

CSA17- Artificial Intelligence

Name :D.Siva Prasad Reddy

Reg :192011401

1. Write the python program to solve 8-Puzzle problem

```
import copy
```

```
from heapq import heappush, heappop
```

```
n = 3
```

```
rows = [ 1, 0, -1, 0 ]
```

```
cols = [ 0, -1, 0, 1 ]
```

```
class priorityQueue:
```

```
    def __init__(self):
```

```
        self.heap = []
```

```
    # Inserting a new key 'key'
```

```
    def push(self, key):
```

```
        heappush(self.heap, key)
```

```
    def pop(self):
```

```
        return heappop(self.heap)
```

```
    def empty(self):
```

```
        if not self.heap:
```

```
        return True
    else:
        return False
```

```
class nodes:
```

```
    def __init__(self, parent, mats, empty_tile_posi,
                  costs, levels):
```

```
        self.parent = parent
```

```
        self.mats = mats
```

```
        self.empty_tile_posi = empty_tile_posi
```

```
        self.costs = costs
```

```
        self.levels = levels
```

```
    def __lt__(self, nxt):
        return self.costs < nxt.costs
```

```
def calculateCosts(mats, final) -> int:
```

```
    count = 0
    for i in range(n):
        for j in range(n):
            if ((mats[i][j]) and
                (mats[i][j] != final[i][j])):
```

```
count += 1
```

```
return count
```

```
def newNodes(mats, empty_tile_posi, new_empty_tile_posi,  
            levels, parent, final) -> nodes:
```

```
new_mats = copy.deepcopy(mats)
```

```
x1 = empty_tile_posi[0]
```

```
y1 = empty_tile_posi[1]
```

```
x2 = new_empty_tile_posi[0]
```

```
y2 = new_empty_tile_posi[1]
```

```
new_mats[x1][y1], new_mats[x2][y2] = new_mats[x2][y2],  
new_mats[x1][y1]
```

```
costs = calculateCosts(new_mats, final)
```

```
new_nodes = nodes(parent, new_mats, new_empty_tile_posi,  
                  costs, levels)
```

```
return new_nodes
```

```
def printMatsrix(mats):
```

```
for i in range(n):
```

```
    for j in range(n):
```

```
        print("%d " % (mats[i][j]), end = " ")
```

```
    print()
```

```

def isSafe(x, y):

    return x >= 0 and x < n and y >= 0 and y < n

def printPath(root):

    if root == None:
        return

    printPath(root.parent)
    printMatsrix(root.mats)
    print()

def solve(initial, empty_tile_posi, final):

    pq = priorityQueue()

    costs = calculateCosts(initial, final)
    root = nodes(None, initial,
                 empty_tile_posi, costs, 0)

    pq.push(root)

    while not pq.empty():

        minimum = pq.pop()

        if minimum.costs == 0:

```

```
printPath(minimum)
return
```

```
for i in range(n):
    new_tile_posi = [
        minimum.empty_tile_posi[0] + rows[i],
        minimum.empty_tile_posi[1] + cols[i], ]

    if isSafe(new_tile_posi[0], new_tile_posi[1]):

        child = newNodes(minimum.mats,
                          minimum.empty_tile_posi,
                          new_tile_posi,
                          minimum.levels + 1,
                          minimum, final,)

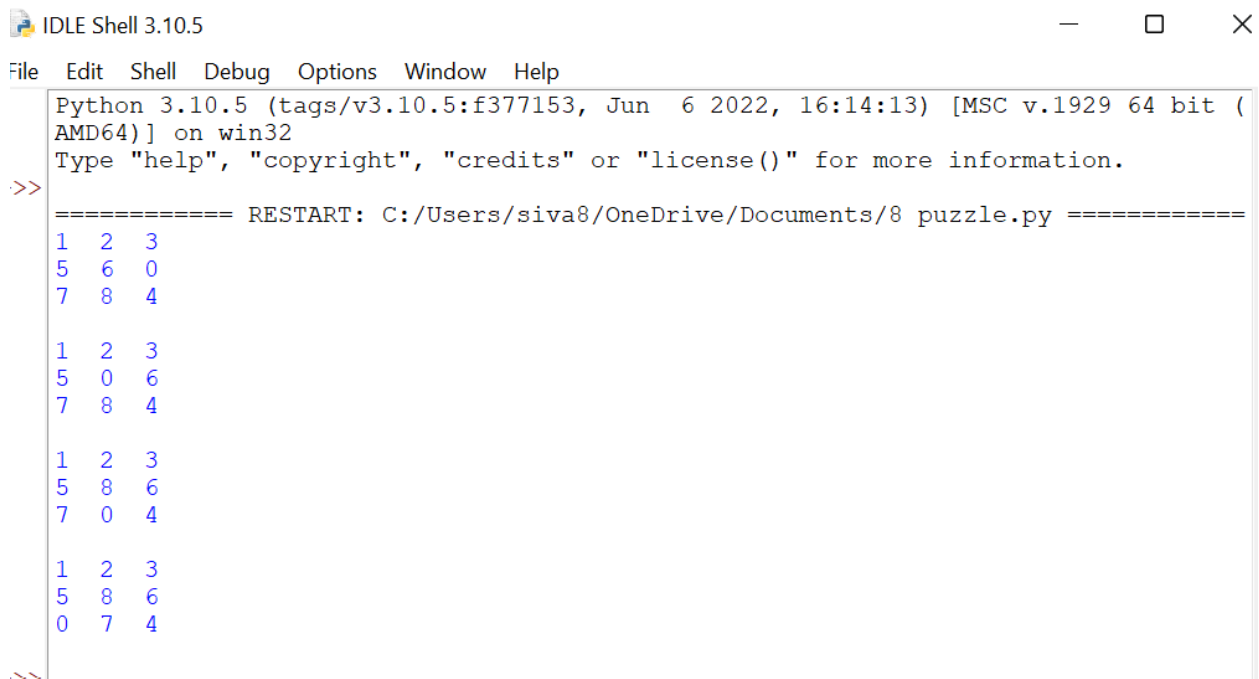
        pq.push(child)
```

```
initial = [ [ 1, 2, 3 ],
            [ 5, 6, 0 ],
            [ 7, 8, 4 ] ]
```

```
final = [ [ 1, 2, 3 ],
          [ 5, 8, 6 ],
          [ 0, 7, 4 ] ]
```

```
empty_tile_posi = [ 1, 2 ]
```

solve(initial, empty_tile_posi, final)



```
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>
===== RESTART: C:/Users/siva8/OneDrive/Documents/8 puzzle.py =====
1 2 3
5 6 0
7 8 4

1 2 3
5 0 6
7 8 4

1 2 3
5 8 6
7 0 4

1 2 3
5 8 6
0 7 4
```

2. Write the python program to solve 8-Queen problem

```
print ("Enter the number of queens")
```

```
N = int(input())
```

```
board = [[0]*N for _ in range(N)]
```

```
def attack(i, j):
```

```
    for k in range(0,N):
```

```
        if board[i][k]==1 or board[k][j]==1:
```

```
            return True
```

```
    for k in range(0,N):
```

```
        for l in range(0,N):
```

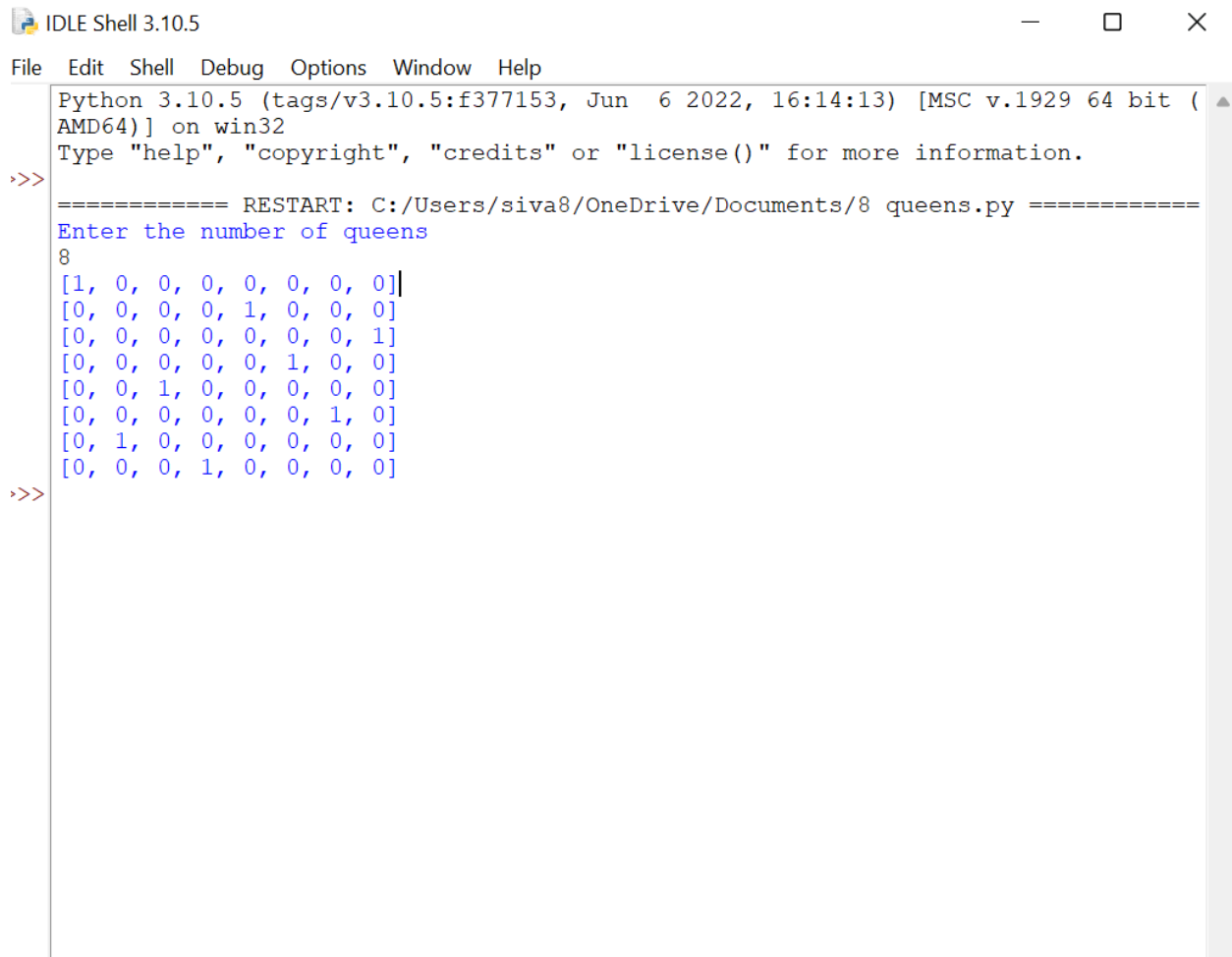
```

        if (k+l==i+j) or (k-l==i-j):
            if board[k][l]==1:
                return True
    return False

def N_queens(n):
    if n==0:
        return True
    for i in range(0,N):
        for j in range(0,N):
            if (not(attack(i,j))) and (board[i][j]!=1):
                board[i][j] = 1
                if N_queens(n-1)==True:
                    return True
                board[i][j] = 0
    return False

N_queens(N)
for i in board:
    print (i)

```



```
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/siva8/OneDrive/Documents/8 queens.py =====
Enter the number of queens
8
[1, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 1, 0, 0]
[0, 0, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 0]
[0, 1, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 0, 0, 0]
>>>
```

3. Write the python program for Water Jug Problem

def waterJugSolver(amt1, amt2):

if (amt1 == aim and amt2 == 0) or (amt2 == aim and amt1 == 0):

print(amt1, amt2)

return True

if visited[(amt1, amt2)] == False:

print(amt1, amt2)


```

        visited[(amt1, amt2)] = True

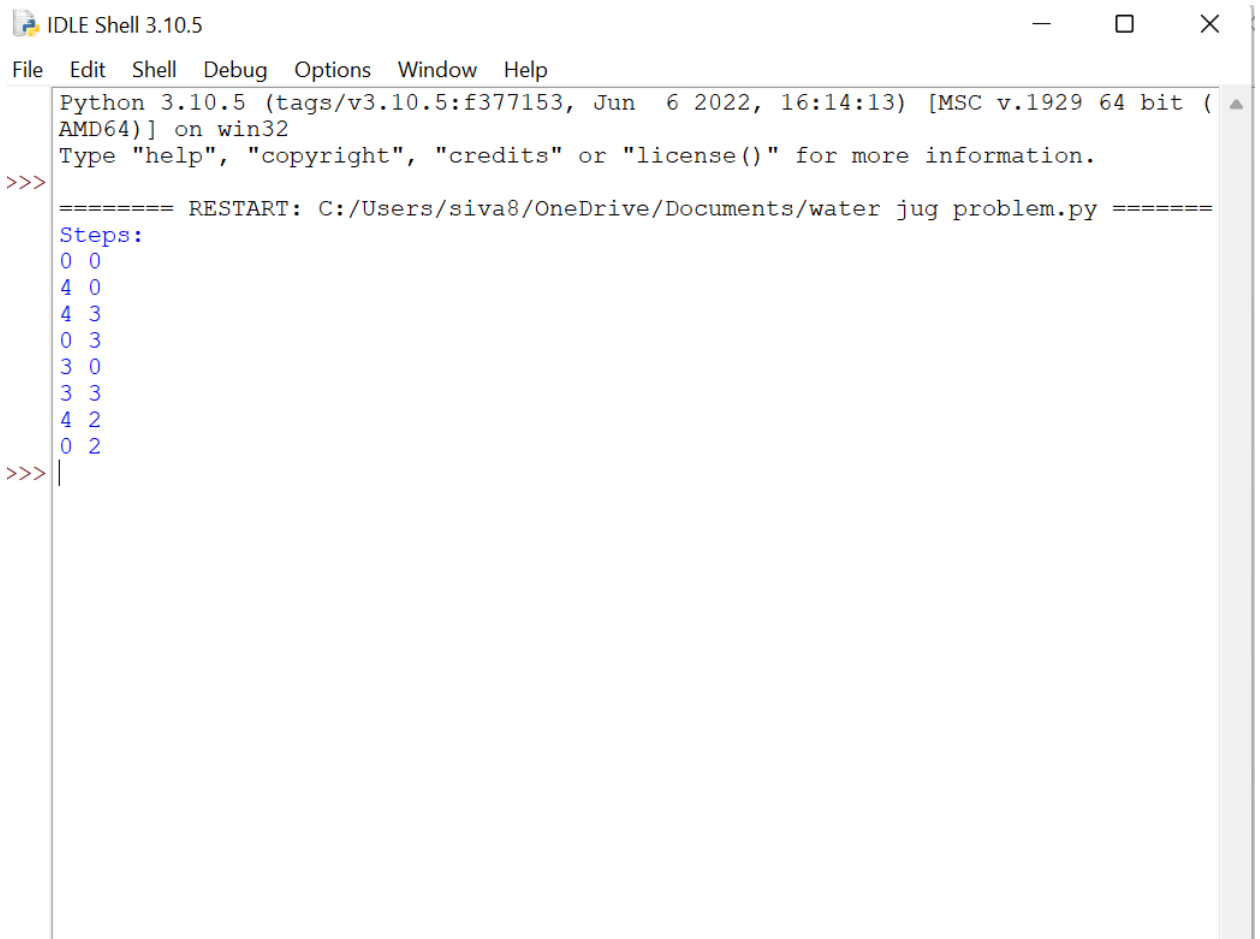
    return (waterJugSolver(0, amt2) or
            waterJugSolver(amt1, 0) or
            waterJugSolver(jug1, amt2) or
            waterJugSolver(amt1, jug2) or
            waterJugSolver(amt1 + min(amt2, (jug1-
amt1)),
                                amt2 - min(amt2, (jug1-amt1))) or
            waterJugSolver(amt1 - min(amt1, (jug2-
amt2)),
                                amt2 + min(amt1, (jug2-amt2))))

    else:
        return False

print("Steps: ")

waterJugSolver(0, 0)

```



```
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/siva8/OneDrive/Documents/water_jug_problem.py =====
Steps:
0 0
4 0
4 3
0 3
3 0
3 3
4 2
0 2
>>> |
```

4. Write the python program for Cript-Arithmetic problem

def isSolvable(words, result):

mp = [-1]*(26)

used = [0]*(10)

Hash = [0]*(26)

CharAtfront = [0]*(26)

uniq = ""

```

for word in range(len(words)):
    for i in range(len(words[word])):

        ch = words[word][i]
        Hash[ord(ch) - ord('A')] += pow(10, len(words[word]) - i - 1)

    if mp[ord(ch) - ord('A')] == -1:

        mp[ord(ch) - ord('A')] = 0

    uniq += str(ch)
    if i == 0 and len(words[word]) > 1:

        CharAtfront[ord(ch) - ord('A')] = 1
for i in range(len(result)):

    ch = result[i]

    Hash[ord(ch) - ord('A')] -= pow(10, len(result) - i - 1)

    if mp[ord(ch) - ord('A')] == -1:

```

```
mp[ord(ch) - ord('A')] = 0
```

```
uniq += str(ch)
```

```
if i == 0 and len(result) > 1:
```

```
CharAtfront[ord(ch) - ord('A')] = 1
```

```
mp = [-1]*(26)
```

```
return True
```

```
def solve(words, i, S, p, used, Hash, CharAtfront):
```

```
    if i == len(words):
```

```
        return S == 0
```

```
    ch = words[i]
```

```
val = mp[ord(words[i]) - ord('A')]
```

```
if val != -1:
```

```
    return solve(words, i + 1, S + val * Hash[ord(ch) - ord('A')], mp,  
used, Hash, CharAtfront)
```

```
x = False
```

```
for l in range(10):
```

```
    if CharAtfront[ord(ch) - ord('A')] == 1 and l == 0:
```

```
        continue
```

```
    if used[l] == 1:
```

```
        continue
```

```
mp[ord(ch) - ord('A')] = l
```

```
used[l] = 1
```

```
x |= solve(words, i + 1, S + l * Hash[ord(ch) - ord('A')], mp, used,  
Hash, CharAtfront)
```

```
mp[ord(ch) - ord('A')] = -1
```

```
used[l] = 0
```

```
return x
```

```
arr = [ "SIX", "SEVEN", "SEVEN" ]
```

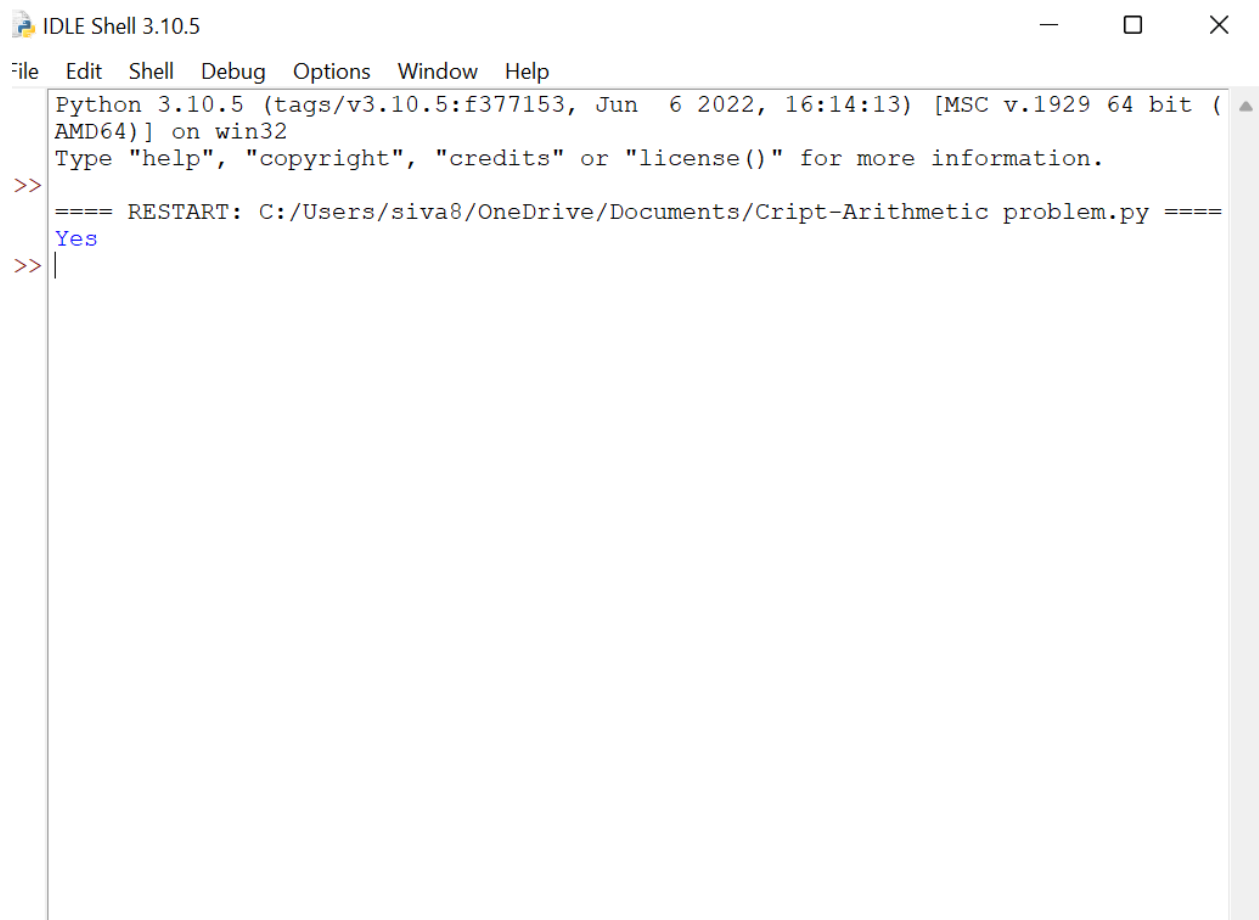
```
S = "TWENTY"
```

```
if isSolvable(arr, S):
```

```
    print("Yes")
```

```
else:
```

```
    print("No")
```



The screenshot shows the IDLE Shell 3.10.5 window. The title bar reads "IDLE Shell 3.10.5". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The shell area displays the following text:

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>
==== RESTART: C:/Users/siva8/OneDrive/Documents/Cript-Arithmetic problem.py ====
Yes
>> |
```

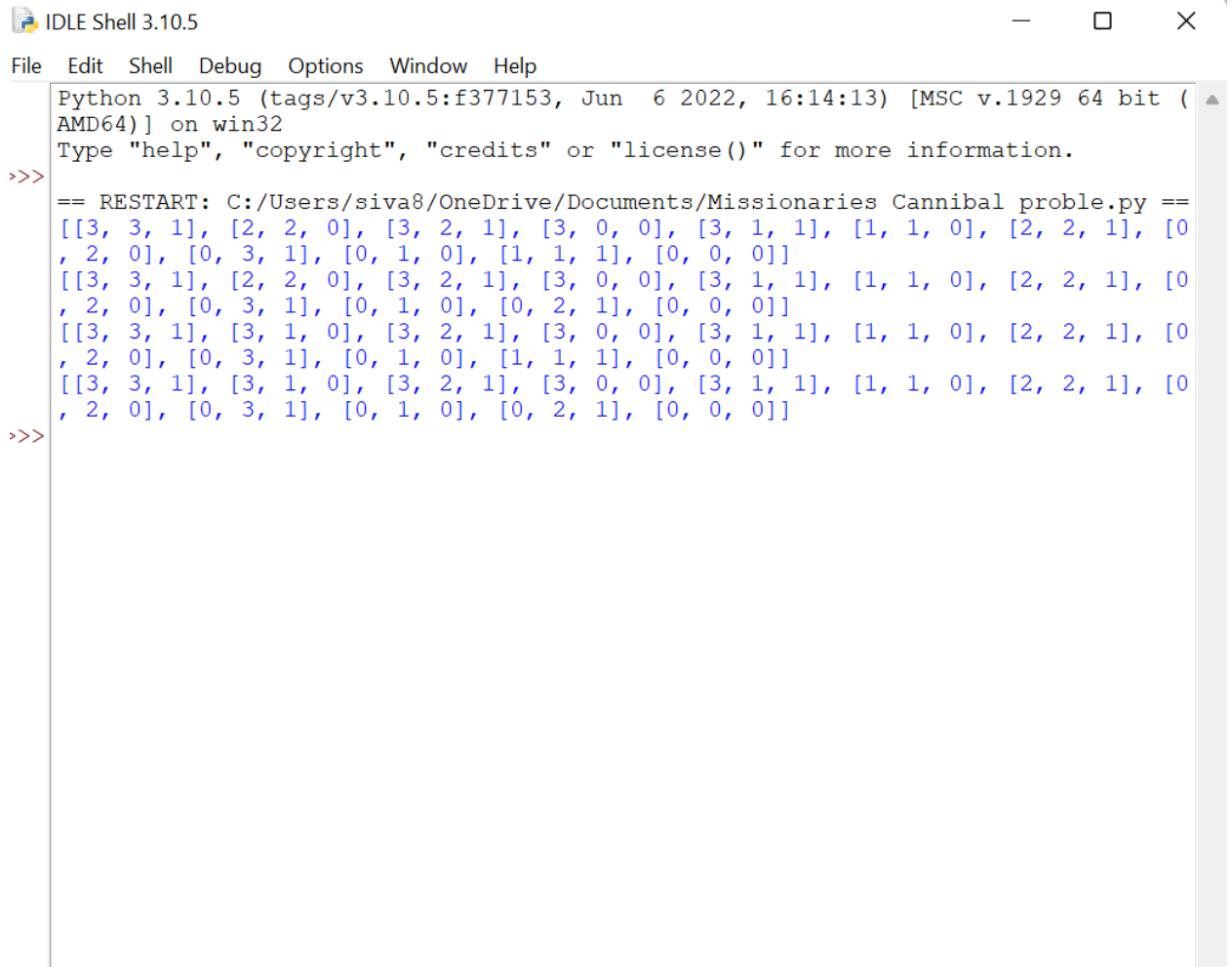
5. Write the python program for Missionaries Cannibal problem

```
start,end =[3,3,1],[0,0,0]
def do_action(state,action):
    if state[2] == 1:
        return [state[i] - action[i] for i in range(3)]
    else:
        return [state[i] + action[i] for i in range(3)]
def is_legal(state):
    if 0 <= state[0] <= 3 and 0 <= state[1] <= 3:
        return True
    else:
        return False
def is_bank_safe(bank):
    if bank[1] > bank[0] and bank[0] != 0:
        return False
    else:
        return True
def is_state_safe(state):
    other_bank = [start[i]-state[i] for i in range(3)]
    if is_bank_safe(state) and is_bank_safe(other_bank) :
        return True
    else:
        return False
def next_possible_actions(state):
    actions = [[1,0,1],[0,1,1],[1,1,1],[2,0,1],[0,2,1]]
    moves = []
    for i in actions:
        j = do_action(state,i)
        if is_legal(j) and is_state_safe(j):
```



```
        moves.append(j)
    return moves
solutions = []
def solve(next_action,path):
    _path = path.copy()
    if next_action == end:
        _path.append(next_action)
        solutions.append(_path)
        return
    elif next_action in path:
        return
    else:
        _path.append(next_action)
        for i in next_possible_actions(next_action):
            solve(i,_path)

solve([3,3,1],[])
print(*solutions,sep="\n")
```

A screenshot of the IDLE Shell 3.10.5 window. The title bar says "IDLE Shell 3.10.5". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The shell area shows the following text: "Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32", "Type 'help', 'copyright', 'credits' or 'license()' for more information.", and a prompt ">>>". Below the prompt, the output of a script is shown: "== RESTART: C:/Users/siva8/OneDrive/Documents/Missionaries Cannibal proble.py ==", followed by five lines of lists of lists of integers, each line representing a state in a problem solution. The lists are: [[3, 3, 1], [2, 2, 0], [3, 2, 1], [3, 0, 0], [3, 1, 1], [1, 1, 0], [2, 2, 1], [0, 2, 0], [0, 3, 1], [0, 1, 0], [1, 1, 1], [0, 0, 0]], [[3, 3, 1], [2, 2, 0], [3, 2, 1], [3, 0, 0], [3, 1, 1], [1, 1, 0], [2, 2, 1], [0, 2, 0], [0, 3, 1], [0, 1, 0], [0, 2, 1], [0, 0, 0]], [[3, 3, 1], [3, 1, 0], [3, 2, 1], [3, 0, 0], [3, 1, 1], [1, 1, 0], [2, 2, 1], [0, 2, 0], [0, 3, 1], [0, 1, 0], [1, 1, 1], [0, 0, 0]], [[3, 3, 1], [3, 1, 0], [3, 2, 1], [3, 0, 0], [3, 1, 1], [1, 1, 0], [2, 2, 1], [0, 2, 0], [0, 3, 1], [0, 1, 0], [0, 2, 1], [0, 0, 0]], and a final prompt ">>>".

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: C:/Users/siva8/OneDrive/Documents/Missionaries Cannibal proble.py ==
[[3, 3, 1], [2, 2, 0], [3, 2, 1], [3, 0, 0], [3, 1, 1], [1, 1, 0], [2, 2, 1], [0, 2, 0], [0, 3, 1], [0, 1, 0], [1, 1, 1], [0, 0, 0]]
[[3, 3, 1], [2, 2, 0], [3, 2, 1], [3, 0, 0], [3, 1, 1], [1, 1, 0], [2, 2, 1], [0, 2, 0], [0, 3, 1], [0, 1, 0], [0, 2, 1], [0, 0, 0]]
[[3, 3, 1], [3, 1, 0], [3, 2, 1], [3, 0, 0], [3, 1, 1], [1, 1, 0], [2, 2, 1], [0, 2, 0], [0, 3, 1], [0, 1, 0], [1, 1, 1], [0, 0, 0]]
[[3, 3, 1], [3, 1, 0], [3, 2, 1], [3, 0, 0], [3, 1, 1], [1, 1, 0], [2, 2, 1], [0, 2, 0], [0, 3, 1], [0, 1, 0], [0, 2, 1], [0, 0, 0]]
>>>
```

6. Write the python program for Vacuum Cleaner problem

```
import random

def display(room):
    print(room)

room = [
    [1, 1, 1, 1],
    [1, 1, 1, 1],
    [1, 1, 1, 1],
    [1, 1, 1, 1],
]
```

```
print("All the rooom are dirty")
display(room)
```

```
x =0
```

```
y= 0
```

```
while x < 4:
```

```
    while y < 4:
```

```
        room[x][y] = random.choice([0,1])
```

```
        y+=1
```

```
    x+=1
```

```
    y=0
```

```
print("Before cleaning the room I detect all of these random  
dirts")
```

```
display(room)
```

```
x =0
```

```
y= 0
```

```
z=0
```

```
while x < 4:
```

```
    while y < 4:
```

```
        if room[x][y] == 1:
```

```
            print("Vaccum in this location now,",x, y)
```

```
            room[x][y] = 0
```

```
            print("cleaned", x, y)
```

```
            z+=1
```

```
        y+=1
```

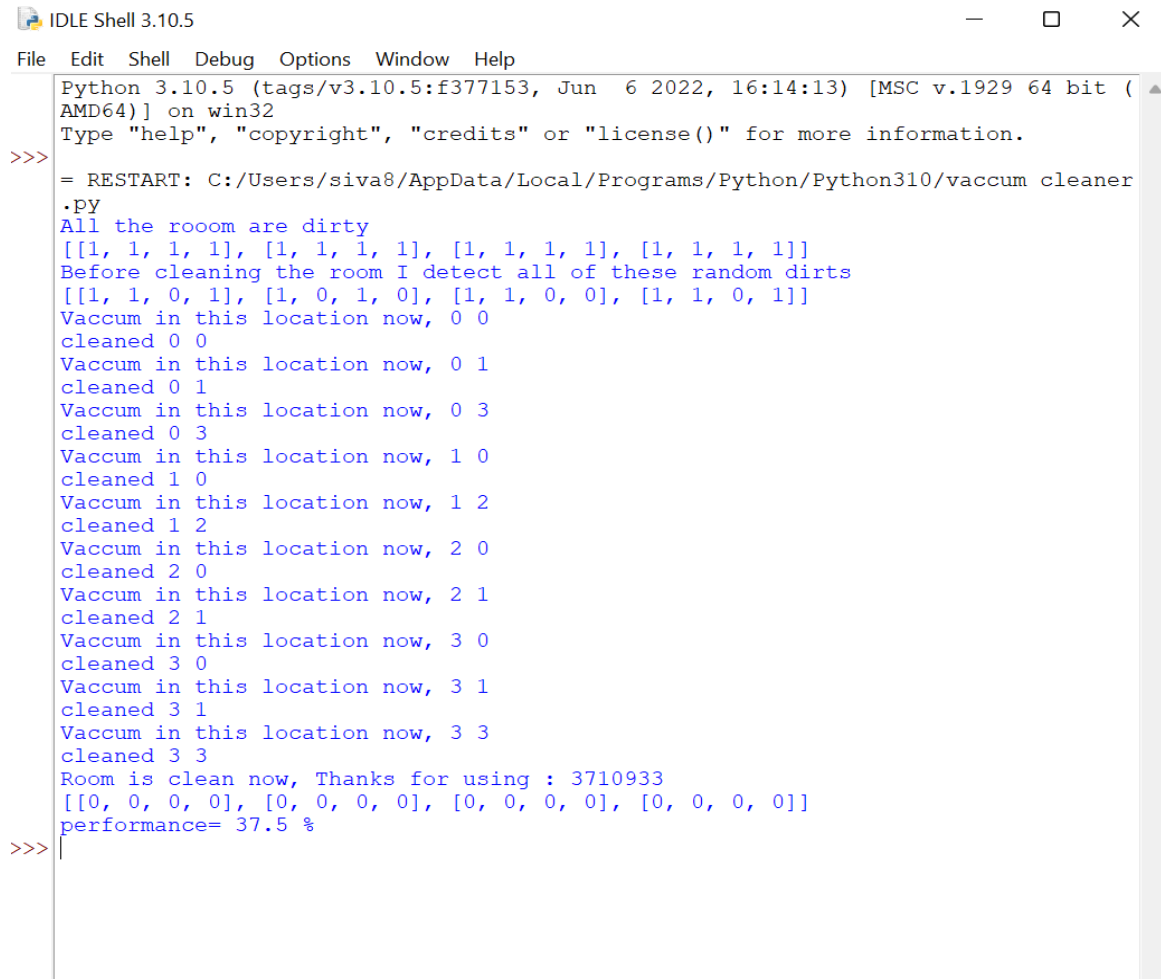
```
    x+=1
```

```
    y=0
```

```
pro= (100-((z/16)*100))
```

```
print("Room is clean now, Thanks for using : 3710933")
```

```
display(room)
print('performance=',pro,'%')S
```



The screenshot shows an IDLE Shell 3.10.5 window. The title bar reads "IDLE Shell 3.10.5". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The shell window displays the output of a Python script. The first line of the script is a docstring: "Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32". The second line is a prompt to type "help", "copyright", "credits" or "license()". The third line is a restart command: "= RESTART: C:/Users/siva8/AppData/Local/Programs/Python/Python310/vacuum cleaner.py". The script then prints "All the rooom are dirty" (note the typo). It displays a 3x3 grid of dirt levels: [[1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1]]. It then prints "Before cleaning the room I detect all of these random dirts" and another 3x3 grid: [[1, 1, 0, 1], [1, 0, 1, 0], [1, 1, 0, 0], [1, 1, 0, 1]]. The script then iterates through the grid, printing "Vaccum in this location now, x y" and "cleaned x y" for each cell. The final output is "Room is clean now, Thanks for using : 3710933" and a performance report: "[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]] performance= 37.5 %". The prompt ">>>" is visible at the bottom of the shell window.

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/siva8/AppData/Local/Programs/Python/Python310/vacuum cleaner.py
All the rooom are dirty
[[1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1]]
Before cleaning the room I detect all of these random dirts
[[1, 1, 0, 1], [1, 0, 1, 0], [1, 1, 0, 0], [1, 1, 0, 1]]
Vaccum in this location now, 0 0
cleaned 0 0
Vaccum in this location now, 0 1
cleaned 0 1
Vaccum in this location now, 0 3
cleaned 0 3
Vaccum in this location now, 1 0
cleaned 1 0
Vaccum in this location now, 1 2
cleaned 1 2
Vaccum in this location now, 2 0
cleaned 2 0
Vaccum in this location now, 2 1
cleaned 2 1
Vaccum in this location now, 3 0
cleaned 3 0
Vaccum in this location now, 3 1
cleaned 3 1
Vaccum in this location now, 3 3
cleaned 3 3
Room is clean now, Thanks for using : 3710933
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
performance= 37.5 %
>>>
```