

**A**  
**Lab Record**  
**On**  
**Python Programming With Problem**  
**Master Of Computer Application - I Sem**



**RUNGTA INTERNATIONAL SKILLS UNIVERSITY**

**SESSION: 2025-26**

**Submitted To:-**  
**Ms. KAVITA KANWAR**

**Submitted By:-**  
**TANU CHAUDHARY**  
**Ref No. RU-25-11704**

**RUNGTA INTERNATIONAL SKILLS  
UNIVERSITY,CG  
SCHOOL OF INFORMATION TECHNOLOGY**

# INDEX

<b>S. No</b>	<b>Name of Practical</b>	<b>Submission Date</b>	<b>Remarks</b>
<b>10</b>	Write a python program using math module to calculate: a) Square root of a number b) Factorial of a number c) Power of a number		
<b>11</b>	Write a python program to find the area and circumference of a circle using math module, take input from the user.		
<b>12</b>	Create a list of student and select CR randomly, then shuffle the list and print it.		
<b>13</b>	Write a python program to generate a random password of 12 characters using UPPERCASE, lowercase, digits and special symbols(!, @, #, \$, *). Password should start with Capital letter or digits.		
<b>14</b>	Create a package named shapes that contains two modules: circle.py and rectangle.py. Each module should compute area and perimeter of the shape. Write the complete directory structure and sample code.		
<b>15</b>	Write a program that asks the user for an index and prints the element at that index from a predefined list. Handle: <ul style="list-style-type: none"><li>• IndexError (index out of range)</li><li>• TypeError (if user enters a non-integer index)</li><li>• Print the specific exception message using except Exception as e.</li></ul>		
<b>16</b>	Implement a supervised learning model to classify flowers in the Iris dataset using decision tree classifier. Print the accuracy of the model.		
<b>17</b>	Use Support Vector Machine (SVM) on the breast cancer datasets to classify malignant vs benign tumors.		
<b>18</b>	Perform Principal Component Analysis (PCA) on the digits dataset and reduce the dimension to 2. Print the explained variance ratio.		
<b>19</b>	Apply K-Means clustering on the Iris Dataset and print cluster labels.		

## AIM 10:

**Write a python program using math module to calculate:**

- **Square root of a number**
- **Factorial of a number**
- **Power of a number (take input from the user)**

## Source Code:

MCA-> TANU CHAUDHARY-> PYTHON-> LAB-RECORD > Lab\_10.ipynb > base = float(input("Enter the base number: "))  
Generate + Code + Markdown | Run All ⚡ Restart ✖ Clear All Outputs | Jupyter Variables ⚙️ Outline ⋮

```
[6] import math  
      ✓ 0.0s
```

```
[7] num = float(input("Enter a number to find its square root: "))  
  
sqrt_result = math.sqrt(num)  
print("Square root of", num, "is:", sqrt_result)  
      ✓ 2.6s
```

... Square root of 4.0 is: 2.0

```
[8] n = int(input("Enter a number to find its factorial: "))  
  
fact_result = math.factorial(n)  
print("Factorial of", n, "is:", fact_result)  
      ✓ 1.5s
```

... Factorial of 5 is: 120

```
[9] ▶ base = float(input("Enter the base number: "))  
  
power = float(input("Enter the power: "))  
  
power_result = math.pow(base, power)  
print(base, "raised to the power", power, "is:", power_result)  
      ✓ 3.6s
```

... 2.0 raised to the power 3.0 is: 8.0

## AIM 11:

**Write a python program to find the area and circumference of a circle using math module, take input from the user.**

### Source Code:

```
MCA- > TANU CHAUDHARY- > PYTHON- > LAB-RECORD > Lab_11.ipynb > radius = f
❖ Generate + Code + Markdown | ⏪ Run All ⏴ Restart ✖ Clear All Outputs
```

```
[11] import math
[11] ✓ 0.0s
```

```
▷ v [12] radius = float(input("Enter the radius of the circle: "))
      print("Entered radius:", radius)
[12] ✓ 1.5s
... Entered radius: 2.0
```

```
[13] area = math.pi * radius * radius
      print("Area of the circle is:", area)
[13] ✓ 0.0s
... Area of the circle is: 12.566370614359172
```

```
[14] circumference = 2 * math.pi * radius
      print("Circumference of the circle is:", circumference)
[14] ✓ 0.0s
... Circumference of the circle is: 12.566370614359172
```

## AIM 12:

**Create a list of student and select CR randomly, then shuffle the list and print it.**

### Source Code:

MCA- > TANU CHAUDHARY- > PYTHON- > LAB-RECORD > Lab\_12.ipynb > cr = random.choice(students)

Generate + Code + Markdown | Run All Restart Clear All Outputs Jupyter Variables Outline ...

Generate + Code + Markdown

```
import random
```

[1] ✓ 0.0s

```
students = ["Tanu", "Akriti", "Rimsa", "Shreya", "Purwa", "Khushika", "Sneha", "Anjali"]  
print("Student list:", students)
```

[2] ✓ 0.0s

... Student list: ['Tanu', 'Akriti', 'Rimsa', 'Shreya', 'Purwa', 'Khushika', 'Sneha', 'Anjali']

▶ ✓ cr = random.choice(students)  
print("Selected CR is:", cr)

[10] ✓ 0.0s

... Selected CR is: Tanu

```
random.shuffle(students)  
print("Shuffled student list:", students)
```

[4] ✓ 0.0s

... Shuffled student list: ['Akriti', 'Purwa', 'Tanu', 'Khushika', 'Shreya', 'Sneha', 'Rimsa', 'Anjali']

### AIM 13:

**Write a python program to generate a random password of 12 characters using UPPERCASE, lowercase, digits and special symbols(!, @, #, \$, \*). Password should start with Capital letter or digits.**

### Source Code:

```
MCA- > TANU CHAUDHARY- > PYTHON- > LAB-RECORD > Lab_13.ipynb > print("Ge
❖ Generate + Code + Markdown | ▶ Run All ⚡ Restart ✖ Clear All Outputs
[84] ✓ 0.0s
import random
import string

[85] ✓ 0.0s
uppercase = string.ascii_uppercase
lowercase = string.ascii_lowercase
digits = string.digits
symbols = "!@#$*"

[86] ✓ 0.0s
first_char = random.choice(uppercase + digits)
password = first_char

[87] ✓ 0.0s
all_chars = uppercase + lowercase + digits + symbols

for i in range(11):
    password += random.choice(all_chars)

[88] ✓ 0.0s
print("Generated Password:", password)
... Generated Password: 8xXqIoloCsM@
```

## AIM 14:

Create a package named shapes that contains two modules: circle.py and rectangle.py. Each module should compute area and perimeter of the shape. Write the complete directory structure and sample code.

### Source Code:

MCA- > TANU CHAUDHARY- > PYTHON- > LAB-RECORD > Shapes > circle.py > ...

```
1 import math
2
3 def area(radius):
4     return math.pi * radius * radius
5
6 def perimeter(radius):
7     return 2 * math.pi * radius
```

MCA- > TANU CHAUDHARY- > PYTHON- > LAB-RECORD > Shapes > rectangle.py > ...

```
1 def area(length, breadth):
2     return length * breadth
3
4 def perimeter(length, breadth):
5     return 2 * (length + breadth)
```

MCA- > TANU CHAUDHARY- > PYTHON- > LAB-RECORD > Lab\_14.py > ...

```
1 from Shapes import circle, rectangle
2
3 # Circle
4 r = float(input("Enter radius of the circle: "))
5
6 print("Circle Area:", circle.area(r))
7 print("Circle Perimeter:", circle.perimeter(r))
8
9 # Rectangle
10 l = float(input("Enter length of the rectangle: "))
11 b = float(input("Enter breadth of the rectangle: "))
12
13 print("Rectangle Area:", rectangle.area(l, b))
14 print("Rectangle Perimeter:", rectangle.perimeter(l, b))
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER

```
PS C:\Users\ASUS\Documents> & C:/Users/ASUS/AppData/Local/Microsoft
Enter radius of the circle: 2
Circle Area: 12.566370614359172
Circle Perimeter: 12.566370614359172
Enter length of the rectangle: 3
Enter breadth of the rectangle: 4
Rectangle Area: 12.0
Rectangle Perimeter: 14.0
```

### AIM 15:

Write a program that asks the user for an index and prints the element at that index from a predefined list. Handle:

- **IndexError** (index out of range)
- **TypeError** (if user enters a non-integer index)
- Print the specific exception message using except Exception as e.

### Source Code:

```
MCA- > TANU CHAUDHARY- > PYTHON- > LAB-RECORD > Lab_15.ipynb > try:  
Generate + Code + Markdown | Run All Restart Clear All Output
```

```
[4]     items = [10, 20, 30, 40, 50]  
         print("List of items:", items)  
✓ 0.0s  
... List of items: [10, 20, 30, 40, 50]
```

```
try:  
    index = int(input("Enter the index: "))  
    print("Entered index is:", index)  
    print("Element at index", index, "is:", items[index])  
  
except IndexError:  
    print("Error: Index out of range")  
  
except ValueError:  
    print("Error: Index must be an integer")  
  
except Exception as e:  
    print("Error message:", e)  
✓ 1.5s  
... Entered index is: 5  
Error: Index out of range
```

## AIM 16:

**Implement a supervised learning model to classify flowers in the Iris dataset using decision tree classifier. Print the accuracy of the model.**

### Source Code:

```
MCA- > TANU CHAUDHARY- > PYTHON- > LAB-RECORD > Lab_16.ipynb > X_train, X_test, y_train, y_test = train_test_split(  
↳ Generate + Code + Markdown | Run All ⏪ Restart ⏴ Clear All Outputs | Jupyter Variables ⏴ Outline  
[1] import pandas as pd  
from sklearn.datasets import load_iris  
from sklearn.model_selection import train_test_split  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.metrics import accuracy_score  
[1] ✓ 0.9s  
  
[2] ▶ v  
    iris = load_iris()  
    X = iris.data  
    y = iris.target  
  
    df = pd.DataFrame(X, columns=iris.feature_names)  
    df['target'] = y  
    df.head()  
[2] ✓ 0.0s  
...  
    sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  target  
0           5.1          3.5            1.4           0.2          0  
1           4.9          3.0            1.4           0.2          0  
2           4.7          3.2            1.3           0.2          0  
3           4.6          3.1            1.5           0.2          0  
4           5.0          3.6            1.4           0.2          0  
  
[3] ▶ v  
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)  
    X_train.shape, X_test.shape  
[3] ✓ 0.0s  
... ((120, 4), (30, 4))
```

```
[5]     dt_model = DecisionTreeClassifier(random_state=42)
          dt_model.fit(X_train, y_train)
```

✓ 0.0s

... ▾ DecisionTreeClassifier ⓘ ?

► Parameters

```
y_pred = dt_model.predict(X_test)
y_pred
```

✓ 0.0s

```
... array([1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0, 2,
          0, 2, 2, 2, 2, 0, 0])
```

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy of Decision Tree Classifier:", accuracy)
```

✓ 0.0s

... Accuracy of Decision Tree Classifier: 1.0

## AIM 17:

**Use Support Vector Machine (SVM) on the breast cancer datasets to classify malignant vs benign tumors.**

**Source Code:**

```
MCA->TANU CHAUDHARY->PYTHON->LAB-RECORD > Lab_17.ipynb > svm_model = SVC(kernel='linear', random_state=42)
↳ Generate + Code + Markdown | ⚡ Run All ⚡ Restart ✎ Clear All Outputs | 📁 Jupyter Variables ⚡ Outline ...
    % Generate + Code + Markdown
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
[1] ✓ 1.0s Python 3.11.9

> v
cancer = load_breast_cancer()
X = cancer.data
y = cancer.target

df = pd.DataFrame(X, columns=cancer.feature_names)
df['target'] = y
df.head()
[2] ✓ 0.0s Python

...
mean mean mean mean mean mean mean concave mean mean mean
radius texture perimeter area smoothness compactness concavity points symmetry fractal dimension ... texture perimeter area smoothness compactness concavity worst worst worst worst
1 17.99 10.38 122.80 1001.0 0.11840 0.27760 0.3001 0.14710 0.2419 0.07871 ... 17.33 184.60 2019.0 0.1622 0.6656 0.7119 0.2654 0.4601 0.11890 0
2 20.57 17.77 132.90 1326.0 0.08474 0.07864 0.0869 0.07017 0.1812 0.05667 ... 23.41 158.80 1956.0 0.1238 0.1866 0.2416 0.1860 0.2750 0.08902 0
! 19.69 21.25 130.00 1203.0 0.10960 0.15990 0.1974 0.12790 0.2069 0.05999 ... 25.53 152.50 1709.0 0.1444 0.4245 0.4504 0.2430 0.3613 0.08758 0
! 11.42 20.38 77.58 386.1 0.14250 0.28390 0.2414 0.10520 0.2597 0.09744 ... 26.50 98.87 567.7 0.2098 0.8663 0.6869 0.2575 0.6638 0.17300 0
! 20.29 14.34 135.10 1297.0 0.10030 0.13280 0.1980 0.10430 0.1809 0.05883 ... 16.67 152.20 1575.0 0.1374 0.2050 0.4000 0.1625 0.2364 0.07678 0
rows x 31 columns

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
X_train.shape, X_test.shape
[3] ✓ 0.0s Python

... ((455, 30), (114, 30))
```

```
▷ <  svm_model = SVC(kernel='linear', random_state=42)
      svm_model.fit(X_train, y_train)
[4]   ✓  0.8s
```

... ▾ SVC ⓘ ⓘ  
▶ Parameters

```
    y_pred = svm_model.predict(X_test)
    y_pred
[5]   ✓  0.0s
...
array([1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1,
       0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1,
       0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0,
       0, 1, 0, 0])
```

```
▷ <  accuracy = accuracy_score(y_test, y_pred)
      print("Accuracy of SVM Classifier:", accuracy)

      print("\nClassification Report:\n", classification_report(y_test, y_pred))
      print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
[6]   ✓  0.0s
```

... Accuracy of SVM Classifier: 0.956140350877193

Classification Report:				
	precision	recall	f1-score	support
0	0.97	0.91	0.94	43
1	0.95	0.99	0.97	71
accuracy			0.96	114
macro avg	0.96	0.95	0.95	114
weighted avg	0.96	0.96	0.96	114

Confusion Matrix:  
[[39 4]  
 [ 1 70]]

## AIM 18:

Perform Principal Component Analysis (PCA) on the digits dataset and reduce the dimension to 2. Print the explained variance ratio.

### Source Code:

```
MCA- > TANU CHAUDHARY- > PYTHON- > LAB-RECORD > Lab_18.ipynb > plt.figure(figsize=(8,6))
↳ Generate + Code + Markdown | ⏪ Run All ⏴ Restart ✖ Clear All Outputs | Jupyter Variab
    ↳ Generate + Code
```

```
[1] import matplotlib.pyplot as plt
     from sklearn.datasets import load_digits
     from sklearn.decomposition import PCA
     ✓ 1.5s
```

```
[2] digits = load_digits()
     X = digits.data
     y = digits.target

     X.shape, y.shape
     ✓ 0.0s
... ((1797, 64), (1797,))
```

```
[3] pca = PCA(n_components=2)
     X_pca = pca.fit_transform(X)

     X_pca.shape
     ✓ 0.0s
... (1797, 2)
```

```
▷ ▾ [4] explained_variance = pca.explained_variance_ratio_
     print("Explained variance ratio:", explained_variance)
     print("Total variance explained by 2 components:", explained_variance.sum())
     ✓ 0.0s
... Explained variance ratio: [0.14890594 0.13618771]
     Total variance explained by 2 components: 0.2850936482369929
```

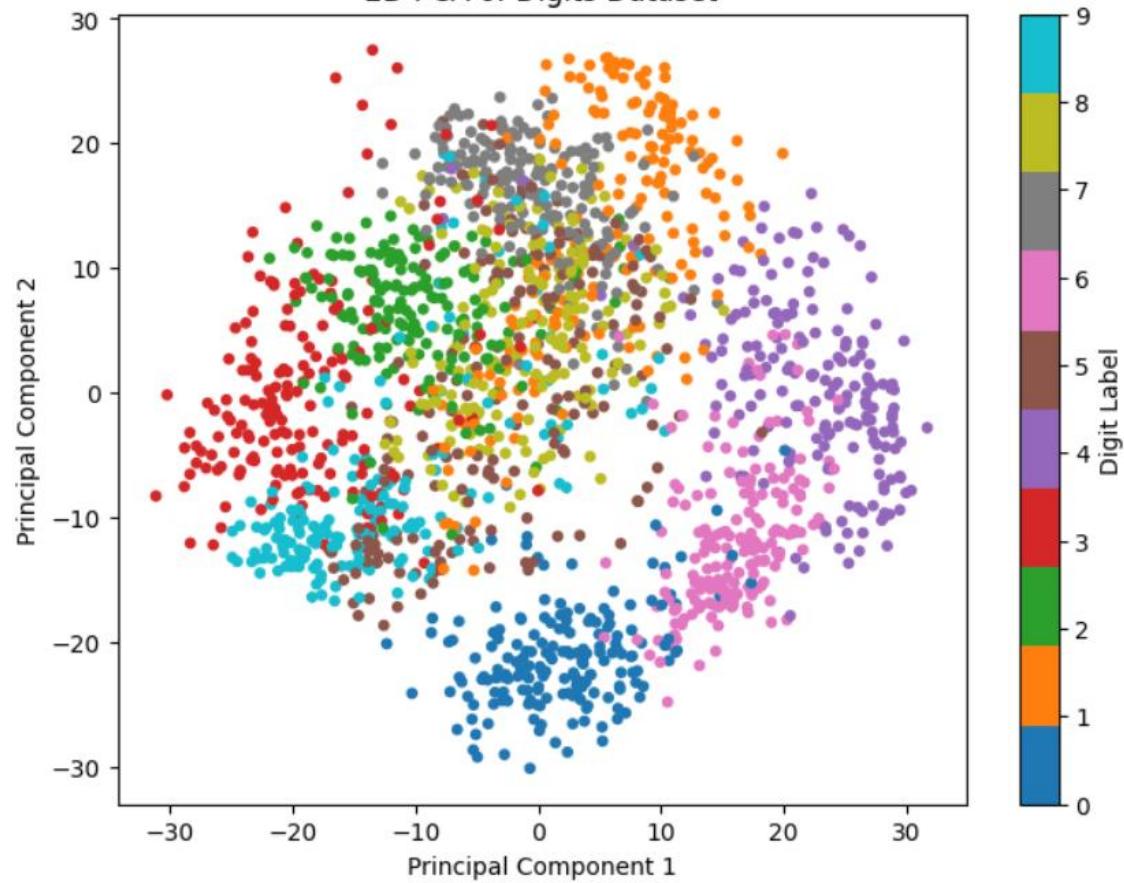
▷ ▾

```
plt.figure(figsize=(8,6))
plt.scatter(X_pca[:,0], X_pca[:,1], c=y, cmap='tab10', s=15)
plt.colorbar(label='Digit Label')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('2D PCA of Digits Dataset')
plt.show()
```

[5] ✓ 3.5s

...

2D PCA of Digits Dataset



## AIM 19:

Apply K-Means clustering on the Iris Dataset and print cluster labels.

Source Code:

The screenshot shows a Jupyter Notebook interface with the following steps:

- Step 1:** Imports pandas, loads the Iris dataset, and initializes KMeans.  
Code: 

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
```

  
Timing: [1] ✓ 18.2s
- Step 2:** Loads the Iris dataset into a DataFrame and prints the first few rows.  
Code: 

```
iris = load_iris()
X = iris.data
y = iris.target

df = pd.DataFrame(X, columns=iris.feature_names)
df['target'] = y
df.head()
```

  
Timing: [2] ✓ 0.0s  
Output: A table showing the first 5 rows of the Iris dataset:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

- Step 5:** Trains the KMeans model with 3 clusters.  
Code: 

```
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)
```

  
Timing: [5] ✓ 0.0s  
Output: A dropdown menu for the KMeans object is shown, with "Parameters" expanded.

```
[7] df['cluster'] = labels  
df.head(10)  
[7]: 0.0s  
...  


|   | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) | target | cluster |
|---|-------------------|------------------|-------------------|------------------|--------|---------|
| 0 | 5.1               | 3.5              | 1.4               | 0.2              | 0      | 1       |
| 1 | 4.9               | 3.0              | 1.4               | 0.2              | 0      | 1       |
| 2 | 4.7               | 3.2              | 1.3               | 0.2              | 0      | 1       |
| 3 | 4.6               | 3.1              | 1.5               | 0.2              | 0      | 1       |
| 4 | 5.0               | 3.6              | 1.4               | 0.2              | 0      | 1       |
| 5 | 5.4               | 3.9              | 1.7               | 0.4              | 0      | 1       |
| 6 | 4.6               | 3.4              | 1.4               | 0.3              | 0      | 1       |
| 7 | 5.0               | 3.4              | 1.5               | 0.2              | 0      | 1       |
| 8 | 4.4               | 2.9              | 1.4               | 0.2              | 0      | 1       |
| 9 | 4.9               | 3.1              | 1.5               | 0.1              | 0      | 1       |


```