The objective of this section to make the code more robust to more robot and less manual config.

Suggestion

First, we remove the default configuration.

First, we going to introduce about numpy random,

# First task:

We also need to import numpy as np

And to make things consistent for all, we set the

np.random.seed(0)

next, we can start

Replace the code

def draw(self,canvas):  
  
 points = [ (self.x + 30\*math.sin(self.theta)) - 30\*math.sin((math.pi/2.0)-self.theta), \  
 (self.y - 30\*math.cos(self.theta)) - 30\*math.cos((math.pi/2.0)-self.theta), \  
 (self.x - 30\*math.sin(self.theta)) - 30\*math.sin((math.pi/2.0)-self.theta), \  
 (self.y + 30\*math.cos(self.theta)) - 30\*math.cos((math.pi/2.0)-self.theta), \  
 (self.x - 30\*math.sin(self.theta)) + 30\*math.sin((math.pi/2.0)-self.theta), \  
 (self.y + 30\*math.cos(self.theta)) + 30\*math.cos((math.pi/2.0)-self.theta), \  
 (self.x + 30\*math.sin(self.theta)) + 30\*math.sin((math.pi/2.0)-self.theta), \  
 (self.y - 30\*math.cos(self.theta)) + 30\*math.cos((math.pi/2.0)-self.theta) \  
 ]  
 canvas.create\_polygon(points, fill="blue", tags=self.name)  
  
 self.sensorPositions = [ (self.x + 20\*math.sin(self.theta)) + 30\*math.sin((math.pi/2.0)-self.theta), \  
 (self.y - 20\*math.cos(self.theta)) + 30\*math.cos((math.pi/2.0)-self.theta), \  
 (self.x - 20\*math.sin(self.theta)) + 30\*math.sin((math.pi/2.0)-self.theta), \  
 (self.y + 20\*math.cos(self.theta)) + 30\*math.cos((math.pi/2.0)-self.theta) \  
 ]  
  
 centre1PosX = self.x  
 centre1PosY = self.y  
 canvas.create\_oval(centre1PosX-15,centre1PosY-15, \  
 centre1PosX+15,centre1PosY+15, \  
 fill="gold",tags=self.name)  
 batteryText = canvas.create\_text(self.x,self.y,text=str(self.battery),tags=self.name)  
  
 wheel1PosX = self.x - 30\*math.sin(self.theta)  
 wheel1PosY = self.y + 30\*math.cos(self.theta)  
 canvas.create\_oval(wheel1PosX-3,wheel1PosY-3, \  
 wheel1PosX+3,wheel1PosY+3, \  
 fill="red",tags=self.name)  
  
 wheel2PosX = self.x + 30\*math.sin(self.theta)  
 wheel2PosY = self.y - 30\*math.cos(self.theta)  
 canvas.create\_oval(wheel2PosX-3,wheel2PosY-3, \  
 wheel2PosX+3,wheel2PosY+3, \  
 fill="green",tags=self.name)  
  
 sensor1PosX = self.sensorPositions[0]  
 sensor1PosY = self.sensorPositions[1]  
 sensor2PosX = self.sensorPositions[2]  
 sensor2PosY = self.sensorPositions[3]  
 canvas.create\_oval(sensor1PosX-3,sensor1PosY-3, \  
 sensor1PosX+3,sensor1PosY+3, \  
 fill="yellow",tags=self.name)  
 canvas.create\_oval(sensor2PosX-3,sensor2PosY-3, \  
 sensor2PosX+3,sensor2PosY+3, \  
 fill="yellow",tags=self.name)

# Task 2

Modify the register, by removing the needs fors

In the positional of register,

We remove

def register(canvas,robot\_configurations,colors):

to

def register(canvas):

Remember, at this stage, we aim to make thing can be expanded for number of robot, and we want to avoid manually assign the robot configuration.

Similarly, the changes we make to the method draw, so, we should modify the function caller.

From

for i, (config,clrx) in enumerate(zip(robot\_configurations,colors)):  
 robot\_name='c'  
 bot=Bot(robot\_name)  
 bot.draw(i, canvas, config, condition, colors, my\_th)

to

for i in range(0,noOfBots):  
 bot = Bot("Bot"+str(i))  
 bot.draw(canvas)

if you notice, no we can simply define the noOfBots. In fact, this is something similar to how we use to generate the number of dirts.

If you manage to made the changes, u should get the following graph:  
  
A screenshot of a computer

Description automatically generated

# Task 3

Let make it move, by changing something