Program

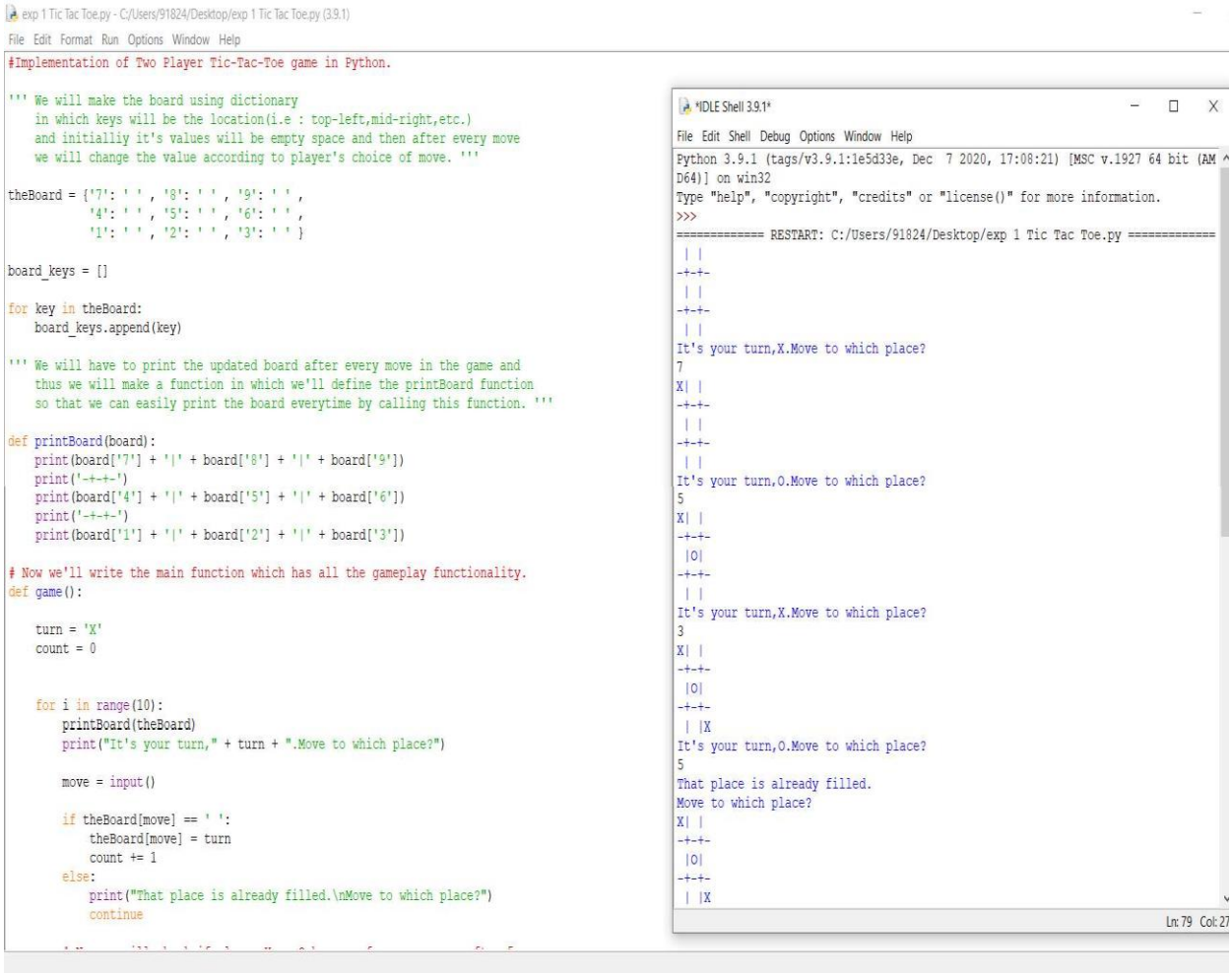
```
#Implementation of Two Player Tic-Tac-Toe game in Python.
theBoard = {'7': ' ' , '8': ' ' , '9': ' ' ,
          '4': ' ' , '5': ' ' , '6': ' ' ,
          '1': ' ' , '2': ' ' , '3': ' ' }
board_keys = []
for key in theBoard:
   board_keys.append(key)
def printBoard(board):
print(board['7'] + '|' + board['8'] + '|' + board['9'])
print('-+-+-')
print(board['4'] + '|' + board['5'] + '|' + board['6'])
print('-+-+-')
print(board['1'] + '|' + board['2'] + '|' + board['3'])
def game():
turn = 'X'
count = 0
for i in range(10):
printBoard(theBoard)
print("It's your turn," + turn + ".Move to which place?")

move = input()

if theBoard[move] == ' ':
theBoard[move] = turn
count += 1
```

```
else:
print("That place is already filled.\nMove to which place?")
continue

if count >= 5:
if theBoard['7'] == theBoard['8'] == theBoard['9'] != ' ': # across the top
printBoard(theBoard)
print("\nGame Over.\n")
print(" **** " +turn + " won. ****")
break
elif theBoard['4'] == theBoard['5'] == theBoard['6'] != ' ': # across the middle
printBoard(theBoard)
print("\nGame Over.\n")
print(" **** " +turn + " won. ****")
break
elif theBoard['1'] == theBoard['2'] == theBoard['3'] != ' ': # across the bottom
printBoard(theBoard)
print("\nGame Over.\n")
print(" **** " +turn + " won. ****")
break
elif theBoard['1'] == theBoard['4'] == theBoard['7'] != ' ': # down the left side
printBoard(theBoard)
print("\nGame Over.\n")
print(" **** " +turn + " won. ****")
break
elif theBoard['2'] == theBoard['5'] == theBoard['8'] != ' ': # down the middle
```

```
printBoard(theBoard)
print("\nGame Over.\n")
print(" **** " +turn + " won. ****")
break
elif theBoard['3'] == theBoard['6'] == theBoard['9'] != ' ': # down the right side
printBoard(theBoard)
print("\nGame Over.\n")
print(" **** " +turn + " won. ****")
break
elif theBoard['7'] == theBoard['5'] == theBoard['3'] != ' ': # diagonal
printBoard(theBoard)
print("\nGame Over.\n")
print(" **** " +turn + " won. ****")
break
elif theBoard['1'] == theBoard['5'] == theBoard['9'] != ' ': # diagonal
printBoard(theBoard)
print("\nGame Over.\n")
print(" **** " +turn + " won. ****")
break
if count == 9:
print("\nGame Over.\n")
print("It's a Tie!!")
if turn =='X':
turn = 'O'
else:
turn = 'X'
```

```
restart = input("Do want to play Again?(y/n)")
if restart == "y" or restart == "Y":
for key in board_keys:
theBoard[key] = " "
game()
if__name__== "_main_":
game()
```

Output:



exp 1 Tic Tac Toe.py - C:/Users/91824/Desktop/exp 1 Tic Tac Toe.py (3.9.1)

File Edit Format Run Options Window Help

```
#Implementation of Two Player Tic-Tac-Toe game in Python.

''' We will make the board using dictionary
    in which keys will be the location(i.e : top-left,mid-right,etc.)
    and initialliy it's values will be empty space and then after every move
    we will change the value according to player's choice of move. '''

theBoard = {'7': ' ' , '8': ' ' , '9': ' ' ,
            '4': ' ' , '5': ' ' , '6': ' ' ,
            '1': ' ' , '2': ' ' , '3': ' ' }

board_keys = []

for key in theBoard:
    board_keys.append(key)

''' We will have to print the updated board after every move in the game and
    thus we will make a function in which we'll define the printBoard function
    so that we can easily print the board everytime by calling this function. '''

def printBoard(board):
    print(board['7'] + '|' + board['8'] + '|' + board['9'])
    print('-+-+-')
    print(board['4'] + '|' + board['5'] + '|' + board['6'])
    print('-+-+-')
    print(board['1'] + '|' + board['2'] + '|' + board['3'])

# Now we'll write the main function which has all the gameplay functionality.
def game():

    turn = 'X'
    count = 0


    for i in range(10):
        printBoard(theBoard)
        print("It's your turn," + turn + ".Move to which place?")

        move = input()

        if theBoard[move] == ' ':
            theBoard[move] = turn
            count += 1
        else:
            print("That place is already filled.\nMove to which place?")
            continue
```

IDLE Shell 3.9.1

File Edit Shell Debug Options Window Help

```
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec  7 2020, 17:08:21) [MSC v.1927 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============ RESTART: C:/Users/91824/Desktop/exp 1 Tic Tac Toe.py ============
 | |
-+-+-
 | |
-+-+-
 | |
It's your turn,X.Move to which place?
7
X| |
-+-+-
 | |
-+-+-
 | |
It's your turn,O.Move to which place?
5
X| |
-+-+-
 |O|
-+-+-
 | |
It's your turn,X.Move to which place?
3
X| |
-+-+-
 |O|
-+-+-
 | |X
It's your turn,O.Move to which place?
5
That place is already filled.
Move to which place?
X| |
-+-+-
 |O|
-+-+-
 | |X
```

Ln: 79 Col: 27

Program

```
import random
import math

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

def index(item, seq):
if item in seq:
return seq.index(item)
else:
return -1

class EightPuzzle:

def__init__(self):
        # heuristic value
        self._hval = 0
        # search depth of current instance
        self._depth = 0
        # parent node in search path
        self._parent = None
        self.adj_matrix = []
for i in range(3):
            self.adj_matrix.append(_goal_state[i][:])

def__eq__(self, other):
if self.__class__!= other._class_:
return False
else:
return self.adj_matrix == other.adj_matrix

def__str__(self):
res = ''
for row in range(3):
res += ' '.join(map(str, self.adj_matrix[row]))
res += '\r\n'
return res

def _clone(self):
        p = EightPuzzle()
```

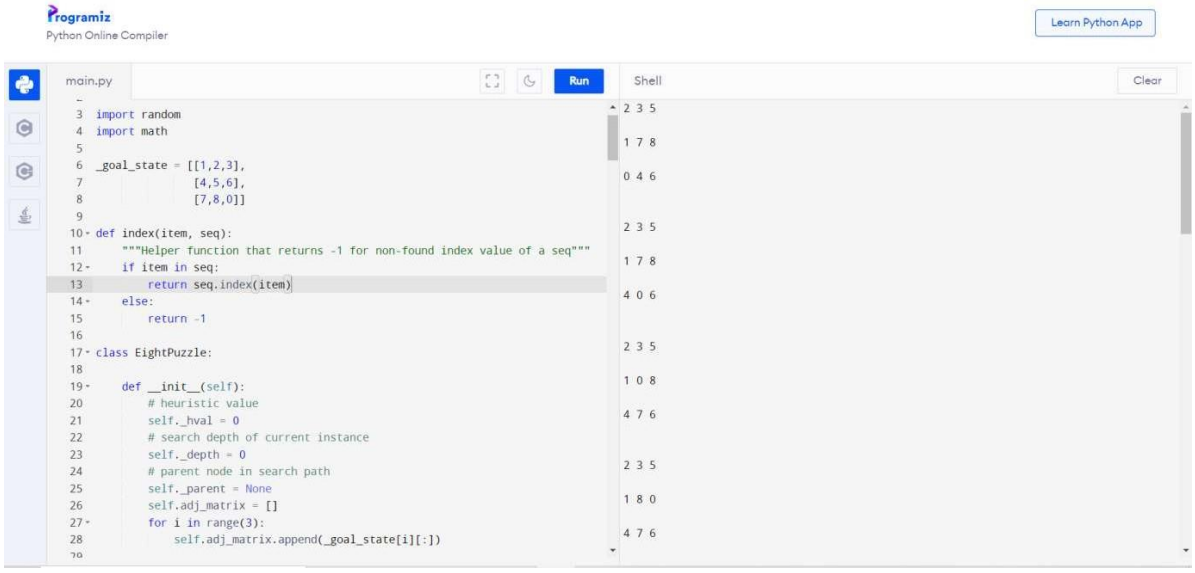
```
for i in range(3):
        p.adj_matrix[i] = self.adj_matrix[i][:]
return p

def _get_legal_moves(self):
row, col = self.find(0)
free = []
if row > 0:
free.append((row - 1, col))
if col > 0:
free.append((row, col - 1))
if row < 2:
free.append((row + 1, col))
if col < 2:
free.append((row, col + 1))
return free
def _generate_moves(self):
free = self._get_legal_moves()
zero = self.find(0)

def swap_and_clone(a, b):
        p = self._clone()
p.swap(a,b)
        p._depth = self._depth + 1
        p._parent = self
return p

return map(lambda pair: swap_and_clone(zero, pair), free)

def _generate_solution_path(self, path):
if self._parent == None:
return path
else:
path.append(self)
return self._parent._generate_solution_path(path)

def is_solved(puzzle):
return puzzle.adj_matrix == _goal_state

openl = [self]
closedl = []
      move_count = 0
```

```
while len(openl) > 0:
        x = openl.pop(0)
        move_count += 1
if (is_solved(x)):
if len(closedl) > 0:
return x._generate_solution_path([]), move_count
else:
return [x]
succ = x._generate_moves()
        idx_open = idx_closed = -1
for move in succ:
            # have we already seen this node?
            idx_open = index(move, openl)
            idx_closed = index(move, closedl)
hval = h(move)
fval = hval + move._depth
if idx_closed == -1 and idx_open == -1:
                move._hval = hval
openl.append(move)
elif idx_open > -1:
copy = openl[idx_open]
if fval < copy._hval + copy._depth:
                    # copy move's values over existing
                    copy._hval = hval
                    copy._parent = move._parent
                    copy._depth = move._depth
elif idx_closed > -1:
copy = closedl[idx_closed]
if fval < copy._hval + copy._depth:
                    move._hval = hval
closedl.remove(copy)
openl.append(move)

        closedl.append(x)
openl = sorted(openl, key=lambda p: p._hval + p._depth)

def shuffle(self, step_count):
for i in range(step_count):
row, col = self.find(0)
free = self._get_legal_moves()
target = random.choice(free)
self.swap((row, col), target)
```

```
row, col = target

def find(self, value):
if value < 0 or value > 8:
raise Exception("value out of range")

for row in range(3):
for col in range(3):
if self.adj_matrix[row][col] == value:
return row, col

def peek(self, row, col):
return self.adj_matrix[row][col]


def poke(self, row, col, value):
        self.adj_matrix[row][col] = value

def swap(self, pos_a, pos_b):
temp = self.peek(*pos_a)
self.poke(pos_a[0], pos_a[1], self.peek(*pos_b))
self.poke(pos_b[0], pos_b[1], temp)

def heur(puzzle, item_total_calc, total_calc):
    t = 0
for row in range(3):
for col in range(3):
val = puzzle.peek(row, col) - 1
            target_col = val % 3
            target_row = val / 3

            # account for 0 as blank
if target_row < 0:
                target_row = 2

            t += item_total_calc(row, target_row, col, target_col)

return total_calc(t)

def h_manhattan(puzzle):
return heur(puzzle,
lambda r, tr, c, tc: abs(tr - r) + abs(tc - c),
```

```
lambda t : t)

def h_manhattan_lsq(puzzle):
return heur(puzzle,
lambda r, tr, c, tc: (abs(tr - r) + abs(tc - c))**2,
lambda t: math.sqrt(t))

def h_linear(puzzle):
return heur(puzzle,
lambda r, tr, c, tc: math.sqrt(math.sqrt((tr - r)**2 + (tc - c)**2)),
lambda t: t)

def h_linear_lsq(puzzle):
return heur(puzzle,
lambda r, tr, c, tc: (tr - r)**2 + (tc - c)**2,
lambda t: math.sqrt(t))

def h_default(puzzle):
return 0

def main():
    p = EightPuzzle()
p.shuffle(20)
print p

path, count = p.solve(h_manhattan)
path.reverse()
for i in path:
print i

print "Solved with Manhattan distance exploring", count, "states"
path, count = p.solve(h_manhattan_lsq)
print "Solved with Manhattan least squares exploring", count, "states"
path, count = p.solve(h_linear)
print "Solved with linear distance exploring", count, "states"
path, count = p.solve(h_linear_lsq)
print "Solved with linear least squares exploring", count, "states"
#    path, count = p.solve(heur_default)
#    print "Solved with BFS-equivalent in", count, "moves"

if__name__== "_main_":
main()
```

Output:


Programiz
Python Online Compiler
Learn Python App
main.py
Run
Shell
Clear
import random
import math
_goal_state = [[1,2,3],
[4,5,6],
[7,8,0]]
def index(item, seq):
"""Helper function that returns -1 for non-found index value of a seq"""
if item in seq:
return seq.index(item)
else:
return -1
class EightPuzzle:
def __init__(self):
# heuristic value
self._hval = 0
# search depth of current instance
self._depth = 0
# parent node in search path
self._parent = None
self.adj_matrix = []
for i in range(3):
self.adj_matrix.append(_goal_state[i][:])
2 3 5
1 7 8
0 4 6
2 3 5
1 7 8
4 0 6
2 3 5
1 0 8
4 7 6
2 3 5
1 8 0
4 7 6

Program:

```
// CPP program for solving cryptographic puzzles
#include <bits/stdc++.h>
using namespace std;
vector<int>  use(10);
struct node
{
	char c;
	int v;
};
int check(node* nodeArr, const int count, string s1,string s2, string s3)
{
	int val1 = 0, val2 = 0, val3 = 0, m = 1, j, i;
	for (i = s1.length() - 1; i >= 0; i--)
	{
		char ch = s1[i];
		for (j = 0; j < count; j++)
			if (nodeArr[j].c == ch)
				break;

		val1 += m * nodeArr[j].v;
		m *= 10;
	}
	m = 1;
	for (i = s2.length() - 1; i >= 0; i--)
	{
```

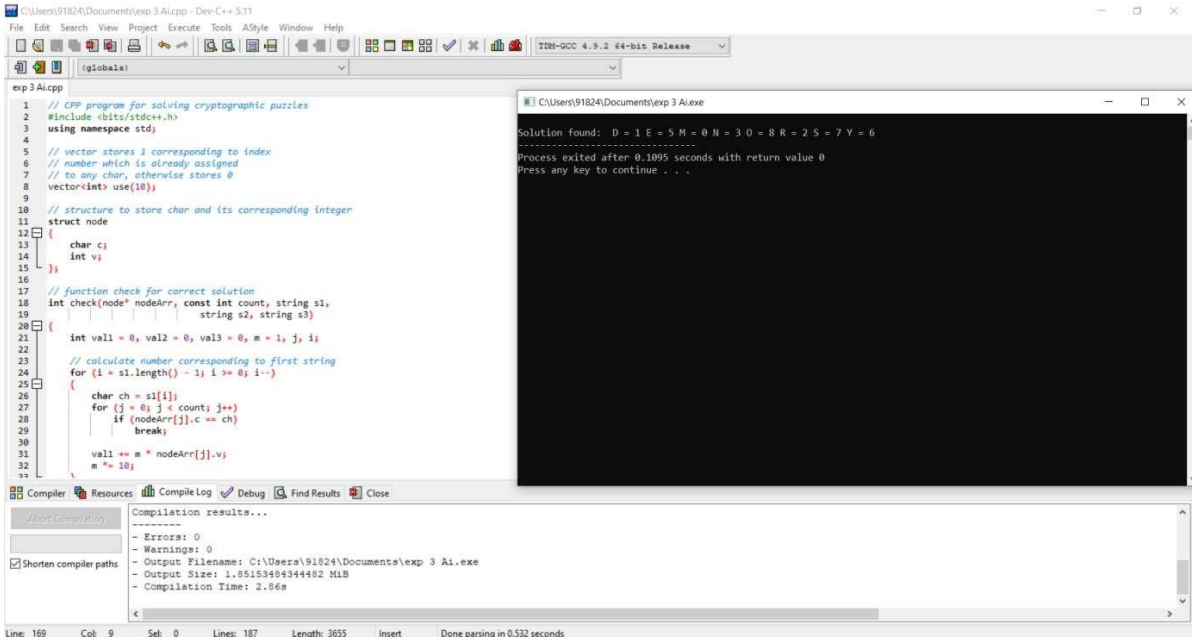
```
            char ch = s2[i];
            for (j = 0; j < count; j++)
                  if (nodeArr[j].c == ch)
                        break;

            val2 += m * nodeArr[j].v;
            m *= 10;
      }
      m = 1;
      for (i = s3.length() - 1; i >= 0; i--)
      {
            char ch = s3[i];
            for (j = 0; j < count; j++)
                  if (nodeArr[j].c == ch)
                        break;

            val3 += m * nodeArr[j].v;
            m *= 10;
      }
      if (val3 == (val1 + val2))
            return 1;
      return 0;
}
bool permutation(const int count, node* nodeArr, int n,string s1, string s2, string s3)
{
      if (n == count - 1)
```

```
{
        for (int i = 0; i < 10; i++)
        {
                if (use[i] == 0)
                {
                        nodeArr[n].v = i;
                        if (check(nodeArr, count, s1, s2, s3) == 1)
                        {
                                cout<< "\nSolution found: ";
                                for (int j = 0; j < count; j++)
                                        cout<< " " << nodeArr[j].c << " = "
                                                <<nodeArr[j].v;
                                return true;
                        }
                }
        }
        return false;
}
for (int i = 0; i < 10; i++)
{
        if (use[i] == 0)
        {
                nodeArr[n].v = i;
                use[i] = 1;
                if (permutation(count, nodeArr, n + 1, s1, s2, s3))
                use[i] = 0;
```

```
            }
        }
        return false;
}

bool solveCryptographic(string s1, string s2,string s3)
{
        int count = 0;
        int l1 = s1.length();
        int l2 = s2.length();
        int l3 = s3.length();
        vector<int> freq(26);
        for (int i = 0; i < l1; i++)
                ++freq[s1[i] - 'A'];

        for (int i = 0; i < l2; i++)
                ++freq[s2[i] - 'A'];

        for (int i = 0; i < l3; i++)
                ++freq[s3[i] - 'A'];
        for (int i = 0; i < 26; i++)
                if (freq[i] > 0)
                        count++;
        if (count > 10)
        {
                cout<< "Invalid strings";
```

```
            return 0;
        }
        node nodeArr[count];
        for (int i = 0, j = 0; i < 26; i++)
        {
            if (freq[i] > 0)
            {
                nodeArr[j].c = char(i + 'A');
                j++;
            }
        }
        return permutation(count, nodeArr, 0, s1, s2, s3);
}
int main()
{
        string s1 = "SEND";
        string s2 = "MORE";
        string s3 = "MONEY";

        if (solveCryptographic(s1, s2, s3) == false)
            cout<< "No solution";
        return 0;}
```

Output:


C:\Users\91824\Documents\exp 3 Ai.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
exp 3 Ai.cpp
// CPP program for solving cryptographic puzzles
#include <bits/stdc++.h>
using namespace std;
// vector stores 1 corresponding to index
// number which is already assigned
// to any char, otherwise stores 0
vector<int> use(10);
// structure to store char and its corresponding integer
struct node
{
char c;
int v;
};
// function check for correct solution
int check(node* nodeArr, const int count, string s1,
string s2, string s3)
{
int val1 = 0, val2 = 0, val3 = 0, m = 1, j, i;
// calculate number corresponding to first string
for (i = s1.length() - 1; i >= 0; i--)
{
char ch = s1[i];
for (j = 0; j < count; j++)
if (nodeArr[j].c == ch)
break;
val1 += m * nodeArr[j].v;
m *= 10;
C:\Users\91824\Documents\exp 3 Ai.exe
Solution found: D = 1 E = 5 M = 0 N = 3 O = 8 R = 2 S = 7 Y = 6
Process exited after 0.1095 seconds with return value 0
Press any key to continue . . .
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Compilation results...
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\91824\Documents\exp 3 Ai.exe
- Output Size: 1.85153484344482 MiB
- Compilation Time: 2.86s
Line: 169 Col: 9 Sel: 0 Lines: 187 Length: 3655 Insert Done parsing in 0.532 seconds

Program:

```
# DFS algorithm in Python
# DFS algorithm
def dfs(graph, start, visited=None):
if visited is None:
visited = set()
visited.add(start)

print(start)

for next in graph[start] - visited:
dfs(graph, next, visited)
return visited


graph = {'0': set(['1', '2']),
        '1': set(['0', '3', '4']),
        '2': set(['0']),
        '3': set(['1']),
        '4': set(['2', '3'])}

dfs(graph, '0')
```

Output for DFS:

dfs.py - C:/Users/91824/AppData/Local/Programs/Python/Python39/dfs.py (3.9.1)

File Edit Format Run Options Window Help

```
# DFS algorithm in Python


# DFS algorithm
def dfs(graph, start, visited=None):
    if visited is None:
        visited = set()
    visited.add(start)

    print(start)

    for next in graph[start] - visited:
        dfs(graph, next, visited)
    return visited


graph = {'0': set(['1', '2']),
         '1': set(['0', '3', '4']),
         '2': set(['0']),
         '3': set(['1']),
         '4': set(['2', '3'])}

dfs(graph, '0')
```

IDLE Shell 3.9.1

File Edit Shell Debug Options Window Help

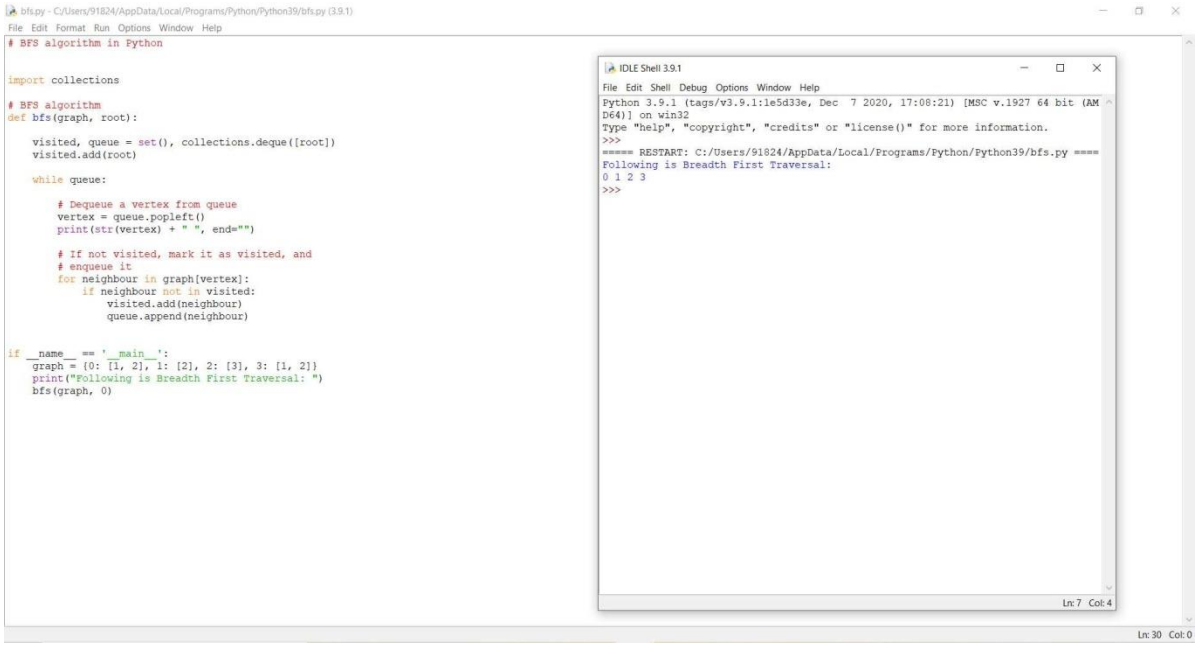
```
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec  7 2020, 17:08:21) [MSC v.1927 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/91824/AppData/Local/Programs/Python/Python39/dfs.py ====
0
1
4
2
3
3
2
>>>
```

Ln: 12 Col: 4

Program:

```
# BFS algorithm in Python
import collections
def bfs(graph, root):
visited, queue = set(), collections.deque([root])
visited.add(root)
while queue:
vertex = queue.popleft()
print(str(vertex) + " ", end="")
for neighbour in graph[vertex]:
if neighbour not in visited:
visited.add(neighbour)
queue.append(neighbour)
if__name__== '__main__':
graph = {0: [1, 2], 1: [2], 2: [3], 3: [1, 2]}
print("Following is Breadth First Traversal: ")
bfs(graph, 0)
```

Output for BFS:



Program:

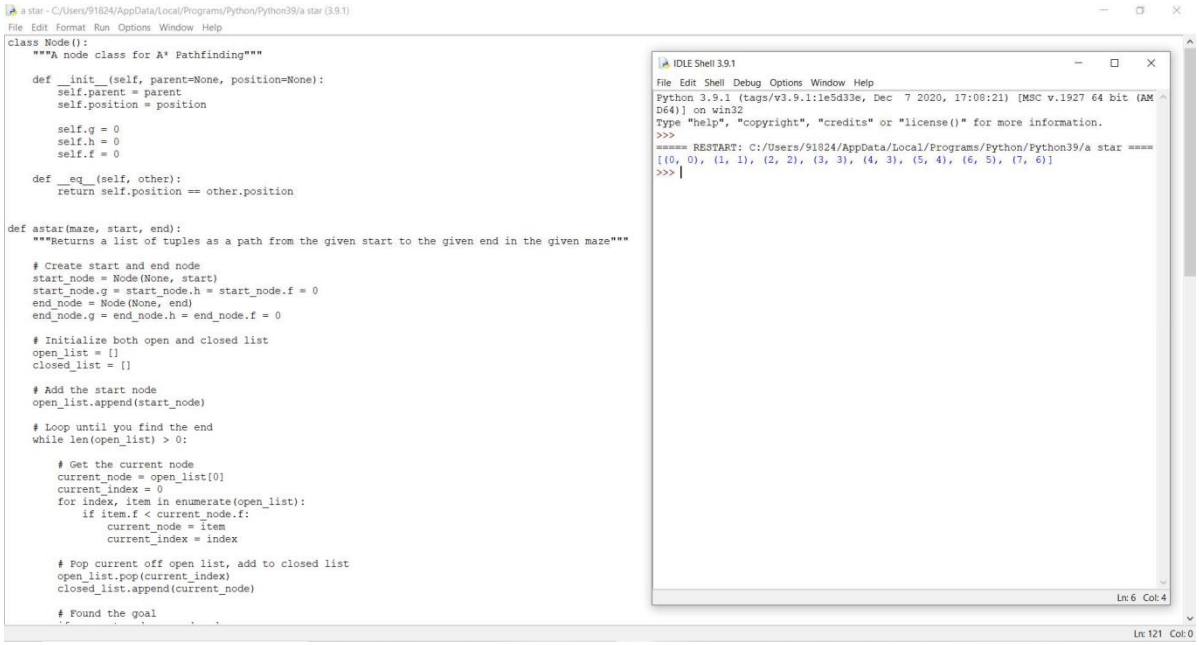
```
class Node():
def__init_(self, parent=None, position=None):
        self.parent = parent
        self.position = position
        self.g = 0
        self.h = 0
        self.f = 0
def__eq__(self, other):
return self.position == other.position
def astar(maze, start, end):
    start_node = Node(None, start)
    start_node.g = start_node.h = start_node.f = 0
    end_node = Node(None, end)
    end_node.g = end_node.h = end_node.f = 0
    open_list = []
    closed_list = []
    open_list.append(start_node)
while len(open_list) > 0:
        current_node = open_list[0]
        current_index = 0
```

```
for index, item in enumerate(open_list):

if item.f < current_node.f:

            current_node = item

            current_index = index

        open_list.pop(current_index)

        closed_list.append(current_node)

if current_node == end_node:

path = []

current = current_node

while current is not None:

path.append(current.position)

current = current.parent

return path[::-1]

children = []

for new_position in [(0, -1), (0, 1), (-1, 0), (1, 0), (-1, -1), (-1, 1), (1, -1), (1, 1)]:

            node_position = (current_node.position[0] + new_position[0],
current_node.position[1] + new_position[1])

if node_position[0] > (len(maze) - 1) or node_position[0] < 0 or node_position[1] >
(len(maze[len(maze)-1]) -1) or node_position[1] < 0:

continue

if maze[node_position[0]][node_position[1]] != 0:

continue
```

```
            new_node = Node(current_node, node_position)

children.append(new_node)

for child in children:

for closed_child in closed_list:

if child == closed_child:

continue

            child.g = current_node.g + 1

            child.h = ((child.position[0] - end_node.position[0]) ** 2) +
((child.position[1] - end_node.position[1]) ** 2)

            child.f = child.g + child.h

for open_node in open_list:

if child == open_node and child.g > open_node.g:

continue

            open_list.append(child)

def main():

maze = [[0, 0, 0, 0, 1, 0, 0, 0, 0, 0],

            [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],

            [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],

            [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],

            [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],

            [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

            [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
```

```
        [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
start = (0, 0)
end = (7, 6)
path = astar(maze, start, end)
print(path)


if__name__== '_main_':
main()
```

Output:


a star - C:/Users/91824/AppData/Local/Programs/Python/Python39/a star (3.9.1)
File Edit Format Run Options Window Help
class Node():
"""A node class for A* Pathfinding"""
def __init__(self, parent=None, position=None):
self.parent = parent
self.position = position
self.g = 0
self.h = 0
self.f = 0
def __eq__(self, other):
return self.position == other.position
def astar(maze, start, end):
"""Returns a list of tuples as a path from the given start to the given end in the given maze"""
# Create start and end node
start_node = Node(None, start)
start_node.g = start_node.h = start_node.f = 0
end_node = Node(None, end)
end_node.g = end_node.h = end_node.f = 0
# Initialize both open and closed list
open_list = []
closed_list = []
# Add the start node
open_list.append(start_node)
# Loop until you find the end
while len(open_list) > 0:
# Get the current node
current_node = open_list[0]
current_index = 0
for index, item in enumerate(open_list):
if item.f < current_node.f:
current_node = item
current_index = index
# Pop current off open list, add to closed list
open_list.pop(current_index)
closed_list.append(current_node)
# Found the goal
Ln: 121 Col: 0
IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/91824/AppData/Local/Programs/Python/Python39/a star ====
[(0, 0), (1, 1), (2, 2), (3, 3), (4, 3), (5, 4), (6, 5), (7, 6)]
>>>
Ln: 6 Col: 4

Program:

```
import sys
import random
class TicTacToeGame:
def___init_(self, rows:int, columns:int, goal:int, max_depth:int=4):
        self.state = []
        self.tiles = {}
        self.inverted_tiles = {}
tile = 0
for y in range(rows):
row = []
for x in range(columns):
row += '.'
tile += 1
self.tiles[tile] = (y, x)
                self.inverted_tiles[(y, x)] = tile
self.state.append(row)
        self.goal = goal
        self.vectors = [(1,0), (0,1), (1,1), (-1,1)]
        self.rows = rows
        self.columns = columns
        self.max_row_index = rows - 1
        self.max_columns_index = columns - 1
        self.max_depth = max_depth
        self.winning_positions = []
        self.get_winning_positions()
```

```
        self.player = random.choice(['X', 'O'])
def get_winning_positions(self):
for y in range(self.rows):
for x in range(self.columns):
for vector in self.vectors:
sy, sx = (y, x)
dy, dx = vector
counter = 0
positions = []
while True:
positions.append(self.inverted_tiles.get((sy, sx)))
if (len(positions) == self.goal):
                        self.winning_positions.append(positions)
break
sy += dy
sx += dx
if(sy < 0 or abs(sy) > self.max_row_index or sx < 0 or abs(sx) >
self.max_columns_index):
break
def play(self):
result = None
print('Starting board')
while True:
            self.print_state()
if (self.player == 'X'):
print('Player X moving (AI) ...')
```

```
max, py, px, depth = self.max(-sys.maxsize, sys.maxsize)
print('Depth: {0}'.format(depth))
if(depth > self.max_depth):
py, px = self.get_best_move()
self.state[py][px] = 'X'
result = self.game_ended()
if(result != None):
break
                self.player = 'O'
elif (self.player == 'O'):
print('Player O moving (Human) ...')
min, py, px, depth = self.min(-sys.maxsize, sys.maxsize)
print('Depth: {0}'.format(depth))
if(depth > self.max_depth):
py, px = self.get_best_move()
print('Recommendation: {0}'.format(self.inverted_tiles.get((py, px))))
number = int(input('Make a move (tile number): '))
tile = self.tiles.get(number)
if(tile != None):
py, px = tile
self.state[py][px] = 'O'
result = self.game_ended()
if(result != None):
break
                    self.player = 'X'
else:
```

```
print('Move is not legal, try again.')
        self.print_state()
print('Winner is player: {0}'.format(result))
def get_best_move(self):
heuristics = {}
        empty_cells = []
for y in range(self.rows):
for x in range(self.columns):
if (self.state[y][x] == '.'):
                    empty_cells.append((y, x))
for empty in empty_cells:
number = self.inverted_tiles.get(empty)
for win in self.winning_positions:
if(number in win):
                    player_x = 0
                    player_o = 0
                    start_score = 1
for box in win:
y, x = self.tiles[box]
if(self.state[y][x] == 'X'):
                        player_x += start_score if self.player == 'X' else start_score * 2
                        start_score *= 10
elif (self.state[y][x] == 'O'):
                        player_o += start_score if self.player == 'O' else start_score * 2
                        start_score *= 10
if(player_x == 0 or player_o == 0):
```

```
score = max(player_x, player_o) + start_score
if(heuristics.get(number) != None):
heuristics[number] += score
else:
heuristics[number] = score
        best_move = random.choice(empty_cells)
        best_count = -sys.maxsize
for key, value in heuristics.items():
if(value > best_count):
                best_move = self.tiles.get(key)
                best_count = value
return best_move
def game_ended(self) -> str:
result = self.player_has_won()
if(result != None):
return result
for y in range(self.rows):
for x in range(self.columns):
if (self.state[y][x] == '.'):
return None
return 'It is a tie!'
def player_has_won(self) -> str:
for y in range(self.rows):
for x in range(self.columns):
for vector in self.vectors:
sy, sx = (y, x)
```

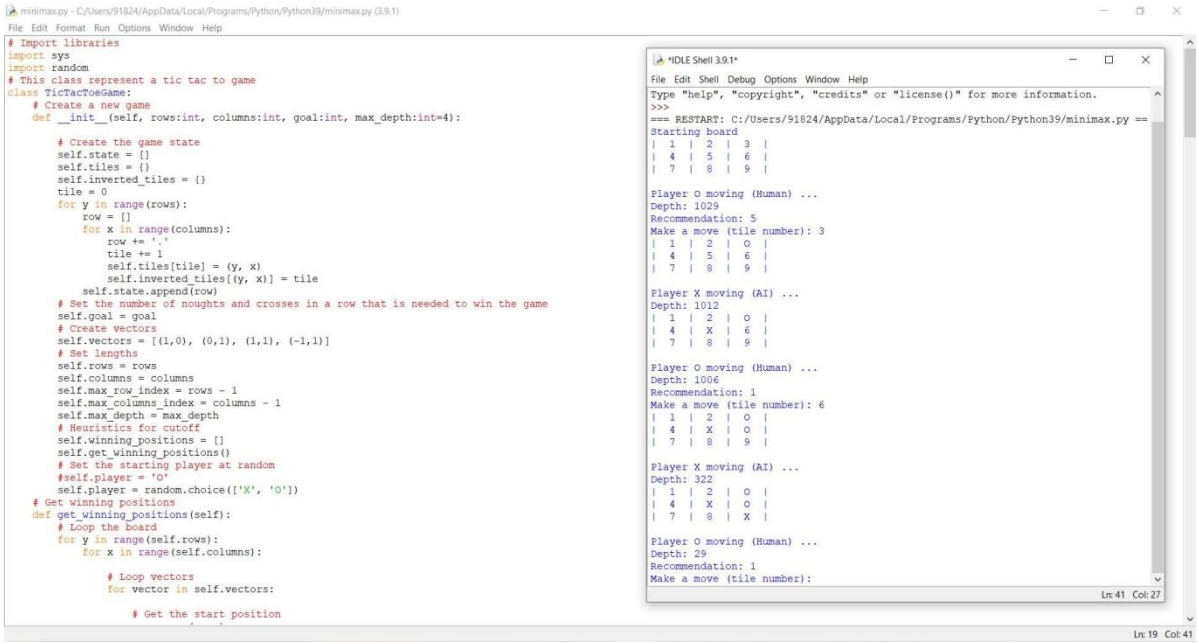
```
dy, dx = vector
steps = 0
                player_x = 0
                player_o = 0
while steps < self.goal:
steps += 1
if(self.state[sy][sx] == 'X'):
                        player_x += 1
elif(self.state[sy][sx] == 'O'):
                        player_o += 1
sy += dy
sx += dx
if(sy < 0 or abs(sy) > self.max_row_index or sx < 0 or abs(sx) >
self.max_columns_index):
break
if(player_x >= self.goal):
return 'X'
elif(player_o >= self.goal):
return 'O'
return None
def min(self, alpha:int=-sys.maxsize, beta:int=sys.maxsize, depth:int=0):


    min_value = sys.maxsize
by = None
bx = None
result = self.game_ended()
```

```
if(result != None):
if result == 'X':
return 1, 0, 0, depth
elif result == 'O':
return -1, 0, 0, depth
elif result == 'It is a tie!':
return 0, 0, 0, depth
elif(depth > self.max_depth):
return 0, 0, 0, depth
for y in range(self.rows):
for x in range(self.columns):
if (self.state[y][x] == '.'):
                self.state[y][x] = 'O'
max, max_y, max_x, depth = self.max(alpha, beta, depth + 1)
if (max < min_value):
                    min_value = max
by = y
bx = x
                self.state[y][x] = '.'
if (min_value <= alpha):
return min_value, bx, by, depth
if (min_value < beta):
beta = min_value
return min_value, by, bx, depth
def max(self, alpha:int=-sys.maxsize, beta:int=sys.maxsize, depth:int=0):
        max_value = -sys.maxsize
```

```
by = None
bx = None
        # Check if the game has ended
result = self.game_ended()
if(result != None):
if result == 'X':
return 1, 0, 0, depth
elif result == 'O':
return -1, 0, 0, depth
elif result == 'It is a tie!':
return 0, 0, 0, depth
elif(depth > self.max_depth):
return 0, 0, 0, depth
for y in range(self.rows):
for x in range(self.columns):
if (self.state[y][x] == '.'):
                    self.state[y][x] = 'X'
min, min_y, min_x, depth = self.min(alpha, beta, depth + 1)
if (min > max_value):
                        max_value = min
by = y
bx = x
                    self.state[y][x] = '.'
if (max_value >= beta):
return max_value, bx, by, depth
if (max_value > alpha):
```

```
alpha = max_value
return max_value, by, bx, depth
def print_state(self):
for y in range(self.rows):
print('| ', end='')
for x in range(self.columns):
if (self.state[y][x] != '.'):
print(' {0}  | '.format(self.state[y][x]), end='')
else:
                digit = str(self.inverted_tiles.get((y,x))) if
len(str(self.inverted_tiles.get((y,x)))) > 1 else ' ' + str(self.inverted_tiles.get((y,x)))
print('{0}  | '.format(digit), end='')
print()
print()
def main():
game = TicTacToeGame(3, 3, 3, 1000)
game.play()
if__name__== "_main__": main()
```

Output:


minimax.py - C:/Users/91824/AppData/Local/Programs/Python/Python39/minimax.py (3.9.1)
File Edit Format Run Options Window Help
# Import libraries
import sys
import random
# This class represent a tic tac to game
class TicTacToeGame:
# Create a new game
def __init__(self, rows:int, columns:int, goal:int, max_depth:int=4):
# Create the game state
self.state = []
self.tiles = {}
self.inverted_tiles = {}
tile = 0
for y in range(rows):
row = []
for x in range(columns):
row += '.'
tile += 1
self.tiles[tile] = (y, x)
self.inverted_tiles[(y, x)] = tile
self.state.append(row)
# Set the number of noughts and crosses in a row that is needed to win the game
self.goal = goal
# Create vectors
self.vectors = [(1,0), (0,1), (1,1), (-1,1)]
# Set lengths
self.rows = rows
self.columns = columns
self.max_row_index = rows - 1
self.max_columns_index = columns - 1
self.max_depth = max_depth
# Heuristics for cutoff
self.winning_positions = []
self.get_winning_positions()
# Set the starting player at random
#self.player = 'O'
self.player = random.choice(['X', 'O'])
# Get winning positions
def get_winning_positions(self):
# Loop the board
for y in range(self.rows):
for x in range(self.columns):
# Loop vectors
for vector in self.vectors:
# Get the start position
Ln: 19 Col: 41
IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=== RESTART: C:/Users/91824/AppData/Local/Programs/Python/Python39/minimax.py ==
Starting board
1	2	3
4	5	6
7	8	9
Player O moving (Human) ...
Depth: 1029
Recommendation: 5
Make a move (tile number): 3
1	2	O
4	5	6
7	8	9
Player X moving (AI) ...
Depth: 1012
1	2	O
4	X	6
7	8	9
Player O moving (Human) ...
Depth: 1006
Recommendation: 1
Make a move (tile number): 6
1	2	O
4	X	O
7	8	9
Player X moving (AI) ...
Depth: 322
1	2	O
4	X	O
7	8	X
Player O moving (Human) ...
Depth: 29
Recommendation: 1
Make a move (tile number):
Ln: 41 Col: 27

Program:

```
def get_index_comma(string):
        index_list = list()
        par_count = 0
    for i in range(len(string)):
    if string[i] == ',' and par_count == 0:
            index_list.append(i)
    elif string[i] == '(':
            par_count += 1
    elif string[i] == ')':
            par_count -= 1
    return index_list
def is_variable(expr):
    for i in expr:
    if i == '(':
    return False
    return True
def process_expression(expr):
    expr = expr.replace(' ', '')
    index = None
    for i in range(len(expr)):
    if expr[i] == '(':
    index = i
    break
        predicate_symbol = expr[:index]
    expr = expr.replace(predicate_symbol, '')
```

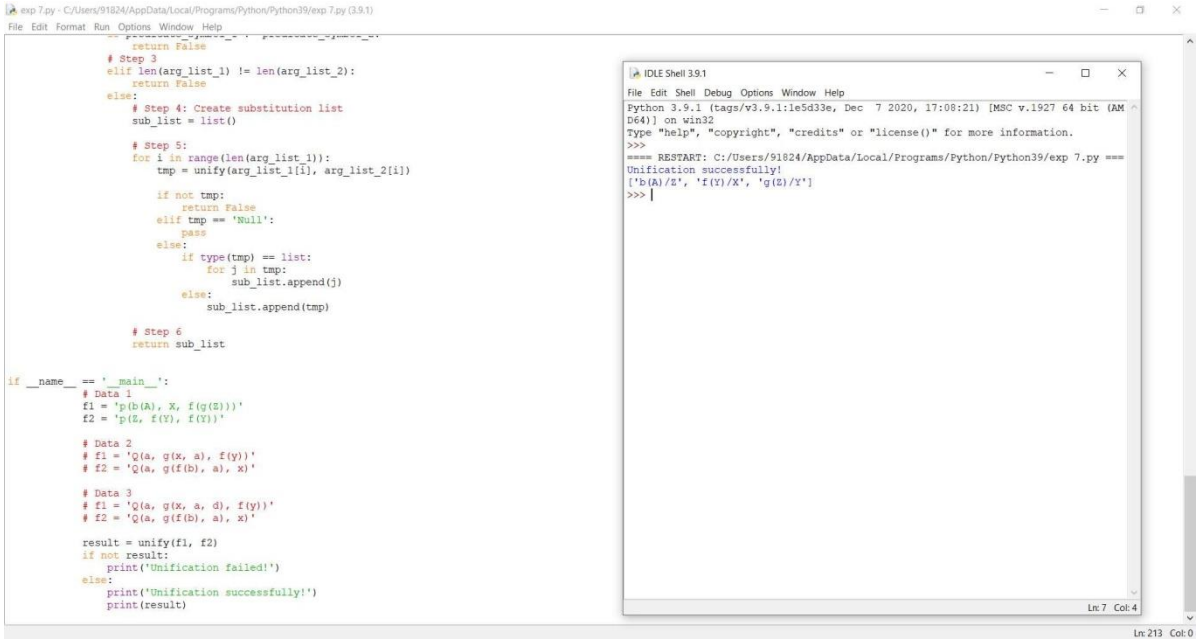
```
        expr = expr[1:len(expr) - 1]
            arg_list = list()
        indices = get_index_comma(expr)
        if len(indices) == 0:
                arg_list.append(expr)
        else:
                arg_list.append(expr[:indices[0]])
        for i, j in zip(indices, indices[1:]):
                    arg_list.append(expr[i + 1:j])
                arg_list.append(expr[indices[len(indices) - 1] + 1:])
        return predicate_symbol, arg_list
def get_arg_list(expr):
                        _, arg_list = process_expression(expr)
        flag = True
        while flag:
        flag = False
        for i in arg_list:
        if not is_variable(i):
        flag = True
                    _, tmp = process_expression(i)
        for j in tmp:
        if j not in arg_list:
                            arg_list.append(j)
                    arg_list.remove(i)
        return arg_list
```

```
def check_occurs(var, expr):
        arg_list = get_arg_list(expr)
    if var in arg_list:
    return True
    return False
def unify(expr1, expr2):
    if is_variable(expr1) and is_variable(expr2):
    if expr1 == expr2:
    return 'Null'
    else:
    return False
    elif is_variable(expr1) and not is_variable(expr2):
    if check_occurs(expr1, expr2):
    return False
    else:
    tmp = str(expr2) + '/' + str(expr1)
    return tmp
    elif not is_variable(expr1) and is_variable(expr2):
    if check_occurs(expr2, expr1):
    return False
    else:
    tmp = str(expr1) + '/' + str(expr2)
    return tmp
    else:
        predicate_symbol_1, arg_list_1 = process_expression(expr1)
```

```
        predicate_symbol_2, arg_list_2 = process_expression(expr2)

        # Step 2
if predicate_symbol_1 != predicate_symbol_2:
return False
        # Step 3
elif len(arg_list_1) != len(arg_list_2):
return False
else:
            # Step 4: Create substitution list
            sub_list = list()

            # Step 5:
for i in range(len(arg_list_1)):
tmp = unify(arg_list_1[i], arg_list_2[i])

if not tmp:
return False
elif tmp == 'Null':
pass
else:
if type(tmp) == list:
for j in tmp:
                        sub_list.append(j)
else:
                    sub_list.append(tmp)
```

```
                # Step 6
        return sub_list


if__name__== '__main__':
            # Data 1
            f1 = 'p(b(A), X, f(g(Z)))'
            f2 = 'p(Z, f(Y), f(Y))'

            # Data 2
            # f1 = 'Q(a, g(x, a), f(y))'
            # f2 = 'Q(a, g(f(b), a), x)'

            # Data 3
            # f1 = 'Q(a, g(x, a, d), f(y))'
            # f2 = 'Q(a, g(f(b), a), x)'

        result = unify(f1, f2)
        if not result:
        print('Unification failed!')
        else:
        print('Unification successfully!')
        print(result)
```

Output:


exp 7.py - C:/Users/91824/AppData/Local/Programs/Python/Python39/exp 7.py (3.9.1)
File Edit Format Run Options Window Help
return False
# Step 3
elif len(arg_list_1) != len(arg_list_2):
return False
else:
# Step 4: Create substitution list
sub_list = list()
# Step 5:
for i in range(len(arg_list_1)):
tmp = unify(arg_list_1[i], arg_list_2[i])
if not tmp:
return False
elif tmp == 'Null':
pass
else:
if type(tmp) == list:
for j in tmp:
sub_list.append(j)
else:
sub_list.append(tmp)
# Step 6
return sub_list
if __name__ == '__main__':
# Data 1
f1 = 'p(b(A), X, f(g(Z)))'
f2 = 'p(Z, f(Y), f(Y))'
# Data 2
# f1 = 'Q(a, g(x, a), f(y))'
# f2 = 'Q(a, g(f(b), a), x)'
# Data 3
# f1 = 'Q(a, g(x, a, d), f(y))'
# f2 = 'Q(a, g(f(b), a), x)'
result = unify(f1, f2)
if not result:
print('Unification failed!')
else:
print('Unification successfully!')
print(result)
Ln: 213 Col: 0
IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:/Users/91824/AppData/Local/Programs/Python/Python39/exp 7.py ====
Unification successfully!
['b(A)/Z', 'f(Y)/X', 'g(Z)/Y']
>>>
Ln: 7 Col: 4

Program:

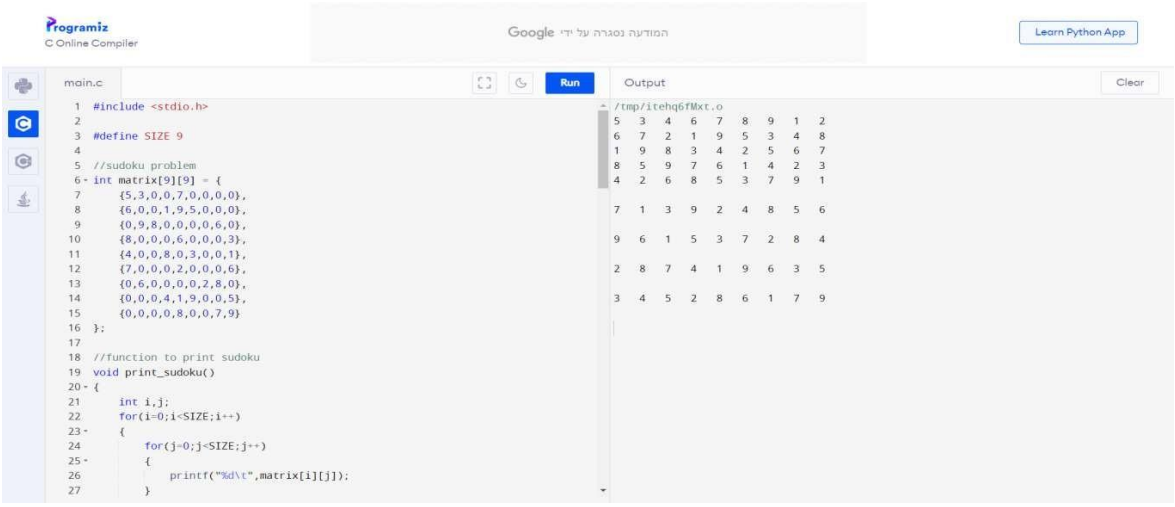
```
#include <stdio.h>
#define SIZE 9
int matrix[9][9] = {
    {5,3,0,0,7,0,0,0,0},
    {6,0,0,1,9,5,0,0,0},
    {0,9,8,0,0,0,0,6,0},
    {8,0,0,0,6,0,0,0,3},
    {4,0,0,8,0,3,0,0,1},
    {7,0,0,0,2,0,0,0,6},
    {0,6,0,0,0,0,2,8,0},
    {0,0,0,4,1,9,0,0,5},
    {0,0,0,0,8,0,0,7,9}
};
void print_sudoku()
{
int i,j;
for(i=0;i<SIZE;i++)
    {
for(j=0;j<SIZE;j++)
        {
printf("%d\t",matrix[i][j]);
        }
printf("\n\n");
    }
}
```

```
int number_unassigned(int *row, int *col)
{
int num_unassign = 0;
int i,j;
for(i=0;i<SIZE;i++)
   {
for(j=0;j<SIZE;j++)
      {
if(matrix[i][j] == 0)
         {
            *row = i;
            *col = j;
            num_unassign = 1;
            return num_unassign;
         }
      }
   }
return num_unassign;
}
int is_safe(int n, int r, int c)
{
int i,j;
for(i=0;i<SIZE;i++)
   {
if(matrix[r][i] == n)
return 0;
```

```
    }
for(i=0;i<SIZE;i++)
    {
if(matrix[i][c] == n)
return 0;
    }
int row_start = (r/3)*3;
int col_start = (c/3)*3;
for(i=row_start;i<row_start+3;i++)
    {
for(j=col_start;j<col_start+3;j++)
        {
if(matrix[i][j]==n)
return 0;
        }
    }
return 1;
}
int solve_sudoku()
{
int row;
int col;
if(number_unassigned(&row, &col) == 0)
return 1;
int n,i;
for(i=1;i<=SIZE;i++)
```

```
if(is_safe(i, row, col))
        {
matrix[row][col] = i;
if(solve_sudoku())
return 1;
matrix[row][col]=0;
        }
    }
return 0;
}

int main()
{
if (solve_sudoku())
        print_sudoku();
else
printf("No solution\n");
return 0;
}
```

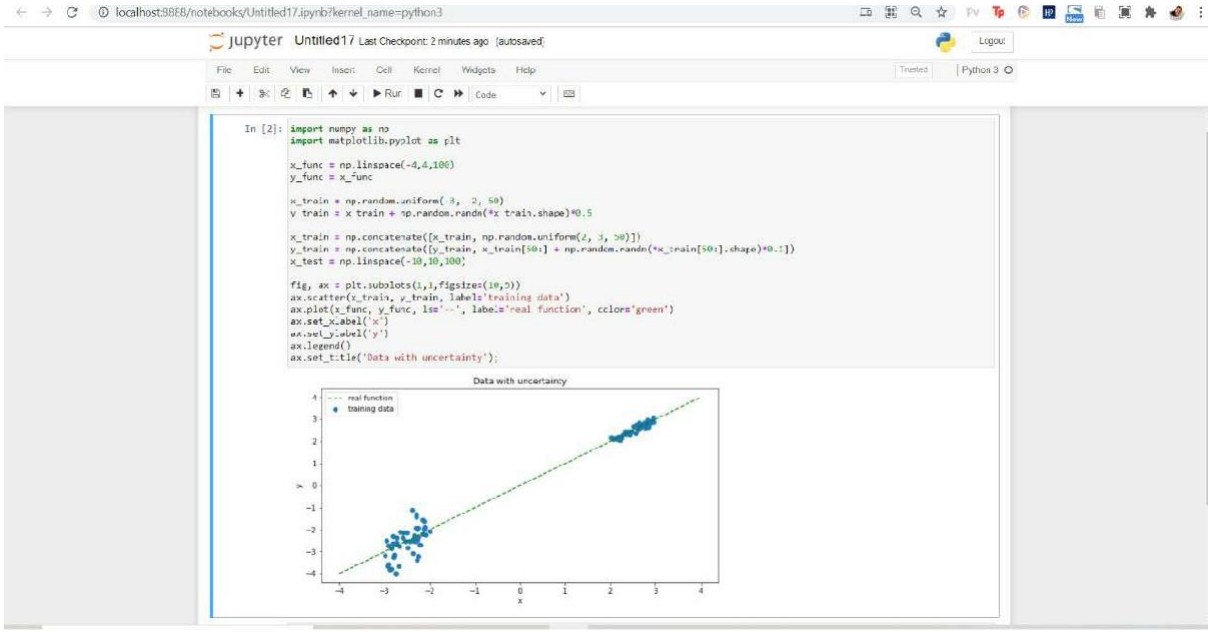
Output:


Programiz
C Online Compiler
Google המודעה נסגרה על ידי
Learn Python App
main.c
Run
Output
Clear
#include <stdio.h>
#define SIZE 9
//sudoku problem
int matrix[9][9] = {
{5,3,0,0,7,0,0,0,0},
{6,0,0,1,9,5,0,0,0},
{0,9,8,0,0,0,0,6,0},
{8,0,0,0,6,0,0,0,3},
{4,0,0,8,0,3,0,0,1},
{7,0,0,0,2,0,0,0,6},
{0,6,0,0,0,0,2,8,0},
{0,0,0,4,1,9,0,0,5},
{0,0,0,0,8,0,0,7,9}
};
//function to print sudoku
void print_sudoku()
{
int i,j;
for(i=0;i<SIZE;i++)
{
for(j=0;j<SIZE;j++)
{
printf("%d\t",matrix[i][j]);
}
/tmp/itehq6fMxt.o
5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9

Program:

```
# prediction
y_pred_without_dropout = model_without_dropout.predict(x_test)
y_pred_with_dropout = model_with_dropout.predict(x_test)

# plotting
fig, ax = plt.subplots(1,1,figsize=(10,5))
ax.scatter(x_train, y_train, s=10, label='train data')
ax.plot(x_test, x_test, ls='--', label='test data', color='green')
ax.plot(x_test, y_pred_without_dropout, label='predicted ANN - R2
{:.2f}'.format(r2_score(x_test, y_pred_without_dropout)), color='red')
ax.plot(x_test, y_pred_with_dropout, label='predicted ANN Dropout - R2
{:.2f}'.format(r2_score(x_test, y_pred_with_dropout)), color='black')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.legend()
ax.set_title('test data');
```

Output:


localhost:8888/notebooks/Untitled17.ipynb?kernel_name=python3
jupyter Untitled17 Last Checkpoint: 2 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [2]:
import numpy as np
import matplotlib.pyplot as plt
x_func = np.linspace(-4,4,100)
y_func = x_func
x_train = np.random.uniform(-3, 2, 50)
y_train = x_train + np.random.randn(*x_train.shape)*0.5
x_train = np.concatenate([x_train, np.random.uniform(2, 3, 50)])
y_train = np.concatenate([y_train, x_train[50:] + np.random.randn(*x_train[50:].shape)*0.1])
x_test = np.linspace(-10,10,100)
fig, ax = plt.subplots(1,1,figsize=(10,5))
ax.scatter(x_train, y_train, label='training data')
ax.plot(x_func, y_func, ls='--', label='real function', color='green')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.legend()
ax.set_title('Data with uncertainty');
Data with uncertainty
real function
training data
y
x

Program:

```
#PREDICATE - ON, ONTABLE, CLEAR, HOLDING, ARMEMPTY
classPREDICATE:
def__str__(self):
pass
def__repr__(self):
pass
def__eq__(self, other) :
pass
def__hash__(self):
pass
defget_action(self, world_state):
pass


#OPERATIONS - Stack, Unstack, Pickup, Putdown
classOperation:
def_str_(self):
pass
def_repr_(self):
pass
def_eq_(self, other) :
pass
defprecondition(self):
pass
defdelete(self):
pass
defadd(self):
pass


classON(PREDICATE):

def_init_(self, X, Y):
   self.X=X
   self.Y=Y
```

```
def str (self):
return"ON({X},{Y})".format(X=self.X,Y=self.Y)

def__repr__(self):
return self.__str__()

def__eq__(self, other) :
return self.__dict___== other.__dict___and self.__class___== other.__class___

def__hash__(self):
returnhash(str(self))

defget_action(self, world_state):
returnStackOp(self.X,self.Y)


classONTABLE(PREDICATE):

def init (self, X):
    self.X=X

def str (self):
return"ONTABLE({X})".format(X=self.X)

def__repr__(self):
return self.__str__()

def__eq__(self, other) :
return self.__dict___== other.__dict___and self.__class___== other.__class___

def hash (self):
returnhash(str(self))

defget_action(self, world_state):
returnPutdownOp(self.X)


classCLEAR(PREDICATE):
```

```
def init (self, X):
   self.X=X

def str (self):
return"CLEAR({X})".format(X=self.X)
   self.X=X

def__repr__(self):
return self.__str__()

def__eq__(self, other) :
return self.__dict___== other.__dict___and self.__class___== other.__class___

def__hash__(self):
returnhash(str(self))

defget_action(self, world_state):
for predicate in world_state:
#If Block is on another block, unstack
ifisinstance(predicate,ON) and predicate.Y==self.X:
returnUnstackOp(predicate.X, predicate.Y)
returnNone


classHOLDING(PREDICATE):

def init (self, X):
   self.X=X

def str (self):
return"HOLDING({X})".format(X=self.X)

def__repr__(self):
return self.__str__()

def__eq__(self, other) :
return self.__dict___== other.__dict___and self.__class___== other.__class___

def__hash__(self):
```

```
returnhash(str(self))

defget_action(self, world_state):
X= self.X
#If block is on table, pick up
ifONTABLE(X) in world_state:
returnPickupOp(X)
#If block is on another block, unstack
else:
for predicate in world_state:
ifisinstance(predicate,ON) and predicate.X==X:
returnUnstackOp(X,predicate.Y)


classARMEMPTY(PREDICATE):

def__init__(self):
pass

def__str__(self):
return"ARMEMPTY"

def__repr__(self):
return self.__str__()

def__eq__(self, other) :
return self.__dict___== other.__dict___and self.__class___== other.__class___

def hash (self):
returnhash(str(self))

defget_action(self, world_state=[]):
for predicate in world_state:
ifisinstance(predicate,HOLDING):
returnPutdownOp(predicate.X)
returnNone
classStackOp(Operation):
```

```
def_init_(self, X, Y):
   self.X=X
   self.Y=Y

def_str_(self):
return"STACK({X},{Y})".format(X=self.X,Y=self.Y)

def__repr__(self):
return self.__str__()

def__eq__(self, other) :
return self.__dict___== other.__dict___and self.__class___== other._class___

defprecondition(self):
return [ CLEAR(self.Y) , HOLDING(self.X) ]

defdelete(self):
return [ CLEAR(self.Y) , HOLDING(self.X) ]

defadd(self):
return [ ARMEMPTY() , ON(self.X,self.Y) ]


classUnstackOp(Operation):

def_init_(self, X, Y):
   self.X=X
   self.Y=Y

def_str_(self):
return"UNSTACK({X},{Y})".format(X=self.X,Y=self.Y)

def__repr__(self):
return self.__str__()

def__eq__(self, other) :
return self.__dict___== other.__dict___and self.__class___== other._class___
```

```
defprecondition(self):
return [ ARMEMPTY() , ON(self.X,self.Y) , CLEAR(self.X) ]

defdelete(self):
return [ ARMEMPTY() , ON(self.X,self.Y) ]

defadd(self):
return [ CLEAR(self.Y) , HOLDING(self.X) ]


classPickupOp(Operation):

def_init_(self, X):
   self.X=X

def_str_(self):
return"PICKUP({X})".format(X=self.X)

def__repr__(self):
return self.__str__()

def__eq__(self, other) :
return self.__dict___== other.__dict___and self.__class___== other._class___

defprecondition(self):
return [ CLEAR(self.X) , ONTABLE(self.X) , ARMEMPTY() ]

defdelete(self):
return [ ARMEMPTY() , ONTABLE(self.X) ]

defadd(self):
return [ HOLDING(self.X) ]


classPutdownOp(Operation):

def_init_(self, X):
   self.X=X
```

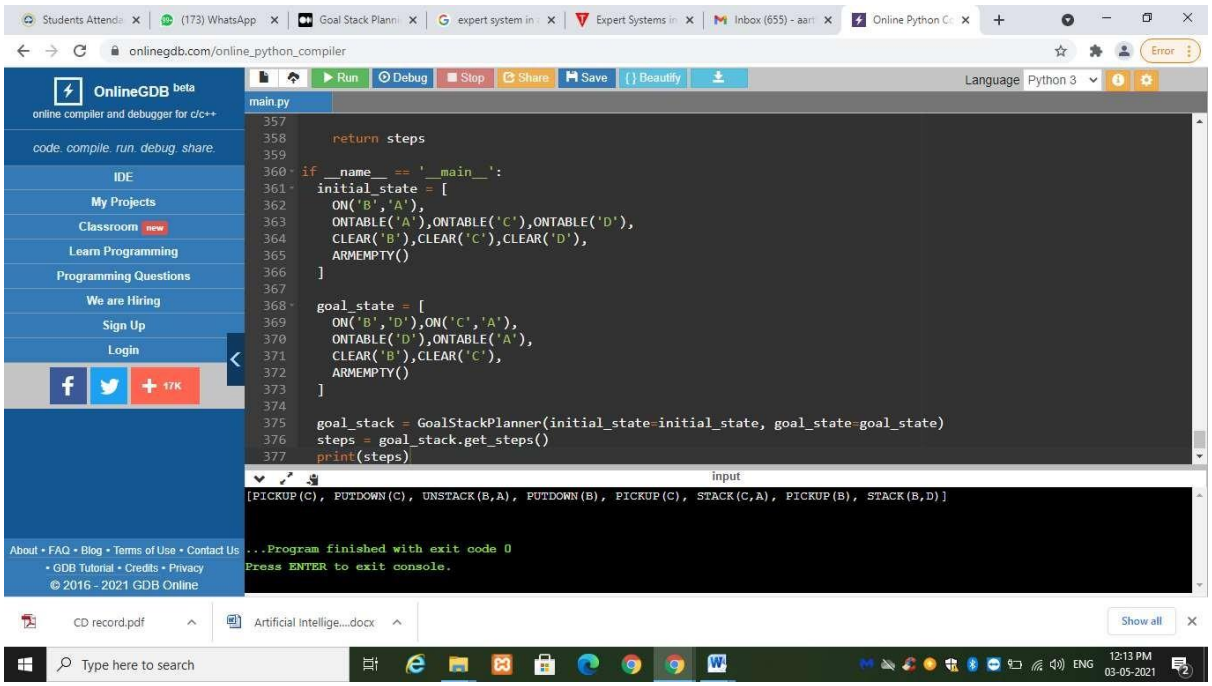
```
def_str_(self):
return"PUTDOWN({X})".format(X=self.X)

def__repr__(self):
return self.__str__()

def__eq__(self, other) :
return self.__dict___== other.__dict___and self.__class___== other.__class___

defprecondition(self):
return [ HOLDING(self.X) ]

defdelete(self):
return [ HOLDING(self.X) ]

defadd(self):
return [ ARMEMPTY() , ONTABLE(self.X) ]


defisPredicate(obj):
  predicates = [ON, ONTABLE, CLEAR, HOLDING, ARMEMPTY]
for predicate in predicates:
ifisinstance(obj,predicate):
returnTrue
returnFalse

defisOperation(obj):
  operations = [StackOp, UnstackOp, PickupOp, PutdownOp]
for operation in operations:
ifisinstance(obj,operation):
returnTrue
returnFalse

defarm_status(world_state):
for predicate in world_state:
ifisinstance(predicate, HOLDING):
return predicate
returnARMEMPTY()
```

```
classGoalStackPlanner:

def_init_(self, initial_state, goal_state):
    self.initial_state = initial_state
    self.goal_state = goal_state

defget_steps(self):

#Store Steps
    steps = []

#Program Stack
    stack = []

#World State/Knowledge Base
    world_state = self.initial_state.copy()

#Initially push the goal_state as compound goal onto the stack
    stack.append(self.goal_state.copy())

#Repeat until the stack is empty
whilelen(stack)!=0:

#Get the top of the stack
      stack_top = stack[-1]

#If Stack Top is Compound Goal, push its unsatisfied goals onto stack
iftype(stack_top) is list:
        compound_goal = stack.pop()
for goal in compound_goal:
if goal notin world_state:
            stack.append(goal)

#If Stack Top is an action
elifisOperation(stack_top):

#Peek the operation
        operation = stack[-1]
```

```
        all_preconditions_satisfied =True

#Check if any precondition is unsatisfied and push it onto program stack
for predicate in operation.delete():
if predicate notin world_state:
            all_preconditions_satisfied =False
            stack.append(predicate)

#If all preconditions are satisfied, pop operation from stack and execute it
if all_preconditions_satisfied:

           stack.pop()
           steps.append(operation)

for predicate in operation.delete():
            world_state.remove(predicate)
for predicate in operation.add():
            world_state.append(predicate)


#If Stack Top is a single satisfied goal
elif stack_top in world_state:
        stack.pop()

#If Stack Top is a single unsatisfied goal
else:
        unsatisfied_goal = stack.pop()

#Replace Unsatisfied Goal with an action that can complete it
        action = unsatisfied_goal.get_action(world_state)

        stack.append(action)
#Push Precondition on the stack
for predicate in action.precondition():
if predicate notin world_state:
            stack.append(predicate)

return steps
```

```
if__name__=='__main__':
  initial_state = [
ON('B','A'),
ONTABLE('A'),ONTABLE('C'),ONTABLE('D'),
CLEAR('B'),CLEAR('C'),CLEAR('D'),
ARMEMPTY()
  ]

  goal_state = [
ON('B','D'),ON('C','A'),
ONTABLE('D'),ONTABLE('A'),
CLEAR('B'),CLEAR('C'),
ARMEMPTY()
  ]

  goal_stack =GoalStackPlanner(initial_state=initial_state,
goal_state=goal_state)
  steps = goal_stack.get_steps()
print(steps)
```

Output:

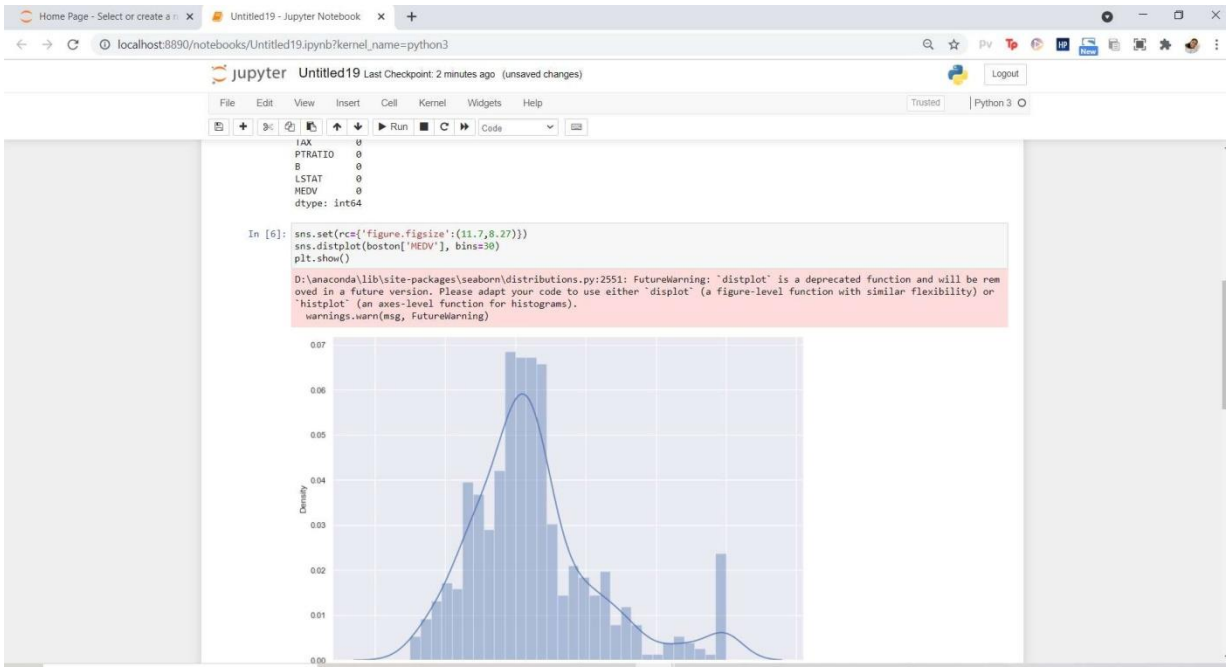

onlinegdb.com/online_python_compiler
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
We are Hiring
Sign Up
Login
Run
Debug
Stop
Share
Save
{} Beautify
Language Python 3
main.py
return steps
if __name__ == '__main__':
initial_state = [
ON('B','A'),
ONTABLE('A'),ONTABLE('C'),ONTABLE('D'),
CLEAR('B'),CLEAR('C'),CLEAR('D'),
ARMEMPTY()
]
goal_state = [
ON('B','D'),ON('C','A'),
ONTABLE('D'),ONTABLE('A'),
CLEAR('B'),CLEAR('C'),
ARMEMPTY()
]
goal_stack = GoalStackPlanner(initial_state=initial_state, goal_state=goal_state)
steps = goal_stack.get_steps()
print(steps)
input
[PICKUP(C), PUTDOWN(C), UNSTACK(B,A), PUTDOWN(B), PICKUP(C), STACK(C,A), PICKUP(B), STACK(B,D)]
...Program finished with exit code 0
Press ENTER to exit console.
About • FAQ • Blog • Terms of Use • Contact Us
• GDB Tutorial • Credits • Privacy
© 2016 - 2021 GDB Online
CD record.pdf
Artificial Intellige....docx
Show all
Type here to search
ENG
12:13 PM
03-05-2021

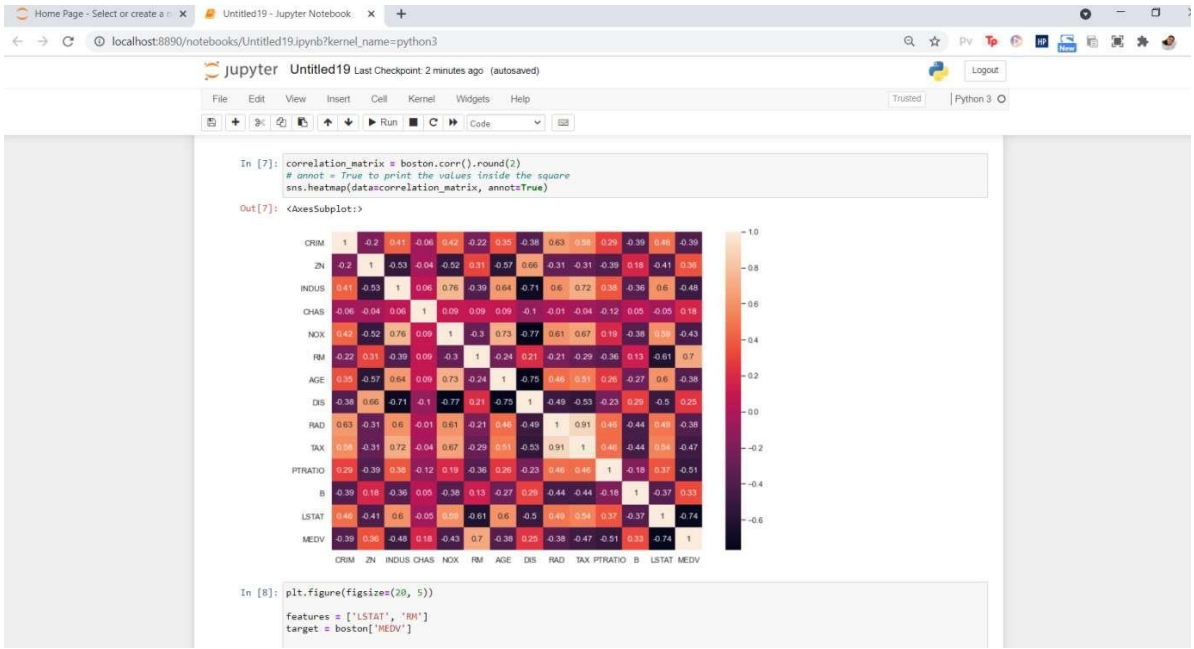
Program:

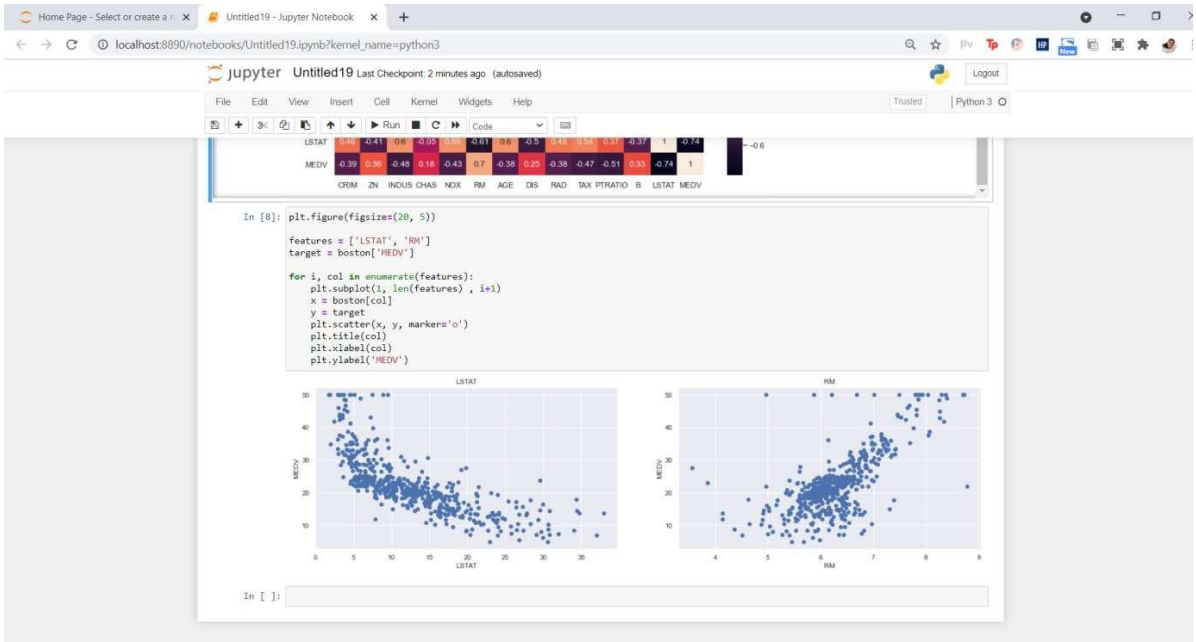
```
import numpy as np
import matplotlib.pyplot as plt

import pandas as pd
import seaborn as sns

%matplotlib inline
from sklearn.datasets import load_boston
boston_dataset = load_boston()
boston = pd.DataFrame(boston_dataset.data,
columns=boston_dataset.feature_names)
boston.head()
boston['MEDV'] = boston_dataset.target
boston.isnull().sum()
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.distplot(boston['MEDV'], bins=30)
plt.show()
correlation_matrix = boston.corr().round(2)
# annot = True to print the values inside the square
sns.heatmap(data=correlation_matrix, annot=True)
plt.figure(figsize=(20, 5))

features = ['LSTAT', 'RM']
target = boston['MEDV']
```

```
for i, col in enumerate(features):
plt.subplot(1, len(features) , i+1)
    x = boston[col]
    y = target
    plt.scatter(x, y, marker='o')
plt.title(col)
plt.xlabel(col)
plt.ylabel('MEDV')
```

Output:


Home Page - Select or create a n
Untitled19 - Jupyter Notebook
localhost:8890/notebooks/Untitled19.ipynb?kernel_name=python3
jupyter Untitled19 Last Checkpoint: 2 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
TAX 0
PTRATIO 0
B 0
LSTAT 0
MEDV 0
dtype: int64
In [6]:
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.distplot(boston['MEDV'], bins=30)
plt.show()
D:\anaconda\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)
Density
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00


Home Page - Select or create a n
Untitled19 - Jupyter Notebook
localhost:8890/notebooks/Untitled19.ipynb?kernel_name=python3
jupyter Untitled19 Last Checkpoint: 2 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [7]:
correlation_matrix = boston.corr().round(2)
# annot = True to print the values inside the square
sns.heatmap(data=correlation_matrix, annot=True)
Out[7]:
<AxesSubplot:>
CRIM ZN INDUS CHAS NOX RM AGE DIS RAD TAX PTRATIO B LSTAT MEDV
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-0.4
-0.6
In [8]:
plt.figure(figsize=(20, 5))
features = ['LSTAT', 'RM']
target = boston['MEDV']

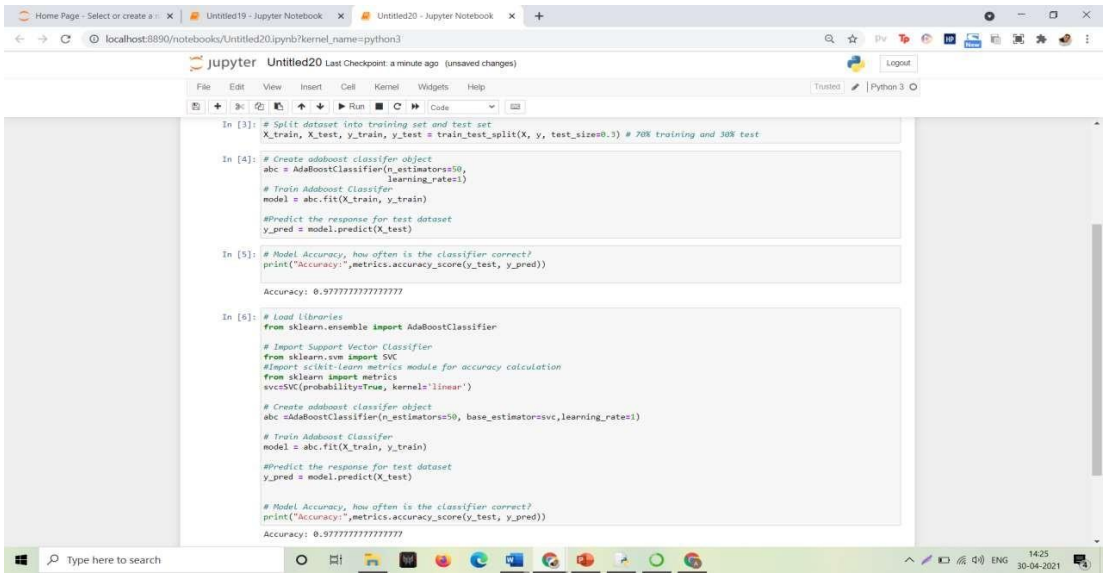

Home Page - Select or create a n
Untitled19 - Jupyter Notebook
localhost:8890/notebooks/Untitled19.ipynb?kernel_name=python3
jupyter Untitled19 Last Checkpoint: 2 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
LSTAT -0.41 0.6 -0.61 0.6 -0.5 0.54 0.37 -0.37 1 -0.74
MEDV -0.39 0.36 -0.48 0.18 -0.43 0.7 -0.38 0.25 -0.38 -0.47 -0.51 0.33 -0.74 1
CRIM ZN INDUS CHAS NOX RM AGE DIS RAD TAX PTRATIO B LSTAT MEDV
0.6
In [8]:
plt.figure(figsize=(20, 5))
features = ['LSTAT', 'RM']
target = boston['MEDV']
for i, col in enumerate(features):
plt.subplot(1, len(features) , i+1)
x = boston[col]
y = target
plt.scatter(x, y, marker='o')
plt.title(col)
plt.xlabel(col)
plt.ylabel('MEDV')
LSTAT
MEDV
RM
In [ ]:

Program:

```
# Load libraries
from sklearn.ensemble import AdaBoostClassifier
from sklearn import datasets
# Import train_test_split function
from sklearn.model_selection import train_test_split
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Load data
iris = datasets.load_iris()
X = iris.data
y = iris.target
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) # 70% training and 30% test
# Create adaboost classifer object
abc = AdaBoostClassifier(n_estimators=50,
                         learning_rate=1)
# Train Adaboost Classifer
model = abc.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = model.predict(X_test)
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
# Load libraries
```

```
from sklearn.ensemble import AdaBoostClassifier

# Import Support Vector Classifier
from sklearn.svm import SVC
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
svc=SVC(probability=True, kernel='linear')

# Create adaboost classifer object
abc =AdaBoostClassifier(n_estimators=50, base_estimator=svc,learning_rate=1)

# Train Adaboost Classifer
model = abc.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = model.predict(X_test)
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Output:


Home Page - Select or create a n
Untitled19 - Jupyter Notebook
Untitled20 - Jupyter Notebook
localhost:8890/notebooks/Untitled20.ipynb?kernel_name=python3
jupyter Untitled20 Last Checkpoint: a minute ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [3]:
Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) # 70% training and 30% test
In [4]:
# Create adaboost classifer object
abc = AdaBoostClassifier(n_estimators=50,
learning_rate=1)
# Train Adaboost Classifer
model = abc.fit(X_train, y_train)
#Predict the response for test dataset
y_pred = model.predict(X_test)
In [5]:
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
Accuracy: 0.9777777777777777
In [6]:
# Load libraries
from sklearn.ensemble import AdaBoostClassifier
# Import Support Vector Classifier
from sklearn.svm import SVC
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
svc=SVC(probability=True, kernel='linear')
# Create adaboost classifer object
abc =AdaBoostClassifier(n_estimators=50, base_estimator=svc,learning_rate=1)
# Train Adaboost Classifer
model = abc.fit(X_train, y_train)
#Predict the response for test dataset
y_pred = model.predict(X_test)
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
Accuracy: 0.9777777777777777
Type here to search
ENG
14:25
30-04-2021

Program

```
# import the necessary libraries

import nltk

import string

import re
```

1.Text Lowercase:

```
def text_lowercase(text):
        return text.lower()
input_str = "Hey, did you know that the summer break is coming? Amazing right !! It's only 5 more days !!"
text_lowercase (input_str)
```

Input: “Hey, did you know that the summer break is coming? Amazing right!! It’s only 5 more days!!”

Output: “hey, did you know that the summer break is coming? amazing right!! it’s only 5 more days!!”

2. Remove numbers:

```
# Remove numbers
def remove_numbers(text):
        result = re.sub(r'\d+', '', text)
        return result
input_str = "There are 3 balls in this bag, and 12 in the other one."
remove_numbers(input_str)
```

Input: “There are 3 balls in this bag, and 12 in the other one.”

Output: ‘There are balls in this bag, and in the other one.’

3.Convert numbers into words

```
# import the inflect library
import inflect
p = inflect.engine()
```

```
# convert number into words
def convert_number(text):
    # split string into list of words
    temp_str = text.split()
    # initialise empty list
    new_string = []

    for word in temp_str:
            # if word is a digit, convert the digit
            # to numbers and append into the new_string list
            if word.isdigit():
                    temp = p.number_to_words(word)
                    new_string.append(temp)

            # append the word as it is
            else:
                    new_string.append(word)

    # join the words of new_string to form a string
    temp_str = ' '.join(new_string)
    return temp_str

input_str = 'There are 3 balls in this bag, and 12 in the other one.'
convert_number(input_str)
```

Input: “There are 3 balls in this bag, and 12 in the other one.”

Output: “There are three balls in this bag, and twelve in the other one.”

4.Remove punctuation:

```
# remove punctuation
def remove_punctuation(text):
    translator = str.maketrans('', '', string.punctuation)
    return text.translate(translator)

input_str = "Hey, did you know that the summer break is coming? Amazing right
!! It's only 5 more days !!"
remove_punctuation(input_str)
```

Input: “Hey, did you know that the summer break is coming? Amazing right!! It’s only 5 more days!!”
Output: “Hey did you know that the summer break is coming Amazing right Its only 5 more days”

5. Remove whitespaces:

```
# remove whitespace from text
def remove_whitespace(text):
    return " ".join(text.split())

input_str = " we don't need the given questions"
remove_whitespace(input_str)
```

Input: " we don't need the given questions"
Output: "we don't need the given questions"

6. Remove default stopwords:

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

# remove stopwords function
def remove_stopwords(text):
    stop_words = set(stopwords.words("english"))
    word_tokens = word_tokenize(text)
    filtered_text = [word for word in word_tokens if word not in stop_words]
    return filtered_text

example_text = "This is a sample sentence and we are going to remove the stopwords from this."
remove_stopwords(example_text)
```

The NLTK library has a set of stopwords and we can use these to remove stopwords from our text and return a list of word tokens.

```
LIST OF ENGLISH STOPWORDS IN NLTK:

their, few, wasn't, has, m, or, did, isn, very, themselves, you've, you'd, do,
between, other, t, shan, yourself, does, ours, i, it, should, what, himself, so
me, itself, there, weren, most, her, mustn, hers, doesn, won, doesn't, hasn, s,
y, wouldn't, didn't, him, couldn, after, a, will, ain, than, for, being, which,
during, ll, my, isn't, its, any, hadn't, his, then, don, of, shouldn't, out, ou
r, have, such, o, nor, too, re, should've, needn't, same, she's, but, weren't,
all, against, down, don't, can, you, under, where, wouldn, only, been, aren't,
haven, that, doing, if, up, d, needn, ma, yours, shan't, wasn, because, about,
those, he, are, was, at, hasn't, over, until, had, with, you're, below, have
n't, mightn, here, own, off, both, whom, while, as, ourselves, they, further, m
ightn't, these, from, to, them, she, who, were, more, am, why, your, aren, had
n, in, won't, yourselves, no, me, didn, an, so, before, is, on, now, each, how,
be, theirs, shouldn, mustn't, above, herself, just, you'll, the, through, agai
n, once, having, by, when, myself, we, it's, this, that'll, couldn't, ve, and,
into, not,
```

Input: "This is a sample sentence and we are going to remove the stopwords from this"
Output: ['This', 'sample', 'sentence', 'going', 'remove', 'stopwords']

7. Stemming:

Stemming is the process of getting the root form of a word. Stem or root is the part to which inflectional affixes (-ed, -ize, -de, -s, etc.) are added. The stem of a word is created by removing the prefix or suffix of a word. So, stemming a word may not result in actual words.

Example:

books ---> book
looked ---> look
denied ---> deni
flies ---> fli

	words	stemmed words
0	connect	connect
1	connected	connect
2	connection	connect
3	connections	connect
4	connects	connect

	words	stemmed words
0	friend	friend
1	friends	friend
2	friended	friend
3	friendly	friendli

```
from nltk.stem.porter import PorterStemmer
from nltk.tokenize import word_tokenize
stemmer = PorterStemmer()

# stem words in the list of tokenized words
def stem_words(text):
    word_tokens = word_tokenize(text)
    stems = [stemmer.stem(word) for word in word_tokens]
    return stems
```

```
text = 'data science uses scientific methods algorithms and many types of
processes'
stem_words(text)
```

Input: ‘data science uses scientific methods algorithms and many types of processes’
Output: [‘data’, ‘scienc’, ‘use’, ‘scientif’, ‘method’, ‘algorithm’, ‘and’, ‘mani’, ‘type’, ‘of’, ‘process’]

8. Lemmatization:

Lemmatization also converts a word to its root form. The only difference is that lemmatization ensures that the root word belongs to the language.

```
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
lemmatizer = WordNetLemmatizer()
# lemmatize string
def  lemmatize_word(text):
    word_tokens = word_tokenize(text)
    # provide context i.e. part-of-speech
    lemmas = [lemmatizer.lemmatize(word, pos ='v') for word in word_tokens]
    return lemmas

text = 'data science uses scientific methods algorithms and many types of
processes'
lemmatize_word(text)
```

Input: ‘data science uses scientific methods algorithms and many types of processes’
Output: [‘data’, ‘science’, ‘use’, ‘scientific’, ‘methods’, ‘algorithms’, ‘and’, ‘many’, ‘type’, ‘of’, ‘process’]

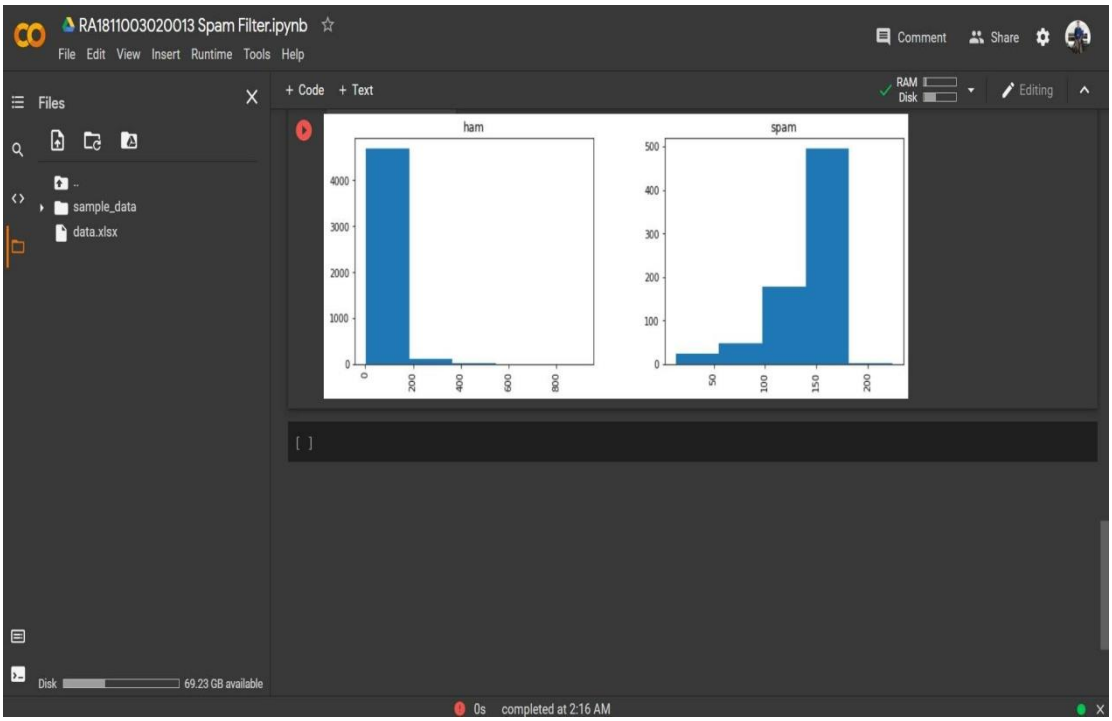
Program:

```
1 import pandas as pd
2 import numpy as np
3 import string
4 import seaborn as sns
5 import matplotlib.pyplot as plt
6 from nltk.corpus import stopwords
7 from sklearn.feature_extraction.text import CountVectorizer
8 from sklearn.feature_extraction.text import TfidfTransformer
9 from sklearn.model_selection import train_test_split
10 from sklearn.svm import SVC
11 from collections import Counter
12 from sklearn.metrics import classification_report,confusion_matrix
13 from sklearn.model_selection import GridSearchCV
14 %matplotlib inline
15 # Load data
16 data = pd.read_excel('data.xlsx')
17 # Rename names columns
18 data.columns = ['label', 'messages']
19 data["length"] = data["messages"].apply(len)
20 data.sort_values(by='length', ascending=False).head(10)
21 data.hist(column = 'length', by ='label',figsize=(12,4), bins = 5)
22 def transform_message(message):
23 message_not_punc = [] # Message without punctuation
24 i = 0
25 for punctuation in message:
```

```
26 if punctuation not in string.punctuation:
27 message_not_punc.append(punctuation)
28 # Join words again to form the string.
29 message_not_punc = ''.join(message_not_punc)
30 # Remove any stopwords for message_not_punc, but first we should
32 # to transform this into the list.
33 message_clean = list(message_not_punc.split(" "))
34 while i <= len(message_clean):
35 for mess in message_clean:
36 if mess.lower() in stopwords.words('english'):
message_clean.remove(mess)
38 i =i +1
39 return message_clean
40 vectorization = CountVectorizer(analyzer = transform_message )
41 X = vectorization.fit(data['messages'])
42 X_transform = X.transform([data['messages']])
43 # TF-IDF
44 tfidf_transformer = TfidfTransformer().fit(X_transform)
45 X_tfidf = tfidf_transformer.transform(X_transform)
46  print(X_tfidf.shape)
47 # Classification Model
48 X_train, X_test, y_train, y_test = train_test_split(X_tfidf, data['messages'],
test_size=0.30, random_state = 50)
49 clf = SVC(kernel='linear').fit(X_train, y_train)
50 # Test model
51 predictions = clf.predict(X_test)
```

```
print('predicted', predictions)
# Is our model reliable?
print (classification_report(y_test, predictions))
print(confusion_matrix(y_test,predictions))
```

Output:


RA1811003020013 Spam Filter.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
Files
+ Code
+ Text
RAM
Disk
Editing
sample_data
data.xlsx
ham
spam
[ ]
Disk
69.23 GB available
0s
completed at 2:16 AM

Program:

```
import json
from typing import Tuple, List

import cv2
import editdistance
from path import Path

from dataloader_iam import DataLoaderIAM, Batch
from model import Model, DecoderType
from preprocessor import Preprocessor


class FilePaths:
    """Filenames and paths to data."""
    fn_char_list = '../model/charList.txt'
    fn_summary = '../model/summary.json'
    fn_corpus = '../data/corpus.txt'


def get_img_height() -> int:
    """Fixed height for NN."""
    return 32


def get_img_size(line_mode: bool = False) -> Tuple[int, int]:
    """Height is fixed for NN, width is set according to training mode
(single words or text lines)."""
    if line_mode:
        return 256, get_img_height()
    return 128, get_img_height()


def write_summary(average_train_loss: List[float], char_error_rates:
List[float], word_accuracies: List[float]) -> None:
    """Writes training summary file for NN."""
    with open(FilePaths.fn_summary, 'w') as f:
```

```
        json.dump({'averageTrainLoss': average_train_loss, 'charErrorRates':
char_error_rates, 'wordAccuracies': word_accuracies}, f)


def char_list_from_file() -> List[str]:
    with open(FilePaths.fn_char_list) as f:
        return list(f.read())


def train(model: Model,
          loader: DataLoaderIAM,
          line_mode: bool,
          early_stopping: int = 25) -> None:
    """Trains NN."""
    epoch = 0  # number of training epochs since start
    summary_char_error_rates = []
    summary_word_accuracies = []

    train_loss_in_epoch = []
    average_train_loss = []

    preprocessor = Preprocessor(get_img_size(line_mode),
data_augmentation=True, line_mode=line_mode)
    best_char_error_rate = float('inf')  # best validation character error rate
    no_improvement_since = 0  # number of epochs no improvement of
character error rate occurred
    # stop training after this number of epochs without improvement
    while True:
        epoch += 1
        print('Epoch:', epoch)

        # train
        print('Train NN')
        loader.train_set()
        while loader.has_next():
            iter_info = loader.get_iterator_info()
            batch = loader.get_next()
            batch = preprocessor.process_batch(batch)
            loss = model.train_batch(batch)
```

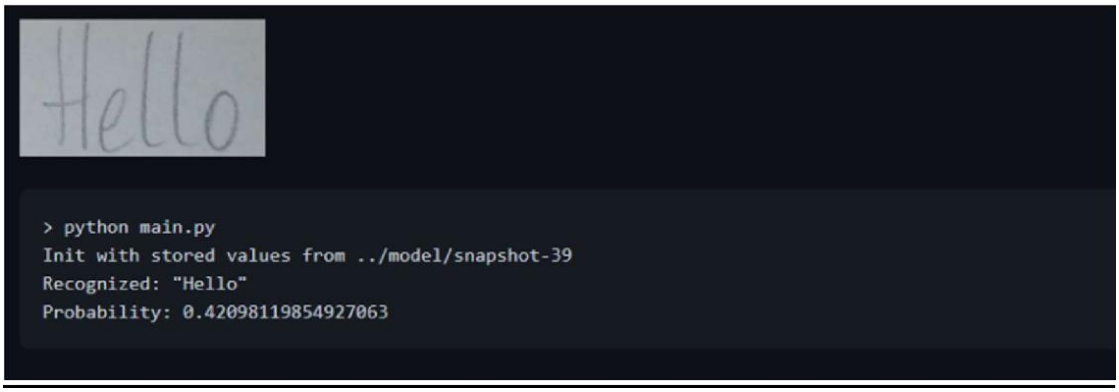
```
            print(f'Epoch: {epoch} Batch: {iter_info[0]}/{iter_info[1]} Loss:
{loss}')
            train_loss_in_epoch.append(loss)

        # validate
        char_error_rate, word_accuracy = validate(model, loader,
line_mode)

        # write summary
        summary_char_error_rates.append(char_error_rate)
        summary_word_accuracies.append(word_accuracy)
        average_train_loss.append((sum(train_loss_in_epoch)) /
len(train_loss_in_epoch))
        write_summary(average_train_loss, summary_char_error_rates,
summary_word_accuracies)

        # reset train loss list
        train_loss_in_epoch = []

        # if best validation accuracy so far, save model parameters
        if char_error_rate < best_char_error_rate:
            print('Character error rate improved, save model')
            best_char_error_rate = char_error_rate
            no_improvement_since = 0
            model.save()
        else:
            print(f'Character error rate not improved, best so far:
{best_char_error_rate * 100.0}%')
            no_improvement_since += 1

        # stop training if no more improvement in the last x epochs
        if no_improvement_since >= early_stopping:
            print(f'No more improvement for {early_stopping} epochs.
Training stopped.')
            break


def validate(model: Model, loader: DataLoaderIAM, line_mode: bool) ->
Tuple[float, float]:
    """Validates NN."""
```

```
    print('Validate NN')
    loader.validation_set()
    preprocessor = Preprocessor(get_img_size(line_mode),
line_mode=line_mode)
    num_char_err = 0
    num_char_total = 0
    num_word_ok = 0
    num_word_total = 0
    while loader.has_next():
        iter_info = loader.get_iterator_info()
        print(f'Batch: {iter_info[0]} / {iter_info[1]}')
        batch = loader.get_next()
        batch = preprocessor.process_batch(batch)
        recognized, _ = model.infer_batch(batch)

        print('Ground truth -> Recognized')
        for i in range(len(recognized)):
            num_word_ok += 1 if batch.gt_texts[i] == recognized[i] else 0
            num_word_total += 1
            dist = editdistance.eval(recognized[i], batch.gt_texts[i])
            num_char_err += dist
            num_char_total += len(batch.gt_texts[i])
            print('[OK]' if dist == 0 else '[ERR:%d]' % dist, '"' +
batch.gt_texts[i] + '"', '->',
                  '"' + recognized[i] + '"')

    # print validation result
    char_error_rate = num_char_err / num_char_total
    word_accuracy = num_word_ok / num_word_total
    print(f'Character error rate: {char_error_rate * 100.0}%. Word
accuracy: {word_accuracy * 100.0}%.')
    return char_error_rate, word_accuracy


def infer(model: Model, fn_img: Path) -> None:
    """Recognizes text in image provided by file path."""
    img = cv2.imread(fn_img, cv2.IMREAD_GRAYSCALE)
    assert img is not None
```

```
    preprocessor = Preprocessor(get_img_size(), dynamic_width=True,
padding=16)
    img = preprocessor.process_img(img)

    batch = Batch([img], None, 1)
    recognized, probability = model.infer_batch(batch, True)
    print(f'Recognized: "{recognized[0]}"')
    print(f'Probability: {probability[0]}')


def parse_args() -> argparse.Namespace:
    """Parses arguments from the command line."""
    parser = argparse.ArgumentParser()

    parser.add_argument('--mode', choices=['train', 'validate', 'infer'],
default='infer')
    parser.add_argument('--decoder', choices=['bestpath', 'beamsearch',
'wordbeamsearch'], default='bestpath')
    parser.add_argument('--batch_size', help='Batch size.', type=int,
default=100)
    parser.add_argument('--data_dir', help='Directory containing IAM
dataset.', type=Path, required=False)
    parser.add_argument('--fast', help='Load samples from LMDB.',
action='store_true')
    parser.add_argument('--line_mode', help='Train to read text lines
instead of single words.', action='store_true')
    parser.add_argument('--img_file', help='Image used for inference.',
type=Path, default='../data/word.png')
    parser.add_argument('--early_stopping', help='Early stopping epochs.',
type=int, default=25)
    parser.add_argument('--dump', help='Dump output of NN to CSV
file(s).', action='store_true')

    return parser.parse_args()


def main():
    """Main function."""

    # parse arguments and set CTC decoder
```

```
    args = parse_args()
    decoder_mapping = {'bestpath': DecoderType.BestPath,
                       'beamsearch': DecoderType.BeamSearch,
                       'wordbeamsearch': DecoderType.WordBeamSearch}
    decoder_type = decoder_mapping[args.decoder]

    # train the model
    if args.mode == 'train':
        loader = DataLoaderIAM(args.data_dir, args.batch_size,
fast=args.fast)
        # when in line mode, take care to have a whitespace in the char list
        char_list = loader.char_list
        if args.line_mode and ' ' not in char_list:
            char_list = [' '] + char_list
        # save characters and words
        with open(FilePaths.fn_char_list, 'w') as f:
            f.write(''.join(char_list))
        with open(FilePaths.fn_corpus, 'w') as f:
            f.write(' '.join(loader.train_words + loader.validation_words))
        model = Model(char_list, decoder_type)
        train(model, loader, line_mode=args.line_mode,
early_stopping=args.early_stopping)
    # evaluate it on the validation set
    elif args.mode == 'validate':
        loader = DataLoaderIAM(args.data_dir, args.batch_size,
fast=args.fast)
        model = Model(char_list_from_file(), decoder_type,
must_restore=True)
        validate(model, loader, args.line_mode)
    # infer text on test image
    elif args.mode == 'infer':
        model = Model(char_list_from_file(), decoder_type,
must_restore=True, dump=args.dump)
        infer(model, args.img_file)

if__name__== '_main_':
    main()
```

Output :


Hello
> python main.py
Init with stored values from ../model/snapshot-39
Recognized: "Hello"
Probability: 0.42098119854927063