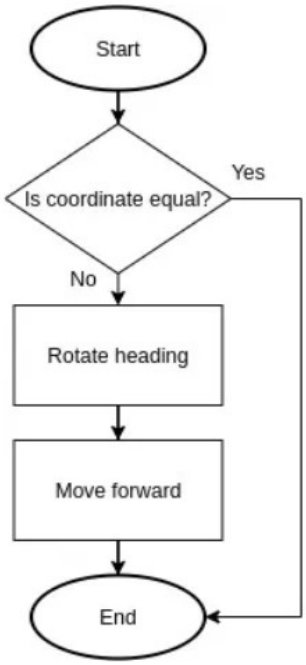
1. Drive distance $d$
2. Rotate by angle $\alpha$
3. Go to step 1

Where you fill in the precise python code that makes it work in Webot.
Extra: You might want to turn on some LED light, or similar, on the robot to indicate that it has completed the task

## Exercise 2.

Leave three boxes on the board that block the direct route from the starting position to the corner with the red ball. But remove the other three boxes.
Make changes to the control script, so that the robot goes to the red ball as fast as possible.

Again it is your job to program a simple Python script that makes this possible in the Webot simulator.



Algorithm

## Exercise 3.

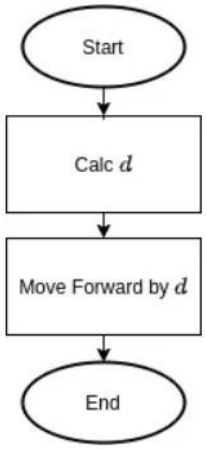
Now, all six boxes must remain on the board…
Make changes to the control script, so that the robot goes to the red ball as fast as possible.

During the testing phase of this, you can remove some of the boxes. But for your final solution the setup must be as indicated in the picture above.

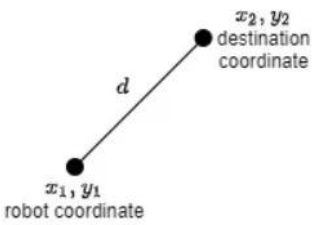
Now it might be necessary for the robot to have map of where the boxes (the obstacles) are located? Or the robot might have different goal states baseed on where it is. Probably it also need to have some random behaviours to get out of trouble (being stuck somewhere).

### Move Forward by Specific Distance

For the detail about how to move forward, below is the algorithm. Simply calculate the distance `d` from the robot coordinate to destination coordinate. Then move forward by d.




Distance `d` is simply calculated by using euclidean distance calculation.




$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

euclidean distance

ps.
Feel three to ask ChatGPT for advice. But remember that you must understand all of the code that

you get from ChatGPT. So, if parts of code from ChatGPT is difficult to understand, you must ask chatGPT questions until you understand the code. You can't use code from ChatGPT in your program that you do not understand.

### Exercise 4. Optional.

Now try to move the boxes around … Or add a few more boxes. There should still be a path to the target though.

- Can your robot still move to the target?
- Make changes to your script that makes it possible for the robot to still find the target.