## Question 1: Functions

### 1. Line b/w two points

```
def line_bw_points(A, B):
    """
    Parameters:
    ----------
    A: first point, tuple
    B: second point, tuple
    Returns:
    --------
    tuple: (x coefficient, y coeffiecient, slop of line)
    """
    x_coff = B[1] - A[1]
    y_coff = A[0] - B[0]
    c = x_coff*A[0] + y_coff*A[1]
    # print(f"the line b/w {A} and {B} is {x_coff}x + {y_coff}y = {c}")
    return (x_coff, y_coff, c)
```

### 2. Distance b/w two points

```
import numpy as np
def distance_bw_points(A, B):
    """
    Arguments:
    ----------
    A: first point, tuple
    B: second point, tuple
    Returns:
    --------
    float: distance
    """
    A = np.array(A)
    B = np.array(B)

    dist = np.linalg.norm(A - B)
    return dist
```

### 3. Perpendicular distance b/w a point and a line segment

```
#| column: margin
#| message: false
def perpendicular_dis(A, line):
    """
    Arguments:
    ----------
    A: a point, tuple
    line: points of lines, tuple
    Returns:
    --------
    dis: float
    """
    return abs((line[0] * A[0] + line[1] * A[1] + line[2])) /
          np.sqrt(np.square(line[0]) + np.square(line[1]))
```

### 4. Distance b/w a point and a polygon

```
from sympy import Polygon, Point
def distance_from_polygon(A, poly_ver):
    """
    Arguments:
    ----------
    A: a point
    poly: vertices of polygon, list of tuples
    Returns:
    --------
    float
    """

    poly = Polygon(*[Point(i) for i in poly_ver])

    return poly.distance(Point(A[0], A[1]))
```

### 5. Tangent vector to a polygon

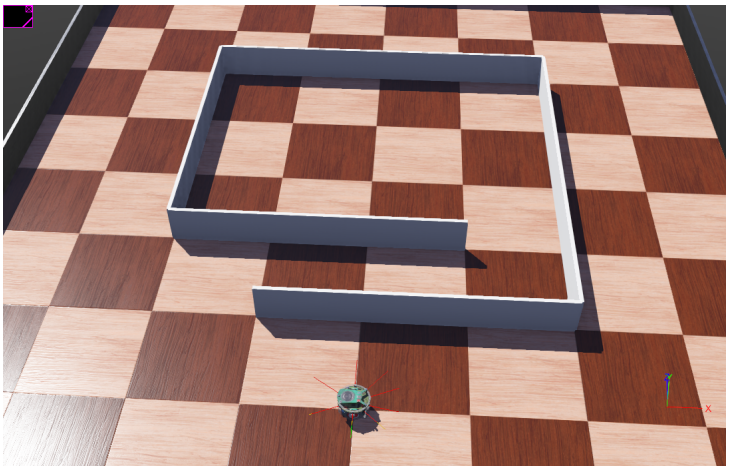
### 6. Intersection of two polygons

```
def intersection_of_polygons(poly_1, poly_2):
    """
    Argunments:
    ----------
    ploy_1: vertices of polygon 1, list of tuples
    ploy_2: vertices of polygon 2, list of tuples
    Returns:
    --------
    list of points of intersection
    """
    poly_1 = Polygon(*[Point(i) for i in poly_1])
    poly_2 = Polygon(*[Point(i) for i in poly_2])

    return poly_1.intersection(poly_2)
```

## Question 2: Incomplete Bug0 :(

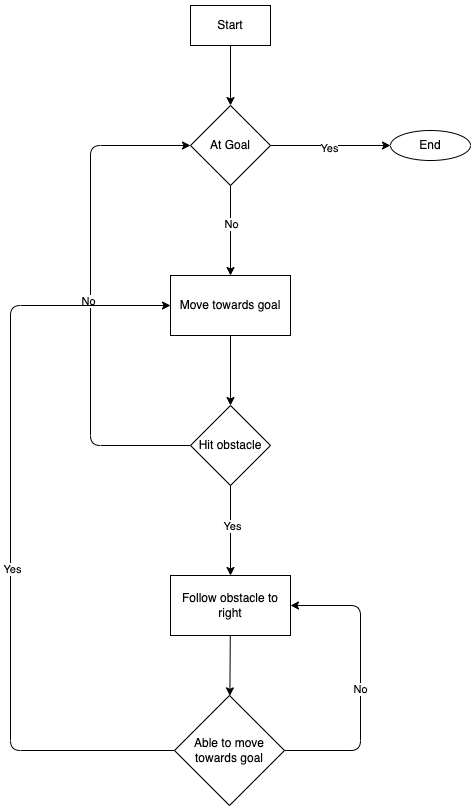
Enviornment:



Code: Link

video: Link

## Question 3:




## Question 4: Incompletness of Bug0

An algorithm is complete if in finite time,

1. It finds a solution, if it exits
2. It terminates with no solution, of no solution exits.

For finite number of obstacles bug0 shouls find a path if not means it stuck in a infinite loop. Since number of obstacles are finite resulting in infinite loop implies the robot must hit same obstacle more than once. which is true for a enviornment used in question 1.