Programação Orientada Objeto

https://dontpad.com/catolicaoop2025

https://meet.google.com/pxg-ozsa-qex?pli=1

Hoje dia 26 de agosto 2025, na aula de hoje foi deixado anteriormente 2 desafios para ser feito, o primeiro desafio foi de “animar” fazer uma animação do elevado passando pelos andares por onde ele passar, sem que eu dei o comando de ele estar no 1 andar ir direto para o andar 10, ele ter que passar do andares entre 1a10 antes de chegar no andar desejado, bom o segundo desafio foi de fazer a abstração de um modelo de robô doméstico que faz limpeza, e fazer também as funções que esse robô teria.

Assim que a aula começou foi nos apresentado o problema que foi passado e entender o'que será feito, apresentou também a biblioteca do python para simplificar matrix.

foi passado um novo desafio com base no código feita na aula juntamente com o professor, fazer um algoritmo para o robô “limpar” todo o ambiente em que ele se encontra

CODIGO:

```
import time

class Environment:
    #inicializador da classe
    def __init__(self, qtdLin: int, qtdCol: int):
        self.qtdLines = qtdLin
        self.qtdCols = qtdCol
        self.matrix = [[1 for _ in range(qtdCol)]for _ in
range(qtdLin)]

    #Permite alterar o valor de uma posicao (linha, coluna) da matriz
    def setPosition(self, lin: int, col: int, value: int):
        self.matrix[lin][col] = value

    #Permite reiniciar toda a matriz com um determinado valor
    def restartWith(self, value: int):
        for line in range(self.qtdLines):
            for colun in range(self.qtdCols):
```

```
                self.matrix[line][colun] = value

    #redefine o comportamento utilizdo para imprimir os dados do objeto
no print
    def __str__(self):
        result = ""
        for line in self.matrix:
            for value in line:
                if (value == 0):
                    result += "  "
                elif (value == 1):
                    result += "# "
                else:
                    result += "O "
            result += "\n"
        return result

    #Testa se uma posicao é ou nao valida na matriz
    def isValidPosition(self, line: int, colun: int):
        return line >= 0 and line < self.qtdLines and colun >= 0 and
colun < self.qtdCols

    #Verifica se existe sujeira no ambiente
    def isAllClean(self):
        for line in range(self.qtdLines):
            for colun in range(self.qtdCols):
                if self.matrix[line][colun] == 1:
                    return False
        return True

class CleanBot:
    def __init__(self, line: int, col: int, environment: Environment,
charge: int = 0,):
        self.lin = line
        self.col =  col
        self.charge = charge
        self.environment = environment

    def startBot(self):
        count = 1
        while (not self.environment.isAllClean()):
            print(f"Round {count}")
            print(environment)
```

```
            time.sleep(2)
            count += 1
        print("O ambiente esta totalmente limpo")



#Testes com o objeto
environment = Environment(qtdLin = 20, qtdCol = 20)
environment.restartWith(value = 1)

bot = CleanBot(line = 3, col = 3, environment = environment)
bot.startBot()
```