VACUUM CLEANER AGENT

    # initializing goal\_state

    # 0 indicates Clean and 1 indicates Dirty

    goal\_state = {'A': '0', 'B': '0'}

    cost = 0

    location\_input = input("Enter Location of Vacuum: ")  # user input of location vacuum is placed

    status\_input = input("Enter status of " + location\_input+":" )  # user input if location is dirty or clean

    status\_input\_complement = input("Enter status of other room: ")

    print("Initial Location Condition" + str(goal\_state))

    if location\_input == 'A':  # Location A is Dirty.

        print("Vacuum is placed in Location A.")

        if status\_input == '1':

            print("Location A is Dirty.")

            # suck the dirt and mark it as clean

            goal\_state['A'] = '0'

            cost += 1  # cost for suck

            print("Cost for CLEANING A: " + str(cost))

            print("Location A has been Cleaned.")

        if status\_input\_complement == '1':  # if B is Dirty

            print("Location B is Dirty.")

            print("Moving right to the Location B. ")

            cost += 1  # cost for moving right

            print("COST for moving RIGHT: " + str(cost))

            # suck the dirt and mark it as clean

            goal\_state['B'] = '0'

            cost += 1  # cost for suck

            print("COST for SUCK: " + str(cost))

            print("Location B has been Cleaned. ")

        else:

            print("No action" + str(cost))

            # suck and mark clean

            print("Location B is already clean.")

    elif status\_input == '0':

        print("Location A is already clean!")

        if status\_input\_complement == '1':  # if B is Dirty

            print("Location B is Dirty.")

            print("Moving RIGHT to the Location B. ")

            cost += 1  # cost for moving right

            print("COST for moving RIGHT: " + str(cost))

            # suck the dirt and mark it as clean

            goal\_state['B'] = '0'

            cost += 1  # cost for suck

            print("Cost for SUCK: " + str(cost))

            print("Location B has been Cleaned. ")

        else:

            print("No action " + str(cost))

            print(cost)

            # suck and mark clean

            print("Location B is already clean.")

    else:

        print("Vacuum is placed in location B")

        # Location B is Dirty.

        if status\_input == '1':

            print("Location B is Dirty.")

            # suck the dirt and mark it as clean

            goal\_state['B'] = '0'

            cost += 1  # cost for suck

            print("COST for CLEANING " + str(cost))

            print("Location B has been Cleaned.")

        if status\_input\_complement == '1':  # if A is Dirty

            print("Location A is Dirty.")

            print("Moving LEFT to the Location A. ")

            cost += 1  # cost for moving right

            print("COST for moving LEFT" + str(cost))

            # suck the dirt and mark it as clean

            goal\_state['A'] = '0'

            cost += 1  # cost for suck

            print("COST for SUCK " + str(cost))

            print("Location A has been Cleaned.")

        else:

            print(cost)

            # suck and mark clean

            print("Location B is already clean.")

        if status\_input\_complement == '1':  # if A is Dirty

            print("Location A is Dirty.")

            print("Moving LEFT to the Location A. ")

            cost += 1  # cost for moving right

            print("COST for moving LEFT " + str(cost))

            # suck the dirt and mark it as clean

            goal\_state['A'] = '0'

            cost += 1  # cost for suck

            print("Cost for SUCK " + str(cost))

            print("Location A has been Cleaned. ")

        else:

            print("No action " + str(cost))

            # suck and mark clean

            print("Location A is already clean.")

    # done cleaning

    print("GOAL STATE: ")

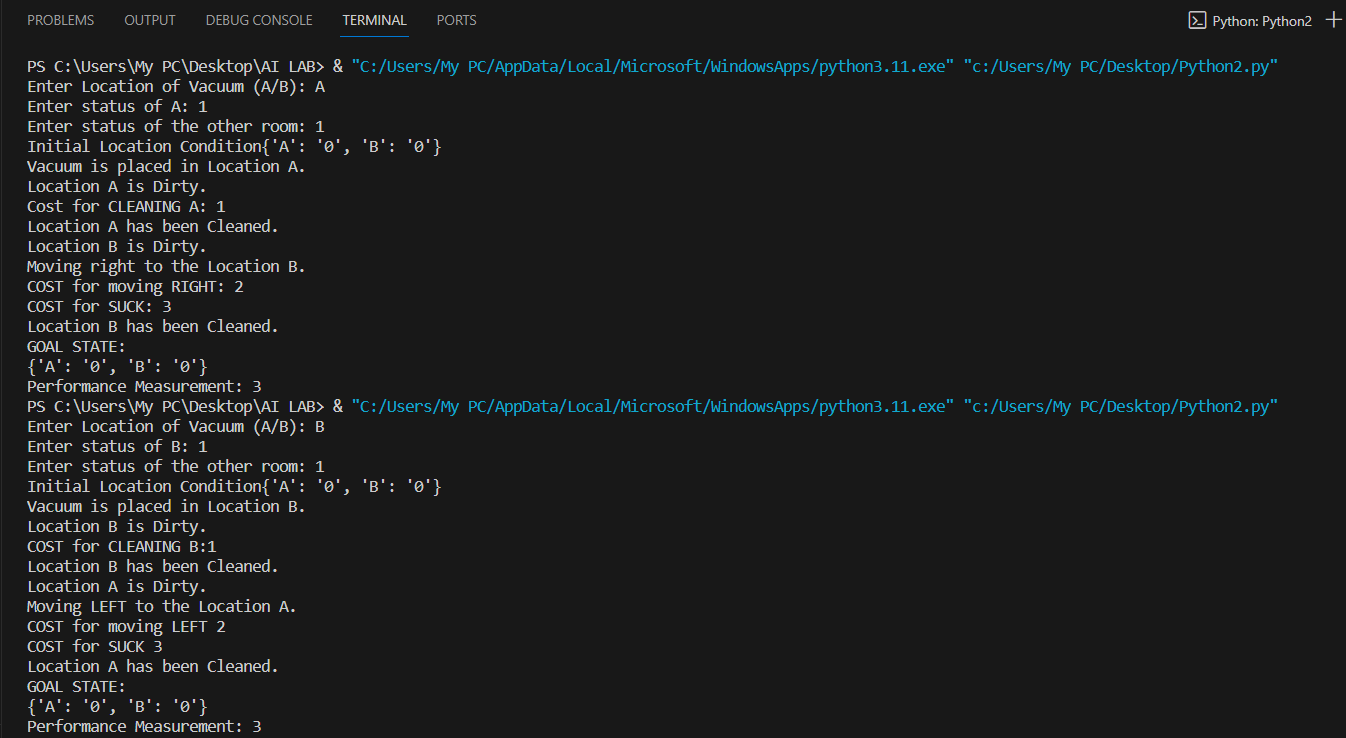
    print(goal\_state)

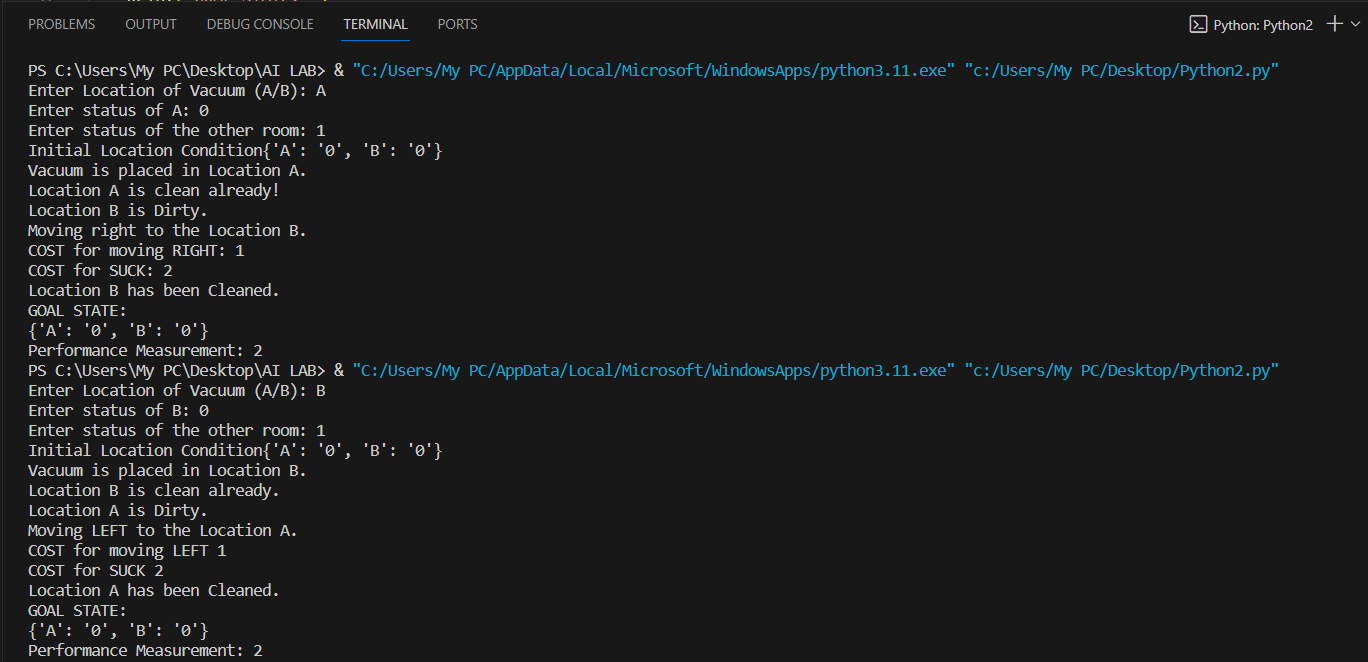
    print("Performance Measurement: " + str(cost))

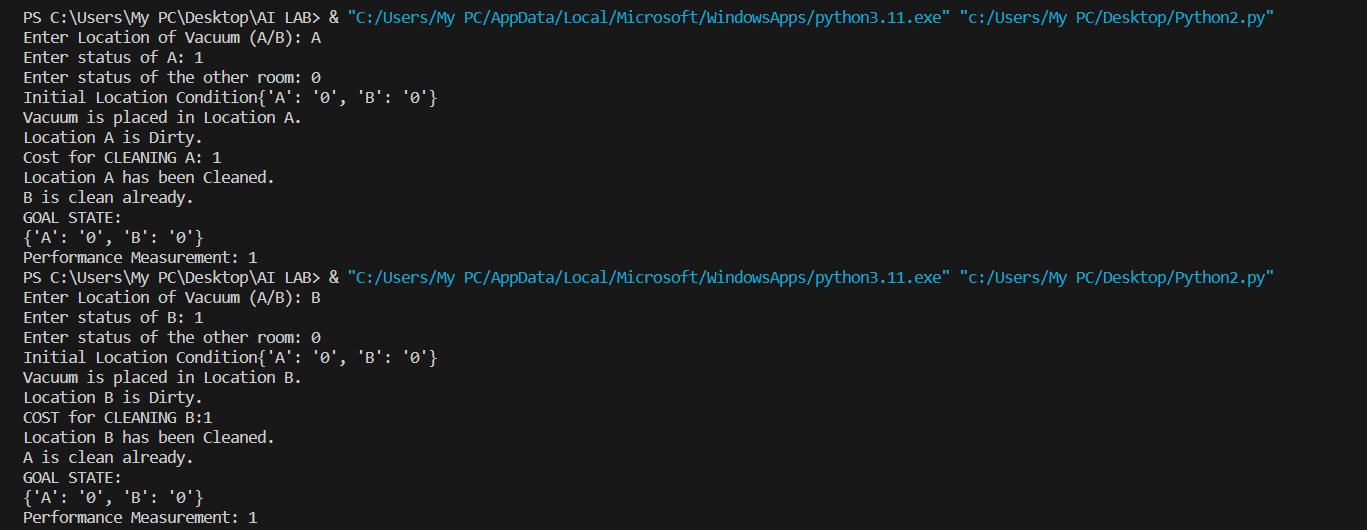
# Output:

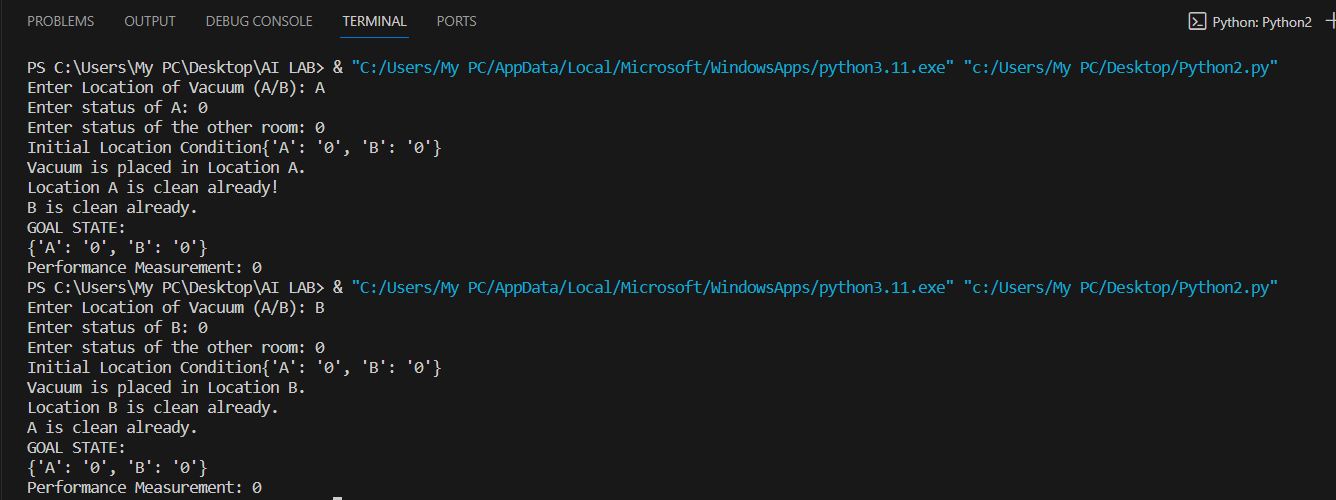
vacuum\_world()

OUTPUT:









All the 8 possible outputs have been analysed properly. All the cases considered have given the expected output.