CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY

DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH

Department of Computer Science & Engineering

Name: Rushik R. Rathod
ID: 20DCS103
Semester: V
Subject Code: CS341
Subject Name: Artificial Intelligence
Academic year: 2022-23

PRACTICAL 1

Aim	Write Programs to demonstrate knowledge of Python Basics.

1.1 Write a program to demonstrate variable creation in python

PROGRAM CODE:

```
name = str(input("Enter your name: "))
print("Name is: " + str(name))
```

OUTPUT:

```
PS F:\B.Tech. Sem 5\AI Practical> python -u "f:\B.Tech. Sem 5\AI Practical\1_1.py"
Enter your name: Rushik Rathod
Name is: Rushik Rathod
```

CONCLUSION:

From this practical, I learned to create variables in Python language.

1.2 Write a program to demonstrate command input in python

PROGRAM CODE:

```
name = input("Enter your name: ")
age = int(input("Enter your age: "))
print("Name is " + name)
```

```
print("Age is " + str(age))
```

OUTPUT:

```
PS F:\B.Tech. Sem 5\AI Practical> python -u "f:\B.Tech. Sem 5\AI Practical\1_2.py"
Enter your name: Rushik Rathod
Enter your age: 20
Name is Rushik Rathod
Age is 20
```

CONCLUSION:

From this practical, I learned to take input from this user and to display it on the output.

1.3 Write a program to demonstrate numbers and strings in python

PROGRAM CODE:

```
a = float(input("Enter your percentage: "))
name = input("Enter your full name: ")
fname = name.split(" ")
print("\nPercentage is " + str(a) + " for first name " + fname[0] + " last name " + fname[1])

string1 = "Hello, Good Morning!"

print("\nSlice Hello from string " + string1 + " is " + string1[0:5])

print("\nReversing string " + string1 + " = " + string1[::-1])
```

OUTPUT:

```
PS F:\B.Tech. Sem 5\AI Practical> python -u "f:\B.Tech. Sem 5\AI Practical\1_3.py"
Enter your percentage: 95
Enter your full name: Rushik Rathod

Percentage is 95.0 for first name Rushik last name Rathod

Slice Hello from string Hello, Good Morning! is Hello

Reversing string Hello, Good Morning! = !gninroM dooG ,olleH
```

CONCLUSION:

From this practical, I learned to use string and numbers in python.

1.4 Write a program to demonstrate operators in python

PROGRAM CODE:

```
a = float(input("Enter number 1: "))
b = float(input("Enter number 2: "))
c = a+b
print(str(a) + " + " + str(b)+ " = " + str(c))
```

OUTPUT:

```
PS F:\B.Tech. Sem 5\AI Practical> python -u "f:\B.Tech. Sem 5\AI Practical\1_4.py"
Enter number 1: 10
Enter number 2: 20
10.0 + 20.0 = 30.0
```

CONCLUSION:
From this practical, I learned to implement operators effectively in the code base.

1.5 Write a program to demonstrate decision making in python

PROGRAM CODE:

```
age = int(input("Enter your age: "))

if age >= 19:
    print("\nYou are eligible to vote.\n")
else:
    print("\nYou are not eligible to vote.\n")
```

OUTPUT:

```
PS F:\B.Tech. Sem 5\AI Practical> python -u "f:\B.Tech. Sem 5\AI Practical\1_5.py"
Enter your age: 20

You are eligible to vote.
```

CONCLUSION:

In this practical we have used if and else loop to make decisions according to the input of a user.

1.6 Write a program to demonstrate control structures in python

PROGRAM CODE:

```
for i in range(0, 9):
    if (i == 5):
        print("Skipping 5 with the help of continue keyword.")
        continue
    print(i)
    if (i == 8):
        print("Break when number count is equal to 8.")
        break
```

OUTPUT:

```
PS F:\B.Tech. Sem 5\AI Practical> python -u "f:\B.Tech. Sem 5\AI Practical\1_6.py"
0
1
2
3
4
Skipping 5 with the help of continue keyword.
6
7
8
Break when number count is equal to 8.
```

CONCLUSION:

In this practical, I learned to control the output using continue and break statements.

1.7 Write a program to demonstrate lists and dictionary in python

PROGRAM CODE:

```
a = [10, 20, 30, 40, 50, 11, 12, 13, 14, 15]
print(a)

print("Sorting the list\n")
a.sort()
print(a)
```

```
print("Removing last element from sorted list\n")
a.pop()
print(a)

print("appending element: 20 at last\n")
a.append(20)
print(a)

print("appending element:120, 200 at last\n")
a.extend([120, 200])
print(a)

b = {
    "Name": "Rushik",
    "Age": 20,
    "Semester": 4
}
print(b)

print("Upadte semester to fifth\n")
b.update({"Semester": 5})
print(b)
print("Adding city : Anand to dictionary\n")
b["city"] = "Anand"
print("Printing value of city " + b["city"])

print("Removing city from dictionary\n")
b.popitem()
print(b)
```

OUTPUT:

```
PS F:\B.Tech. Sem 5\AI Practical> python -u "f:\B.Tech. Sem 5\AI Practical\1_7.py"
[10, 20, 30, 40, 50, 11, 12, 13, 14, 15]
Sorting the list

[10, 11, 12, 13, 14, 15, 20, 30, 40, 50]
Removing last element from sorted list

[10, 11, 12, 13, 14, 15, 20, 30, 40]
appending element: 20 at last

[10, 11, 12, 13, 14, 15, 20, 30, 40, 20]
appending element:120, 200 at last

[10, 11, 12, 13, 14, 15, 20, 30, 40, 20, 120, 200]
{'Name': 'Rushik', 'Age': 20, 'Semester': 4}
Upadte semester to fifth

{'Name': 'Rushik', 'Age': 20, 'Semester': 5}
Adding city : Anand to dictionary

Printing value of city Anand
Removing city from dictionary

{'Name': 'Rushik', 'Age': 20, 'Semester': 5}
```

CONCLUSION:

From this practical, I learned to utilize dictionary and lists effectively to get desired output in python programming language.

PRACTICAL 2

Aim	**Write a program to solve Tower of Hanoi problem in python.**

PROGRM CODE:

```
counter = 0

def TOH(n, s, a, d):
    global counter
    counter = counter + 1
    if (n == 1):
        print(s + " -> " + d)
        return

    TOH(n-1, s, d, a)
    print(s + " -> " + d)
    TOH(n-1, a, s, d)

n = int(input("Enter the number of disks: "))
print("Tower of Hanoi for " + str(n) + " disks")
TOH(n, "s", "a", "d")
print("Cost is: " + str(counter))
```

OUTPUT:

```
PS F:\B.Tech. Sem 5\AI Practical> python -u "f:\B.Tech. Sem 5\AI Practical\2.py"
Enter the number of disks: 3
Tower of Hanoi for 3 disks
s -> d
s -> a
d -> a
s -> d
a -> s
a -> d
s -> d
Cost is: 7
```

CONCLUSION:
From this practical, I learned to implement Tower of Hanoi using recursive approach to the solution with python language.

PRACTICAL 3

Aim	Write a program to solve Monkey Banana problem in Prolog

PROGRAM CODE:

```
move(state(middle,onbox,middle,hasnot),
   grasp,
   state(middle,onbox,middle,has)).
move(state(P,onfloor,P,H),
   climb,
   state(P,onbox,P,H)).
move(state(P1,onfloor,P1,H),
   drag(P1,P2),
   state(P2,onfloor,P2,H)).
move(state(P1,onfloor,B,H),
   walk(P1,P2),
   state(P2,onfloor,B,H)).
canget(state(_,_,_,has)).
canget(State1) :-
   move(State1,_,State2),
   canget(State2).
```

OUTPUT:

```
% f:/B.Tech. Sem 5/AI Practical/3_Monkey_Banana.pl compiled 0.00 sec, 6 clauses
?- canget(state(atdoor,onfloor,atwindow,hasnot)).
true .

?-
% f:/B.Tech. Sem 5/AI Practical/3_Monkey_Banana.pl compiled 0.00 sec, -2 clauses
?- canget(state(atdoor,onbox,atwindow,have)).
false.
```

CONCLUSION:

In this practical we learned about the monkey banana problem through prolog. As we can see from the output that if the monkey is at Door and does not have a banana then he can get it but if he already has it then in that case we can see the program returns false.

PRACTICAL 4

Aim	Write Programs to demonstrate knowledge of Prolog Basics.

4.1 Write a program in prolog to implement simple facts and Queries.

PROGRAM CODE:

```
meeting(ram,kishan).
meeting(kishan,ram).
meeting(lakhan,balram).

brothers(X,Y):-
meeting(X,Y),
meeting(Y,X),write('yes they are brothers').

friendship(X,Y):-
meeting(Y,X),write('they are friends').

friendship(X,Y):-
meeting(X,Y),write('they are friends').
```

OUTPUT:

```
% f:/B.Tech. Sem 5/AI Practical/4_1_Simple_Facts_Queries.pl compiled 0.00 sec, 6
clauses
?- brothers(ram,kishan).
yes they are brothers
true.

?- friendship(lakhan,balram).
they are friends
true.
```

CONCLUSION:

From this practical, I learned to implement simple queries and facts in prolog.

4.2 Write a program in prolog to implement phone list which stores name, phone number and birthdays of friends and family members. Write a query to get a list of people whose birthdays are in the current month.

PROGRAM CODE:

```
store('Ayush',1231231230,'August').
store('Jay',1425258258,'August').
store('Yash',1475456545,'January').
store('Karan',1425875465,'May').
store('Om',1478524752,'July').
store('Vraj',47545447544,'August').
store('Mit',25465582555,'August').

birth(X,Y,Z) :- store(X,Y,Z), write(month).
```

OUTPUT:

```
% f:/B.Tech. Sem 5/AI Practical/4_2_Phonelist.pl compiled 0.00 sec, 8 clauses
?- birth(X,Y,'August').
month
X = 'Ayush',
Y = 1231231230 ;
month
X = 'Jay',
Y = 1425258258 ;
month
X = 'Vraj',
Y = 47545447544 ;
month
X = 'Mit',
Y = 25465582555.
```

CONCLUSION:

From this practical, I learned to store data and fetch them in prolog.

4.3 Write predicates one converts centigrade temperatures to Fahrenheit, the other checks if a temperature is below freezing.

PROGRAM CODE:

```
ctof(C):- F is C*1.8 + 32, F>32 ,write(F).
```

OUTPUT:

```
% f:/B.Tech. Sem 5/AI Practical/4_3_Celsius_To_Fahrenhit.pl compiled 0.00 sec, -2
 clauses
?- ctof(37).
98.60000000000001
true.

?- ctof(50).
122.0
true.
```

CONCLUSION:

In this practical, we learned about the prolog basics, its queries and simple facts. Then we stored the information in the form of phone list and also generated the query as per the requirement. At last we performed a program to convert temperature from Centigrade to Fahrenheit along with the freezing point condition. I learnt how to write a function to check multiple queries subsequently if one turned out to be true.

PRACTICAL 5

Aim	Write Programs to demonstrate knowledge of arithmetic operators in prolog

5.1 Write a program to display Fibonacci series in prolog

PROGRAM CODE:

```
fib(0,0):- !.
fib(1,1):- !.
fib(N,F):-
   N > 1,
   N1 is N-1,
   N2 is N-2,
   fib(N1,F1),
   fib(N2,F2),
   F is (F1+F2).
```

OUTPUT:

```
% f:/B.Tech. Sem 5/AI Practical/5_1_Fibonacci_Series.pl compiled 0.00 sec, 0 clauses
?- fib(0,F).
F = 0.

?- fib(1,F).
F = 1.

?- fib(2,F).
F = 1.

?- fib(3,F).
F = 2.

?- fib(4,F).
F = 3.

?- fib(5,F).
F = 5.

?- fib(6,F).
F = 8.

?- fib(7,F).
F = 13.
```

CONCLUSION:

From this practical, I learned to find fibonacci of a number in prolog.

5.2 Write a program to find factorial of a number in prolog

PROGRAM CODE:

```
factorial(0,1).
factorial(N,F) :-
 N>0,
 N1 is N-1,
 factorial(N1,F1),
 F is N * F1.
```

OUTPUT:

```
% f:/B.Tech. Sem 5/AI Practical/5_2_Factorial_Number.pl compiled 0.00 sec, 0 cla
ses
?- factorial(5,F).
F = 120
```

CONCLUSION:

From this practical, I learned to find factorial of a number in prolog.

PRACTICAL 6

Aim	Write a prolog program for medical diagnosis system of childhood diseases.

PROGRAM CODE:

```
symptom('Parth', 'High Temp').
symptom('Parth', 'Sneezing').
symptom('Parth', 'Sore Throat').

symptom('Manan', 'Itching').
symptom('Manan', 'Blisters').
symptom('Manan', 'Weakness').

symptom('Ayush', 'Yellow Eyes').
symptom('Ayush', 'Abdominal Pain').
symptom('Ayush', 'Vomitting').

symptom('a', 'Fever').
symptom('a', 'Weakness').
symptom('a', 'Sizziness').

symptom('b', 'Fever').
symptom('b', 'Sore Throat').
symptom('b', 'Joint Pain').

diagnose(X, 'Cold') :- symptom(X,'High Temp'), symptom(X, 'Sneezing'), symptom(X, 'Sore
Throat').
diagnose(X, 'Chickenpox'):- symptom(X,'Itching'), symptom(X, 'Blisters'), symptom(X,
'Weakness').
diagnose(X, 'Jaundice') :- symptom(X, 'Yellow Eyes'), symptom(X, 'Abdominal Pain'),
symptom(X, 'Vomitting').
diagnose(X, 'COVID-19') :- symptom(X, 'Fever'), symptom(X, 'Sore Throat'), symptom(X,
'Joint Pain').
diagnose(X, 'Malaria') :- symptom(X, 'Fever'), symptom(X, 'Weakness'), symptom(X,
'Sizziness').
```

OUTPUT:

```
% f:/B.Tech. Sem 5/AI Practical/6_Medical_Diagnosis.pl compiled 0.00 sec, 0 clause
?- diagnose('b',Y).
Y = 'COVID-19' ;
false.

?- diagnose('Ayush',Y).
Y = 'Jaundice' ;
false.

?- diagnose(X,'Cold').
X = 'Parth' ;
false.
```

CONCLUSION:

From this practical, I learned to write a prolog program for medical diagnosis of childhood diseases.

PRACTICAL 7

Aim	**Write a program which contains three predicates: male, female, parent. Make rules for following family relations: father, mother, grandfather, grandmother, brother, sister, uncle, aunt, nephew and niece, cousin.**

PROGRAM CODE:

```
male('Champak').
male('Jethalal').
male('Tapu').
male('Iyer').
male('Parth').

female('Sarla').
female('Daya').
female('Sonu').
female('Babita').
female('Devi').

parent('Champak','Jethalal').
parent('Sarla','Jethalal').
parent('Champak', 'Babita').
parent('Sarla','Babita').

parent('Jethalal','Tapu').
parent('Daya','Tapu').
parent('Jethalal','Sonu').
parent('Daya','Sonu').

parent('Babita','Parth').
parent('Iyer','Parth').
parent('Babita','Devi').
parent('Iyer','Devi').

father(X,Y):-male(X),parent(X,Y).
mother(X,Y):-female(X),parent(X,Y).

brother(X,Y):-male(X),father(Z,X),parent(Z,Y),X\==Y.
sister(X,Y):-female(X),father(Z,X),parent(Z,Y),X\==Y.
grandparent(X,Y):-parent(X,Z),parent(Z,Y).

grandfather(X,Y):-male(X),grandparent(X,Y).
```

```
grandmother(X,Y):-female(X),grandparent(X,Y).

uncle(X,Y):-brother(X,Z),parent(Z,Y).
aunt(X,Y):-sister(X,Z),parent(Z,Y).

niece(X,Y):-female(X),aunt(Y,X).
niece(X,Y):-female(X),uncle(Y,X).

nephew(X,Y):-male(X),aunt(Y,X).
nephew(X,Y):-male(X),uncle(Y,X).

cousin(X,Y):-aunt(Z,X),parent(Z,Y),X\==Y.
cousin(X,Y):-uncle(Z,X),parent(Z,Y), X\==Y.
```

OUTPUT:

```
% f:/B.Tech. Sem 5/AI Practical/7_Family_Tree.pl compiled 0.02 sec, 0 clause
?- uncle(X,Y).
X = 'Jethalal',
Y = 'Parth' ;
X = 'Jethalal',
Y = 'Devi' ;
false.

?- father(X,Y).
X = 'Champak',
Y = 'Jethalal' ;
X = 'Champak',
Y = 'Babita' ;
X = 'Jethalal',
Y = 'Tapu' ;
X = 'Jethalal',
Y = 'Sonu' ;
X = 'Iyer',
Y = 'Parth' ;
X = 'Iyer',
Y = 'Devi' ;
false.

?- sister(X,Y).
X = 'Sonu',
Y = 'Tapu' ;
X = 'Babita',
Y = 'Jethalal' ;
X = 'Devi',
Y = 'Parth' ;
false.
```

CONCLUSION:

From this practical, I learned to build family tree and make relations between those facts..

PRACTICAL 8

Aim	Write a program to perform following operations on lists in prolog.

8.1 Print member of a list
8.2 Write list
8.3 Membership
8.4 Concatenation
8.5 Add an item
8.6 Delete an item
8.7 Sub list
8.8 Permutations
8.9 Append list
8.10 Finding nth element

PROGRAM CODE:

```
list_member(X, [X|_]).
list_member(X, [_|TAIL]) :- list_member(X, TAIL).
list_length([],0).
list_length([_|TAIL],N) :- list_length(TAIL,N1), N is N1 + 1.
print_list([X|TAIL]) :- write(X), nl, print_list(TAIL).
list_concat([],L,L).
list_concat([X1|L1],L2,[X1|L3]) :- list_concat(L1,L2,L3).
list_delete(X, [X], []).
list_delete(X,[X|L1], L1).
list_delete(X, [Y|L2], [Y|L1]) :- list_delete(X,L2,L1).
list_insert(X,L,R) :- list_delete(X,R,L).
list_append(A,T,T) :- list_member(A,T),!.
list_append(A,T,[A|T]).
list_perm([],[]).
list_perm(L,[X|P]) :- list_delete(X,L,L1),list_perm(L1,P).
index_of(X, [X|_], 1).
index_of(X, [_|TAIL], N) :- index_of(X, TAIL, N1), N is N1+1.
element_at(X,[X|_],1).
element_at(X,[_|L],K) :- element_at(X,L,K1), K is K1 + 1.
list_divide([],[],[]).
list_divide([X],[X],[]).
list_divide([X,Y|Tail], [X|List1],[Y|List2]):- list_divide(Tail,List1,List2).
```

OUTPUT:

```
% f:/B.Tech. Sem 5/AI Practical/8_List_Operations.pl compiled 0.00 sec, 22 clause
?- list_member(5,[1,2,3,4,5]).
true ;
false.

?- list_length([1,2,3,4,5,a,b,c],LENGTH).
LENGTH = 8.

?- list_concate([1,2,3],[a,b,c],C).
Correct to: "list_concat([1,2,3],[a,b,c],C)"? yes
C = [1, 2, 3, a, b, c].

?-
% f:/B.Tech. Sem 5/AI Practical/8_List_Operations.pl compiled 0.00 sec, 0 clauses
?- list_member(5,[1,2,3,4,5]).
true ;
false.

?- list_length([1,2,3,4,5,a,b,c],LENGTH).
LENGTH = 8.

?- list_concat([1,2,3],[a,b,c],C).
C = [1, 2, 3, a, b, c].

?- list_delete(2,[1,2,3],C).
C = [1, 3] ;
false.

?- list_insert(a,[1,2,3],R).
R = [a, 1, 2, 3] ;
R = [1, a, 2, 3] ;
R = [1, 2, a, 3] ;
R = [1, 2, 3, a] ;
R = [1, 2, 3, a] ;
false.

?- list_append(a,[1,2,3],R).
R = [a, 1, 2, 3].

?- index_of(b,[a,b,c],I).
I = 2 ;
false.
```

CONCLUSION:

From this practical, I learned to implement various operations on lists using prolog which includes appending, finding nth element, inserting an element and deleting an element, etc.

PRACTICAL 9

Aim	Write a program to demonstrate cut and fail in prolog.
	PROGRAM CODE: a(X) :- b(X),c(X),fail. a(X) :- d(X). b(1). b(4). c(1). c(3). d(4). **OUTPUT:** % f:/B.Tech. Sem 5/AI Practical/9_cut_fail.pl compiled 0.00 sec, 7 clauses ?- a(X). X = 4. **CONCLUSION:** From this practical, I learned to implement cut and fail in prolog.

PRACTICAL 10

Aim	**Design Depth First Search Tree and Breadth First Search Tree for Water-Jug Problem in python.**

PROGRAM CODE:

```
def water_jug_dfs(jug1, jug2, target, current1, current2, visited, done):

    if done == True:
        return
    if current1 == target_water:
        print((current1, 0))
        done = True
        return

    if current2 == target_water:
        print((0, current2))
        done = True
        return

    print((current1, current2))
    if (jug1, current2) not in visited:
        visited.append((jug1, current2))
        water_jug_dfs(jug1, jug2, target, jug1, current2, visited, done)
    if (current1, jug2) not in visited:
        visited.append((current1, jug2))
        water_jug_dfs(jug1, jug2, target, current1, jug2, visited, done)
    if (0, current2) not in visited:
        visited.append((0, current2))
        water_jug_dfs(jug1, jug2, target, 0, current2, visited, done)
    if (current1, 0) not in visited:
        visited.append((current1, 0))
        water_jug_dfs(jug1, jug2, target, current1, 0, visited, done)

    vacancy2 = min(jug2 - current2, current1)
    if (current1 - vacancy2, vacancy2) not in visited:
```

```
        visited.append((current1 - vacancy2, vacancy2))
        water_jug_dfs(jug1, jug2, target, current1 -
                    vacancy2, vacancy2, visited, done)

    vacancy1 = min(jug1 - current1, current2)
    if (vacancy1, current2 - vacancy1) not in visited:
        visited.append((vacancy1, current2 - vacancy1))
        water_jug_dfs(jug1, jug2, target, vacancy1,
                    current2 - vacancy1, visited, done)
    return


def water_jug_bfs(jug1, jug2, target_water, current1, current2):
    queue = []
    queue.append((0, 0))
    visited = [(0, 0)]
    while len(queue) > 0:
        node = queue[0]
        queue.pop(0)

        if node[0] == target_water:
            print((node[0], 0))
            break
        if node[1] == target_water:
            print((0, node[1]))
            break
        print(node)
        if (jug1, node[1]) not in visited:
            visited.append((jug1, node[1]))
            queue.append((jug1, node[1]))
        if (node[0], jug2) not in visited:
            visited.append((node[0], jug2))
            queue.append((node[0], jug2))
        if (0, node[1]) not in visited:
            visited.append((0, node[1]))
            queue.append((0, node[1]))
        if (node[0], 0) not in visited:
```

```
            visited.append((node[0], 0))
            queue.append((node[0], 0))

        vacancy2 = min(jug2 - node[1], node[0])
        if (node[0] - vacancy2, vacancy2) not in visited:
            visited.append((node[0] - vacancy2, vacancy2))
            queue.append((node[0] - vacancy2, vacancy2))

        vacancy1 = min(jug1 - node[0], node[1])
        if (vacancy1, node[1] - vacancy1) not in visited:
            visited.append((vacancy1, node[1] - vacancy1))
            queue.append((vacancy1, node[1] - vacancy1))
    return


jug1 = int(input("Enter the capacity of Jug 1 : "))  # 4, 11
jug2 = int(input("Enter the capacity of Jug 2 : "))  # 3, 7
target_water = int(input("Enter the value to fill : "))  # 2, 6

if jug1 < jug2:
    jug1, jug2 = jug2, jug1

current1 = 0
current2 = 0
print("\nSolution with DFS :")
visited_nodes = [(0, 0)]
done = False
water_jug_dfs(jug1, jug2, target_water, current1,
              current2, visited_nodes, done)

current1 = 0
current2 = 0
print("\nSolution with BFS :")
water_jug_bfs(jug1, jug2, target_water, current1, current2)
```

OUTPUT:

```
Enter the capacity of Jug 1 : 11
Enter the capacity of Jug 2 : 7
Enter the value to fill : 6

Solution with DFS :
(0, 0)
(11, 0)
(11, 7)
(0, 7)
(7, 0)
(7, 7)
(4, 3)
(11, 3)
(0, 3)
(3, 0)
(3, 7)
(7, 4)
(11, 4)
(0, 4)
(4, 0)
(4, 7)
(8, 3)
(8, 7)
(8, 0)
(1, 7)
(1, 0)
(0, 1)
(11, 1)
(0, 6)
(3, 4)
(4, 4)
(1, 3)
```

```
Solution with BFS :
(0, 0)
(11, 0)
(0, 7)
(11, 7)
(4, 7)
(7, 0)
(4, 0)
(7, 7)
(0, 4)
(4, 3)
(11, 4)
(11, 3)
(0, 3)
(3, 0)
(8, 3)
(7, 4)
(3, 7)
(8, 7)
(8, 0)
(4, 4)
(3, 4)
(1, 7)
(1, 3)
(1, 0)
(0, 1)
(11, 1)
(0, 6)
```

CONCLUSION:

From this practical, I learned to write a python program to solve water jug problem using two different approaches: Breadth First Search and Depth First search.

PRACTICAL 11

Aim	Write a program to solve 8 puzzle problem using A*Algorithm in python

PROGRAM CODE:

```
class Node:
    def __init__(self, data, level, fval):
        """ Initialize the node with the data, level of the node and the calculated fvalue """
        self.data = data
        self.level = level
        self.fval = fval

    def generate_child(self):
        """ Generate child nodes from the given node by moving the blank space
            either in the four directions {up,down,left,right} """
        x, y = self.find(self.data, '_')
        """ val_list contains position values for moving the blank space in either of
            the 4 directions [up,down,left,right] respectively. """
        val_list = [[x, y-1], [x, y+1], [x-1, y], [x+1, y]]
        children = []
        for i in val_list:
            child = self.shuffle(self.data, x, y, i[0], i[1])
            if child is not None:
                child_node = Node(child, self.level+1, 0)
                children.append(child_node)
        return children

    def shuffle(self, puz, x1, y1, x2, y2):
        """ Move the blank space in the given direction and if the position value are out
            of limits the return None """
        if x2 >= 0 and x2 < len(self.data) and y2 >= 0 and y2 < len(self.data):
            temp_puz = []
            temp_puz = self.copy(puz)
            temp = temp_puz[x2][y2]
            temp_puz[x2][y2] = temp_puz[x1][y1]
```

```
            temp_puz[x1][y1] = temp
            return temp_puz
        else:
            return None

    def copy(self, root):
        """ Copy function to create a similar matrix of the given node"""
        temp = []
        for i in root:
            t = []
            for j in i:
                t.append(j)
            temp.append(t)
        return temp

    def find(self, puz, x):
        """ Specifically used to find the position of the blank space """
        for i in range(0, len(self.data)):
            for j in range(0, len(self.data)):
                if puz[i][j] == x:
                    return i, j

class Puzzle:
    def __init__(self, size):
        """ Initialize the puzzle size by the specified size,open and closed lists to empty """
        self.n = size
        self.open = []
        self.closed = []

    def accept(self):
        """ Accepts the puzzle from the user """
        puz = []
        for i in range(0, self.n):
            temp = input().split(" ")
            puz.append(temp)
        return puz
```

```
    def f(self, start, goal):
        """ Heuristic Function to calculate hueristic value f(x) = h(x) + g(x) """
        return self.h(start.data, goal)+start.level

    def h(self, start, goal):
        """ Calculates the different between the given puzzles """
        temp = 0
        for i in range(0, self.n):
            for j in range(0, self.n):
                if start[i][j] != goal[i][j] and start[i][j] != '_':
                    temp += 1
        return temp

    def process(self):
        """ Accept Start and Goal Puzzle state"""
        print("Enter the start state matrix ")
        start = self.accept()
        print("Enter the goal state matrix ")
        goal = self.accept()
        start = Node(start, 0, 0)
        start.fval = self.f(start, goal)
        """ Put the start node in the open list"""
        self.open.append(start)
        print("\n\n")
        while True:
            cur = self.open[0]
            print("")
            for i in cur.data:
                for j in i:
                    print(j, end=" ")
                print("")
            """ If the difference between current and goal node is 0 we have reached the goal
node"""
            if(self.h(cur.data, goal) == 0):
                break
            for i in cur.generate_child():
                i.fval = self.f(i, goal)
```

```
            self.open.append(i)
        self.closed.append(cur)
        del self.open[0]

        """ sort the opne list based on f value """
        self.open.sort(key=lambda x: x.fval, reverse=False)
puz = Puzzle(3)
puz.process()
```

OUTPUT:

```
PS F:\B.Tech. Sem 5\AI Practical> python -u "f:\B.Tech. Sem 5\AI Practical\11.py"
Enter the start state matrix
1 2 3
_ 4 6
7 5 8
Enter the goal state matrix
1 2 3
4 5 6
7 8 _




1 2 3
_ 4 6
7 5 8

1 2 3
4 _ 6
7 5 8

1 2 3
4 5 6
7 _ 8

1 2 3
4 5 6
7 8 _
```

CONCLUSION:

From this practical, I learned to solve 8 puzzle problem using A* algorithm in python programming language.

PRACTICAL 12

Aim	**Write a program for game Tic-Tac-Toe using MINIMAX Algorithm in python.**

PROGRAM CODE:

```
player, opponent = 'x', 'o'

def isMovesLeft(board):

    for i in range(3):
        for j in range(3):
            if (board[i][j] == '_'):
                return True
    return False


def evaluate(b):

    # Checking for Rows for X or O victory.
    for row in range(3):
        if (b[row][0] == b[row][1] and b[row][1] == b[row][2]):
            if (b[row][0] == player):
                return 10
            elif (b[row][0] == opponent):
                return -10

    # Checking for Columns for X or O victory.
    for col in range(3):

        if (b[0][col] == b[1][col] and b[1][col] == b[2][col]):

            if (b[0][col] == player):
                return 10
            elif (b[0][col] == opponent):
                return -10
```

```
    # Checking for Diagonals for X or O victory.
    if (b[0][0] == b[1][1] and b[1][1] == b[2][2]):

        if (b[0][0] == player):
            return 10
        elif (b[0][0] == opponent):
            return -10

    if (b[0][2] == b[1][1] and b[1][1] == b[2][0]):

        if (b[0][2] == player):
            return 10
        elif (b[0][2] == opponent):
            return -10

    return 0


def minimax(board, depth, isMax):
    score = evaluate(board)

    if (score == 10):
        return score

    if (score == -10):
        return score

    if (isMovesLeft(board) == False):
        return 0

    if (isMax):
        best = -1000

        for i in range(3):
            for j in range(3):
                if (board[i][j] == '_'):
```

```
                board[i][j] = player

                best = max(best, minimax(board,
                                 depth + 1,
                                 not isMax))

                board[i][j] = '_'
        return best

    else:
        best = 1000

        for i in range(3):
            for j in range(3):
                if (board[i][j] == '_'):

                board[i][j] = opponent
                best = min(best, minimax(board, depth + 1, not isMax))
                board[i][j] = '_'
        return best


def findBestMove(board):
    bestVal = -1000
    bestMove = (-1, -1)

    for i in range(3):
        for j in range(3):

            if (board[i][j] == '_'):
                board[i][j] = player
                moveVal = minimax(board, 0, False)
                board[i][j] = '_'

                if (moveVal > bestVal):
                    bestMove = (i, j)
                    bestVal = moveVal
```

```
    print("The value of the best Move is :", bestVal)
    print()
    return bestMove


# Driver code
board = [
        ['x', 'o', 'x'],
        ['o', 'o', 'x'],
        ['_', '_', '_']
]

bestMove = findBestMove(board)

print("The Optimal Move is...")
print("ROW:", bestMove[0], "\nCOL:", bestMove[1])
```

OUTPUT:

```
The value of the best Move is : 10

The Optimal Move is...
ROW: 2
COL: 2
```

CONCLUSION:

From this practical, I learned to write program to solve Tic-Tac-Toe problem using MINIMAX algorithm in Python programming language.

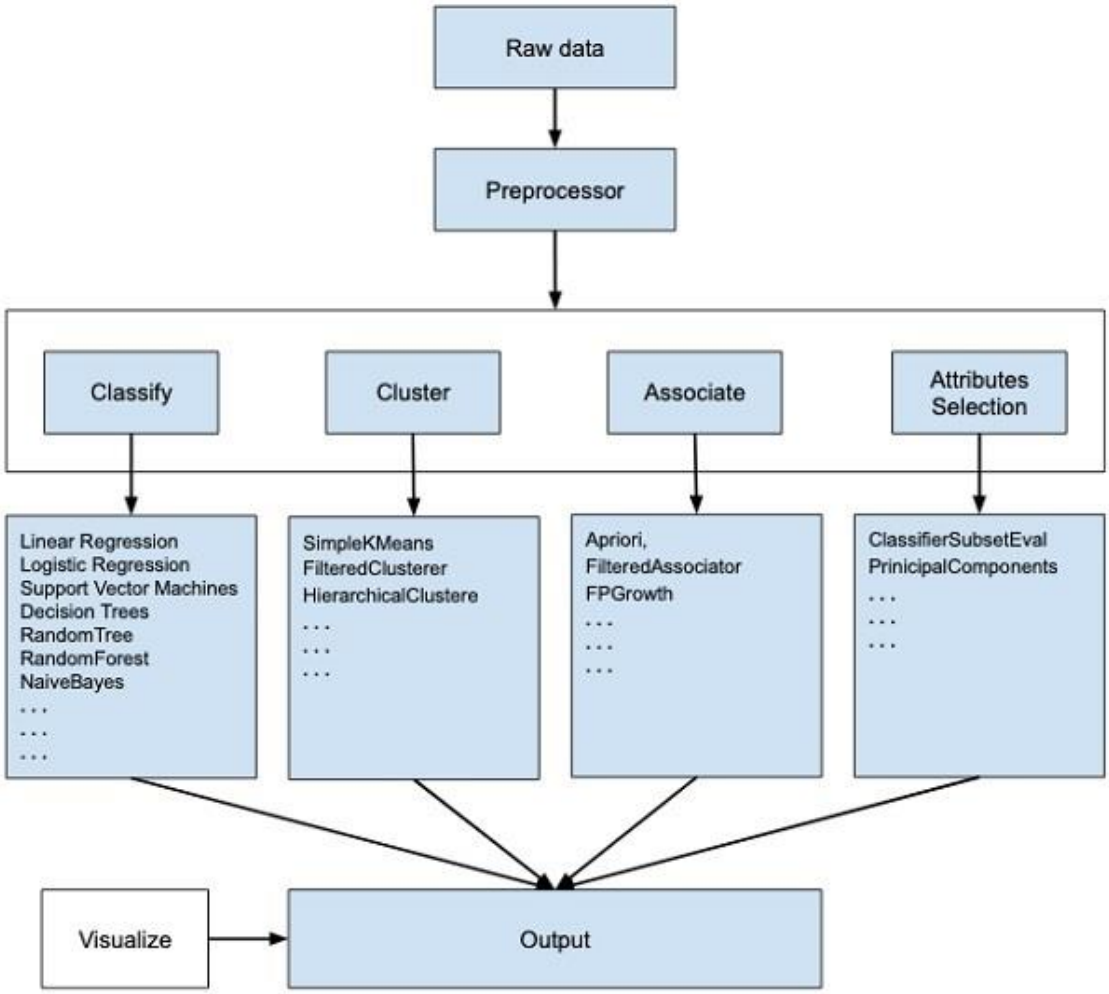
PRACTICAL 13

Aim	**Perform classification on Iris dataset using neural network tools such as WEKA, ORANGE, NEUROINTELLIGENCE, EasyNN.**

THEORY:

WEKA:
WEKA - an open source software provides tools for data preprocessing, implementation of several Machine Learning algorithms, and visualization tools so that you can develop machine learning techniques and apply them to real-world data mining problems.

WEKA offers following functionalities to the developer:



If you observe the beginning of the flow of the image, you will understand that there are many stages in dealing with Big Data to make it suitable for machine learning −

First, you will start with the raw data collected from the field. This data may contain several null values and irrelevant fields. You use the data preprocessing tools provided in WEKA to cleanse the data.

Then, you would save the preprocessed data in your local storage for applying ML algorithms.

Next, depending on the kind of ML model that you are trying to develop you would select one of the options such as **Classify, Cluster**, or **Associate**. The **Attributes Selection** allows the automatic selection of features to create a reduced dataset.

Note that under each category, WEKA provides the implementation of several algorithms. You would select an algorithm of your choice, set the desired parameters and run it on the dataset.

Then, WEKA would give you the statistical output of the model processing. It provides you a visualization tool to inspect the data.

The various models can be applied on the same dataset. You can then compare the outputs of different models and select the best that meets your purpose.

Thus, the use of WEKA results in a quicker development of machine learning models on the whole.

Now that we have seen what WEKA is and what it does, in the next chapter let us learn how to install WEKA on your local computer.

CONCLUSION:

From this practical, I learned about WEKA platform to build Machine Learning models based on Neural Network.

PRACTICAL 14

Aim	Perform sentiment analysis of movie reviews using nltk in python

PROGRAM CODE:

```
import nltk.classify.util
from nltk.classify import NaiveBayesClassifier
from nltk.corpus import movie_reviews
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize


# This is how the Naive Bayes classifier expects the input
def create_word_features(words):
  # Remove all stopwords
  useful_words = [word for word in words if word not in stopwords.words("english")]

  # For each word, we create a dictionary with all the words and True.
  # Why a dictionary? So that words are not repeated. If a word already exists, it won't be
added to the dictionary.
  my_dict = dict([(word, True) for word in useful_words])

  return my_dict


# nltk.download('stopwords')
# nltk.download('movie_reviews')
# nltk.download('punkt')
create_word_features(["python", "is", "better", "than", "r", "and", "r", "is", "better", "than",
"java"])
# We create an empty list called neg_reviews. Next, we loop over all the files in the neg folder.
neg_reviews = []
for fileid in movie_reviews.fileids('neg'):
  # We get all the words in that file.
  words = movie_reviews.words(fileid)
  # Then we use the function we wrote earlier to create word features in the format nltk
expects.
  neg_reviews.append((create_word_features(words), "negative"))

print(neg_reviews[0])
print(len(neg_reviews))

# Let's do the same for the positive reviews. The code is exactly the same:
pos_reviews = []
```

```
for fileid in movie_reviews.fileids('pos'):
   words = movie_reviews.words(fileid)
   pos_reviews.append((create_word_features(words), "positive"))

print(pos_reviews[0])
print(len(pos_reviews))

# We will now create our test and train samples
train_set = neg_reviews[:750] + pos_reviews[:750]
test_set = neg_reviews[750:] + pos_reviews[750:]

print(len(train_set), len(test_set))

# Let’s create our Naive Bayes Classifier, and train it with our training set.
classifier = NaiveBayesClassifier.train(train_set)

# And let’s use our test set to find the accuracy
accuracy = nltk.classify.util.accuracy(classifier, test_set)
print(accuracy * 100)

review_santa = '''

It would be impossible to sum up all the stuff that sucks about this film, so I'll break it down
into what I remember most strongly: a man in an ingeniously fake-looking polar bear costume
(funnier than the "bear" from Hercules in New York); an extra with the most unnatural laugh
you're ever likely to hear; an ex-dope addict martian with tics; kid actors who make sure every
syllable of their lines are slowly and caaarreee-fulll-yyy prrooo-noun-ceeed; a newspaper
headline stating that Santa's been "kidnaped", and a giant robot. Yes, you read that right. A
giant robot.

The worst acting job in here must be when Mother Claus and her elves have been "frozen" by
the "Martians'" weapons. Could they be *more* trembling? I know this was the sixties and
everyone was doped up, but still.
'''
print(review_santa)

words = word_tokenize(review_santa)
words = create_word_features(words)
classifier.classify(words)

review_spirit = '''
Spirited Away' is the first Miyazaki I have seen, but from this stupendous film I can tell he is a
master storyteller. A hallmark of a good storyteller is making the audience empathise or pull
them into the shoes of the central character. Miyazaki does this brilliantly in 'Spirited Away'.
During the first fifteen minutes we have no idea what is going on. Neither does the main
character Chihiro. We discover the world as Chihiro does and it's truly amazing to watch. But
Miyazaki doesn't seem to treat this world as something amazing. The world is filmed just like
```

our workaday world would. The inhabitants of the world go about their daily business as usual as full with apathy as us normal folks. Places and buildings are not greeted by towering establishing shots and majestic music. The fact that this place is amazing doesn't seem to concern Miyazaki.

What do however, are the characters. Miyazaki lingers upon the characters as if they were actors. He infixes his animated actors with such subtleties that I have never seen, even from animation giants Pixar. Twenty minutes into this film and I completely forgot these were animated characters; I started to care for them like they were living and breathing. Miyazaki treats the modest achievements of Chihiro with unashamed bombast. The uplifting scene where she cleanses the River God is accompanied by stirring music and is as exciting as watching gladiatorial combatants fight. Of course, by giving the audience developed characters to care about, the action and conflicts will always be more exciting, terrifying and uplifting than normal, generic action scenes.
'''

```
print(review_spirit)

words = word_tokenize(review_spirit)
words = create_word_features(words)
classifier.classify(words)
```

OUTPUT:

```
In [13]: words = word_tokenize(review_spirit)
         words = create_word_features(words)
         classifier.classify(words)

Out[13]: 'positive'
```

CONCLUSION:

In this practical, we performed sentiment analysis of movie reviews using nltk in python.