

# AI LAB PROGRAMS

## OUTPUT SCREENSHOTS

### 1. Implementation of Tic-Tac-Toe game

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/sumuk/AppData/Local/Programs/Python/Python38/ailaboutputs.py
Player is [O] and computer is [X]
  | |
  | |
  | |
  | |
  | |
# Make your move ! [1-9] : 3
X | | O
  | |
  | |
  | |
# Make your move ! [1-9] : 7
X | | O
  | X |
O | |
  | |
# Make your move ! [1-9] : 9
X | | O
  | X | X
O | | O
  | |
# Make your move ! [1-9] : 8
X | | O
  | X | X
O | O | O
  | |
*** Congratulations ! You won ! ***
>>> |
```

## 2. Solving 8 puzzle problem

1 2 3  
5 6 0  
7 8 4

1 2 3  
5 0 6  
7 8 4

1 2 3  
0 5 6  
7 8 4

1 2 3  
7 5 6  
0 8 4

1 2 3  
7 5 6  
8 0 4

1 2 3  
7 5 6  
8 4 0

1 2 3  
7 4 5  
0 8 6

1 2 3  
7 5 0  
8 4 6

1 2 3  
0 4 5  
7 8 6

1 2 3  
7 0 5  
8 4 6

1 2 3  
4 0 5  
7 8 6

1 2 3  
7 0 5  
8 4 6

1 2 3  
4 5 0  
7 8 6

1 2 3  
7 4 5  
8 0 6

1 2 3  
4 5 6  
7 8 0

### 3. Implementation of vacuum cleaner agent

Python 3.8.2 Shell

File Edit Shell Debug Options Window Help

Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.

>>>

>>>

= RESTART: C:/Users/sumuk/AppData/Local/Programs/Python/Python38/ailaboutputs.py

{'A': 0, 'B': 0}

Vacuum is randomly placed at Location A.

Moving to Location B...

{'A': 0, 'B': 0}

Performance Measurement: -1

>>>

= RESTART: C:/Users/sumuk/AppData/Local/Programs/Python/Python38/ailaboutputs.py

{'A': 1, 'B': 0}

Vacuum is randomly placed at Location A.

Location A is Dirty.

Location A has been Cleaned.

Moving to Location B...

{'A': 0, 'B': 0}

Performance Measurement: 0

>>>

### 4. Implementation of A\* search algorithm

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

D
[5, 3, 4]      R
[2, 6, 8]      [2, 5, 4]
[1, 0, 7]      [1, 3, 6]
               [7, 8, 0]

R
[5, 3, 4]      U
[2, 6, 8]      [2, 5, 4]
[1, 7, 0]      [1, 3, 0]
               [7, 8, 6]

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

L
[5, 3, 4]      I
[2, 0, 6]      [2, 0, 5]
[1, 7, 8]      [1, 3, 4]
               [7, 8, 6]

U
[5, 0, 4]      D
[2, 3, 6]      [2, 3, 5]
[1, 7, 8]      [1, 0, 4]
               [7, 8, 6]

L
[0, 5, 4]      R
[2, 3, 6]      [2, 3, 5]
[1, 7, 8]      [1, 4, 0]
               [7, 8, 6]

D
[2, 5, 4]      U
[0, 3, 6]      [2, 3, 0]
[1, 7, 8]      [1, 4, 5]
               [7, 8, 6]

D
[2, 5, 4]      L
[1, 3, 6]      [2, 0, 3]
[0, 7, 8]      [1, 4, 5]
               [7, 8, 6]

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

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

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

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

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

>>>
```

## 5. Implementing iterative deepening search to solve 8 puzzle problem

0 1 2	1 2 0	1 2 5	<u>1 2 5</u>	1 2 5	1 2 5	1 2 5	1 2 5	2 0 5	2 3 5	1 0 2
3 4 5	3 4 5	3 4 0	3 0 8	3 4 0	3 4 0	3 4 8	3 0 8	1 3 8	6 1 8	3 4 5
6 7 8	6 7 8	6 7 8	6 4 7	6 7 8	6 7 8	6 7 0	6 4 7	6 4 7	4 0 7	6 7 8
1 0 2	1 0 2	1 2 0	1 2 5	1 2 0	1 2 0	1 2 5	1 2 5	0 2 5	2 3 5	0 1 2
3 4 5	3 4 5	3 4 5	3 4 8	3 4 5	3 4 5	3 4 0	3 4 8	1 3 8	6 1 8	3 4 5
6 7 8	6 7 8	6 7 8	6 0 7	6 7 8	6 7 8	6 7 8	6 0 7	6 4 7	0 4 7	6 7 8
0 1 2	0 1 2	1 0 2	1 2 5	1 0 2	1 0 2	1 2 0	1 2 5	1 2 5	2 3 5	2 3 0
3 4 5	3 4 5	3 4 5	3 4 8	3 4 5	3 4 5	3 4 5	3 4 8	0 3 8	0 1 8	6 1 5
6 7 8	6 7 8	6 7 8	6 7 0	6 7 8	6 7 8	6 7 8	6 7 0	6 4 7	6 4 7	4 7 8
1 2 0	1 2 5	0 1 2	1 2 5	0 1 2	0 1 2	1 0 2	1 2 5	1 2 5	2 3 5	2 3 5
3 4 5	3 4 8	3 4 5	3 4 0	3 4 5	3 4 5	3 4 5	3 4 0	3 0 8	1 0 8	6 1 0
6 7 8	6 0 7	6 7 8	6 7 8	6 7 8	6 7 8	6 7 8	6 7 8	6 4 7	6 4 7	4 7 8
1 0 2	1 2 5	1 2 5	1 2 0	2 3 5	2 3 5	0 1 2	1 2 0	1 2 5	2 0 5	2 3 5
3 4 5	3 4 8	0 3 8	3 4 5	0 1 8	6 1 8	3 4 5	3 4 5	3 4 8	1 3 8	6 1 8
6 7 8	6 7 0	6 4 7	6 7 8	6 4 7	0 4 7	6 7 8	6 7 8	6 0 7	6 4 7	4 7 0
0 1 2	1 2 5	1 2 5	1 0 2	2 3 5	2 3 5	2 3 5	1 0 2	1 2 5	0 2 5	2 3 5
3 4 5	3 4 0	3 0 8	3 4 5	1 0 8	0 1 8	6 1 8	3 4 5	3 4 8	1 3 8	6 1 8
6 7 8	6 7 8	6 4 7	6 7 8	6 4 7	6 4 7	4 0 7	6 7 8	6 7 0	6 4 7	4 0 7
1 2 5	1 2 0	1 2 5	0 1 2	2 0 5	2 3 5	2 3 5	0 1 2	1 2 5	1 2 5	2 3 5
3 4 0	3 4 5	3 4 8	3 4 5	1 3 8	1 0 8	6 1 8	3 4 5	3 4 0	0 3 8	6 1 8
6 7 8	6 7 8	6 0 7	6 7 8	6 4 7	6 4 7	0 4 7	6 7 8	6 7 8	6 4 7	0 4 7
1 2 0	1 0 2	1 2 5	2 3 5	0 2 5	2 0 5	2 3 5	2 3 5	1 2 0	1 2 5	2 3 5
3 4 5	3 4 5	3 4 8	1 0 8	1 3 8	1 3 8	0 1 8	6 1 8	3 4 5	3 0 8	0 1 8
6 7 8	6 7 8	6 7 0	6 4 7	6 4 7	6 4 7	6 4 7	4 7 0	6 7 8	6 4 7	6 4 7
1 0 2	0 1 2	1 2 5	2 0 5	1 2 5	0 2 5	2 3 5	2 3 5	1 0 2	1 2 5	2 3 5
3 4 5	3 4 5	3 4 0	1 3 8	0 3 8	1 3 8	1 0 8	6 1 8	3 4 5	3 4 8	1 0 8
6 7 8	6 7 8	6 7 8	6 4 7	6 4 7	6 4 7	6 4 7	4 0 7	6 7 8	6 0 7	6 4 7
0 1 2	1 2 5	1 2 0	0 2 5	1 2 5	1 2 5	2 0 5	2 3 5	0 1 2	1 2 5	2 0 5
3 4 5	3 0 8	3 4 5	1 3 8	3 0 8	0 3 8	1 3 8	6 1 8	3 4 5	3 4 8	1 3 8
6 7 8	6 4 7	6 7 8	6 4 7	6 4 7	6 4 7	6 4 7	0 4 7	6 7 8	6 7 0	6 4 7
1 2 5	1 2 5	1 0 2	1 2 5	1 2 5	1 2 5	0 2 5	2 3 5	2 3 5	1 2 5	0 2 5
3 4 8	3 4 8	3 4 5	0 3 8	3 4 8	3 0 8	1 3 8	0 1 8	6 1 0	3 4 0	1 3 8
6 7 0	6 0 7	6 7 8	6 4 7	6 0 7	6 4 7	6 4 7	6 4 7	4 7 8	6 7 8	6 4 7
1 2 5	1 2 5	0 1 2	1 2 5	1 2 5	1 2 5	1 2 5	2 3 5	2 3 5	1 2 0	1 2 5
3 4 0	3 4 8	3 4 5	3 0 8	3 4 8	3 4 8	0 3 8	1 0 8	6 1 8	3 4 5	0 3 8
6 7 8	6 7 0	6 7 8	6 4 7	6 7 0	6 0 7	6 4 7	6 4 7	4 7 0	6 7 8	6 4 7

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

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

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

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

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

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

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

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

1 2 5
3 4 0
6 7 8

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

1 2 0
3 4 5
6 7 8

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

1 0 2
3 4 5
6 7 8

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

0 1 2
3 4 5
6 7 8

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

Total number of moves: 209  
Total searching time: 0.13 seconds

## 6. Create a knowledge base using propositional logic and show that the given query entails the knowledge base or not

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/sumuk/AppData/Local/Programs/Python/Python38/ailaboutputs.py
Enter rule :pvq
Enter the Query : p
*****Truth Table Reference*****
kb alpha
*****
True True
-----
False False
-----
True False
-----
The Knowledge Base does not entail query
>>>
= RESTART: C:/Users/sumuk/AppData/Local/Programs/Python/Python38/ailaboutputs.py
Enter rule :p^q
Enter the Query : p
*****Truth Table Reference*****
kb alpha
*****
True True
-----
False False
-----
False False
-----
False True
-----
The Knowledge Base entails query
>>> |
```

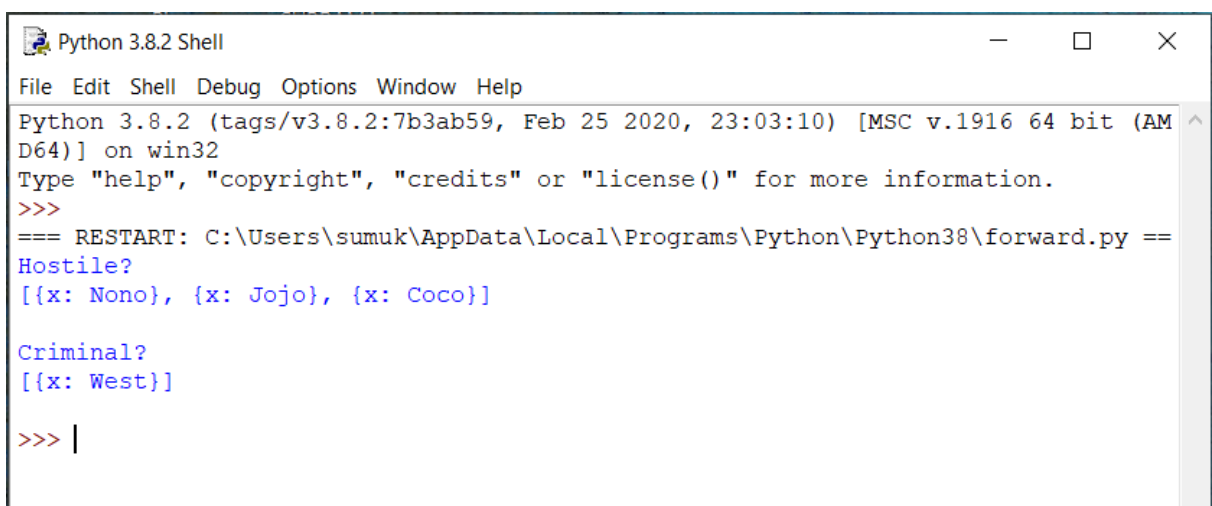
## 7. Convert the given first order logic statement into conjunctive normal form (CNF)

```
>>>
= RESTART: C:/Users/sumuk/AppData/Local/Programs/Python/Python38/fol_to_cnf.py =
^ for and, + for or, ! for not, > for implies, = for biconditional
Enter the expressiona>(b^c)
Applying implication elimination
!(a)+((b^c))
>>>
= RESTART: C:/Users/sumuk/AppData/Local/Programs/Python/Python38/fol_to_cnf.py =
^ for and, + for or, ! for not, > for implies, = for biconditional
Enter the expressiona>(b+c)
Applying implication elimination
!(a)+((b+c))
>>>
```

## 8. Implementation of unification in first order logic

```
= RESTART: C:/Users/sumuk/AppData/Local/Programs/Python/Python38/unificationAI.py
=====PROGRAM FOR UNIFICATION=====
Enter Number of Predicates:2
Enter Predicate 1 :
p
Enter No.of Arguments for Predicate p :
2
Enter argument 1 :
a
Enter argument 2 :
b
Enter Predicate 2 :
p
Enter No.of Arguments for Predicate p :
2
Enter argument 1 :
c
Enter argument 2 :
b
=====PREDICATES ARE=====
p (a,b)
p (c,b)
=====SUBSTITUTION IS=====
c / a
Do you want to continue(y/n): |
```

## 9. Create a knowledge base consisting of first order logic statements and prove the query using forward reasoning



```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=== RESTART: C:\Users\sumuk\AppData\Local\Programs\Python\Python38\forward.py ==
Hostile?
[{x: Nono}, {x: Jojo}, {x: Coco}]

Criminal?
[{x: West}]

>>> |
```

10. Demonstrate decision tree learning for a given set of training examples and test data

```
Dataset Length: 625
Dataset Shape: (625, 5)
Dataset:      0  1  2  3  4
0  B  1  1  1  1
1  R  1  1  1  2
2  R  1  1  1  3
3  R  1  1  1  4
4  R  1  1  1  5
Results Using Entropy:
Predicted values:
['R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L'
 'L' 'R' 'L' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'L' 'L'
 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'L'
 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'L' 'L' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'R' 'R' 'L' 'R' 'L'
 'R' 'R' 'L' 'L' 'L' 'R' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'R' 'R' 'R' 'R' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L'
 'L' 'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R'
 'L' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'R'
 'R' 'R' 'L' 'L' 'L' 'R' 'R' 'R']
Confusion Matrix: [[ 0  6  7]
 [ 0 63 22]
 [ 0 20 70]]
Accuracy : 70.74468085106383

Report :
           precision    recall  f1-score   support

      B      0.00      0.00      0.00        13
      L      0.71      0.74      0.72        85
      R      0.71      0.78      0.74        90

 accuracy          0.71        188
  macro avg      0.47      0.51      0.49        188
 weighted avg      0.66      0.71      0.68        188
```