Pgm:1

#python code for Breadth First Search (BFS)

import matplotlib.pyplot as plt

import networkx as nx

graph = {

'5' : ['3','7'],

'3' : ['2', '4'],

'7' : ['8'],

'2' : [],

'4' : ['8'],

'8' : []

}

visited = [] # List for visited nodes.

queue = [] #Initialize a queue

def bfs(visited, graph, node): #function for BFS

visited.append(node)

queue.append(node)

while queue: # Creating loop to visit each node

m = queue.pop(0)

print (m, end = " ")

for neighbour in graph[m]:

if neighbour not in visited:

visited.append(neighbour)

queue.append(neighbour)

# Driver Code

print("Following is the Breadth-First Search")

bfs(visited, graph, '5') # function calling

# Visualizing the graph

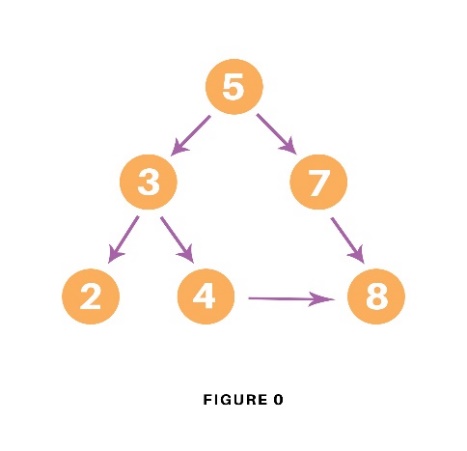
G = nx.Graph(graph)

pos = nx.spring\_layout(G) # Positions for all nodes

nx.draw(G, pos, with\_labels=True, node\_size=1000, node\_color="lightblue", font\_size=12, font\_weight="bold")

plt.title("Graph Visualization")

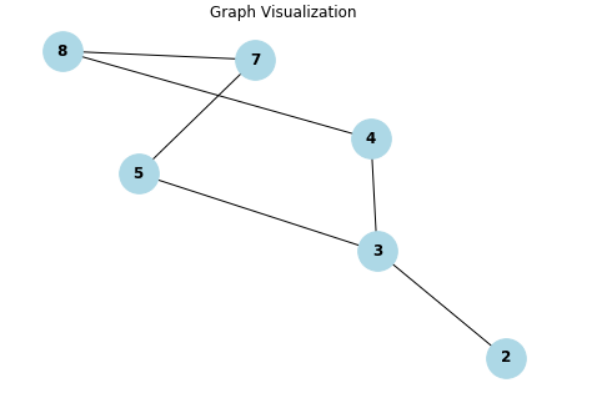
plt.show()



OUTPUT:

Following is the Breadth-First Search

5 3 7 2 4 8



Pgm2:

# Using a Python dictionary to act as an adjacency list

#python code for Depth First Search(DFS)

import matplotlib.pyplot as plt

import networkx as nx

graph = {

'5' : ['3','7'],

'3' : ['2', '4'],

'7' : ['8'],

'2' : [],

'4' : ['8'],

'8' : []

}

visited = set() # Set to keep track of visited nodes of graph.

def dfs(visited, graph, node): #function for dfs

if node not in visited:

print (node)

visited.add(node)

for neighbour in graph[node]:

dfs(visited, graph, neighbour)

# Driver Code

print("Following is the Depth-First Search")

dfs(visited, graph, '5')

# Visualizing the graph

G = nx.Graph(graph)

pos = nx.spring\_layout(G) # Positions for all nodes

nx.draw(G, pos, with\_labels=True, node\_size=1000, node\_color="lightblue", font\_size=12, font\_weight="bold")

plt.title("Graph Visualization")

plt.show()

OUTPUT:

Following is the Depth-First Search

5

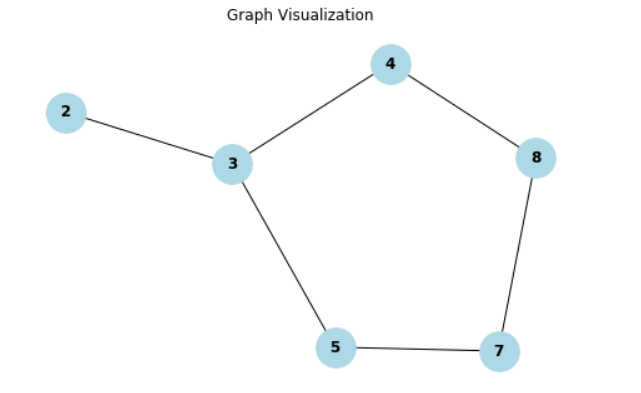
3

2

4

8

7



Pgm 3:

#Python code for TOWER OF HANOI

# Creating a recursive function

def tower\_of\_hanoi(disks, source, auxiliary, target):

if (disks == 1):

print('Move disk 1 from rod {} to rod {}.'.format(source, target))

return

# function call itself

tower\_of\_hanoi(disks - 1, source, target, auxiliary)

print('Move disk {} from rod {} to rod {}.'.format(disks, source, target))

tower\_of\_hanoi(disks - 1, auxiliary, source, target)

disks = int(input('Enter the number of disks: '))

# We are referring source as A, auxiliary as B, and target as C

tower\_of\_hanoi(disks, 'A', 'B', 'C') # Calling the function

OUTPUT:

Enter the number of disks: 3

Move disk 1 from rod A to rod C.

Move disk 2 from rod A to rod B.

Move disk 1 from rod C to rod B.

Move disk 3 from rod A to rod C.

Move disk 1 from rod B to rod A.

Move disk 2 from rod B to rod C.

Move disk 1 from rod A to rod C.

Pgm 4:

#python code for creating a Simple AI ChatBot

print("How are you?")

print("Are you working?")

print("What is your name?")

print("what did you do yesterday?")

print("Quit")

while True:

question = input("Enter one question from above list:")

question = question.lower()

if question in ['hi']:

print("Hello")

elif question in ['how are you?','how do you do?']:

print("I am fine")

elif question in ['are you working?','are you doing any job?']:

print("yes. I'am working in KLU")

elif question in ['what is your name?']:

print("My name is Emilia")

name=input("Enter your name?")

print("Nice name and Nice meeting you", name)

elif question in ['what did you do yesterday?']:

print("I saw Bahubali 5 times")

elif question in ['quit']:

break

else:

print("I don't understand what you said")

Pgm 5:

#python code for linear regression

import numpy as np

import matplotlib.pyplot as pt

from sklearn.linear\_model import LinearRegression

x=np.array([1,2,3,4,5,6,-7,8,11,25,30]).reshape(-1,1)

y=np.array([2,3,4,5,6,7,8,9,10,12,15])

model=LinearRegression()

model.fit(x,y)

y\_pred=model.predict(x)

pt.scatter(x,y,color='red',label='Actual Data')

pt.plot(x,y\_pred,color='blue',label='Linear Regression Line ')

pt.title('Line Of Regression')

pt.xlabel('x')

pt.ylabel('y')

pt.legend()

pt.show()

print('Intercept:',model.intercept\_)

print('Slope:',model.coef\_[0])