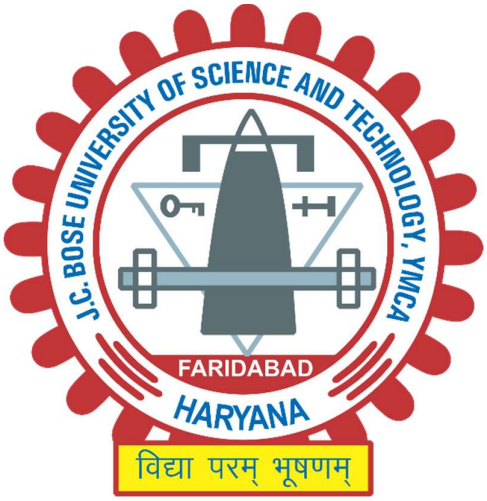
AI/ML LAB FILE

J.C. BOSE UNIVERSITY OF SCIENCE AND TECHNOLOGY, YMCA



Submitted To: **Dr. Manjit Sir**

Submitted By: **Mukesh Aggarwal**

Class: **MCA 2nd Semester**

Roll No.:**24001602043**

INDEX

PROGRAM:

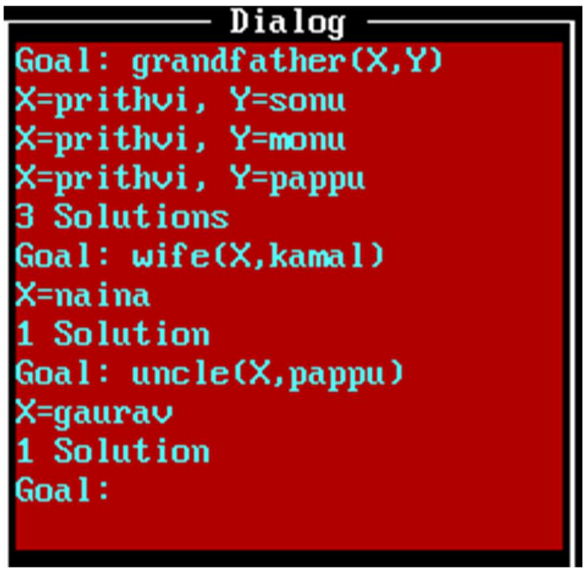
1. Write a Prolog program to implement Family Tree.

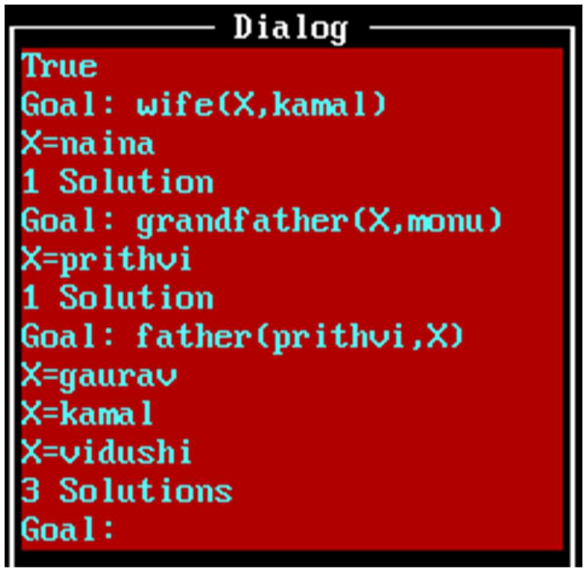
domains
name=symbol

predicates
father(name,name)
grandfather(name,name)
wife(name,name)
daughter(name,name)
uncle(name,name)
nephew(name,name)
male(name)

clauses
father(prithvi,gaurav).
father(prithvi,kamal).
father(prithvi,vidushi).
father(gaurav,sonu).
father(gaurav,monu).
father(kamal,pappu).
wife(chhayaa,prithvi).
wife(rani,gaurav).
wife(naina,kamal).
wife(vidushi,vidvan).
daughter(tanu,gaurav).
daughter(deepti,vidushi).
male(pappu).
male(monu).
male(sonu).
male(prithvi).
male(gaurav).
grandfather(X,Y):-father(X,Z),father(Z,Y).
uncle(X,Y):-father(Z,X),father(Z,W),father(W,Y),male(X),not(father(X,Y)).
nephew(X,Y):-uncle(Y,X),male(X).

Output: -


Dialog
Goal: grandfather(X,Y)
X=prithvi, Y=sonu
X=prithvi, Y=monu
X=prithvi, Y=pappu
3 Solutions
Goal: wife(X,kamal)
X=naina
1 Solution
Goal: uncle(X,pappu)
X=gaurav
1 Solution
Goal:

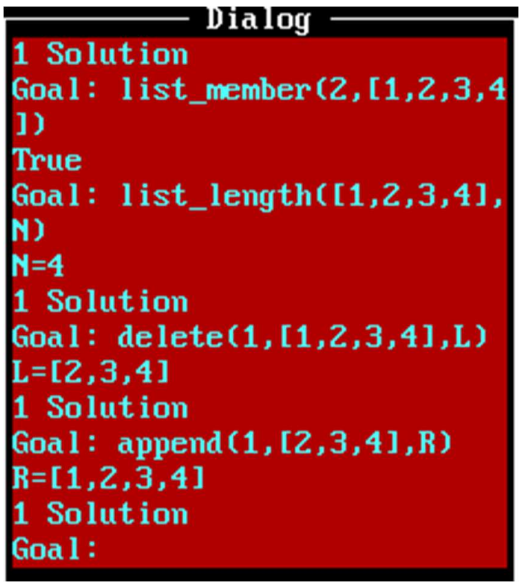

Dialog
True
Goal: wife(X,kamal)
X=naina
1 Solution
Goal: grandfather(X,monu)
X=prithvi
1 Solution
Goal: father(prithvi,X)
X=gaurav
X=kamal
X=vidushi
3 Solutions
Goal:

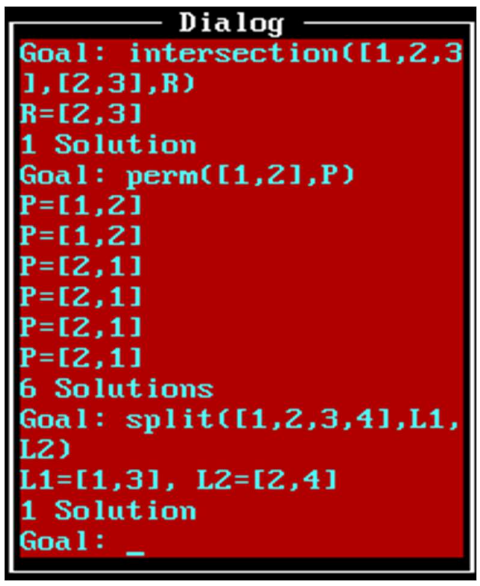
2. Write a Prolog program to make a list and perform following functions {member, length, delete, append into list, insert, concatenation, permutations, subset, union, intersection, split/divide}.

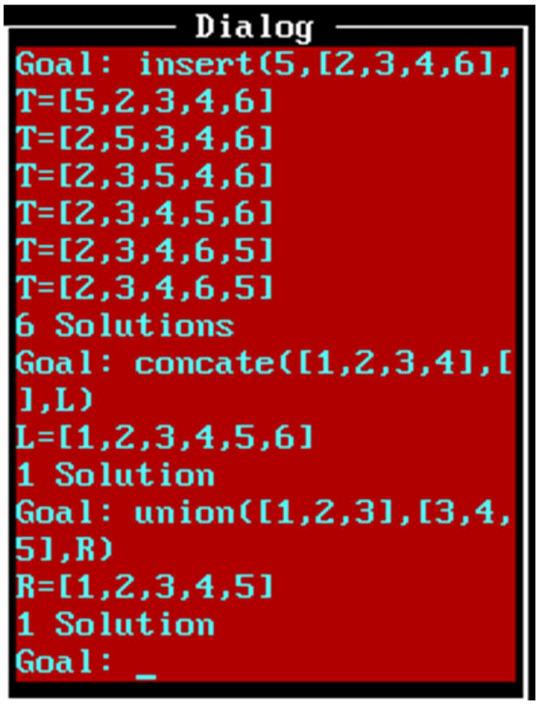
```
domains
list=integer*
member=integer

predicates
list_length(list,integer)
list_member(integer,list)
insert(integer,list,list)
append(integer,list,list)
concate(list,list,list)
perm(list,list)
delete(integer,list,list)
subset(list,list)
union(list,list,list)
intersection(list,list,list)
split(list,list,list)
mergesort(list,list)
merge(list,list,list)

clauses
list_member(X,[X|Y]).
list_member(X,[_|Y]):-list_member(X,Y).
list_length([],0).
list_length([X|Y],N):-list_length(Y,N1),N=N+1.
delete(X,[X],[]).
delete(X,[X|Y],Y).
delete(X,[Y|T],[Y|L1]):-delete(X,T,L1).
append(A,T,T):-list_member(A,T),!.
append(A,T,[A|T]).
insert(A,X,R):-delete(A,R,X).
concate([],L,L).
concate([X1|L1],L2,[X1|L3]):-concate(L1,L2,L3).
```

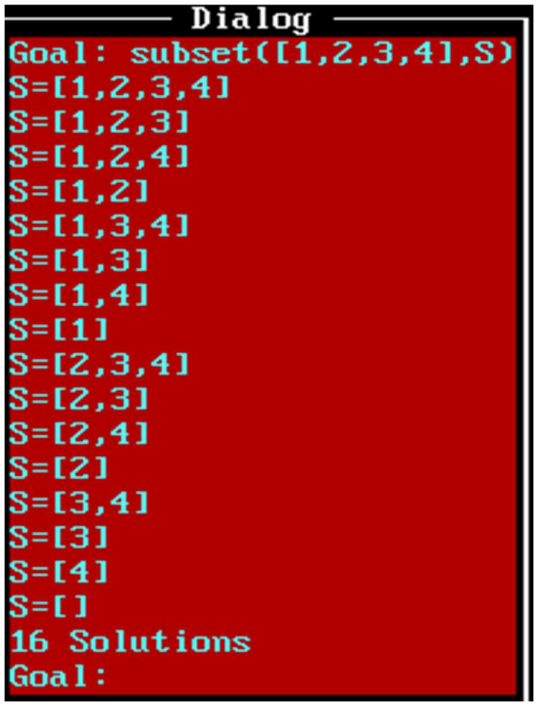
```
perm([],[]).
perm(L,[X|P]):-delete(X,L,L1),perm(L1,P).
subset([],[]).
subset([H|T],[H|Sub]):-subset(T,Sub).
subset([H|T],Sub):-subset(T,Sub).
union([X|Y],Z,W):-list_member(X,Z),union(Y,Z,W).
union([X|Y],Z,[X|W]):-not(list_member(X,Z)),union(Y,Z,W).
union([],Z,Z).
intersection([X|Y],Z,[X|W]):-list_member(X,Z),intersection(Y,Z,W).
intersection([X|Y],Z,W):-not(list_member(X,Z)),intersection(Y,Z,W).
intersection([],L,[]).
split([],[],[]).
split([X],[X],[]).
split([X,Y|T],[X|L1],[Y|L2]):-split(T,L1,L2).
mergesort([],[]).
mergesort([X],[X]).
mergesort([A,B|R],S):-
split([A,B|R],L1,L2),mergesort(L1,S1),mergesort(L2,S2),merge(S1,S2,S).
merge(A,[],A).
merge([],B,B).
merge([A|Ra],[B|Rb],[A|M]):-A<=B,merge(Ra,[B|Rb],M).
merge([A|Ra],[B|Rb],[B|M]):-B<=A,merge([A|Ra],Rb,M).
```

Output: -






Dialog
Goal: insert(5,[2,3,4,6],
T=[5,2,3,4,6]
T=[2,5,3,4,6]
T=[2,3,5,4,6]
T=[2,3,4,5,6]
T=[2,3,4,6,5]
T=[2,3,4,6,5]
6 Solutions
Goal: concate([1,2,3,4],[
],L)
L=[1,2,3,4,5,6]
1 Solution
Goal: union([1,2,3],[3,4,
5],R)
R=[1,2,3,4,5]
1 Solution
Goal: _


Dialog
Goal: subset([1,2,3,4],S)
S=[1,2,3,4]
S=[1,2,3]
S=[1,2,4]
S=[1,2]
S=[1,3,4]
S=[1,3]
S=[1,4]
S=[1]
S=[2,3,4]
S=[2,3]
S=[2,4]
S=[2]
S=[3,4]
S=[3]
S=[4]
S=[]
16 Solutions
Goal:

3. Write a program in Prolog to implement Merge Sort.

```
domains
list=integer*

predicates
split(list,list,list)
mergesort(list,list)
merge(list,list,list)

clauses
split([],[],[]).
split([X],[X],[]).
split([X,Y|T],[X|L1],[Y|L2]):-split(T,L1,L2).
mergesort([],[]).
mergesort([X],[X]).
mergesort([A,B|R],S):-
split([A,B|R],L1,L2),mergesort(L1,S1),mergesort(L2,S2),merge(S1,S2,S).
merge(A,[],A).
merge([],B,B).
merge([A|Ra],[B|Rb],[A|M]):-A<=B,merge(Ra,[B|Rb],M).
merge([A|Ra],[B|Rb],[B|M]):-B<=A,merge([A|Ra],Rb,M).
```

Output: -

4. Write a program in Prolog to implement Tower of Hanoi.

```
domains
list=integer*
member=integer
peg=symbol

predicates
tower_hanoi(integer)
hanoi(integer,peg,peg,peg)

clauses
tower_hanoi(N):-hanoi(N,s,i,d).
hanoi(1,S,I,D):-write("Mov ",1," ",S," To ",D,"\n").
hanoi(N,S,I,D):-hanoi(N1,S,D,I),
N=N1+1,
write("Mov ",N," ",S," To ",D,"\n"),
hanoi(N1,I,S,D).
```

Output: -

5. Write a program in Prolog to implement Road Map.

```
domains
list=integer*
list1=symbol*
member=integer

predicates
road(symbol,symbol,integer)
route(symbol,symbol,list1,integer)

clauses
road(d,f,30).
road(d,n,20).
road(d,r,45).
road(d,g,22).
road(r,j,32).
road(j,g,28).
road(g,s,18).
road(s,p,17).
road(f,p,25).
road(n,gn,15).
route(X,Y,[Y,X],D):-
road(Y,X,D).
route(X,Y,[X,Y],D):-road(X,Y,D).
route(X,Y,[X,Y],D):-road(Y,X,D).
route(X,Y,[X|P],D):-road(X,Z,D1),
route(Z,Y,P,D2),D=D1+D2.
```

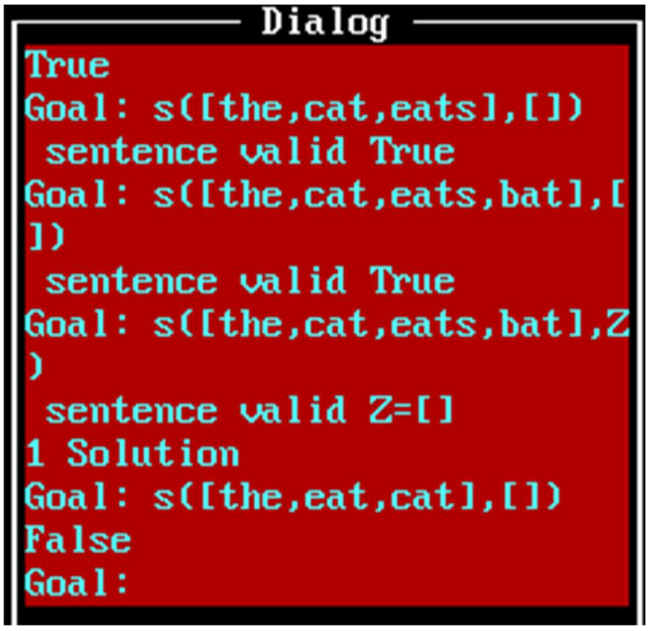
Output: -

```
Goal: route(d,g,P,D).
P=["d","g"], D=22
P=["d","r","j","g"], D=105
P=["d","r","j","g","g","s"],
 D=141
P=["d","r","j","g","s","g"],
 D=141
P=["d","g","g","s"], D=58
P=["d","g","s","g"], D=58
6 Solutions
```

6. Write a program to implement Prolog parser for English sentences using Definite Clause Grammar-style rules.

```
domains
list=symbol*

predicates
s(list,list).
np(list,list).
vp(list,list).
det(list,list).
n(list,list).
v(list,list).
adj(list,list).
length(list,integer).

clauses
length([],0).
length([_|Tail],N):- length(Tail,N1),N=N1+1.
s(A,Z):- np(A,Y),vp(Y,Z),length(Z,0),write(" sentence valid ").
s(A,Z):- np(A,Y1),vp(Y1,Y2),np(Y2,Z),length(Z,0),write(" sentence valid ").
np(A,Z):- det(A,Y),n(Y,Z).
np(A,Z):- adj(A,Y),n(Y,Z).
np(A,Z):- n(A,Z).
vp(A,Z):- v(A,Z).
det([a|X],X).
det([the|X],X).
det([an|X],X).
n([cat|X],X).
n([bat|X],X).
v([eats|X],X).
adj([fat|X],X).
```

Output: -


Dialog
True
Goal: s([the,cat,eats],[])
sentence valid True
Goal: s([the,cat,eats,bat],[
])
sentence valid True
Goal: s([the,cat,eats,bat],Z
)
sentence valid Z=[]
1 Solution
Goal: s([the,eat,cat],[])
False
Goal:

7. Write a program to implement DFA (Deterministic Finite Automaton).

domains

list=symbol*

predicates

start(symbol)
final(symbol)
transition(symbol,symbol,symbol)
dfaaccept(list,symbol)

clauses

start(s0).
final(s3).
final(s4).
transition(s0,a,s1).
transition(s0,b,s2).
transition(s1,c,s3).
transition(s2,d,s4).
transition(s3,d,s3).
transition(s4,c,s4).
dfaaccept([H|T],S):-transition(S,H,I),dfaaccept(T,I).
dfaaccept([],State):-final(State),write("input accepted").
dfaaccept([_|_],state):-NOT(final(State)),write("input rejected").

Output: -

8. Write a program to implement Monkey and Banana problem.

```
move(state(P1,onfloor,B,H),walk(P1,P2),state(P2,onfloor,B,H)).
move(state(P1,onfloor,P1,H),push(P1,P2),state(P2,onfloor,P2,H)).
move(state(P,onfloor,P,H),climb,state(P,onbox,P,H)).
move(state(middle,onbox,middle,hasnot),grasp,state(middle,onbox,middle,has
)).
%canget(state(_, _, _, has)).  % Base case when the monkey has the banana

%canget(State) :-
%    move(State, _ , State1),
%    \+ State = State1,       % Prevent infinite loops (ensure the state changes)
%    canget(State1).

canget(state(_, _, _, has,[])).  % Base case when the monkey has the banana

canget(State,[Action/Subseq]) :-
    move(State,Action,State1),
    \+ State = State1,  % Ensure state changes
    write('Current State: '), write(State), nl,
    write('Action: '), write(Action), nl,
    canget(State1,Subseq).
```

Output: -

9. Write a program in Python to calculate grades of student.

```
def grade(marks):
    if marks>=90:
        print("grade A+")
    elif marks>=80 and marks<90:
        print("grade A")
    elif marks>=75 and marks<80:
        print("grade B+")
    elif marks>=65 and marks<75:
        print("grade B")
    elif marks>=50 and marks<65:
        print("grade C")
    elif marks>=40 and marks<50:
        print("grade D")
    else:
        print("Fail")

def percent(maths,chem,phy,bio):
    total_marks=maths+chem+phy+bio
    average=total_marks/400
    percentage=average*100
    return percentage

a=int(input("Enter number of students="))
x=[]
total_percent=0
for i in range(a):
    print(f"Result of student{i+1}\n")
    print(f"Enter marks of student{i+1}-:" )
    maths=int(input("Enter marks of math="))
    chem=int(input("Enter marks of chem="))
    phy=int(input("Enter marks of phy="))
    bio=int(input("Enter marks of bio="))
    print("grade of maths:")
    grade(maths)
```

```
    print("grade of chem:")
    grade(chem)
    print("grade of phy:")
    grade(phy)
    print("grade of bio:")
    grade(bio)
    print(f"Percentage of student{i+1}-:")
    y=percent(maths,chem,phy,bio)
    x.append(y)
    print(y)
average_percent=sum(x)/len(x)
print(f"average percentage of class={average_percent}")
```

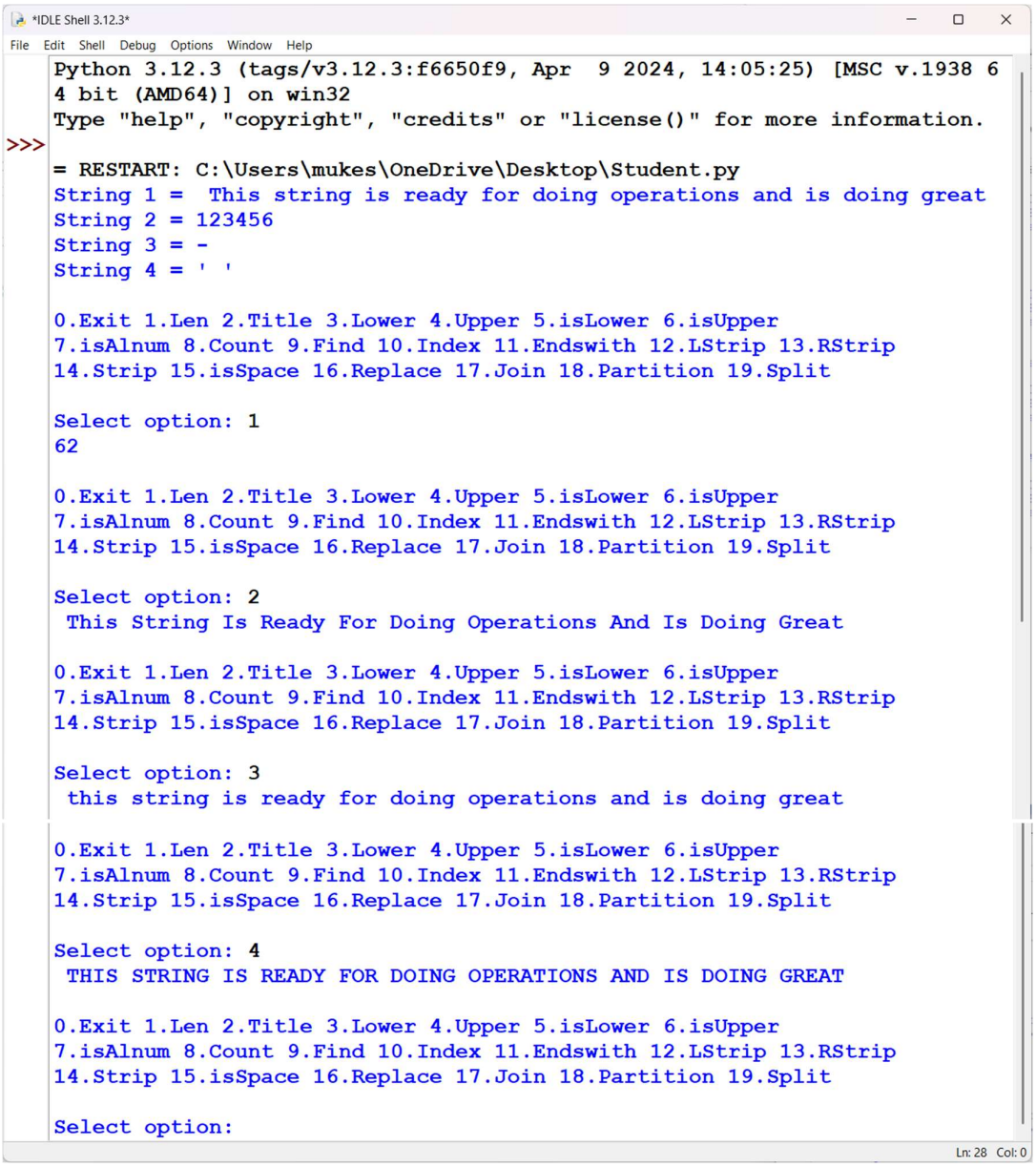
Output: -

10. Write a program in Python to implement string functions.

```
s1 = " This string is ready for doing operations and is doing great "
s2 = "123456"
s3 = "-"
s4 = " "
print(f"String 1 = {s1}\nString 2 = {s2}\nString 3 = {s3}\nString 4 = '{s4}'")
while True:
    op = input("\n0.Exit 1.Len 2.Title 3.Lower 4.Upper 5.isLower 6.isUpper
7.isAlnum 8.Count 9.Find 10.Index 11.Endswith 12.LStrip 13.RStrip
14.Strip 15.isSpace 16.Replace 17.Join 18.Partition 19.Split
\nSelect option: ")

    if op == '0': break
    elif op == '1': print(len(s1))
    elif op == '2': print(s1.title())
    elif op == '3': print(s1.lower())
    elif op == '4': print(s1.upper())
    elif op == '5': print(s1.islower())
    elif op == '6': print(s1.isupper(), s2.isupper())
    elif op == '7': print(s1.isalnum(), s2.isalnum())
    elif op == '8': print(s1.count('is'))
    elif op == '9': print(s1.find('doing',3,20), s1.find('doing'))
    elif op == '10': print(s1.index('doing',3,50), s1.index('doing'))
    elif op == '11': print(s1.endswith('great'))
    elif op == '12': print(s1.lstrip())
    elif op == '13': print(s1.rstrip())
    elif op == '14': print(s1.strip())
    elif op == '15': print(s1.isspace(), s4.isspace())
```

```
elif op == '16': print(s1.replace('is','was'))
elif op == '17': print(s3.join(s1))
elif op == '18': print(s1.partition('is'), s1.partition(' '))
elif op == '19': print(s1.split('is'), s1.split())
else: print("Invalid Option")
```

Output: -

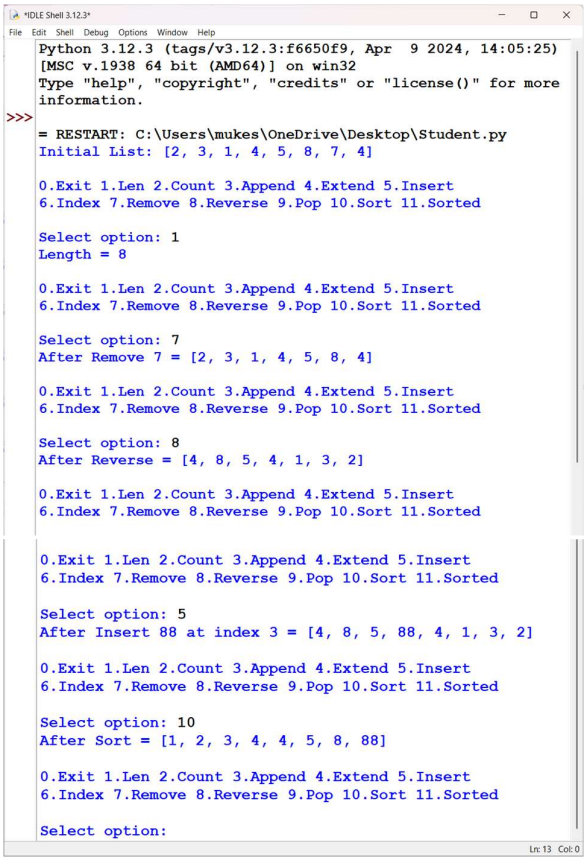


11. Write a program in Python to implement List functions.

```
l = [2, 3, 1, 4, 5, 8, 7, 4]
print("Initial List:", l)

while True:
    op = input("'\n0.Exit 1.Len 2.Count 3.Append 4.Extend 5.Insert
6.Index 7.Remove 8.Reverse 9.Pop 10.Sort 11.Sorted
\nSelect option: '")

    if op == '0':
        break
    elif op == '1':
        print("Length =", len(l))
    elif op == '2':
        print("Count of 4 =", l.count(4))
    elif op == '3':
        l.append([4, 6])
        print("After Append [4,6] =", l)
    elif op == '4':
        l.extend([30, 40])
        print("After Extend [30,40] =", l)
    elif op == '5':
        l.insert(3, 88)
        print("After Insert 88 at index 3 =", l)
    elif op == '6':
        print("Index of 7 =", l.index(7))
    elif op == '7':
        l.remove(7)
```

```
        print("After Remove 7 =", l)
    elif op == '8':
        l.reverse()
        print("After Reverse =", l)
    elif op == '9':
        print("Popped element at index 1 =", l.pop(1))
        print("After Pop =", l)
    elif op == '10':
        l.sort()
        print("After Sort =", l)
    elif op == '11':
        print("Sorted copy =", sorted(l))
    else:
        print("Invalid option")
```

Output: -



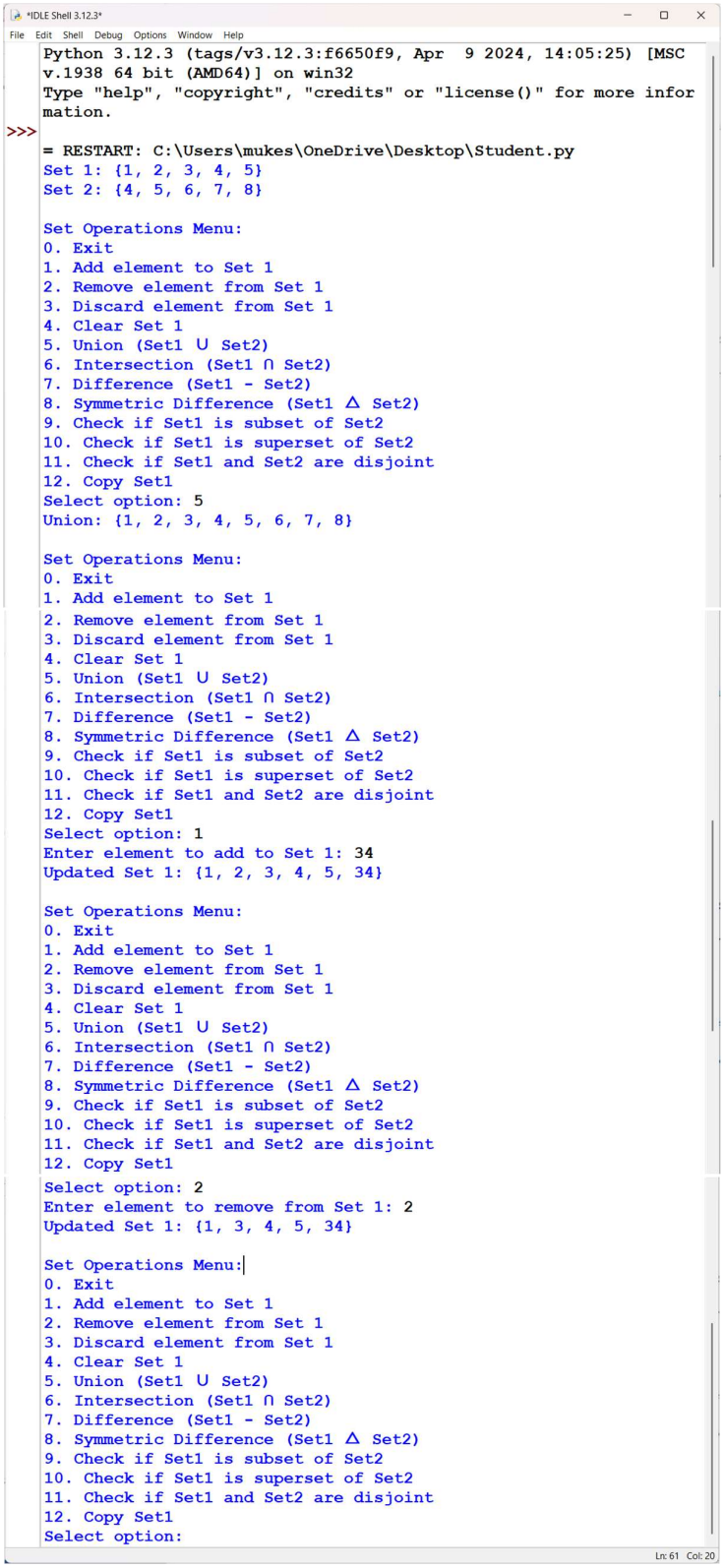
12. Write a program in Python to implement sets functions.

```
# Initial sets
s1 = {1, 2, 3, 4, 5}
s2 = {4, 5, 6, 7, 8}

print("Set 1:", s1)
print("Set 2:", s2)

while True:
    op = input("'\nSet Operations Menu:
0. Exit
1. Add element to Set 1
2. Remove element from Set 1
3. Discard element from Set 1
4. Clear Set 1
5. Union (Set1 ∪ Set2)
6. Intersection (Set1 ∩ Set2)
7. Difference (Set1 - Set2)
8. Symmetric Difference (Set1 △ Set2)
9. Check if Set1 is subset of Set2
10. Check if Set1 is superset of Set2
11. Check if Set1 and Set2 are disjoint
12. Copy Set1
Select option: "')

    if op == '0':
        print("Exiting...")
        break
    elif op == '1':
```

```
    elem = int(input("Enter element to add to Set 1: "))
    s1.add(elem)
    print("Updated Set 1:", s1)
elif op == '2':
    elem = int(input("Enter element to remove from Set 1: "))
    if elem in s1:
        s1.remove(elem)
        print("Updated Set 1:", s1)
    else:
        print("Element not found in Set 1.")
elif op == '3':
    elem = int(input("Enter element to discard from Set 1: "))
    s1.discard(elem)
    print("Updated Set 1:", s1)
elif op == '4':
    s1.clear()
    print("Set 1 is now empty:", s1)
elif op == '5':
    print("Union:", s1.union(s2))
elif op == '6':
    print("Intersection:", s1.intersection(s2))
elif op == '7':
    print("Difference (Set1 - Set2):", s1.difference(s2))
elif op == '8':
    print("Symmetric Difference:", s1.symmetric_difference(s2))
elif op == '9':
    print("Is Set1 subset of Set2?", s1.issubset(s2))
```

```
    elif op == '10':
        print("Is Set1 superset of Set2?", s1.issuperset(s2))
    elif op == '11':
        print("Are Set1 and Set2 disjoint?", s1.isdisjoint(s2))
    elif op == '12':
        s3 = s1.copy()
        print("Copied Set1 to Set3:", s3)
    else:
        print("Invalid option. Please try again.")
```

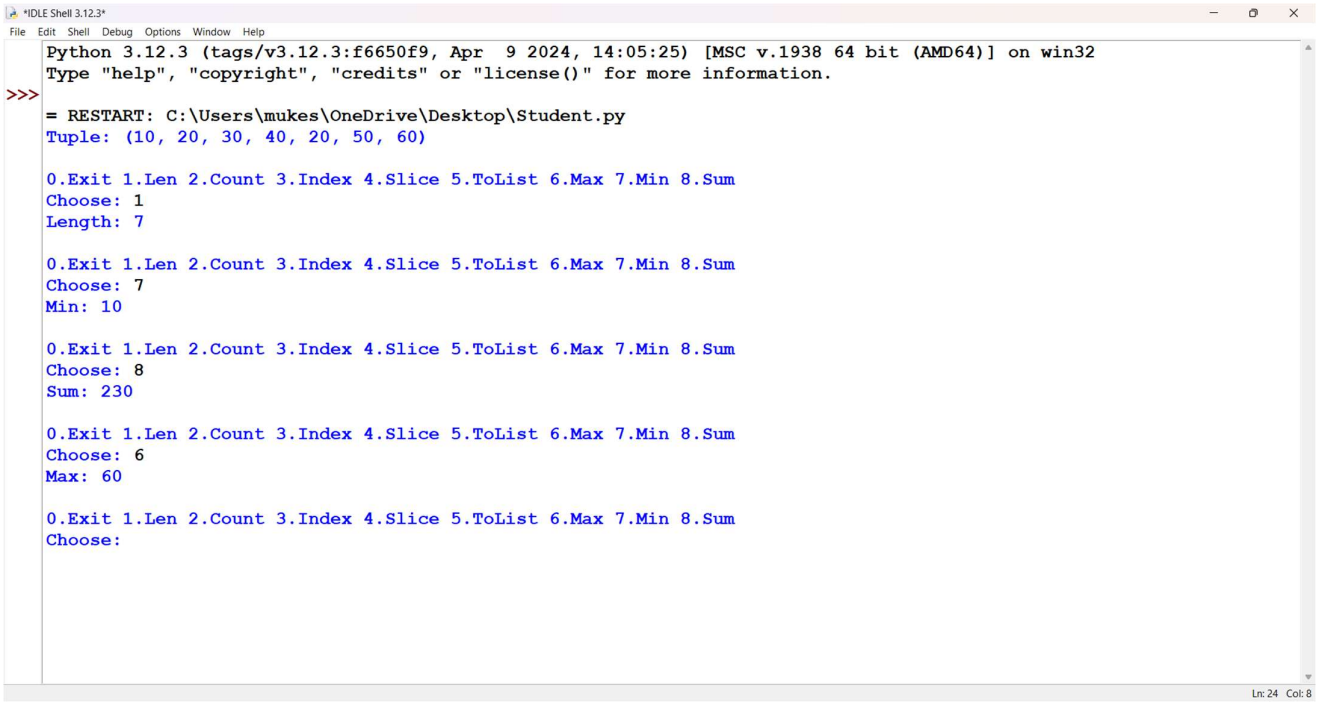
Output: -


IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\mukes\OneDrive\Desktop\Student.py
Set 1: {1, 2, 3, 4, 5}
Set 2: {4, 5, 6, 7, 8}
Set Operations Menu:
0. Exit
1. Add element to Set 1
2. Remove element from Set 1
3. Discard element from Set 1
4. Clear Set 1
5. Union (Set1 ∪ Set2)
6. Intersection (Set1 ∩ Set2)
7. Difference (Set1 - Set2)
8. Symmetric Difference (Set1 Δ Set2)
9. Check if Set1 is subset of Set2
10. Check if Set1 is superset of Set2
11. Check if Set1 and Set2 are disjoint
12. Copy Set1
Select option: 5
Union: {1, 2, 3, 4, 5, 6, 7, 8}
Set Operations Menu:
0. Exit
1. Add element to Set 1
2. Remove element from Set 1
3. Discard element from Set 1
4. Clear Set 1
5. Union (Set1 ∪ Set2)
6. Intersection (Set1 ∩ Set2)
7. Difference (Set1 - Set2)
8. Symmetric Difference (Set1 Δ Set2)
9. Check if Set1 is subset of Set2
10. Check if Set1 is superset of Set2
11. Check if Set1 and Set2 are disjoint
12. Copy Set1
Select option: 1
Enter element to add to Set 1: 34
Updated Set 1: {1, 2, 3, 4, 5, 34}
Set Operations Menu:
0. Exit
1. Add element to Set 1
2. Remove element from Set 1
3. Discard element from Set 1
4. Clear Set 1
5. Union (Set1 ∪ Set2)
6. Intersection (Set1 ∩ Set2)
7. Difference (Set1 - Set2)
8. Symmetric Difference (Set1 Δ Set2)
9. Check if Set1 is subset of Set2
10. Check if Set1 is superset of Set2
11. Check if Set1 and Set2 are disjoint
12. Copy Set1
Select option: 2
Enter element to remove from Set 1: 2
Updated Set 1: {1, 3, 4, 5, 34}
Set Operations Menu:
0. Exit
1. Add element to Set 1
2. Remove element from Set 1
3. Discard element from Set 1
4. Clear Set 1
5. Union (Set1 ∪ Set2)
6. Intersection (Set1 ∩ Set2)
7. Difference (Set1 - Set2)
8. Symmetric Difference (Set1 Δ Set2)
9. Check if Set1 is subset of Set2
10. Check if Set1 is superset of Set2
11. Check if Set1 and Set2 are disjoint
12. Copy Set1
Select option:
Ln: 61 Col: 20

13. Write a program in Python to implement tuple functions.

```
t = (10, 20, 30, 40, 20, 50, 60)
print("Tuple:", t)

while True:
    op = input("'\n0.Exit 1.Len 2.Count 3.Index 4.Slice 5.ToList 6.Max 7.Min
8.Sum\nChoose: '")

    if op == '0': break
    elif op == '1': print("Length:", len(t))
    elif op == '2': print("Count:", t.count(int(input("Value: "))))
    elif op == '3':
        val = int(input("Value: "))
        print("Index:", t.index(val) if val in t else "Not found")
    elif op == '4':
        s, e = int(input("Start: ")), int(input("End: "))
        print("Slice:", t[s:e])
    elif op == '5': print("List:", list(t))
    elif op == '6': print("Max:", max(t))
    elif op == '7': print("Min:", min(t))
    elif op == '8': print("Sum:", sum(t))
    else: print("Invalid")
```

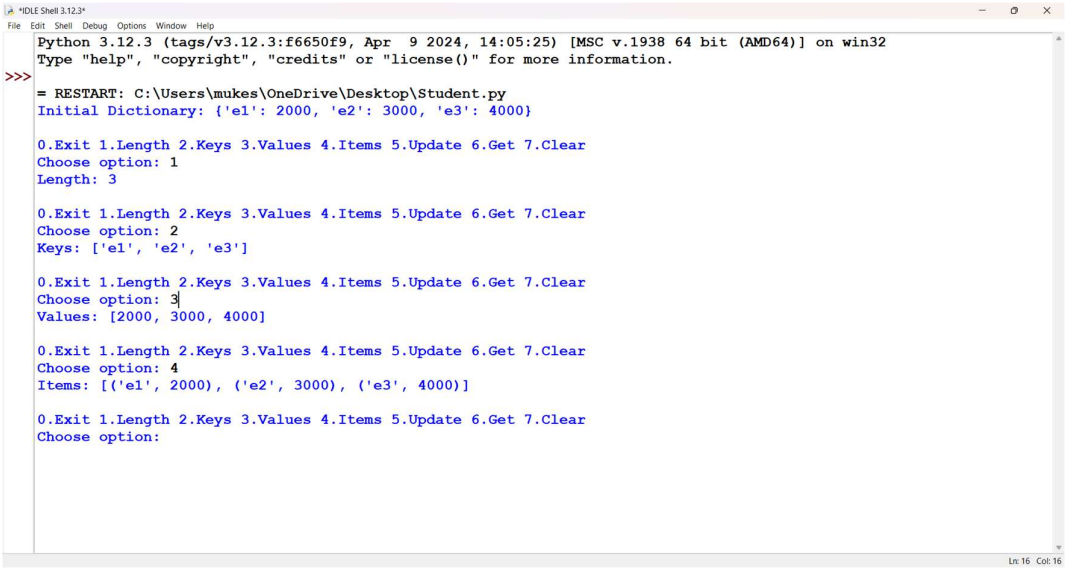
Output: -


IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\mukes\OneDrive\Desktop\Student.py
Tuple: (10, 20, 30, 40, 20, 50, 60)
0.Exit 1.Len 2.Count 3.Index 4.Slice 5.ToList 6.Max 7.Min 8.Sum
Choose: 1
Length: 7
0.Exit 1.Len 2.Count 3.Index 4.Slice 5.ToList 6.Max 7.Min 8.Sum
Choose: 7
Min: 10
0.Exit 1.Len 2.Count 3.Index 4.Slice 5.ToList 6.Max 7.Min 8.Sum
Choose: 8
Sum: 230
0.Exit 1.Len 2.Count 3.Index 4.Slice 5.ToList 6.Max 7.Min 8.Sum
Choose: 6
Max: 60
0.Exit 1.Len 2.Count 3.Index 4.Slice 5.ToList 6.Max 7.Min 8.Sum
Choose:
Ln: 24 Col: 8

14. Write a program in Python to implement dictionary functions.

```
l = {'e1': 2000, 'e2': 3000, 'e3': 4000}
d2 = {'e4': 20050, 'e5': 30800}

print("Initial Dictionary:", l)

while True:
    op = input("'\n0.Exit 1.Length 2.Keys 3.Values 4.Items 5.Update 6.Get 7.Clear
Choose option: "')

    if op == '0': break
    elif op == '1': print("Length:", len(l))
    elif op == '2': print("Keys:", list(l.keys()))
    elif op == '3': print("Values:", list(l.values()))
    elif op == '4': print("Items:", list(l.items()))
    elif op == '5': l.update(d2); print("Updated Dictionary:", l)
    elif op == '6': print("Get 'e2':", l.get('e2'))
    elif op == '7': l.clear(); print("Cleared Dictionary:", l)
    else: print("Invalid Option")
```

Output: -

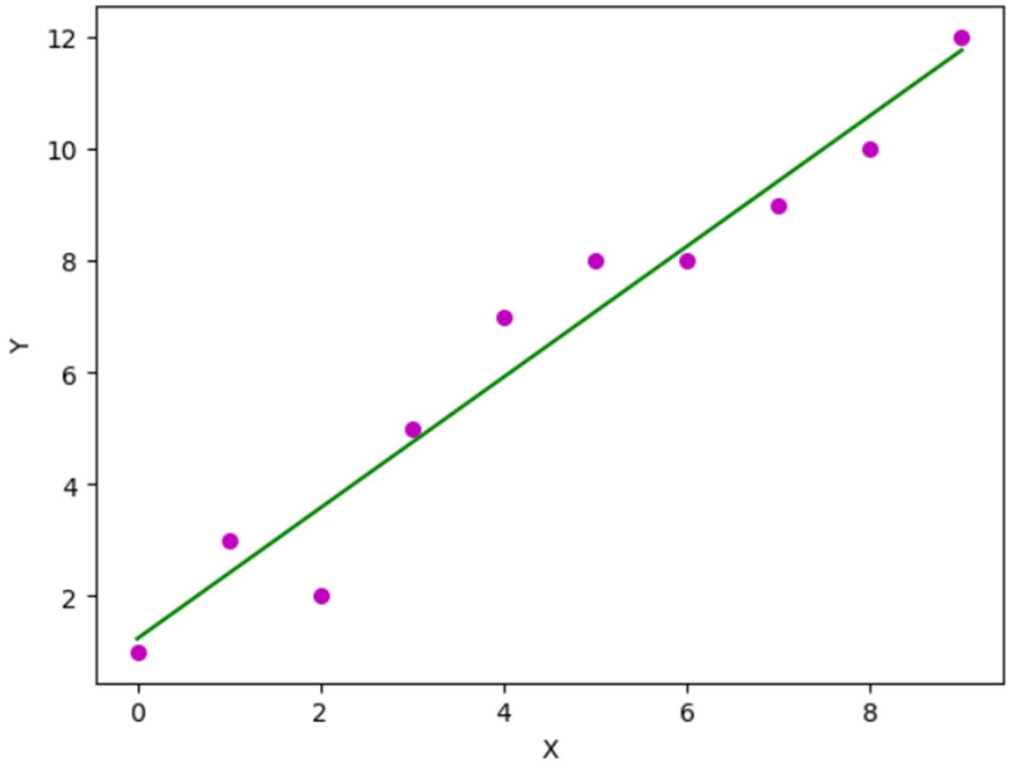


15. Write a program in Python to implement linear regression.

```
import numpy as np
import matplotlib.pyplot as plt
def estimate_coef(x,y):
    n = np.size(x)
    m_x, m_y = np.mean(x), np.mean(y)
    SS_xy = np.sum(x*y) - n*m_x*m_y
    SS_xx = np.sum(x*x) - n*m_x*m_x
    b_1 = SS_xy / SS_xx
    b_0 = m_y - b_1 * m_x
    return b_0,b_1

def plot_reg_line(x,y,b):
    plt.scatter(x,y,color='m',marker='o',s=30)
    y_pred = b[0] + b[1]*x
    plt.plot(x,y_pred,color='g')
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.show()
def main():
    x=np.array([0,1,2,3,4,5,6,7,8,9])
    y=np.array([1,3,2,5,7,8,8,9,10,12])
    b=estimate_coef(x,y)
    print('b_0=',b[0],'b_1=',b[1])
    plot_reg_line(x,y,b)

if __name__ == '__main__':
main()
```

Output: –

```
b_0= 1.2363636363636363 b_1= 1.1696969696969697
```

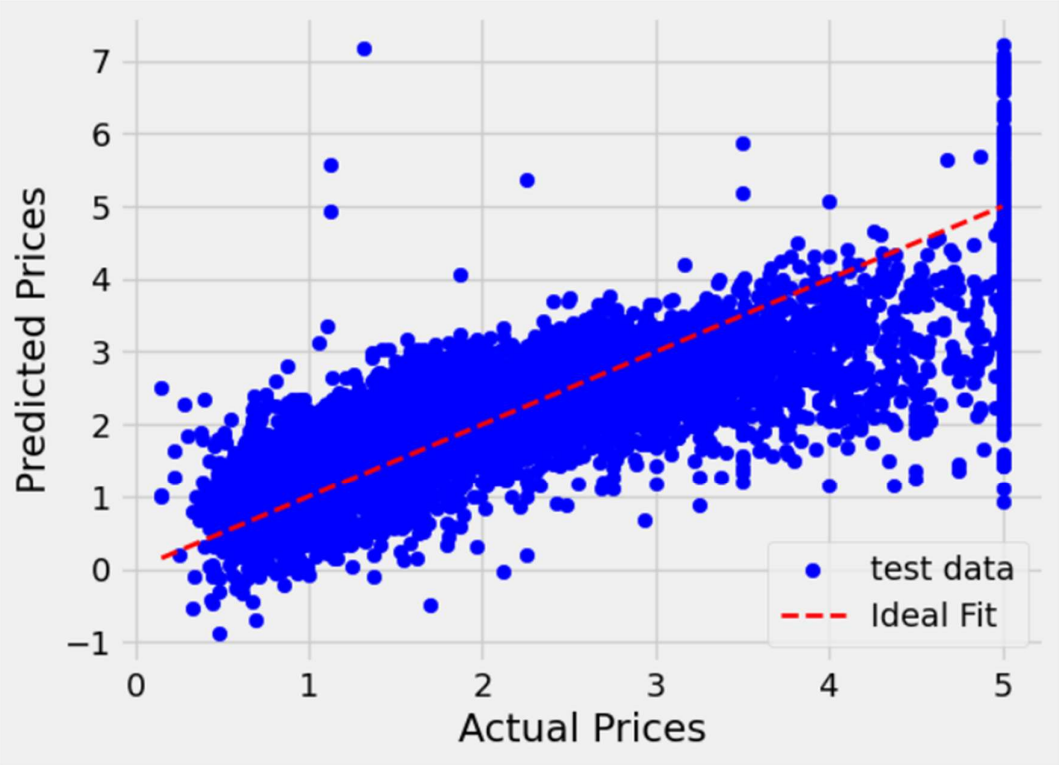


16. Write a program in Python to implement linear regression using California housing dataset.

```
from weakref import ref
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import datasets
boston = datasets.fetch_california_housing(return_X_y=False)
X = boston.data
Y = boston.target
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.4,
random_state=1)

model = LinearRegression()
model.fit(X_train, Y_train)
print('coefficient:', model.coef_)
print('variance score:', model.score(X_test, Y_test))
y_pred = model.predict(X_test)

plt.style.use("fivethirtyeight")
plt.scatter(Y_test, y_pred, color='blue', label='test data')
plt.plot([min(Y_test), max(Y_test)], [min(Y_test), max(Y_test)], color='red',
linestyle='dashed', linewidth=2, label='Ideal Fit')
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.legend()
plt.show()
```

OUTPUT: -

```
coefficient: [ 4.36676927e-01  9.58588561e-03 -9.61859974e-02  5.77800722e-01
 -4.33084487e-06 -3.13805195e-03 -4.22347516e-01 -4.34869554e-01]
variance score: 0.606837323920864
```

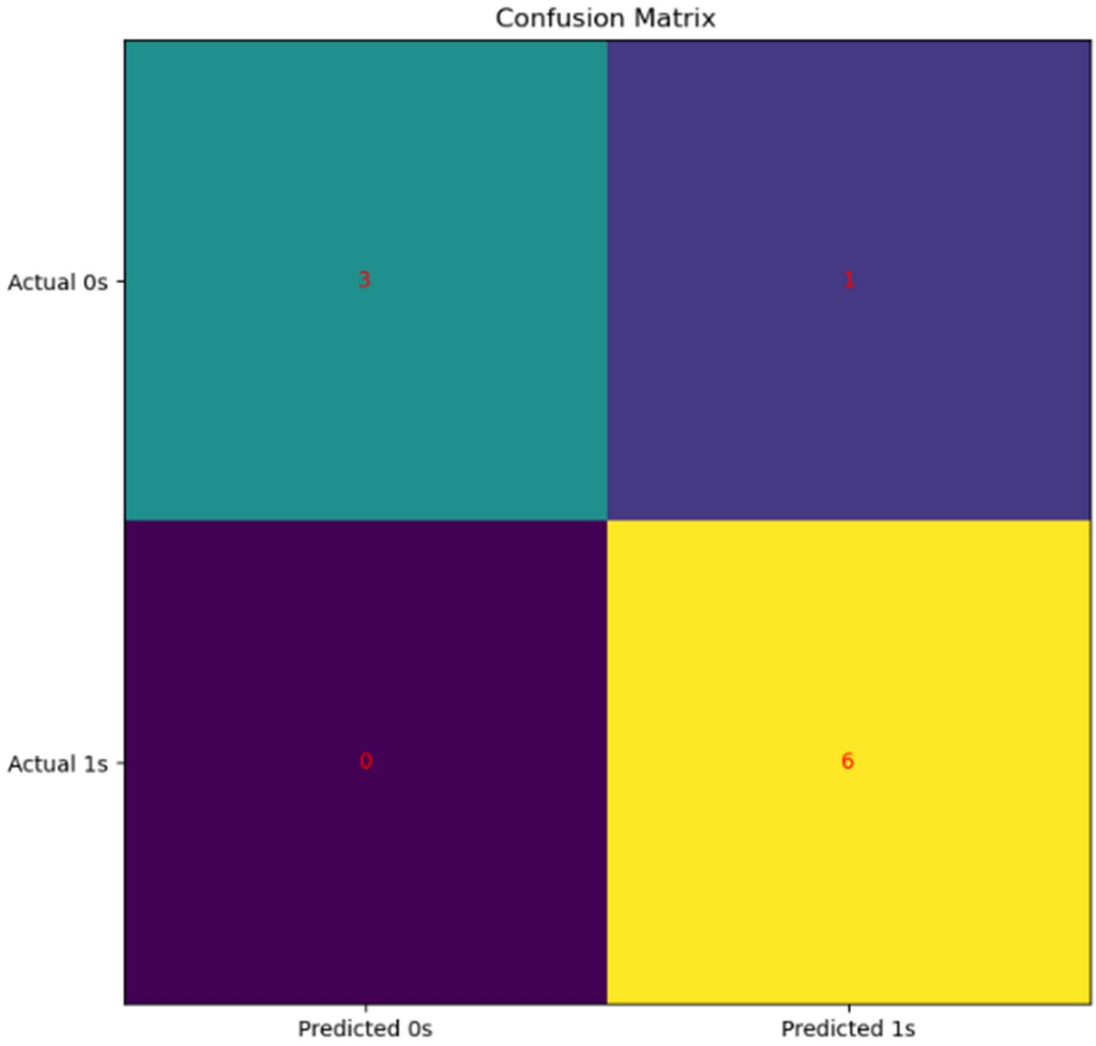


17. Write a program in Python to implement logistic regression.

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

x = np.arange(10).reshape(-1, 1)
y = np.array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1])
model = LogisticRegression(solver='liblinear', random_state=0)
model.fit(x, y)
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=0, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)
model = LogisticRegression(solver='liblinear', random_state=0).fit(x, y)
model.classes_
model.intercept_
model.coef_
model.predict_proba(x)
model.predict(x)
model.score(x, y)
confusion_matrix(y, model.predict(x))
cm = confusion_matrix(y, model.predict(x))

fig, ax = plt.subplots(figsize=(8, 8))
ax.imshow(cm)
```

```
ax.grid(False)
ax.xaxis.set(ticks=(0, 1), ticklabels=('Predicted 0s', 'Predicted 1s'))
ax.yaxis.set(ticks=(0, 1), ticklabels=('Actual 0s', 'Actual 1s'))
ax.set_ylim(1.5, -0.5)
for i in range(2):
    for j in range(2):
        ax.text(j, i, cm[i, j], ha='center', va='center', color='red')
plt.title("Confusion Matrix")
plt.show()
print(classification_report(y, model.predict(x)))
model = LogisticRegression(solver='liblinear', C=10.0, random_state=0)
model.fit(x, y)
model.intercept_
model.coef_
model.predict_proba(x)
model.predict(x)
confusion_matrix(y, model.predict(x))
print(classification_report(y, model.predict(x)))
```

OUTPUT: -



	precision	recall	f1-score	support
0	1.00	0.75	0.86	4
1	0.86	1.00	0.92	6
accuracy			0.90	10
macro avg	0.93	0.88	0.89	10
weighted avg	0.91	0.90	0.90	10

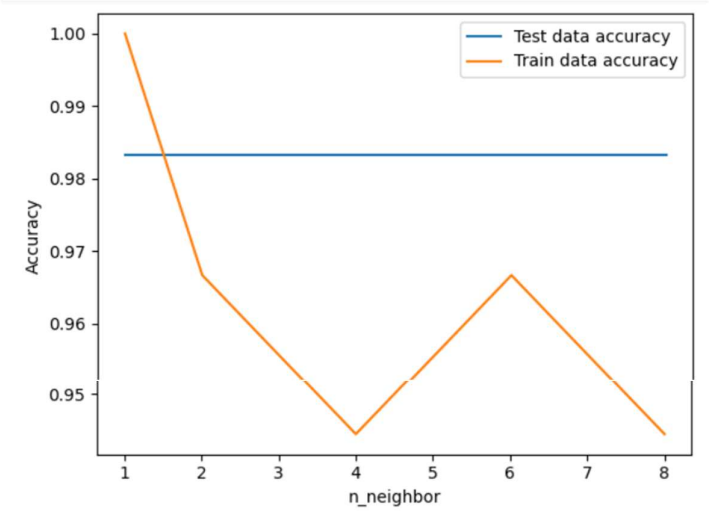
	precision	recall	f1-score	support
0	1.00	1.00	1.00	4
1	1.00	1.00	1.00	6
accuracy			1.00	10
macro avg	1.00	1.00	1.00	10
weighted avg	1.00	1.00	1.00	10

18. Write a program in Python to implement K nearest neighbour.

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score, classification_report

iris=load_iris()
X=iris.data
Y=iris.target
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.4,random_state=42
)

neighbor=np.arange(1,9)
train_accuracy=np.empty(len(neighbor))
test_accuracy=np.empty(len(neighbor))
for i,k in enumerate(neighbor):
    model=KNeighborsClassifier(n_neighbors=k)
    model.fit(x_train,y_train)
    train_accuracy[i]=model.score(x_train,y_train)
    test_accuracy[i]=model.score(x_test,y_test)

plt.plot(neighbor,test_accuracy,label='Test data accuracy')
plt.plot(neighbor,train_accuracy,label='Train data accuracy')
plt.legend()
plt.xlabel('n_neighbor')
plt.ylabel('Accuracy')
```

```
plt.show()
pred=model.predict(x_test)
report = classification_report(y_test,pred)
print(report)
```

Output: -



	precision	recall	f1-score	support
0	1.00	1.00	1.00	23
1	0.95	1.00	0.97	19
2	1.00	0.94	0.97	18
accuracy			0.98	60
macro avg	0.98	0.98	0.98	60
weighted avg	0.98	0.98	0.98	60

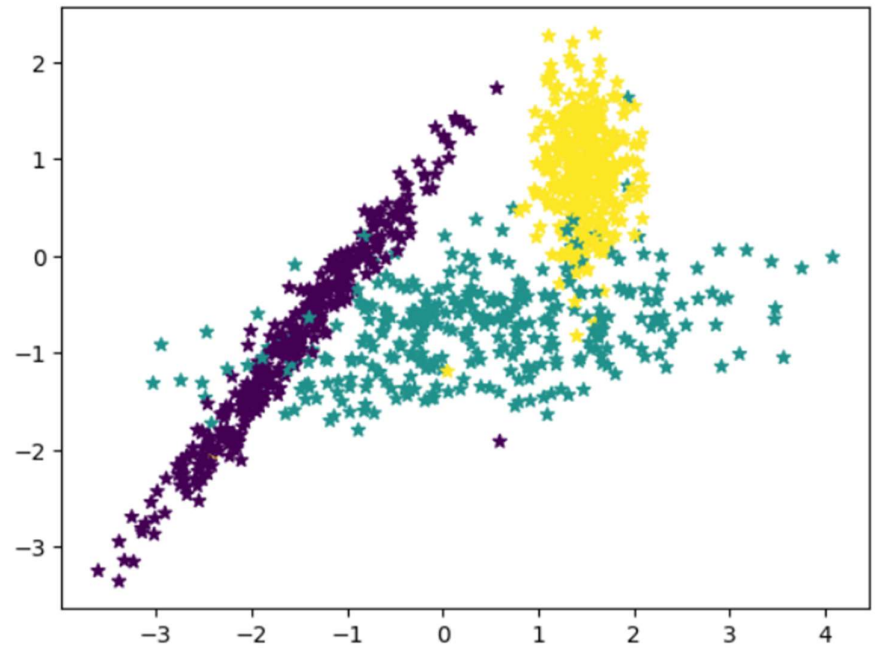
19. Write a program in Python to implement naïve bayes classification.

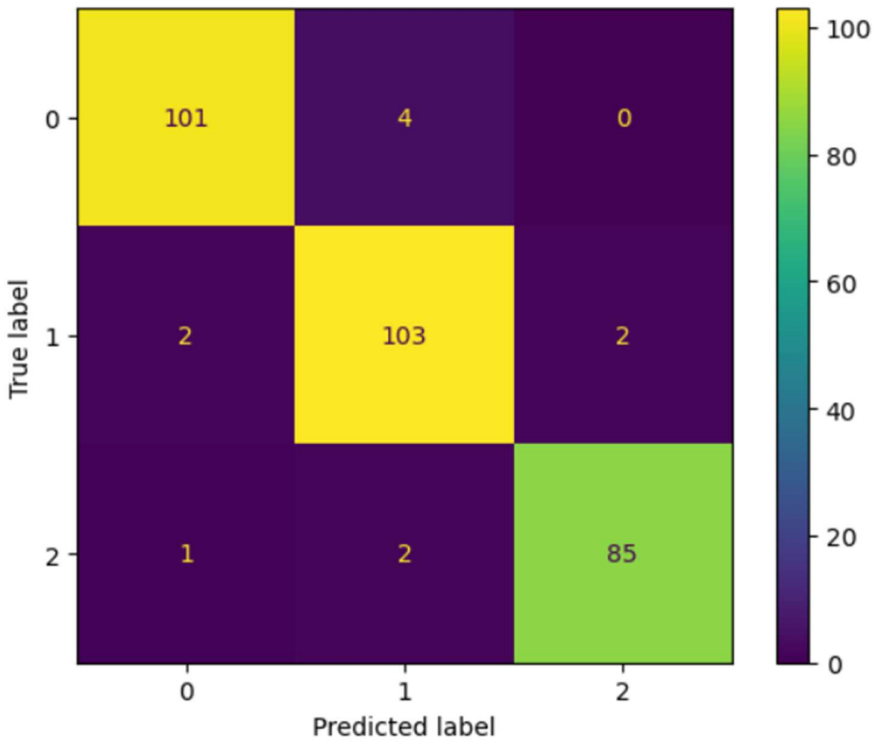
```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import make_classification
import matplotlib.pyplot as plt
from sklearn.metrics import
(accuracy_score,confusion_matrix,ConfusionMatrixDisplay,f1_score)
X,y =
make_classification(n_features=6,n_classes=3,n_samples=1000,n_informative=3
,random_state=1,n_clusters_per_class=1)
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=12
5)

plt.scatter(X[:,0],X[:,1],c=y,marker="*")
model=GaussianNB()
model.fit(X_train,y_train)
predicted=model.predict([X_test[6]])
print("Actual value : ",y_test[6])
print("Predicted value : ",predicted)

y_pred=model.predict(X_test)
accuracy = accuracy_score(y_pred,y_test)
f1 = f1_score(y_pred,y_test,average="weighted")
print("Accuracy : ",accuracy)
print("F1 score : ",f1)
labels = [0,1,2]
cm = confusion_matrix(y_pred,y_test,labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
```

```
disp.plot()
plt.show()
```

Output: -

```
Actual value :  1
Predicted value :  [1]
Accuracy :  0.9633333333333334
F1 score :  0.9633842139017577
```



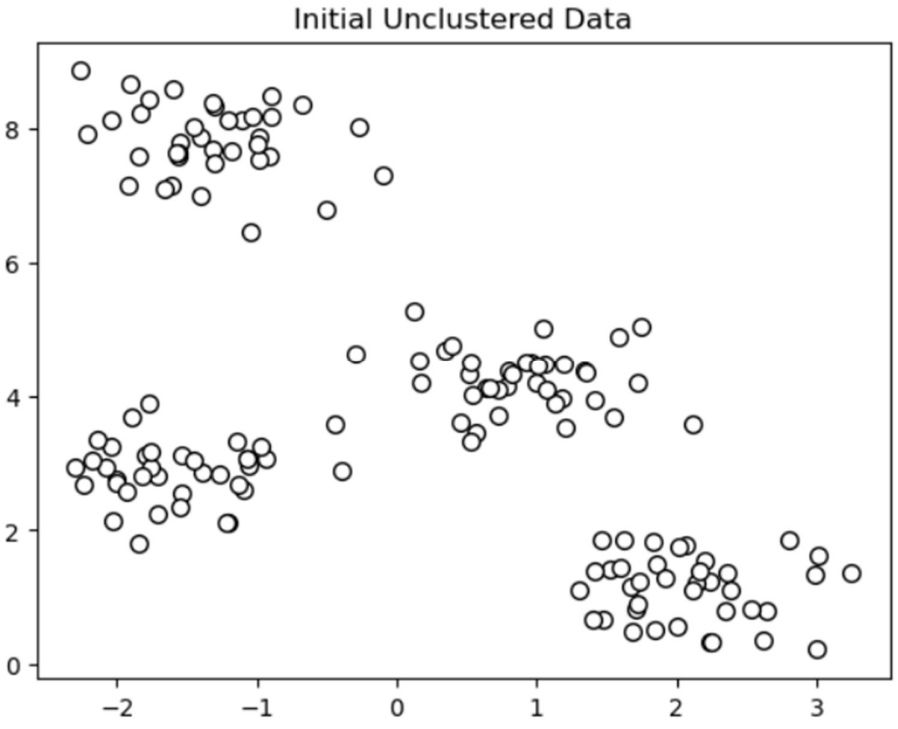


20. Write a program in Python to implement K-Mean Clustering.

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

# Create synthetic data
X, y = make_blobs(n_samples=150, n_features=2, centers=4, cluster_std=0.5, shuffle=True, random_state=0)

# Plot initial data
plt.scatter(X[:, 0], X[:, 1], c='white', marker='o', edgecolor='black', s=50)
plt.title("Initial Unclustered Data")
plt.show()

# Apply KMeans
km = KMeans(n_clusters=4, init='random', n_init=10, max_iter=300, tol=1e-04, random_state=0)
y_km = km.fit_predict(X)

print("KMeans Model:", km)
print("Type of km:", type(km))
print("Shape of cluster centers:", km.cluster_centers_.shape)
print("\nCluster Centers:\n", km.cluster_centers_)
print("\nCluster Labels:", km.labels_)
print("Shape of labels:", km.labels_.shape)

# Plot clusters
plt.scatter(X[y_km == 0, 0], X[y_km == 0, 1], s=50, c='lightgreen', marker='s', edgecolor='black', label='cluster 1')
```

```
plt.scatter(X[y_km == 1, 0], X[y_km == 1, 1], s=50, c='orange', marker='o',
edgecolor='black', label='cluster 2')

plt.scatter(X[y_km == 2, 0], X[y_km == 2, 1], s=50, c='lightblue', marker='v',
edgecolor='black', label='cluster 3')

plt.scatter(X[y_km == 3, 0], X[y_km == 3, 1], s=50, c='pink', marker='^',
edgecolor='black', label='cluster 4')

# Plot centroids

plt.scatter(km.cluster_centers_[:, 0], km.cluster_centers_[:, 1],

        s=100, c='red', marker='*', edgecolor='black', label='centroids')

plt.legend(scatterpoints=1)

plt.grid()

plt.title("K-Means Clustering Result")

plt.show()
```

Output: -



```
KMeans Model: KMeans(init='random', n_clusters=4, n_init=10, random_state=0)
Type of km: <class 'sklearn.cluster._kmeans.KMeans'>
Shape of cluster centers: (4, 2)

Cluster Centers:
 [[-1.32931949  7.83606554]
 [-1.55311219  2.87260114]
 [ 0.88922686  4.24805239]
 [ 2.08356978  1.13724593]]

Cluster Labels: [2 2 0 2 2 3 2 1 1 0 2 3 1 3 0 0 0 1 3 0 2 3 0 2 0 2 0 0 2 2 0 1 2 1 0 0 3
 0 2 1 1 1 2 3 1 1 1 0 3 3 1 0 0 0 2 1 3 2 1 3 3 3 3 1 0 0 2 1 1 2 3 3 3 2
 2 2 1 3 3 1 2 3 0 1 2 3 0 2 2 3 3 1 1 0 1 1 3 2 3 3 3 3 0 0 3 3 3 1 3 3 1
 2 0 2 2 2 1 0 2 1 0 3 2 2 2 0 0 0 2 1 1 1 3 1 0 1 1 1 3 0 0 2 0 3 0 2 2 1
 3 0]
Shape of labels: (150,)
```

21. Write a program in Python to implement decision tree classification.

```
# Decission Tree using sklearn

# Load libraries

import pandas as pd

from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier

from sklearn.model_selection import train_test_split # Import train_test_split function

from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation

from sklearn.datasets import load_iris

import pydot

#col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'label']

# load dataset

#pima = pd.read_csv("diabetes.csv", header=None, names=col_names)

#split dataset in features and target variable

###y = pima.label # Target variable

iris=load_iris()

X=pd.DataFrame(iris.data, columns=iris.feature_names)

y=pd.Categorical.from_codes(iris.target,iris.target_names)

print(X.head())

print(y)

y=pd.get_dummies(y)

print(y)


# 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,
random_state=1) # 70% training and 30% test

# Create Decision Tree classifer object

clf = DecisionTreeClassifier()

#clf = DecisionTreeClassifier(criterion="entropy", max_depth=3) # default value
is "gini"

# Train Decision Tree Classifer

clf = clf.fit(X_train,y_train)

#Predict the response for test dataset

y_pred = clf.predict(X_test)

# Model Accuracy, how often is the classifier correct?

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

#Visualization of  decision tree

from sklearn.tree import export_graphviz

#from sklearn.externals.six import StringIO

from io import StringIO

from IPython.display import Image

from pydot import graph_from_dot_data


dot_data = StringIO()

export_graphviz(clf, out_file=dot_data,

                filled=True, rounded=True,

                special_characters=True,feature_names =
iris.feature_names,class_names=['0','1'])

graph = pydot.graph_from_dot_data(dot_data.getvalue())

graph[0].write_png('diabetes.png')

Image(graph[0].create_png())
```

Output: –

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2
['setosa', 'setosa', 'setosa', 'setosa', 'setosa', ..., 'virginica', 'virginica', 'virginica', 'virginica', 'virginica']
Length: 150
Categories (3, object): ['setosa', 'versicolor', 'virginica']
     setosa  versicolor  virginica
0      True       False      False
1      True       False      False
2      True       False      False
3      True       False      False
4      True       False      False
..      ...         ...        ...
145   False       False       True
146   False       False       True
147   False       False       True
148   False       False       True
149   False       False       True

[150 rows x 3 columns]
Accuracy: 0.9555555555555556
```