J.N.T.U.H. UNIVERSITY COLLEGE OF ENGINEERING, SCIENCE & TECHNOLOGY HYDERABAD

KUKATPALLY, HYDERABAD – 500 085

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY
HYDERABAD
योगः कर्मसु कौशलम्
GATEWAY TO EXCELLENCE

Certificate

Certified that this is the bonafide record of the practical work done during

the academic year ______________________ by

Name ____

Roll Number _______________________ *Class* __________________________________

in the Laboratory of ____

of the Department of ____

Signature of the Staff Member

Signature of the Head of the Department

*Date of Examination*_________________________

Signature of the Examiner/s

Internal Examiner

External Examiner

J.N.T.U.H. UNIVERSITY COLLEGE OF ENGINEERING, SCIENCE &
TECHNOLOGY HYDERABAD

KUKATPALLY, HYDERABAD – 500 085

Name _________________________________ *Roll Number* ________________________

Class __________________ *Year*___________ *Laboratory* _________________________

List of Experiments

S.No.	Name of the Experiment	Date of Experiment	Page Number	Marks	Remarks

J.N.T.U.H. UNIVERSITY COLLEGE OF ENGINEERING, SCIENCE & TECHNOLOGY HYDERABAD

KUKATPALLY, HYDERABAD – 500 085

Name ________________________________ *Roll Number* ________________________

Class __________________ *Year*___________ *Laboratory* _________________________

List of Experiments

S.No.	**Name of the Experiment**	**Date of Experiment**	**Page Number**	**Marks**	**Remarks**

1. Distance between 2 points

Code:

```
x1 = int(input("Enter the x coordinate of 1st point: "))
y1 = int(input("Enter the y coordinate of 1st point: "))
x2 = int(input("Enter the x coordinate of 2nd point: "))
y2 = int(input("Enter the y coordinate of 2nd point: "))
A = str((((x2-x1)*(x2-x1))+((y2-y1)*(y2-y1)))**0.5)
print('The distance between the given points is '+ A)
```

Output:

```
Enter the x coordinate of 1st point: 2
Enter the y coordinate of 1st point: 3
Enter the x coordinate of 2nd point: -4
Enter the y coordinate of 2nd point: 5
The distance between the given points is 6.324555320336759
```

2. Python interpreter as a calculator

Code:

```
a=(int)(input("Enter the first value:"))
b=(int)(input("Enter the second value:"))
c=(int)(input("For addition enter 1\nFor subtraction enter 2\nFor
multiplication enter 3\nFor     division enter 4\n"))
if c==1:
   print(a+b);
if c==2:
   print(a-b);
if c==3:
   print(a*b);
if c==4:
   print(a/b);
```

Output:

```
Enter the first value:        3
Enter the second value:    4
For addition enter 1
For subtraction enter 2
For multiplication enter 3
For division enter 4
1
7
```

3. Fibronacci Series

Code:

```
n = int(input('Enter the length: '))
 a = 0
 b = 1
 print(a)
 print(b)
 while n>0:
 c = a + b
 a = b
 b = c
 print(c)
 n = n-1
```

(or)

```
n=int(input("Enter a no.\n"))
a=0
b=1
c=0
for i in range(0,n+1):
  print(c)
  a=b
  b=c
  c=a+b
```

Output:

```
Enter a no.
8
0
1
1
2
3
```

```
5
8
13
21
```

4. Prime numbers list

Code:

```
n=(int)(input("Enter a number\n"))
for i in range(2,n):
  f=0
  for j in range(1,i):
    if((i%j)==0):
      f=f+1
  if(f<2):
      print(i)
```

Output:

```
Enter a number
9
2
3
5
7
```

5. **Write a function called is_sorted() that takes a list as a parameter and returns True if the list is sorted in ascending order and False otherwise.**

Code:

```
def is_sorted(a):
b = a.copy()
a.sort()
if a == b:
return True
else:
return False
lis = [ ]
n = int(input("Enter the length of the list: "))
for i in range(0,n):
lis.append(input("Enter the word: "))
print(is_sorted(lis))
```

Output:

```
Enter the length of the word: 4
Enter the word: 1
Enter the word: 1.02
Enter the word: abc
Enter the word: -1
False
```

6. **Write a function called has_duplicates that takes a list and returns true if there is any element that appears more than once. It should not modify the original list.**

Code:

```
def has_duplicates(a):
 b = []
 for i in a:
 if i not in b:
 b.append(i)
 print("A: "+str(a))
 print("B: "+str(b))
 if a!=b:
 return True
 else:
 return False
lis = []
n = int(input("Enter the length of list: "))
for i in range(0,n):
 lis.append(input("Enter the word: "))
print(has_duplicates(lis))
```

Output:

```
Enter the length of list: 5
Enter the word: college
Enter the word: study
Enter the word: learn
Enter the word: college
Enter the word: exams
True
```

7. **Write a function called remove_duplicates that takes a list and returns a new list with only the unique elements from the original. Hint: they don’t have to be in the same order.**

Code:

```
def remove_duplicates():
 a = []
 l = int(input("Enter the length of the list: "))
 for i in range(1,l+1):
 a.append(input("Enter the word("+str(i)+"): "))
 print(a)
 k = 0
 i = 0
 c = 0
 while i < (l-k):
 c = c + 1
 for j in range(0,l-k):
 if i!=j:
 #print(a[i],a[j],i,j,k)
 if a[i]==a[j]:
 #print('Same')
 a.remove(a[i])
 #print(a)
 k = k + 1
 break
 elif j==l-k-1:
 i = i + 1
 if c > l:
 break
 print(a)
```

```
remove_duplicates()
```

Output:

```
Enter the length of the list: 9
Enter the word(1): q
Enter the word(2): w
Enter the word(3): e
Enter the word(4): r
Enter the word(5): t
Enter the word(6): r
Enter the word(7): e
Enter the word(8): w
Enter the word(9): y
['q', 'w', 'e', 'r', 't', 'r', 'e', 'w', 'y']
['q', 't', 'r', 'e', 'w', 'y']
```

8. Add a comma between the characters. If the given word is ‘Apple’, it should become ‘A,p,p,l,e’.

Code:

```
a = input("Enter the word: ")
l = len(a)
k = 0
for i in a:
 if k == l-1:
 print(i)
 break
 print(i,end=',')
 k=k+1
```

Output:

```
Enter the word: Apple
A,p,p,l,e
```

9. Remove the given word in all the places in a string.

Code:

```
line = input("Enter the sentence: ")
word = input("Enter the word to be omitted: ")
lin = line.split()
for i in lin:
 if i != word:
 print(i, end = ' ')
```

Output:

```
Enter the sentence: Good evening. This is XYZ from LMNO
Enter the word to be omitted: is
Good evening. This  XYZ from LMNO
```

10. Write a function that takes a sentence as an input parameter and replaces the first letter of every word with the corresponding upper-case letter and the rest of the letter corresponding letters in the lower-case without using a built-in function.

Code:

```
line = input("Enter the sentence: ")
word = input("Enter the word to be omitted: ")
lin = line.split()
for i in lin:
 if i != word:
 print(i, end = ' ')
```

Output:

```
Enter the sentence: This is a nOTE BoOk
This Is A Note Book
```

11. Matrix Operations

Code:

```
import numpy as np
from scipy.linalg import svd
A = np.array(([1,2,3],[4,5,6],[2,0,-1]))
B = np.array(([2,8,0],[8,2,0],[0,-1,-5]))
print("Matrix A:")
print(A)
print("Matrix B:")
print(B)
print("A+B:")
print(A+B)
print("A-B:")
print(A-B)
print("A*B:")
print(A.dot(B))
print("Transpose of A:")
print(A.T)
print("Inverse of A or A^-1:")
print(np.linalg.inv(A))
print("Trace of A:")
print(np.trace(A))
print("|A|:")
print(np.linalg.det(A))
print("Rank of A: ")
print(np.linalg.matrix_rank(A))
print("Eigen values:")
u,v = np.linalg.eig(A)
print(u)
#print("Eigen vectors:")
```

```
#print(v)
print("SVD A:")
U,s,VT = svd(A)
print("U = ")
print(U)
print("s = ")
print(s)
print("VT = ")
print(VT)
```

Output:

```
Matrix A:
[[ 1 2 3]
 [ 4 5 6]
 [ 2 0 -1]]

Matrix B:
[[ 2 8 0]
 [ 8 2 0]
 [ 0 -1 -5]]

A+B:
[[ 3 10 3]
 [12 7 6]
 [ 2 -1 -6]]

A-B:
[[-1 -6 3]
 [-4 3 6]
 [ 2 1 4]]
```

```
A*B:
[[ 18 9 -15]
[ 48 36 -30]
 [ 4 17 5]]

Transpose of A:
[[ 1 4 2]
 [ 2 5 0]
 [ 3 6 -1]]

Inverse of A or A^-1:
[[ 1.66666667 -0.66666667 1. ]
 [-5.33333333 2.33333333 -2. ]
 [ 3.33333333 -1.33333333 1. ]]

Trace of A:
5

|A|:
-2.999999999999999

Rank of A:
3

Eigen values:
[ 7.06347233 -2.25206391 0.18859158]

SVD A:
U =
[[-0.38596417 -0.30466343 -0.87075361]
 [-0.92235572 0.10997264 0.37035921]
 [-0.01707584 0.94608996 -0.32345355]]

s =
[9.50934171 2.35679908 0.1338592 ]

VT =
[[-0.43215807 -0.56614928 -0.70193618]
 [ 0.86023754 -0.02523069 -0.50926888]
 [-0.27061187 0.82391651 -0.49792651]]
```

12. Machine Learning Statistics

Code:

```
import numpy as np
from scipy import stats
data = np.array([1,2,3,4,5,6,7,8,8,9])
print("Mean, Median, Mode: ")
print("Mean = "+ str(np.mean(data)))
print("Median = "+ str(np.median(data)))
print("Mode =", stats.mode(data).mode[0], " (occurs", stats.mode(data).count[0], "times)")
percentile_50 = np.percentile(data, 50)
quartiles = np.percentile(data, [25, 50, 75])
iqr = np.percentile(data, 75) - np.percentile(data, 25)
print("50th Percentile:", percentile_50)
print("Quartiles (Q1, Q2, Q3):", quartiles)
print("Interquartile Range (IQR):", iqr)
mean_abs_deviation = np.mean(np.abs(data - np.mean(data)))
std_deviation = np.std(data)
variance = np.var(data)
print("Mean Absolute Deviation:", mean_abs_deviation)
print("Standard Deviation:", std_deviation)
print("Variance:", variance)
maximum = np.max(data)
minimum = np.min(data)
print("Maximum:", maximum)
print("Minimum:", minimum)
```

Output:

```
Mean, Median, Mode:
Mean = 5.3
Median = 5.5
Mode = 8 (occurs 2 times)
50th Percentile: 5.5
Quartiles (Q1, Q2, Q3): [3.25 5.5 7.75]
Interquartile Range (IQR): 4.5
Mean Absolute Deviation: 2.3
Standard Deviation: 2.6095976701399777
Variance: 6.81
Maximum: 9
Minimum: 1
```

13.Truth Table

Code:

```
def or_oper(p,q):
    return(bool(p) or bool(q))
def and_oper(p,q):
    return(bool(p) and bool(q))
def not_oper(p):
    return(not (bool(p)))
def imply_oper(p,q):
    return(not(bool(p)) or bool(q))
def doubleimply_oper(p,q):
    m=bool(p)
    n=bool(q)
    return((not(m) or n) and (not(n) or m))
lis=["p","q","p or q","p^q","~p","~q","p->q","p<=>q"]
for i in lis:
    print(i,end="\t")
print("\n")
for i in range(1,-1,-1):
    for j in range(1,-1,-1):

A=[bool(i),bool(j),or_oper(i,j),and_oper(i,j),not_oper(i),not_oper(j),imply_oper(i,j)
,doubleimply_oper(i,j)]
        for i in A:
            print(i,end="\t")
        print("\n")
```

Output:

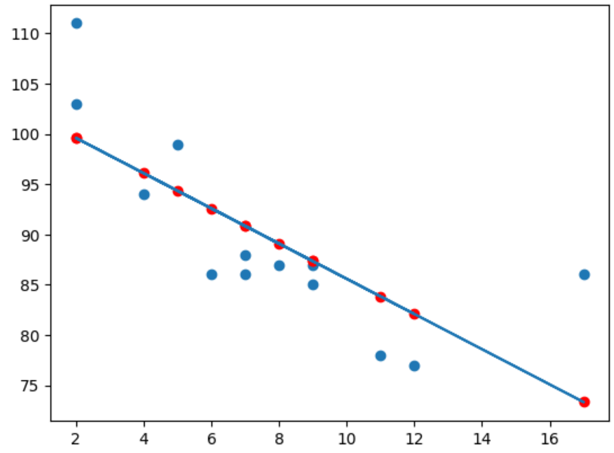
```
p      q      por q  p^q    ~p     ~q     p->q   p<=>q

True   True   True   True   False  False  True   True

True   False  True   False  False  True   False  False

False  True   True   False  True   False  True   False

False  False  False  False  True   True   True   True
```

13. Linear Regression

Code:

```
import matplotlib.pyplot as plt
from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std_err = stats.linregress(x, y)

def fun(x):
  return slope * x + intercept

t= list(map(fun, x))
print(t)
print(r)
plt.scatter(x, y)
plt.plot(x, t)
plt.scatter(x,t,color="red")
plt.show()
```

Output:

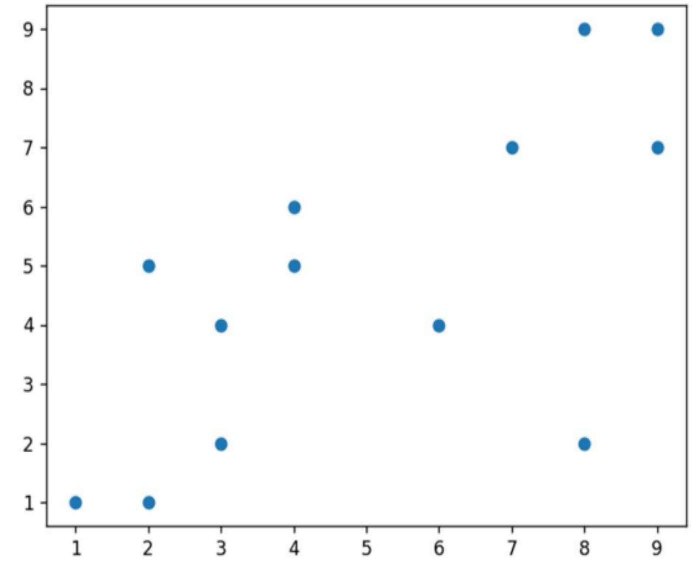


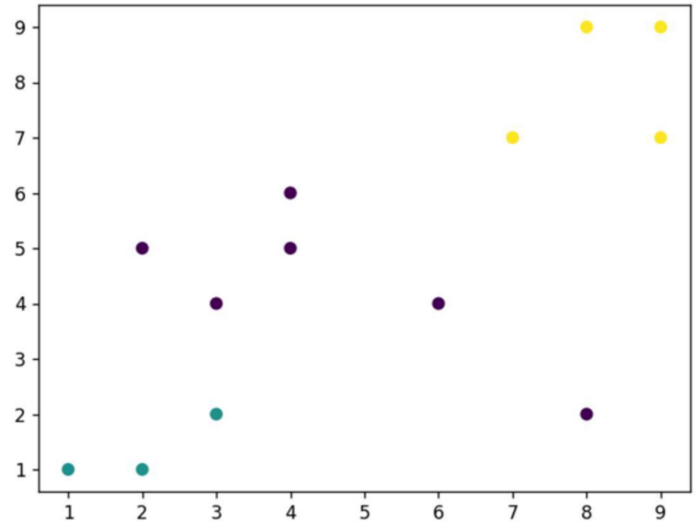
14. K Nearest Neighbor (KNN)

Code:

```
import matplotlib.pyplot as plt

x = [4, 5, 10, 4, 3, 11, 14 , 8, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
classes = [0, 0, 1, 0, 0, 1, 1, 0, 1, 1]

plt.scatter(x, y, c=classes)
plt.show()

from sklearn.neighbors import KNeighborsClassifier

t = list(zip(x, y))
knn = KNeighborsClassifier(n_neighbors=3)

knn.fit(t, classes)

new_x = 8
new_y = 21
new_point = [(new_x, new_y)]

prediction = knn.predict(new_point)

plt.scatter(x + [new_x], y + [new_y], c=classes + [prediction[0]])
plt.text(x=new_x-1.7, y=new_y-0.7, s=f"new point, class: {prediction[0]}")
plt.show()
```

Output:





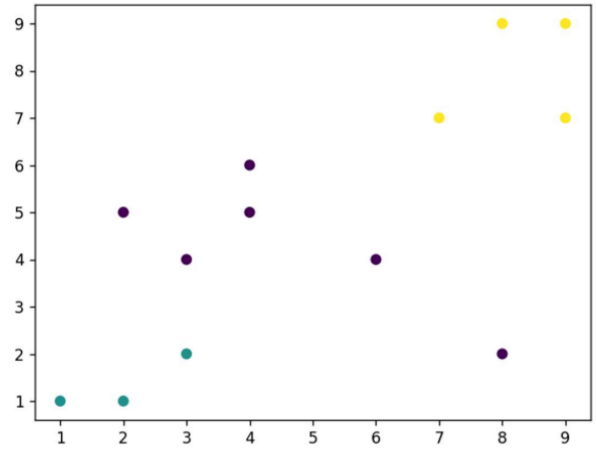
15. K Means Clustering

Code:

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
x = [1,8,9,4,2,7,4,8,2,6,3,9,3]
y = [1,9,7,5,1,7,6,2,5,4,4,9,2]
data = list(zip(x,y))
k = int(input("Number of Clusters: "))
plt.scatter(x,y)
plt.show()
kmeans = KMeans(n_clusters = k)
kmeans.fit(data)
plt.scatter(x,y,c=kmeans.labels_)
plt.show()
```

Output:

```
Number of Clusters: 3
```



16. Gradient Descent

Code:

```
import numpy as np
import matplotlib.pyplot as plt
def gd(x,y):
 m = 0
 c = 0
 n = len(x)
 lr = float(input("Enter the learning rate: "))
 ite = int(input("Enter the number of iterations: "))
 for i in range(ite):
 yp = m*x+c
 md = -(2/n)*sum(x*(y-yp))
 cd = -(2/n)*sum((y-yp))
 cost = (1/n)*sum([val**2 for val in (y-yp)])
 m = m - lr*md
 c = c - lr*cd
 print(f"y = {m}x + {c}, cost = {cost}")
 print(f"Therefore, y = {m}x + {c}, cost = {cost}")
X = np.array([1,2,3,4,5])
Y = np.array([5,7,9,11,13])
gd(X,Y)
```

Output:

```
Enter the learning rate: 0.001
Enter the number of iterations: 90
y = 0.062x + 0.018000000000000002, cost = 89.0
Therefore, y = 0.062x + 0.018000000000000002, cost = 89.0
```

17. Logistic Regression

Code:

```
import numpy as np
from sklearn import linear_model
x = np.array([25,29,30,31,41,42,44,49,50,59,60,62,68,72,79,80,81,84]).reshape(-1,1)
y = np.array([0,0,0,0,0,0,1,1,0,1,0,0,1,0,1,0,1,1])
logr = linear_model.LogisticRegression()
logr.fit(x,y)
tp = int(input("Enter the value to be predicted: "))
predicted = logr.predict(np.array([tp]).reshape(-1,1))
print(f"predicted = {predicted}")
log_odds = logr.coef_
odds = np.exp(log_odds)
print(f"Odds = {odds}")
def logit2prob(logr,x):
 log_odds = logr.coef_ * x + logr.intercept_
 odds = np.exp(log_odds)
 probability = odds / (1 + odds)
 return(probability)
print(f"For all:\n{logit2prob(logr, x)}")
```

Output:

```
Enter the value to be predicted: 99
predicted = [1]
Odds = [[1.06239359]]
For all:
[[0.08365596]
 [0.10418355]
 [0.10996909]
 [0.11603431]
 [0.19383398]
 [0.20346724]
 [0.22379012]
 [0.28067982]
 [0.29306019]
 [0.41681857]
 [0.43160081]
 [0.46150784]
 [0.5520292 ]
 [0.61087002]
 [0.70571069]
 [0.71812228]
 [0.73021053]
 [0.76445557]]
```