**PRACTICAL-2**

**A)AIM: Write a python program to print the multiplication table for the given number.**

**Code:**

x = int(input("Enter Number:"))

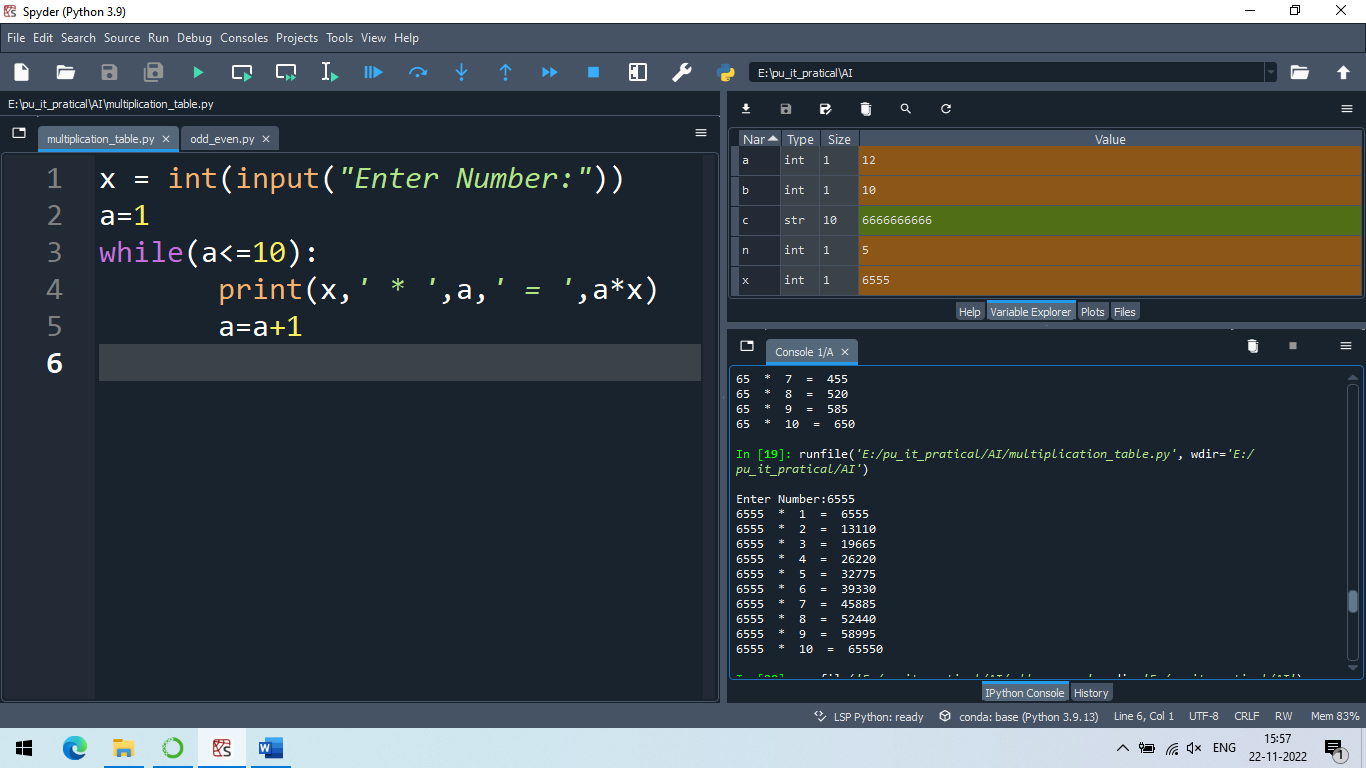
a=1

while(a<=10):

print(x,' \* ',a,' = ',a\*x)

a=a+1

**Output:**



**B)AIM:** **Write a python program to check whether the given number is prime or not.**

**Code:**

x = int(input("Enter Number:"))

if x > 1:

for n in range(2, x):

if (x % n) == 0:

print(x, "is not prime")

print(n, "times", x // n, "is", x)

break

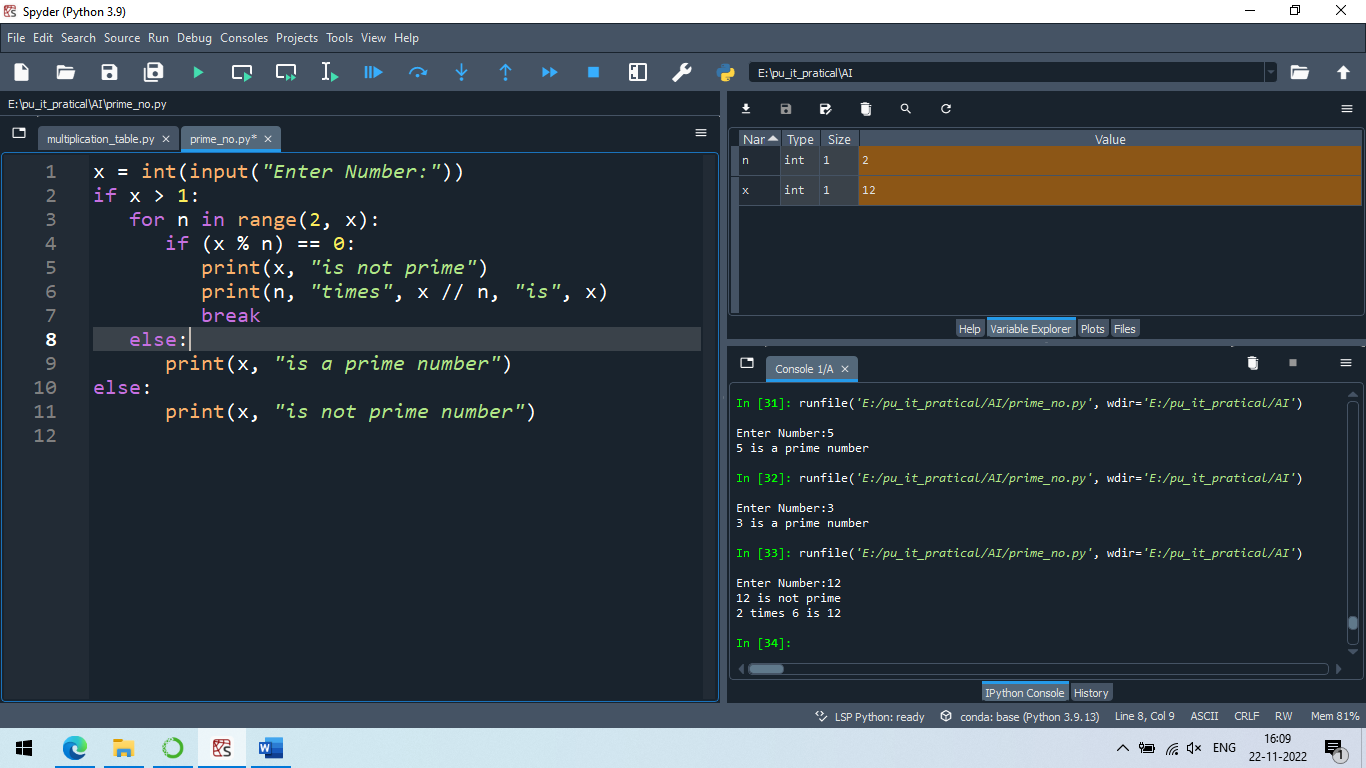
else:

print(x, "is a prime number")

else:

print(x, "is not prime number")

**Output:**



**C) AIM: Write a python program to find factorial of the given number.**

**Code:**

a = int(input("Enter Number:"))

p=1

i=a

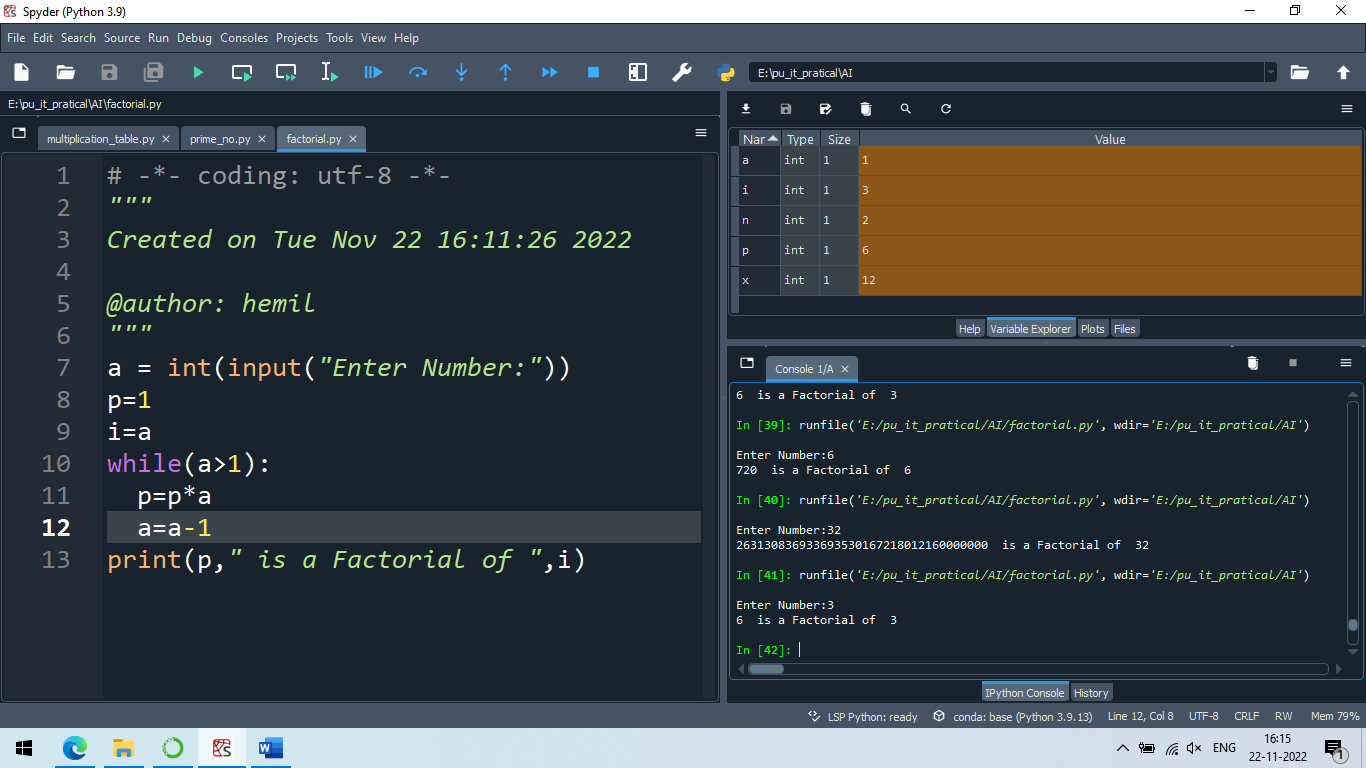
while(a>1):

p=p\*a

a=a-1

print(p," is a Factorial of ",i)

**Output:**



**PRACTICAL-4**

**AIM: A)** **Write a python program to implement List operations (Nested List, Length, Concatenation, Membership, Iteration, Indexing and Slicing)?**

**Code**:

list\_name = ['Hemil','Jay','Avadh',[5,6,8,9],'Tirth','Romit']

print('Nested list')

print(list\_name) #nested list

print('length of list')

print(len(list\_name)) #printing length of list

print(list\_name) #printing original list

list\_name.pop(1) #delete value at index 1

print('after deleting value at index 1')

print(list\_name) #printing list after pop

list\_items = ['raj','krish']

print('Concatenation of 2 list')

print(list\_name+list\_items) #Concatenation of 2 list

print('Membership')

if "Tirth" in list\_name:

print("Yes, 'Tirth' is member the list") # printing the membership of the list

print('Iteration') #Iteration

i = 0

while i < len(list\_name):

print(list\_name[i])

i = i + 1

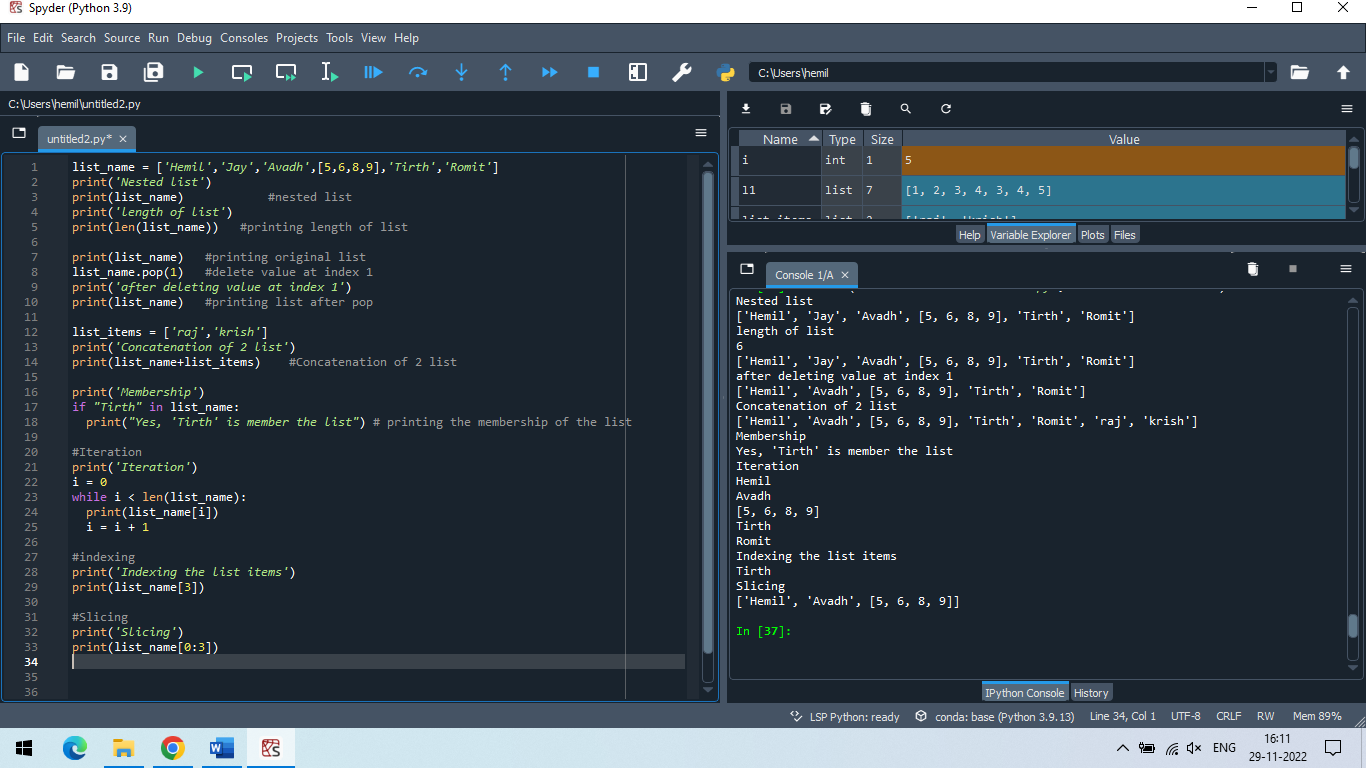
print('Indexing the list items') #indexing

print(list\_name[3])

print('Slicing') #Slicing

print(list\_name[0:3])

**Output A:**



**B) Write a python program to implement List methods (Add, Append, Extend & Delete).**

**Code**:

list\_items=['Hemil', 'Jay', 'Avadh', [5, 6, 8, 9], 'Tirth', 'Romit']

print(list\_items)

#insert

print('insert ronak in list')

list\_items.insert(1,'ronak')

print(list\_items)

#append

print('append list')

list\_items\_object = ['table','furniture']

list\_items.append(list\_items\_object)

print(list\_items)

#extend

print('Extend list')

list\_items\_object = ['table','furniture']

list\_items.extend(list\_items\_object)

print(list\_items)

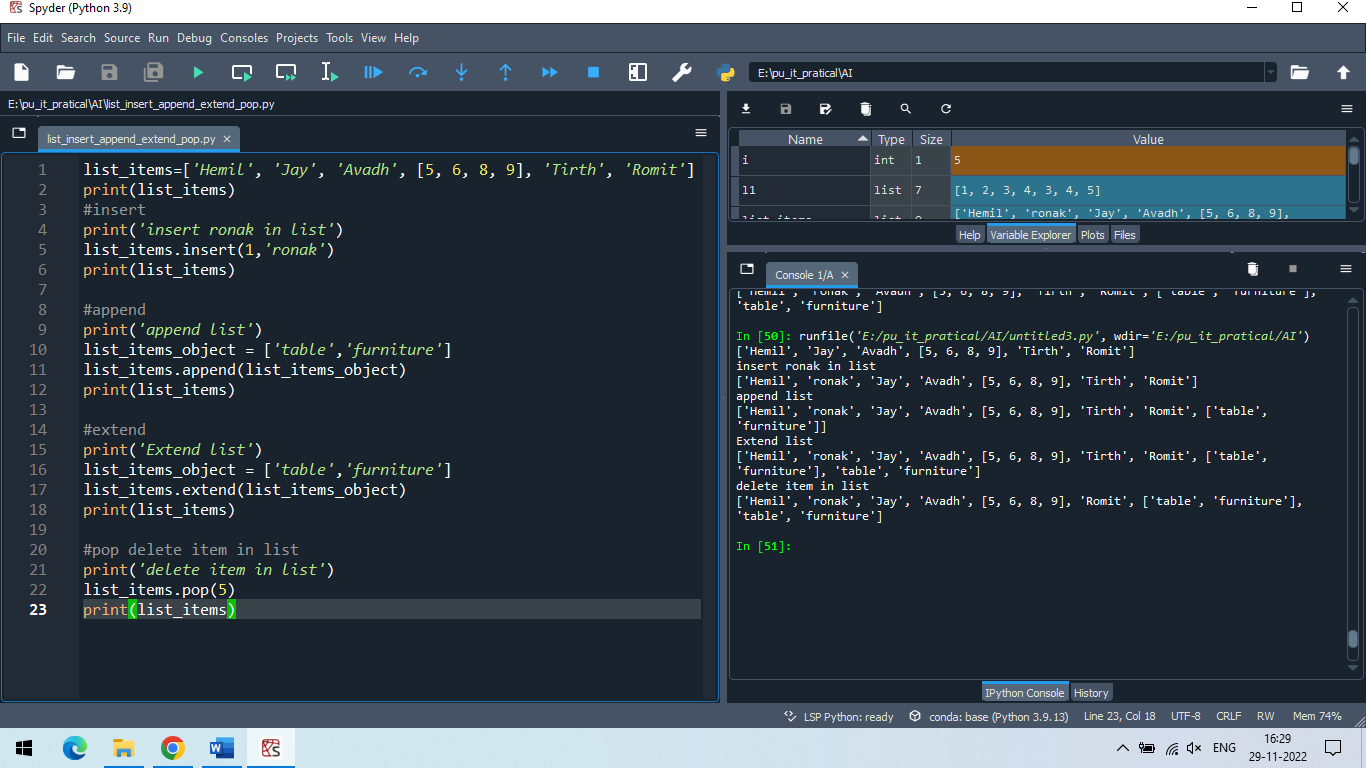
#pop delete item in list

print('delete item in list')

list\_items.pop(5)

print(list\_items)

**Output B:**



**PRACTICAL-5**

**AIM: a) Write a python program to Illustrate Different Set Operations?**

**b) Write a python program to generate Calendar for the given month and year? c) Write a python program to implement Simple Calculator program?**

**Code A :**

even = {'null',2, 4, 6, 8};

odd = {'null',3, 5, 7};

def Union(even,odd):

print("Union of even and odd is",even | odd) # set union

def Intersection(even,odd):

print("Intersection of even and odd is",even & odd) # set intersection

def Difference(even,odd):

print("Difference of even and odd is",even - odd) # set difference

def Symmetric(even,odd):

print("Symmetric difference of even and odd is",even ^ odd) # set symmetric difference

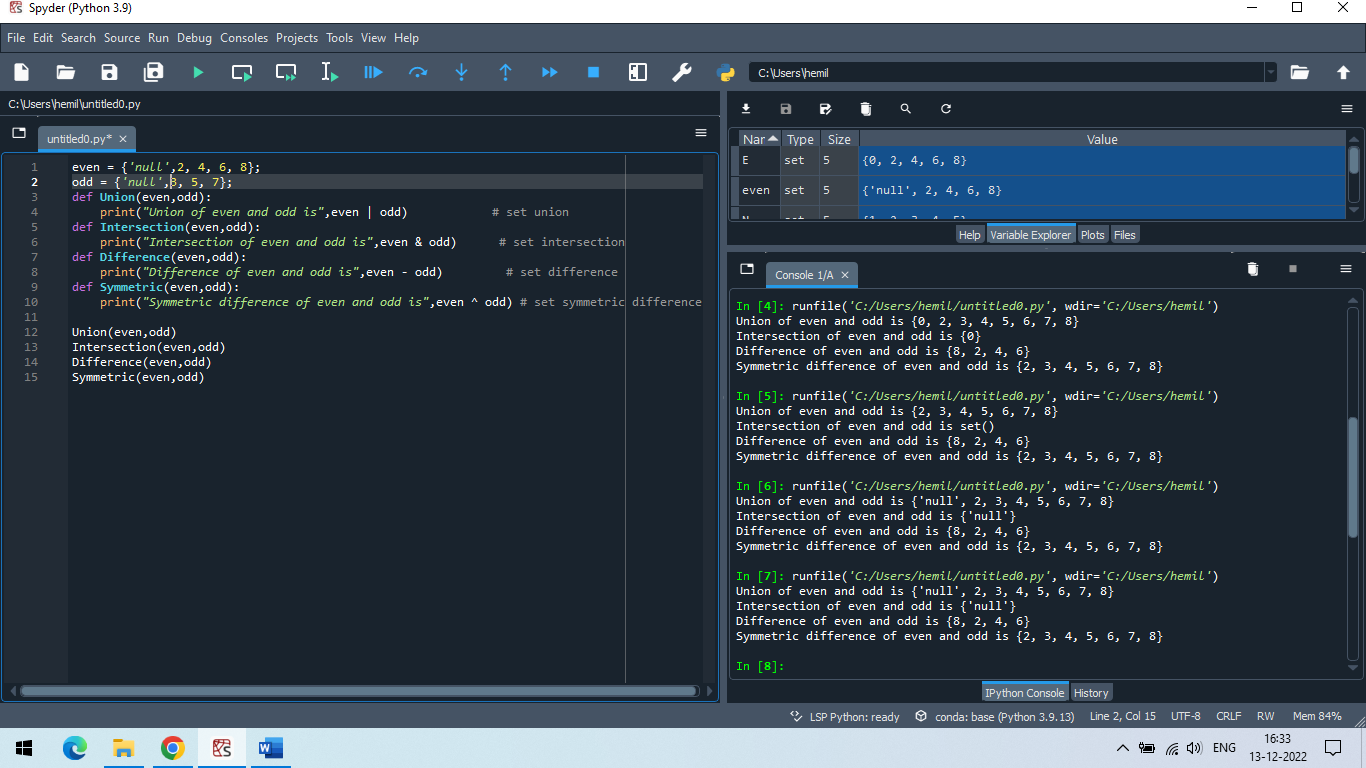
Union(even,odd)

Intersection(even,odd)

Difference(even,odd)

Symmetric(even,odd)

**Output:**



**Code B :**

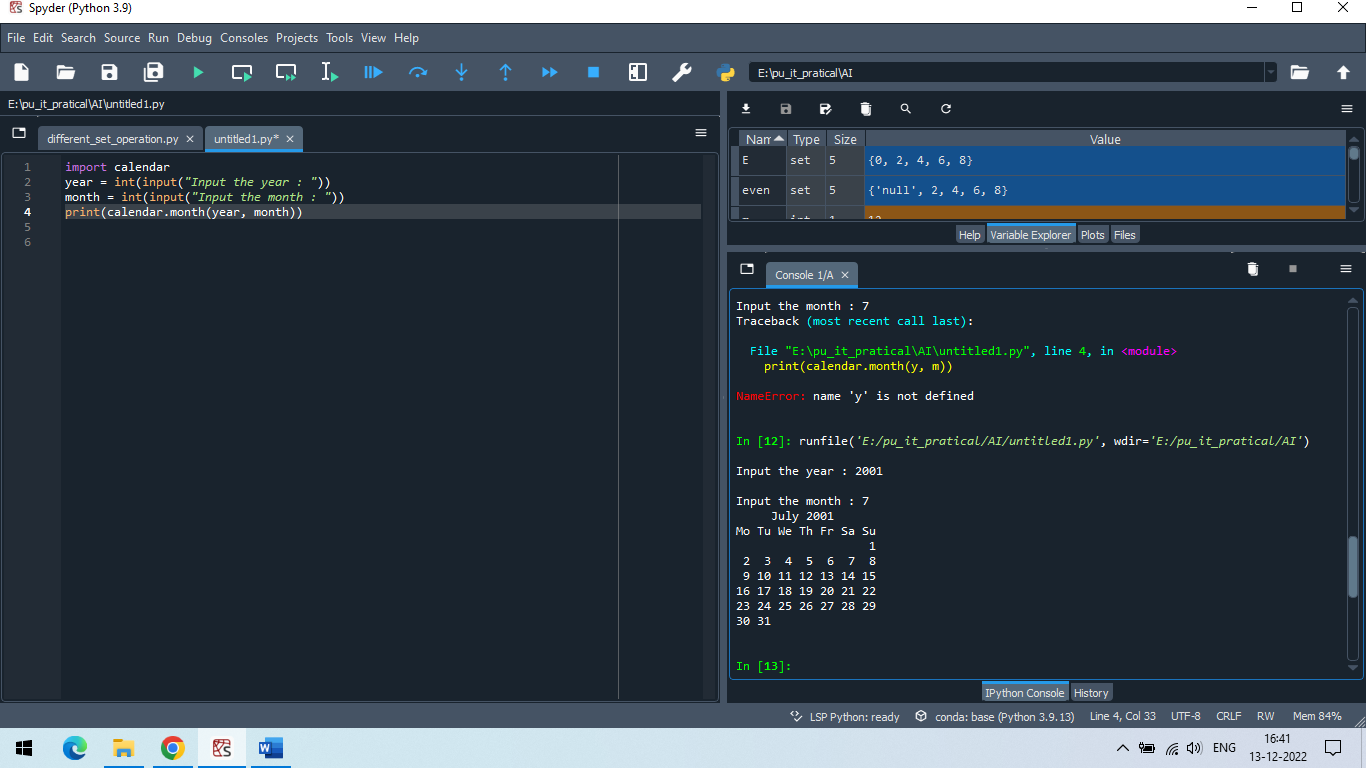
import calendar

year = int(input("Input the year : "))

month = int(input("Input the month : "))

print(calendar.month(year, month))

**Output:**



**Code C :**

def div(a,b):

print(a/b)

def mul(a,b):

print(a\*b)

def add(a,b):

print(a+b)

def sub(a,b):

print(a-b)

def power(a,b):

print(a^b)

operation=0

while operation < 6:

print("1.div 2.mul 3.add 4.sub 5.power 6.exit")

operation = int(input("Enter Operation No: "))

first\_no = int(input("Input 1st no : "))

second\_no = int(input("Input 2nd no : "))

if operation == 1:

div(first\_no, second\_no)

elif operation == 2:

mul(first\_no, second\_no)

elif operation == 3:

add(first\_no, second\_no)

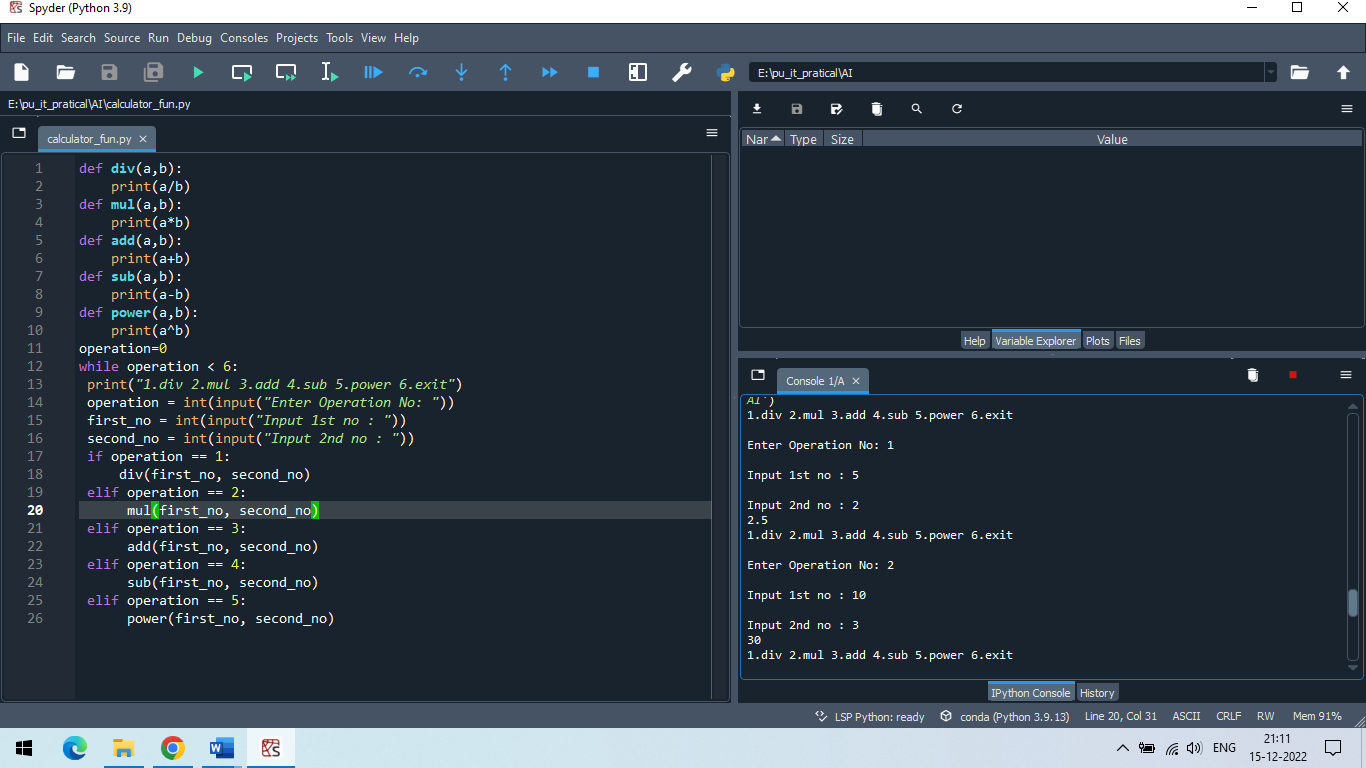
elif operation == 4:

sub(first\_no, second\_no)

elif operation == 5:

power(first\_no, second\_no)

**Output:**



**PRACTICAL-6**

**AIM: a)Write a python program to Add Two Matrices.**

**b) Write a python program to Transpose a Matrix.**

**Code a)**

R = int(input("Enter Matrix rows:"))

C = int(input("Enter Matrix columns:"))

matrix1 = [] # Initialize matrix

# For user input

for i in range(R): # A for loop for row entries

a1 =[]

for j in range(C): # A for loop for column entries

print("Enter Matrix1 Number:",i,j)

a1.append(int(input()))

matrix1.append(a1)

matrix2 = [] # Initialize matrix

# For user input

for i in range(R): # A for loop for row entries

a2 =[]

for j in range(C): # A for loop for column entries

print("Enter Matrix2 Number:",i,j)

a2.append(int(input()))

matrix2.append(a2)

print("Addition of 2 Matrix ",R,"\*",C) # For printing the matrix

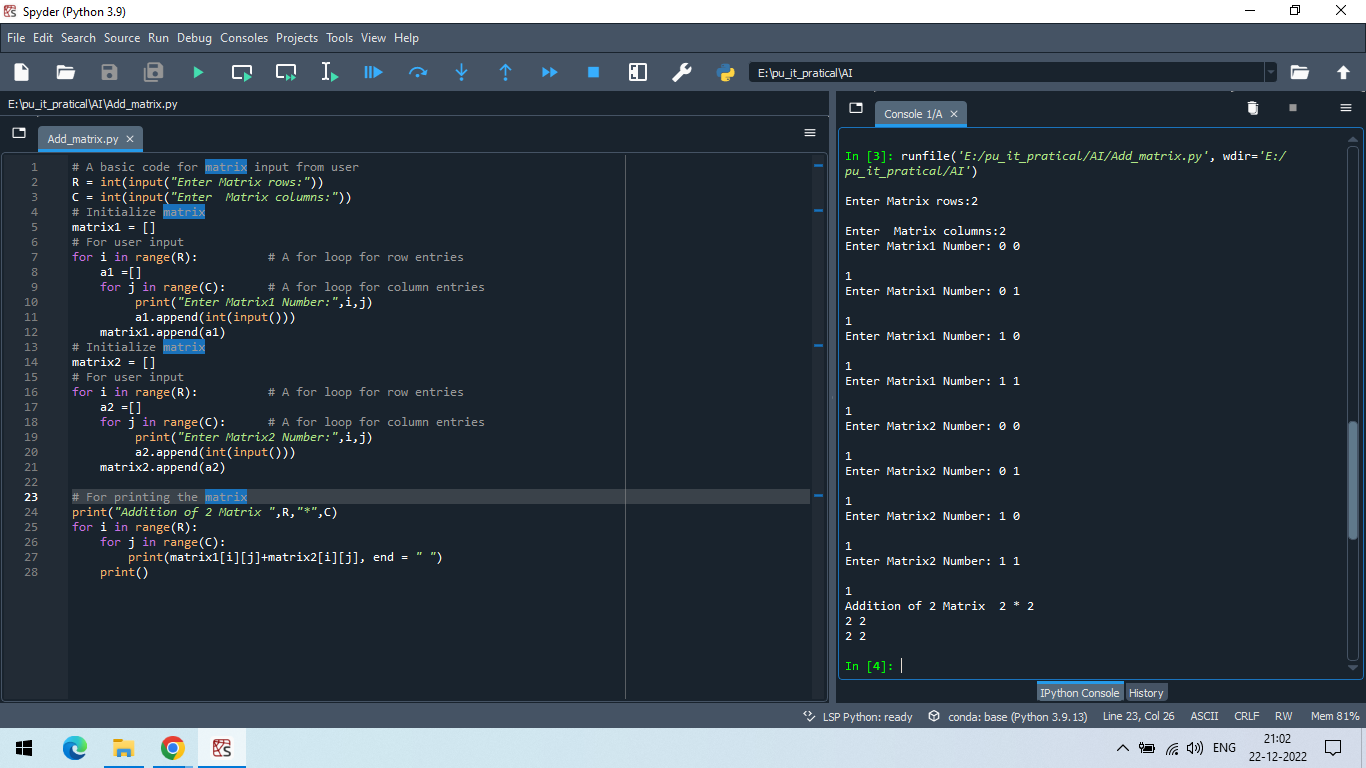
for i in range(R):

for j in range(C):

print(matrix1[i][j]+matrix2[i][j], end = " ")

print()

**Output:**



**Code b)**

R = int(input("Enter Matrix rows:"))

C = int(input("Enter Matrix columns:"))

matrix = [] # Initialize matrix

transpose = []

# For user input

for i in range(R): # A for loop for row entries

a =[]

for j in range(C): # A for loop for column entries

print("Enter Matrix Number:",i,j)

a.append(int(input()))

matrix.append(a)

transpose.append(a)

transpose = [[0 for i in range(R)] for j in range(C)]

# Initialize matrix

for i in range(R):

for j in range(C):

transpose[j][i] = matrix[i][j]

# For printing the matrix

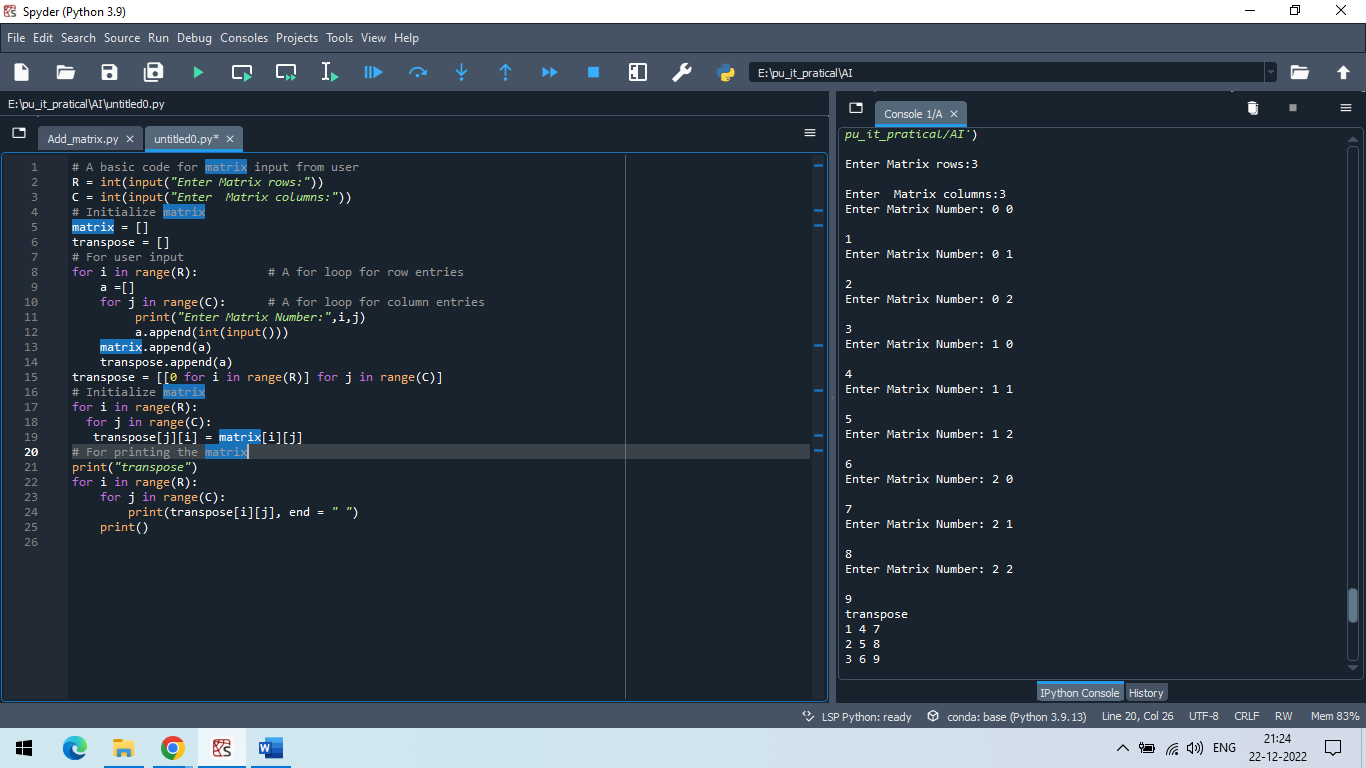
print("transpose")

for i in range(R):

for j in range(C):

print(transpose[i][j], end = " ")

print()

**Output:**  


**PRACTICAL-1**

**AIM: Write a program in prolog to implement simple facts and Queries.**

**Code:**

domains

x,y = symbol

predicates

father(x,y)

mother(x,y)

grandfather(x,y)

grandmother(x,y)

clauses

mother(a,b).

mother(b,c).

mother(m,q).

father(p,q).

father(q,r).

father(f,b).

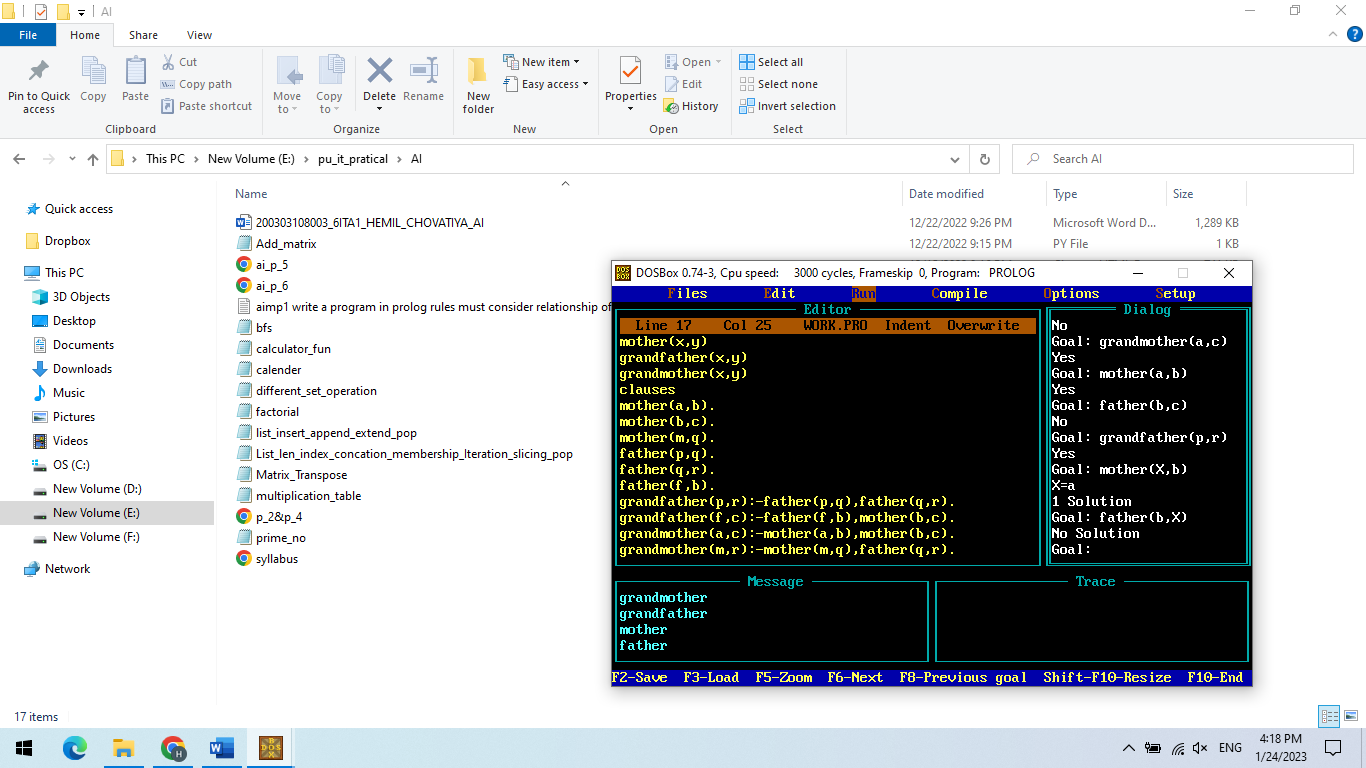
grandfather(p,r):-father(p,q),father(q,r).

grandfather(f,c):-father(f,b),mother(b,c).

grandmother(a,c):-mother(a,b),mother(b,c).

grandmother(m,r):-mother(m,q), father(q,r).

**Output:**



**PRACTICAL-7**

**AIM:** **Write a python program to implement Breadth First Search Traversal?**

**Code:**

from collections import defaultdict

class Graph\_bfs:

def \_\_init\_\_(self):

self.graph\_dict = defaultdict(list)

def edge(self, From, To):

self.graph\_dict[From].append(To)

def bfs(self, start):

visited\_node = [False] \* (len(self.graph\_dict))

queue1 = []

queue1.append(start)

visited\_node[start] = True

while queue1:

start = queue1.pop(0)

print(start, end=" / ")

for i in self.graph\_dict[start]:

if visited\_node[i] == False:

queue1.append(i)

visited\_node[i] = True

b1 = Graph\_bfs()

print('Enter path from vertex 1 and vertex 2 :- ')

while(1):

new = int(input('u want to add path? 1(yes)/0(no) :- '))

k = bool(new)

if(k == False):

break

key = int(input('Enter the vertex 1 :- '))

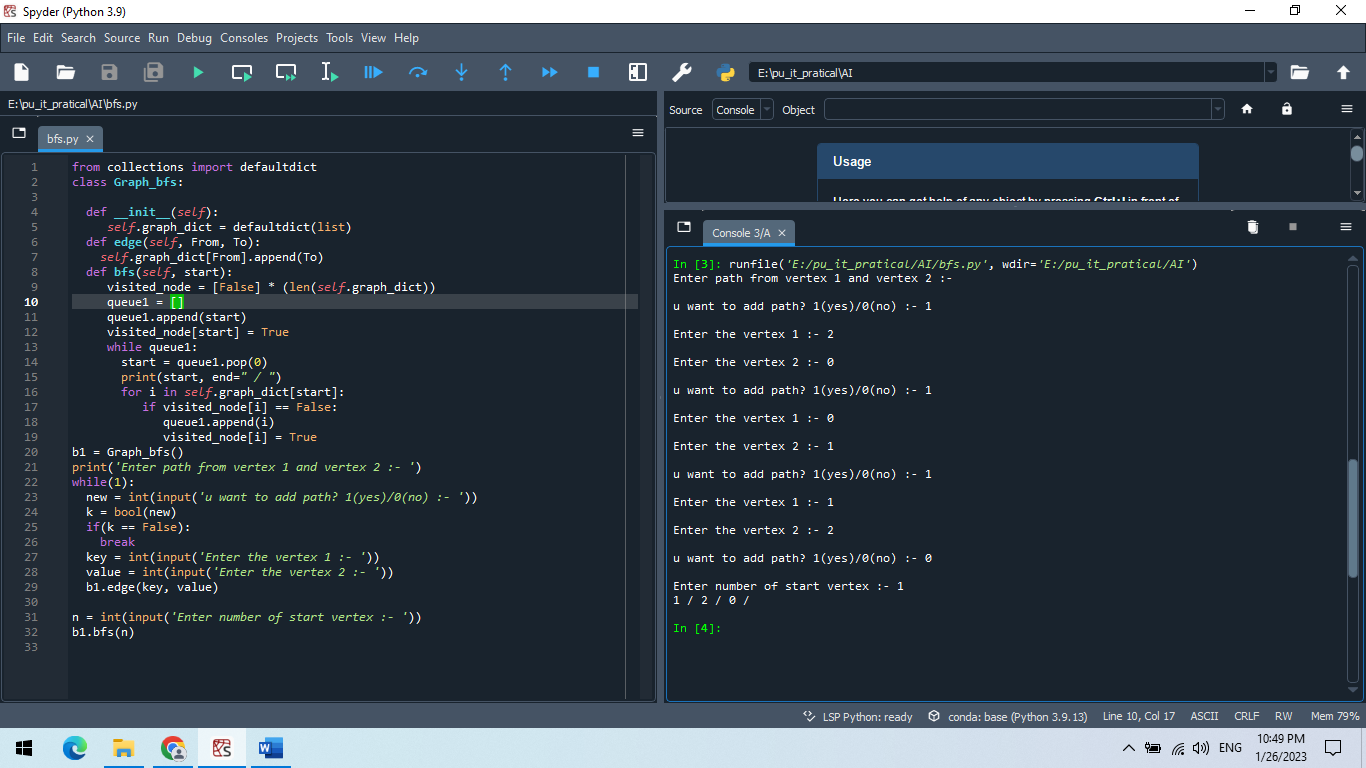
value = int(input('Enter the vertex 2 :- '))

b1.edge(key, value)

n = int(input('Enter number of start vertex :- '))

b1.bfs(n)

**Output:**



**PRACTICAL-8**

**AIM:** **Write a python program to implement Water Jug Problem?**

**Code:**

from collections import defaultdict

jug1, jug2, aim = 4, 3, 2

visited = defaultdict(lambda: False)

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))))

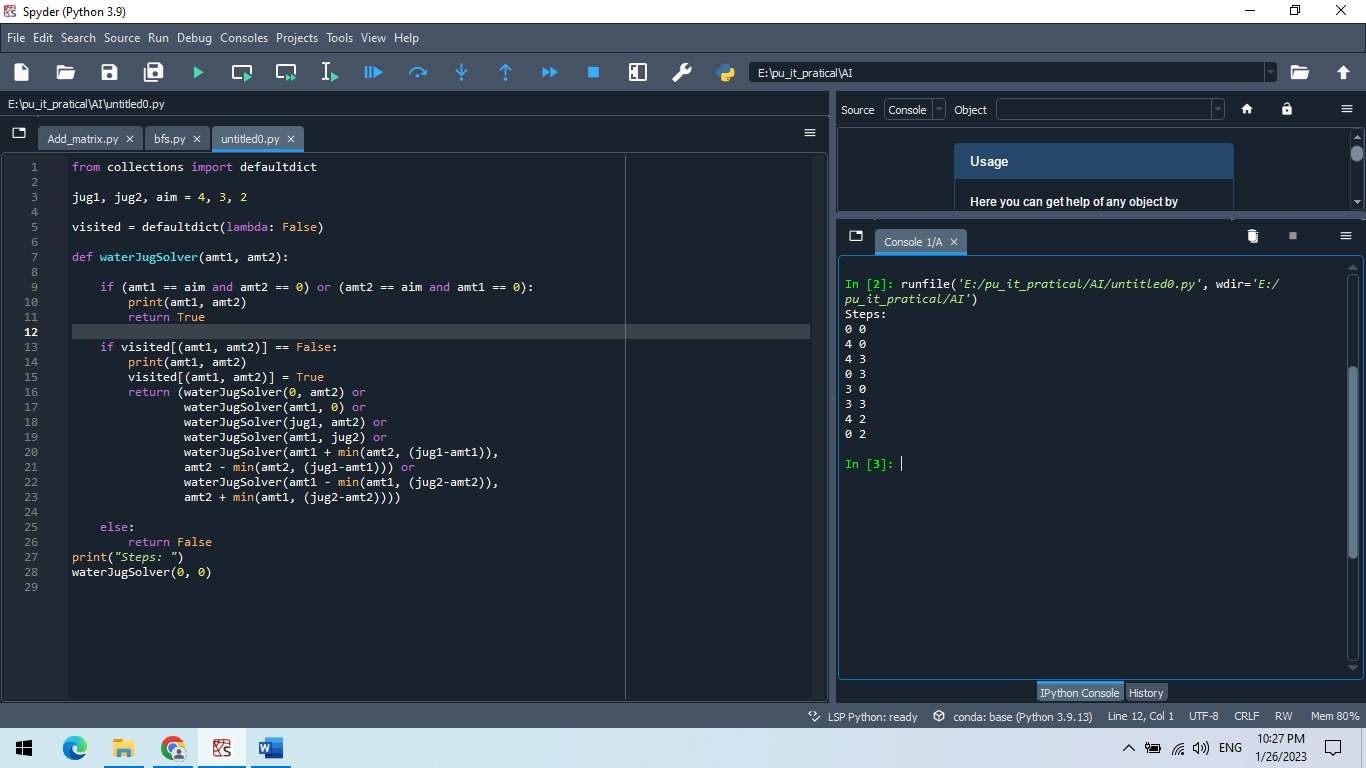
else:

return False

print("Steps: ")

waterJugSolver(0, 0)

**Output:**



**PRACTICAL-9**

**AIM:** **Write a program to implement Tic-Tac-Toe game using python.**

**Code:**

import random

class TicTacToe:

def \_\_init\_\_(self):

self.board = []

def create\_board(self):

for i in range(3):

row = []

for j in range(3):

row.append('-')

self.board.append(row)

def get\_random\_first\_player(self):

return random.randint(0, 1)

def fix\_spot(self, row, col, player):

self.board[row][col] = player

def is\_player\_win(self, player):

win = None

n = len(self.board)

for i in range(n):

win = True

for j in range(n):

if self.board[i][j] != player:

win = False

break

if win:

return win

for i in range(n):

win = True

for j in range(n):

if self.board[j][i] != player:

win = False

break

if win:

return win

win = True

for i in range(n):

if self.board[i][i] != player:

win = False

break

if win:

return win

win = True

for i in range(n):

if self.board[i][n - 1 - i] != player:

win = False

break

if win:

return win

return False

for row in self.board:

for item in row:

if item == '-':

return False

return True

def is\_board\_filled(self):

for row in self.board:

for item in row:

if item == '-':

return False

return True

def swap\_player\_turn(self, player):

return 'X' if player == 'O' else 'O'

def show\_board(self):

for row in self.board:

for item in row:

print(item, end=" ")

print()

def start(self):

self.create\_board()

player = 'X' if self.get\_random\_first\_player() == 1 else 'O'

while True:

print(f"Player {player} turn")

self.show\_board()

row, col = list(

map(int, input("Enter row and column numbers to fix spot: ").split()))

print()

self.fix\_spot(row - 1, col - 1, player)

if self.is\_player\_win(player):

print(f"Player {player} wins the game!")

break

if self.is\_board\_filled():

print("Match Draw!")

break

player = self.swap\_player\_turn(player)

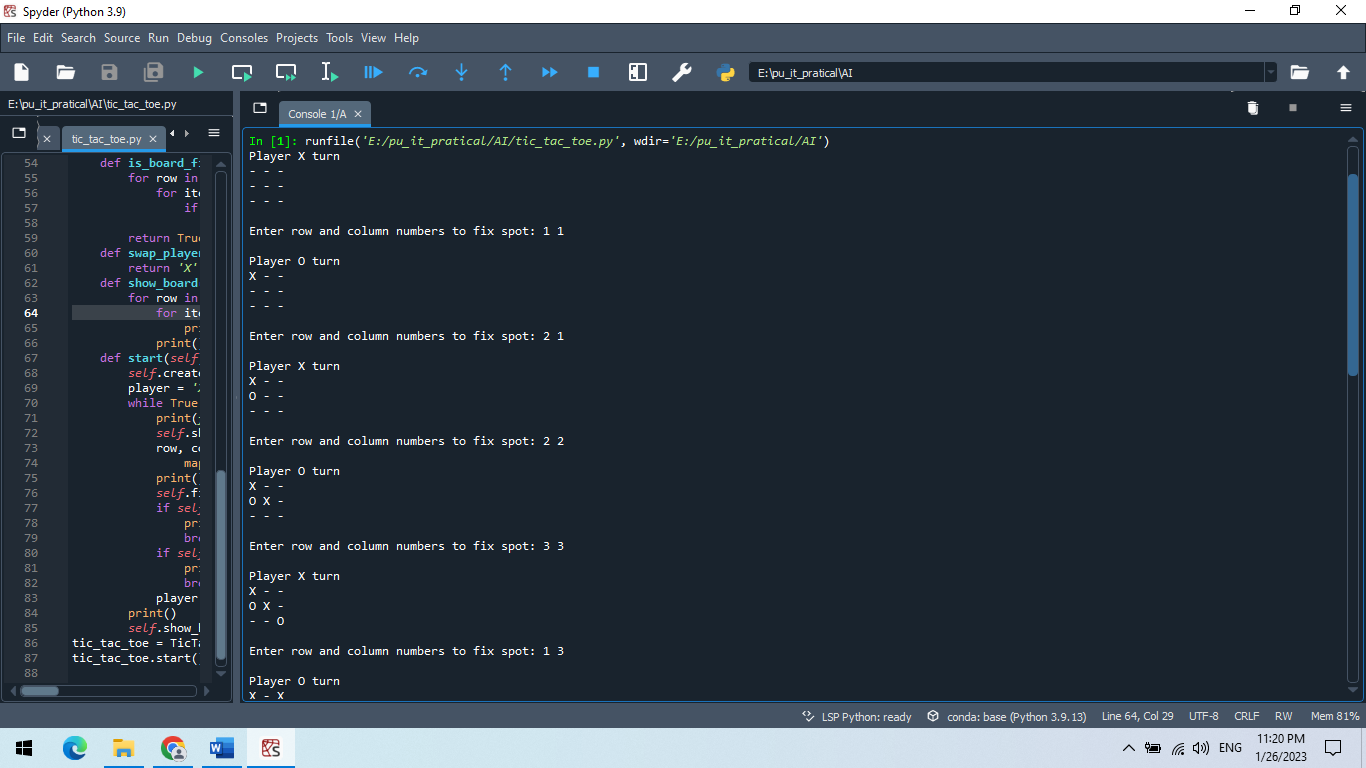
print()

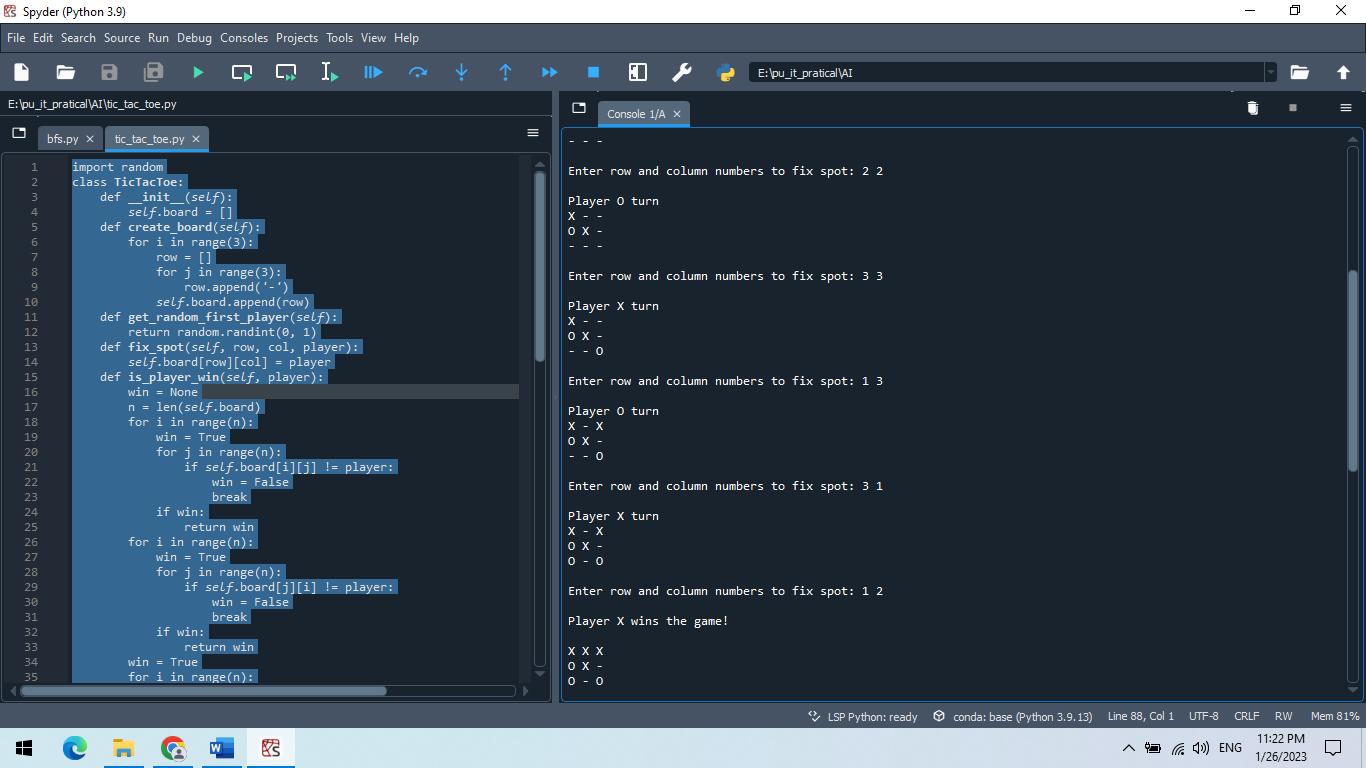
self.show\_board()

tic\_tac\_toe = TicTacToe()

tic\_tac\_toe.start()

**Output:**





**PRACTICAL-3**

**AIM: Write a python program to implement simple Chatbot.**

**Code a:**

import random

hello\_msg=["hello","hii","hi","how are you"]

bye\_msg=["bye","by","see you later","nothing much"]

while True:

ip\_msg=input()

if ip\_msg.lower() in hello\_msg:

reply\_msg=["hyy sir","how can i help you"]

print(random.choice(reply\_msg))

elif ip\_msg.lower() in bye\_msg:

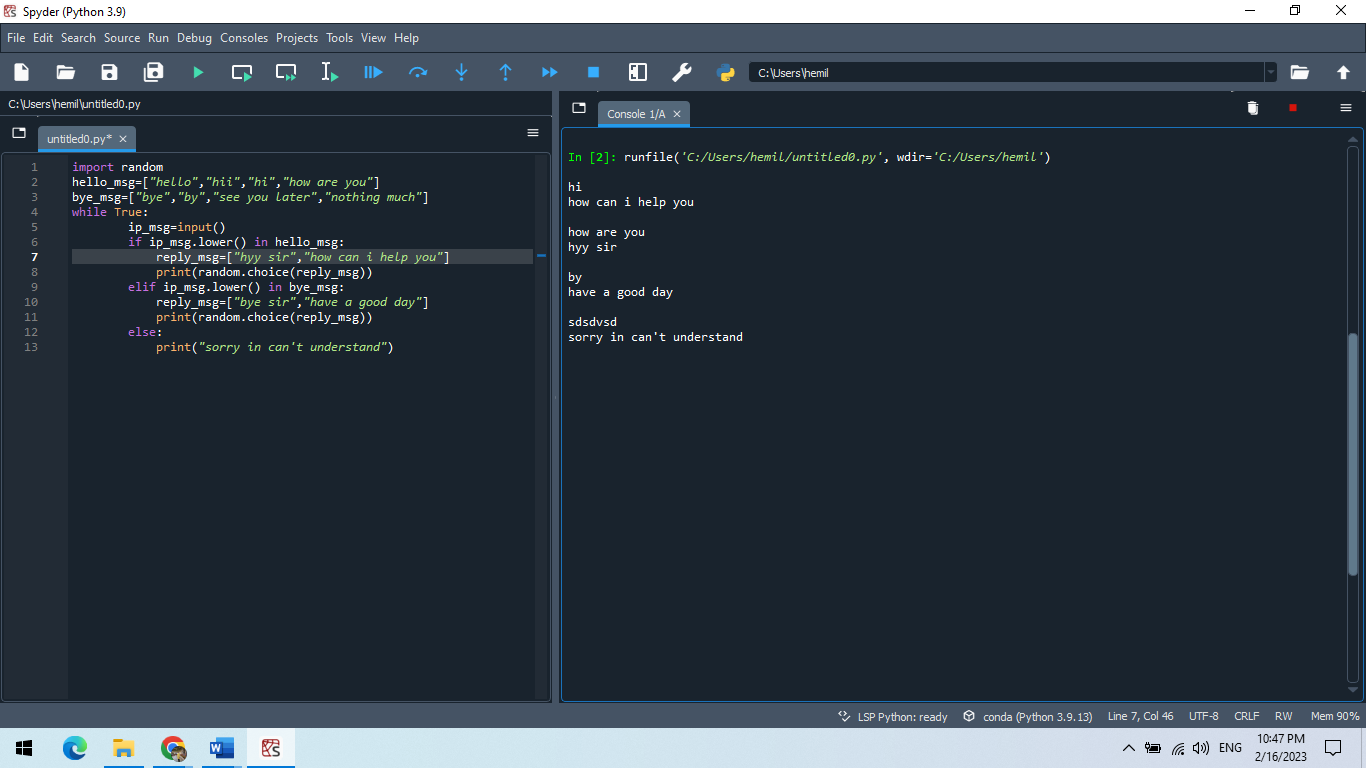
reply\_msg=["bye sir","have a good day"]

print(random.choice(reply\_msg))

else:

print("sorry in can't understand")from collections import defaultdict

**Output:**



**PRACTICAL-10**

**AIM:** **a)Write a python program to remove stop words for a given passage from a text file using NLTK?**

**b)Write a python program to implement stemming for a given sentence using NLTK?**

**c)Write a python program to POS (Parts of Speech) tagging for the give sentence using NLTK?**

**Code a:**

from collections import defaultdict

**Output:**

**Code b:**

from collections import defaultdict

**Output:**

**Code c:**

from collections import defaultdict

**Output:**

**PRACTICAL-11**

**AIM:a)** **Write a python program to implement Lemmatization using NLTK? B)Write a python program to for Text Classification for the give sentence using NLTK?**

**Code a:**

from collections import defaultdict

**Output:**

**Code b:**

from collections import defaultdict

**Output:**