In [6]:

```
#N QWEEN BILL BOARD APPLICATION
############################

def solveNQueens(board):
    n = len(board)

    def isSafe(row, col):

        for i in range(col):
            if board[row][i] == 1:
                return False
        for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
            if board[i][j] == 1:
                return False
        for i, j in zip(range(row, n, 1), range(col, -1, -1)):
            if board[i][j] == 1:
                return False
        return True

    def backtrack(col):

        if col == n:
            return True

        for i in range(n):
            if isSafe(i, col):
                board[i][col] = 1

                if backtrack(col + 1):
                    return True

                board[i][col] = 0


        return False


    for i in range(n):
        for j in range(n):
            if board[i][j] == 0:

                if backtrack(j):

                    for row in board:
                        print(row)
                    return True
                else:
                    return False
```

In [8]:
```python
###CAMEL BANANA
def camel_banana(n_bananas):

    if n_bananas <= 0 or n_bananas % 2 == 1:
        return "Invalid number of bananas"


    bananas_per_trip = n_bananas // 2

    total_trips = 3 * bananas_per_trip


    return total_trips
```

In [ ]:
```python
####CRYPTHEMETIC PUZZLE
def solve_cryptarithmetic(puzzle):

    letters = set(puzzle.replace(' ', ''))

    if len(letters) > 10:
        return "Invalid puzzle: More than 10 unique letters"


    permutations = itertools.permutations(range(10), len(letters))


    for perm in permutations:
        mapping = dict(zip(letters, perm))
        if eval(puzzle.translate(mapping)) == True:
            return mapping

    return "No solution found"
```

In [12]:

```python
#MAP COLORING aplication is scheduling events


import numpy as np

def schedule(events):

    event_colors = np.zeros(len(events), dtype=int)


    adj_matrix = np.zeros((len(events), len(events)), dtype=int)
    for i in range(len(events)):
        for j in range(i+1, len(events)):
            if events[i][1] > events[j][0] and events[j][1] > events[i][0]:
                adj_matrix[i][j] = 1
                adj_matrix[j][i] = 1


    colors = list(range(len(events)))

    if backtrack(adj_matrix, event_colors, colors, 0):

        return event_colors
    else:

        return None

def backtrack(adj_matrix, event_colors, colors, event_idx):

    if event_idx == len(event_colors):
        return True

    for color in colors:

        if is_color_valid(adj_matrix, event_colors, event_idx, color):

            event_colors[event_idx] = color


            if backtrack(adj_matrix, event_colors, colors, event_idx + 1):
                return True


            event_colors[event_idx] = 0

    return False

def is_color_valid(adj_matrix, event_colors, event_idx, color):

    for i in range(adj_matrix.shape[0]):
        if adj_matrix[event_idx][i] == 1 and event_colors[i] == color:
            return False


    return True
```

In [13]:
```
####BFS APPLICATION---SHOORTEST DISTANCE


from collections import deque

def bfs_shortest_path(graph, start, end):

    queue = deque([(start, [start])])
    visited = set([start])


    while queue:

        node, path = queue.popleft()


        if node == end:
            return path


        for neighbor in graph[node]:
            if neighbor not in visited:

                queue.append((neighbor, path + [neighbor]))
                visited.add(neighbor)


    return None
```

In [14]:
```
#DFS APPLICATION BIPARTITE A bipartite graph is a graph in which the vertices


def dfs_bipartite(graph, start, colors):

    for neighbor in graph[start]:
        if neighbor in colors:

            if colors[neighbor] == colors[start]:
                return False
        else:

            colors[neighbor] = 1 - colors[start]
            if not dfs_bipartite(graph, neighbor, colors):
                return False


    return True
```

In [15]:

```python
# BEST FIRST SEARCH ---MIN SPANNING TREE

import heapq

def best_first_search(graph, start):

    heap = [(0, start, None)]


    visited = set()


    tree = {}

    while heap:

        weight, node, parent = heapq.heappop(heap)


        if node in visited:
            continue


        visited.add(node)
        if parent is not None:
            tree[(parent, node)] = weight


        for neighbor, weight in graph[node].items():

            if neighbor not in visited:
                heapq.heappush(heap, (weight, neighbor, node))


    return tree
```

In [23]:

```
##A STAR----8 PUZZLE
import heapq

def best_first_search(start_state, goal_state):

    def heuristic(state):
        distance = 0
        for i in range(3):
            for j in range(3):
                if state[i][j] != 0:
                    x, y = divmod(state[i][j] - 1, 3)
                    distance += abs(x - i) + abs(y - j)
        return distance


    heap = [(heuristic(start_state), 0, start_state)]


    visited = set()


    while heap:

        _, cost, state = heapq.heapp
```

In [24]:

```python
#MINIMAX-----ZEROSUM--ROCK-PAPER -SCISSORS


import random

def minimax(strategy):

    payoff = [[0, -1, 1], [1, 0, -1], [-1, 1, 0]]


    total_score = 0

    # Play multiple rounds of the game
    for _ in range(NUM_ROUNDS):

        opponent_move = random.randint(0, 2)


        player_move = strategy(opponent_move)


        score = payoff[player_move][opponent_move]


        total_score += score


    return total_score / NUM_ROUNDS
```

In [ ]:

```
##alpha beta --chess game tree

def alphabeta(board, player, depth, alpha, beta):
    # Check if the game is over or the maximum depth has been reached
    if game_over(board) or depth == 0:
        return evaluate(board, player)

    # Initialize the best score to the worst possible score for the player
    best_score = -float("inf") if player == MAX_PLAYER else float("inf")

    # Loop through all possible moves
    for move in get_moves(board):
        # Apply the move to the board
        new_board = make_move(board, move, player)

        # Recursively call the Alpha-beta function to get the score of the res
        score = alphabeta(new_board, switch_player(player), depth - 1, alpha,

        # Update the best score based on the player
        if player == MAX_PLAYER:
            best_score = max(best_score, score)
            alpha = max(alpha, best_score)
            if beta <= alpha:
                break
        else:
            best_score = min(best_score, score)
            beta = min(beta, best_score)
            if beta <= alpha:
                break

    # Return the best score
    return best_score
```

In [ ]:
```python
#Logic programming
!pip install kanren
from kanren import run, eq, membero, var, Relation

# Define a friend relation
friend = Relation()

# Define some people
alice, bob, charlie, diana = var(), var(), var(), var()

# Add some friends
facts = (
    friend(alice, bob),
    friend(bob, alice),
    friend(charlie, diana),
    friend(diana, charlie),
    friend(alice, charlie),
    friend(charlie, alice)
)

# Query the friend relation
query = friend(alice, bob)
result = run(1, (bob,), query)

print(result)
```

In [29]:
```python
#supervised---logistic regression-iris dataset


from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

# Load the iris dataset
iris = load_iris()
X = iris.data[:, :2]
y = iris.target

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

# Create a logistic regression model and fit it to the training data
logreg = LogisticRegression()
logreg.fit(X_train, y_train)

# Predict the test set labels and calculate the accuracy
accuracy = logreg.score(X_test, y_test)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 0.79
```

In [41]:

```python
#supervised---Logistic regression-custom csv file  dataset

import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# Load the dataset from a CSV file
df = pd.read_csv("  path")

# Split the data into training and test sets
X = df.drop("target_variable", axis=1)
y = df["target_variable"]
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

# Create a logistic regression model and fit it to the training data
logreg = LogisticRegression()
logreg.fit(X_train, y_train)

# Predict the test set labels and calculate the accuracy
accuracy = logreg.score(X_test, y_test)
print(f"Accuracy: {accuracy:.2f}")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[41], line 8
      5 from sklearn.model_selection import train_test_split
      7 # Load the dataset from a CSV file
----> 8 df = pd.read_csv("")
     10 # Split the data into training and test sets
     11 X = df.drop("target_variable", axis=1)

File ~\anaconda3\lib\site-packages\pandas\util\_decorators.py:211, in deprecate_kwarg.<locals>._deprecate_kwarg.<locals>.wrapper(*args, **kwargs)
    209     else:
    210         kwargs[new_arg_name] = new_arg_value
--> 211 return func(*args, **kwargs)

File ~\anaconda3\lib\site-packages\pandas\util\_decorators.py:331, in deprecate_nonkeyword_arguments.<locals>.decorate.<locals>.wrapper(*args, **kwargs)
    325 if len(args) > num_allow_args:
    326     warnings.warn(
    327         msg.format(arguments=_format_argument_list(allow_args)),
    328         FutureWarning,
    329         stacklevel=find_stack_level(),
    330     )
--> 331 return func(*args, **kwargs)

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:950, in read_csv(filepath_or_buffer, sep, delimiter, header, names, index_col, usecols, squeeze, prefix, mangle_dupe_cols, dtype, engine, converters, true_values, false_values, skipinitialspace, skiprows, skipfooter, nrows, na_values, keep_default_na, na_filter, verbose, skip_blank_lines, parse_dates, infer_datetime_format, keep_date_col, date_parser, dayfirst, cache_dates, iterator, chunksize, compression, thousands, decimal, lineterminator, quotechar, quoting, doublequote, escapechar, comment, encoding, encoding_errors, dialect, error_bad_lines, warn_bad_lines, on_bad_lines, delim_whitespace, low_memory, memory_map, float_precision, storage_options)
    935 kwds_defaults = _refine_defaults_read(
    936     dialect,
    937     delimiter,
   (...)
    946     defaults={"delimiter": ","},
    947 )
    948 kwds.update(kwds_defaults)
--> 950 return _read(filepath_or_buffer, kwds)

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:605, in _read(filepath_or_buffer, kwds)
    602 _validate_names(kwds.get("names", None))
    604 # Create the parser.
--> 605 parser = TextFileReader(filepath_or_buffer, **kwds)
    607 if chunksize or iterator:
    608     return parser

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:1442, in TextFileReader.__init__(self, f, engine, **kwds)
   1439     self.options["has_index_names"] = kwds["has_index_names"]
   1441 self.handles: IOHandles | None = None
-> 1442 self._engine = self._make_engine(f, self.engine)
```

```
File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:1735, in TextFileReader._make_engine(self, f, engine)
   1733     if "b" not in mode:
   1734         mode += "b"
-> 1735 self.handles = get_handle(
   1736     f,
   1737     mode,
   1738     encoding=self.options.get("encoding", None),
   1739     compression=self.options.get("compression", None),
   1740     memory_map=self.options.get("memory_map", False),
   1741     is_text=is_text,
   1742     errors=self.options.get("encoding_errors", "strict"),
   1743     storage_options=self.options.get("storage_options", None),
   1744 )
   1745 assert self.handles is not None
   1746 f = self.handles.handle

File ~\anaconda3\lib\site-packages\pandas\io\common.py:856, in get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text, errors, storage_options)
    851 elif isinstance(handle, str):
    852     # Check whether the filename is to be opened in binary mode.
    853     # Binary mode does not support 'encoding' and 'newline'.
    854     if ioargs.encoding and "b" not in ioargs.mode:
    855         # Encoding
--> 856         handle = open(
    857             handle,
    858             ioargs.mode,
    859             encoding=ioargs.encoding,
    860             errors=errors,
    861             newline="",
    862         )
    863     else:
    864         # Binary mode
    865         handle = open(handle, ioargs.mode)

FileNotFoundError: [Errno 2] No such file or directory: ''
```

In [31]:
```
#unsupervised --kmeans

from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt

# Generate some sample data
X, y = make_blobs(n_samples=1000, centers=3, random_state=42)

# Create a K-means clustering model with 3 clusters
kmeans = KMeans(n_clusters=3)

# Fit the model to the data
kmeans.fit(X)

# Predict the cluster labels for the data
labels = kmeans.predict(X)

# Plot the data points with different colors representing different clusters
plt.scatter(X[:, 0], X[:, 1], c=labels)
plt.show()
```

In [32]:
```
import nltk
import nltk.corpus
#Tokenization
from nltk.tokenize import word_tokenize
chess = "Samay Raina is the best chess streamer in the world"
nltk.download('punkt')
word_tokenize(chess) #Tokenization
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\91967\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
```

Out[32]:
```
['Samay',
 'Raina',
 'is',
 'the',
 'best',
 'chess',
 'streamer',
 'in',
 'the',
 'world']
```

In [33]:
```
#sentence tokenizer
from nltk.tokenize import sent_tokenize
chess2 = "Samay Raina is the best chess streamer in the world. Sagar Sh ah is
sent_tokenize(chess2)
```

Out[33]:
```
['Samay Raina is the best chess streamer in the world.',
 'Sagar Sh ah is  the best chess coach in the world']
```

In [34]:
```
#Checking the number of tokens
len(word_tokenize(chess))
  #Checking the number of tokens
len(word_tokenize(chess))
```

Out[34]:
```
10
```

In [35]:
```
#bigrams and n-grams
astronaut = "Can anybody hear me or am I talking to myself? My mind is  runnin
astronaut_token=(word_tokenize(astronaut))#bigrams and n-grams
astronaut = "Can anybody hear me or am I talking to myself? My mind is  runnin
astronaut_token=(word_tokenize(astronaut))#bigrams and n-grams
```

In [36]:
```
list(nltk.trigrams(astronaut_token))
```

Out[36]:
```
[('Can', 'anybody', 'hear'),
 ('anybody', 'hear', 'me'),
 ('hear', 'me', 'or'),
 ('me', 'or', 'am'),
 ('or', 'am', 'I'),
 ('am', 'I', 'talking'),
 ('I', 'talking', 'to'),
 ('talking', 'to', 'myself'),
 ('to', 'myself', '?'),
 ('myself', '?', 'My'),
 ('?', 'My', 'mind'),
 ('My', 'mind', 'is'),
 ('mind', 'is', 'running'),
 ('is', 'running', 'empty'),
 ('running', 'empty', 'in'),
 ('empty', 'in', 'the'),
 ('in', 'the', 'search'),
 ('the', 'search', 'for'),
 ('search', 'for', 'someone'),
 ('for', 'someone', 'else')]
```

In [ ]:
```

```

In [37]:
```
#Stemming
from nltk.stem import PorterStemmer
my_stem = PorterStemmer()
my_stem.stem("eating")
my_stem.stem("going")
my_stem.stem("shopping")
```

Out[37]:
```
'shop'
```

In [38]:
```
#pos-tagging
tom ="Tom Hanks is the best actor in the world"
tom_token = word_tokenize(tom)
nltk.download('averaged_perceptron_tagger')
nltk.pos_tag(tom_token)
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\91967\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping taggers\averaged_perceptron_tagger.zip.
```

Out[38]:
```
[('Tom', 'NNP'),
 ('Hanks', 'NNP'),
 ('is', 'VBZ'),
 ('the', 'DT'),
 ('best', 'JJS'),
 ('actor', 'NN'),
 ('in', 'IN'),
 ('the', 'DT'),
 ('world', 'NN')]
```

In [39]:
```
#Named entity recognition
from nltk import ne_chunk
president = "Barack Obama was the 44th President of America"
president_token = word_tokenize(president)
president_pos = nltk.pos_tag(president_token)
nltk.download('maxent_ne_chunker')
nltk.download('words')
print(ne_chunk(president_pos))
```

```
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     C:\Users\91967\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping chunkers\maxent_ne_chunker.zip.
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\91967\AppData\Roaming\nltk_data...

(S
  (PERSON Barack/NNP)
  (PERSON Obama/NNP)
  was/VBD
  the/DT
  44th/JJ
  President/NNP
  of/IN
  (GPE America/NNP))

[nltk_data]   Unzipping corpora\words.zip.
```

In [40]:
```
!pip install gTTS
from gtts import gTTS
from IPython.display import Audio
tts = gTTS('Hello everybody, How are you')
tts.save('1.wav')
sound_file = '1.wav'
Audio(sound_file, autoplay=True)
```

```
Collecting gTTS
  Downloading gTTS-2.3.2-py3-none-any.whl (28 kB)
Requirement already satisfied: click<8.2,>=7.1 in c:\users\91967\anaconda3\li
b\site-packages (from gTTS) (8.0.4)
Requirement already satisfied: requests<3,>=2.27 in c:\users\91967\anaconda3
\lib\site-packages (from gTTS) (2.28.1)
Requirement already satisfied: colorama in c:\users\91967\anaconda3\lib\site-
packages (from click<8.2,>=7.1->gTTS) (0.4.6)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\91967\anacon
da3\lib\site-packages (from requests<3,>=2.27->gTTS) (1.26.14)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\91967\anaconda3
\lib\site-packages (from requests<3,>=2.27->gTTS) (2022.12.7)
Requirement already satisfied: idna<4,>=2.5 in c:\users\91967\anaconda3\lib\s
ite-packages (from requests<3,>=2.27->gTTS) (3.4)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\91967\ana
conda3\lib\site-packages (from requests<3,>=2.27->gTTS) (2.0.4)
Installing collected packages: gTTS
Successfully installed gTTS-2.3.2
```

Out[40]:

0:00 / 0:00

In [ ]:

```
import nltk
import nltk.corpus
#Tokenization
from nltk.tokenize import word_tokenize
chess = "Samay Raina is the best chess streamer in the world"
nltk.download('punkt')
word_tokenize(chess) #Tokenization


#sentence tokenizer
from nltk.tokenize import sent_tokenize
chess2 = "Samay Raina is the best chess streamer in the world. Sagar Sh ah is
sent_tokenize(chess2)


#Checking the number of tokens
len(word_tokenize(chess))
  #Checking the number of tokens
len(word_tokenize(chess))


#bigrams and n-grams
astronaut = "Can anybody hear me or am I talking to myself? My mind is  runnin
astronaut_token=(word_tokenize(astronaut))#bigrams and n-grams
astronaut = "Can anybody hear me or am I talking to myself? My mind is  runnin
astronaut_token=(word_tokenize(astronaut))#bigrams and n-grams



#Stemming
from nltk.stem import PorterStemmer
my_stem = PorterStemmer()
my_stem.stem("eating")
my_stem.stem("going")
my_stem.stem("shopping")


#pos-tagging
tom ="Tom Hanks is the best actor in the world"
tom_token = word_tokenize(tom)
nltk.download('averaged_perceptron_tagger')
nltk.pos_tag(tom_token)



#Named entity recognition
from nltk import ne_chunk
president = "Barack Obama was the 44th President of America"
president_token = word_tokenize(president)
president_pos = nltk.pos_tag(president_token)
nltk.download('maxent_ne_chunker')
nltk.download('words')
print(ne_chunk(president_pos))



!pip install gTTS
```

```
from gtts import gTTS
from IPython.display import Audio
tts = gTTS('Hello everybody, How are you')
tts.save('1.wav')
sound_file = '1.wav'
Audio(sound_file, autoplay=True)
```