EXERCISE NO:1	INTRODUCTION TO PYTHON & JUPYTER NOTEBOOK
DATE :	

AIM:

To understand the basic concepts of the python programming language and its applications, and to become familiar with the jupyter Notebook environment.

Apparatus / Software Required

A personal computer (Windows/Linux /Macos):

Internet connection.

* Jupyter notebook (through Anaconda or direct installation of jupyter)

Theary / Background.

Python is a high-level, interpreted programming language widely used for its simple syntax, readability and versatility. It is applied in diverse fields such as much development, data science, artificial intelligence, and automation

Jupyter notebook is an interactive environment that combines code, results and documentation in a single document, making it ideal for learning, research and experimentation

Procedure:

Read the "Python Introduction" section to understand key features such as simplicity, Portability and library support.

Examine the and learn the and comments. "Hello, World! " program and learn the use of the print () function and comments (#,” ”….” ‘’)

Study Python's indentation rules and observe how improper indentation leads to errors.

Review real-world application of Python to understand its importance in current technology

Read about Jupyter Notebook and identify its main features (interactive execution,Combination of code and text).

Familiarize yourself with cells (code/Markdown) and the kernel (execution angine).

Result :

Thus the foundational understanding of python's syntax, features and applications, along with the structure and purpose of Jupyter notebook was achieved

EXERCISE NO:2	INSTALLING PYTHON AND JUPITER NOTEBOOK USING ANACONDA DISTRIBUTION
DATE :	

AIM:

To install Python and jupyter notebook using the anaconda distribution and verify the installation.

Apparatus /Software Required

A personal computer with windows/Linus/Macos.

Internet connection

Anaconda Distribution (latest version)

Theory/Background.

The Anaconda distribution is an free and open source platform that simplifies package management and deployment. It comes with Python, Jupyter Natbook, and pre- installed libraries (NumPy, Pandas, Matplattih. etc.), making it highly suitable for data science and machine laming application

Procedure:

Download the anaconda installer from the official website corresponding to your operating system.

Run the installer and follow on-screen instruction.

Accept the license agreement and choose the 'Just Me' installation type (Recommended).

Select the destination folder (default suggestion)

Proceed with the installation process Until completion.

Launch Jupyter Notebook via Start Menu or by typing jupyter notebook in the command prompt.

In the jupyter Dashboard create a new notebook by selecting python 3 Lipykernel

Test the setup by typing paint ("Hello World!") and executing the cell with ctrl + Enter

Result :

The Anaconda Distribution was successfully installed, providing python, Jupyter Notebook, and essential libraries

EXERCISE NO:3	PRINTING OPTIONS IN PYTHON
DATE :	

AIM:

To write a python program to demonstrate all the printing options.

Procedure:

Step 1: Use the following print option used in Python

printing! the simple message

Printing the message along with the assigned name and age

Printing the assigned message

Printing the content in separate output file output. text).

Step 2 : Print the results.

Source Code :

```
     Paint ('Hello', 'world, sep=',')
    Paint ('This is the end., and = '')
    Print ('My massage.')
    Name age = ‘Alice'
    Age = 30
      Print If "My name is {name}  and I am {age} years old.")
    Name =’Bah'
    Age = 25
Message = “My name is {} and I am {} years old."
        format (name, age) p
paint (message)
name = ‘Charlie’
age = 35
message =” My name is "+ name + and I am"+ Str (age) + "years old."
Print (message)
 Name = ‘David’
Age = 40
Message = “My name is % is and I am % d years old." % (name, age)
Print (message)
```

Output

```
Hello, warlol
This is the end. My message
My name is alice and I am 30 years old
My name is Boh and I am 25 old years
My name is Charlie and I am 35 old years
My name is Damid and I am 40 old years
```

Result :

Thus all the printing option for python has been demonstration successfully.

EXERCISE NO:4	VARIABLES IN PYTHON
DATE :	

AIM:
To learn how to create and use variables in Python to store and manipulate different types of data.

PROCEDURE:
Open Python (IDLE/ Jupyter Notebook /any IDC).
Create variables of different data types (Integer, float, string, Boolean)
Assign values to variables and perform simple operations
Print the variables to absence the stand values.
Use the type () function to check the data type of each variable
Run the program and verify the output.

Program (code):

```
# Exercise: Variables in Pythan
# Creating variables
Product _ name =” laptop” # String variable
Quantity = 5 #Integar variable
Price = 45000. 75 # Float variable
Available = True # Boolean variable
# Performing operations
Total cost = quantity "price
# Displaying variable values
print ("Product", product_name)
print ("Quantity:", quantity)
print ("price per unit:" price)
Print ("Available:", available)
Print ("Total Cost:", total. cost)
#Checking data types
Print ("Data type of product_name:", type (product_name))
print ("Data type of quantity:", type (quantity))
print ("Data type of price:", type (price))
print ("Data type of available:", type (available)
Print ("Data type of total cost:", type (total cost))
```

Result:
The program was accented successfully. Variable of different data types were created, values were assigned, operations were performed and data types were verified.

EXERCISE NO:5	DATA TYPES IN PYTHON
DATE :	

AIM:

To understand how to create assign, and use variables in python for storing and manipulating data.

Procedure

Open Python (IDLE, Jupyter Notebook, or any python IDE).
Create variables of different data types such
as intiger, float, string and hoolan.
Assigner values to the variables
Perform simple arithmetic operations using Variables
Use the print () function to display the variable values.
Use the type () function to check the data type of each variable
Execute the program and observe the output.

Program (code):

```
# Exercise: Variables in Python
# Declaring variables of different data types
Student_ name = ‘john’ # String variable
Age = 22 # Integer variable
Gpa  =8.5 # Float variable
is-present = true # Boolean variable
# performing an operation
Next year - age = age +1
# Displaying variable values
print ("Student Name:", Student_name)
Print ("Age:", age)
print (" GPA: ", gpa)
print ("Is present:", is- present)
print (" Age Neset_ Year _ age:", type Inset year-age))
```

Result

The program was executed successfully. Variables of different data types were created, values were assigned, operations were performed and this data verified. This confirms that python can effectively handle variable in business-related computation.

EXERCISE NO:6	ARITHMETIC OPERATORS IN PYTHON
DATE :	

AIM:

To study and demonstrate the use operators in Python

Procedure:

Open Python (IDLE, Jupyter Notebook, or IDE)

Declare numeric variables and assign values.

Apply arithmetic operators (+, -,*,/,//,%,**) on the variables.

Use the print () function to display results for each operator

Execute the program and verify the output.

Program (Code):

```
# Exercise Arithmetic Operators in Python

# Declaring variables

a = 15

b = 4

# Applying arithmetic operators

Print ("a", a, "b=",b)

Print ("Addition (a+b):", a+b)

Print ("Subtraction (a-b):", a-b)

Print ("Multiplication (a*b):", a*b)

Paint ("Division (a/b):", a/b)

Print ("Floor Division (a/b)=", a//b)

Print ("Modulus (a/b):", a/b)

Print ("Exponentiation (a**b):", a **b)
```

Result

The program was executed successfully Different arithmetic operators in python were applied were an numeric variables and their results displayed.

EXERCISE NO:7	PROGRAM TO FIND AREA OF TRIANGLE
DATE :	

AIM:

To write a python program to find the area of a triangle.

PROCEDURE:

Step 1 : Get the values of breadth and height from the user

Step 2: Use the formula 1/2*b*h to find the area of triangle.

Step 3: Print the results.

Source Code:

```
b = int (input ("Enter breadth of a triangle: '))
h = int (input ("Enter height of a triangle: "))
area = (b*h)12
Print ("The area of triangle is, area)
```

RESULT

Thus the area of a triangle has been found out successfully

EXERCISE NO:8	PROGRAM TO FIND SQUARE ROOT
DATE :	

AIM:

To write a python program to find the square root.

PROCEDURE:

Step 1: Get an integer number

Step 2: Find the square root of the number

Step 3: Print the results

SOURCE CODE:

```
# To take the input from the user
Num =  float (input ('Enter a number :'))
num sqrt = Num**0.5
Paint ('The square root of %0.34 f %0.3f % (num,num_sort))
```

RESULT:

Thus the square root for the given number has been performed successfully.

EXERCISE NO:9	LOGICAL OPERATORS IN PYTHON
DATE :	

AIM:

To study and demonstrate the use of logical operators (and, or, not) in python

PROCEDURE:

Open Python (IDLE, Jupyter Notebook, or any IDE).

Declare boolean variables with values true or false.

Apply logical operators and, or, and not on the variables.

Display the results using the print () function

Execute the program and verify the output.

Program (code):

```
# Exercise: Logical operators in Python
# Declaring Boolean variables
 x= True
y= False
# Applying logical operators
Paint ("x="x," y=", Y)
Print ("x and y : ", x and x) #True only if both are true.
Print ("x or y: ", x or y) # True if at least one is true
Print ("not x:", not x) # Negates the value of x
print ("not y", not y) # Negates the value of y
```

Result:

The program was executed successfully Logical operations and, or, and not were demonstold With horcelean values, and their truth table behaviour was verified.

EXERCISE NO:10	SWAPPING OF TWO NUMBERS USING PYTHON
DATE :	

AIM:

To write a python program to swap two numbers using third variable.

PROCEDURE:

Step 1 :Get the values of x and y.

Step 2: Swap the values of two variables using a third variable called temp.

Step 3: Print the results.

SOURCE CODE:

```
x = 10

y = 50

# Swapping of two variables

# using third variable

Temp = x

X = Y

y = temp

Print ("Value of x:",x)

Print ("Value of Y:", Y)
```

Result:

Thus the two variable has been swapped using a third variable successfully.

EXERCISE NO:11	PYTHON PROGRAM TO FIND THE LARGEST AMONG THREE NUMBERS
DATE :	

AIM:

To write a python program to find the largest among three numbers.

Procedure

Get the value for num 1, num 2 and num 3.

Use the if elif and else statement to find largest number

Paint the result.

Source Code:

```
# Python program to find the largest number among the three input numbers.
# change the value of num1, num2 and num3.
# for a different result.
Num 1 = 10
num 2 = 14
num 3 = 12
if (num1>= num2) and (num1>= num3):
largest = num 1
elif (num2 > = num1) and (num 2>= num 3):
   largest = num 2
else:
   largest = num 3
Print ("The largest number is", largest)
```

Result

Thus, the program is executed successfully.

EXERCISE NO:12	CONTROL STRUCTURES IN PYTHON
DATE :	

AIM:

To study and implement control structures in python, including conditional statements (if, if-else, if - elif-else) and loops (for, while)

Procedure:

Open Python (IDLE, Jupyter Notebook, or any

Write a program to demonstrate:

» Decision making using if, if-else, and if-elif- else.

» Iteration using for loop and while loop.

Print appropriate outputs for each case

Execute the program and verify the results

Program (code):

```
# Exercise Control Structures in Python
# 1. Conditional Statements
num = 10
# if statement
if num >0:
Print ("Number is positive")
# if-else statement
if num % 2 == 0:
Print ("Number is even")
else
print ("Number is odd")
# if-elif-else statemen
If num <o:
print ("Number is Negative")
elif num == 0:
```

```
Print ("Number is Zero")
else:
Print ("Number is positive")
print ("---------------------")
#2. Looping Statements
# for loop
Print ("For loop: Printing numbers from 1 to 5")
for i in range (1,6):
print(i)
print ("-----------------")
# while loop
Print ("While loop: Printing numbers from 1 to 5")
i = 1
while I <= 5
Print (i)
i += 1
```

Result

The program was executed successfully Different control structures in Python (if, if-else, if - elif-else, for loop, and while loop) were demonstrated with correct outputs.

EXERCISE NO:13	PYTHON PROGRAM DISPLAY THE MULTIPLICATION TABLE
DATE :	

AIM:

To write a python to display the multiplication table

PROCEDURE:

Get the value as 12

Use the loop to display the multiplication table.

Paint the result.

SOURCE CODE

```
# Multiplication table (from 1 to 10) in python
num = 12
# To take input from the user
# num = int (input ("Display multiplication table of ?"))
# Iterate 10 times from i = 1 to 10 For i in aange (1,11):
Print (num, 'X', i`=', num +i)
```

Result:

Thus, the program is excecated successfully.

EXERCISE NO:14	FUNCTIONS IN PYTHON
DATE :	

AIM:

To Understand the concept of functions in Python and learn how to define and call them for code reusability.

PROCEDURE:

Open Anaconda Navigator and launch Jypyter Notebook (or any Python IDE).

Guate a new python file are notebook

Define a function using the def keyword.

> Example:

```
def greet (name):
Print ("Hello", name, "welcome to python Programming!")
```

Call the function with different arguments (values).

```
Greet ("Alice")
greet ("Boh")
```

Execute the program and observe the output.

Code Example

```
# Defining a function
def greet (name):
"" This function greets the person by name””
print ("Hello", name. "Welcome to Python Programming!")
# Calling the function with arguments
greet ("Alice")
greet ("Bah")
```

Result:

The concept of functions in Python was successfully implemented. We learned how to define a function, pass parameters, and call the function to achieve reusable and modular code.

EXERCISE NO:15	PYTHON PROGRAM TO MAKE A SIMPLE CALCULATOR
DATE :	

AIM

To write a python program to make a simple calculator.

PROCEDURE

Get instruction in string

Use the function arguments and user

defined function to make a simple Calculator

Paint the result

SOURCE CODE

```
# This function add two numbers
def add (x, y)
return x+y
# This function subtracts two numbers
def subtract (x,y).
return x-x
# This function multiplies two numbers
def multiply (x,y).
return x*y
# This function divides two numbers
def divide (X,Y):
return x/x
Print ("Select operation.")
print ("1.Add")
Print ("2. Subtract")
print ("3. Multiply")
print ("4. Divide")
While true.
# take input from the choice =  input ("Enter choice (1/2/3/4): ")
# check if choice is one of the four options
if choice in ('1', '2',3';' 4').
try
num 1 = float Cinput ("Enter first number!"))
num 2 = float (input ("Enter second number: "))
except value Error:
Print ("Invalid input. Please enter a number)
Continue
if choice == '1':
Print (num 1, "+", nuum 2,"=", add (num1,num 2)
elif choice =='2':
Paint (num1,"-", num 2, "=", subtract (num1, num2))
print (num1,”*”,num2,”*”, multiply (num1,num2)
elif choice == '4':
Print (num1,"/", num 2, "=", divide (num1, num2))
# check if user want another calculation
# break the while loop if answer is no
next- calculation = input ("Let's do next Calculation? (Yes/No ):")
if next_calculation == "no":
break
else:
  paint (invalid Input")
```

Result:

Thus, the program is executed successfully.

EXERCISE NO:16	STRING OPERATIONS IN PYTHON
DATE :	

AIM:

To study and implement various string Operations in Python such as concatenation, repetition, slicing and built-in string functions.

PROCEDURE:

1) Open Python in any IDE (eg., Jupyter Notebook, IDLE or VS Code)

ii) Create a Python file or notebook

Declare string variables and performance the following Operations

Concatenation (+)

Repetition (*)

Inducing and slicing ([])

Common functions (len (), upper (), lower (),strip (), replace (), split £), find (), etc.)

Execute the program and observe the output.

Code Example:

```
# Declaring string variables
str1 ="Hello"
str 2="World"
# Concatenation
result = str1+"”+str2
# Repetition
result 2 = str*3
# slicing
Result3=str2 [0.3]
# string Functions
Lenth=len(str1)
Upper_case=str2.lower()
Lower_case=str2.replace(“world”, “python”)
Split+str=”python programming”. Split()
# Displaying results
Print ("Concatenation: ", result 1)
Print (“Repetition: ", result 2)
Print ("Slicing", result 3)
Print ("Length of str 1:", length)
print ("Upper case:", upper_ case)
Print ("lower Case:", lower - case)
Paint ("Replaced string", replaced)
print ("Split string:", split-str)
```

Result

Various string operation in python such as Concatenation, repetition, slicing, and built-in string functions were successfully implemented.

EXERCISE NO:17	PYTHON PROGRAMM TO CHECK WHETHER A STRING PALINDROME OR NOT
DATE :	

AIM:

To write a python program to find the sum of python program

of natural numbers

PROCEDURE:

i) Get the instruction in string

Use the string method to check a string is palindrome or not

iii) Paint the result.

SOURCE CODE:

```
# Program to check if a string palindrome or not
My _ str =’aIbohPhoBIA'
# Make it suitable for comparison
My_str= str. case fold ()
# Reverse the string
rev_str = reversed (my_str)
# check if the string is equal to its reverse
        if list (my_str) == list (rev_str):
       Print ("The string is a palindrome.")
else:
      Print ("This string is not a palindrome")
```

RESULT

Thus, the program is executed successfully.

EXERCISE NO:18	PYTHON PROGRAM TO COMPUTE A THE POWER NUMBER
DATE :	

AIM:

To write a python program to computer a power of a number.

PROCEDURE:

get the value for base and exponent

ii) Use the while lamp the computer the power of a number.

Print the result.

SOURCE CODE:

```
base = 3
exponent = 4
result = 1
while exponent! -o:
   result = 1
    exponent =1
print (“ Answer =”+ str (result))
```

RESULT:

Thus the program is executed successfully.

EXERCISE NO:19	PYTHON PROGRAM TO COUNT THE NOMBER OF DIGITS PRESENT IN A NUMBER
DATE :	

AIM:

To write a python program to compute a power of a number.

PROCEDURE:

Get the value of 3452

use the while loop to count digits present In a number

iii) Paint the result

SOURCE CODE:

```
Num = 3452

Count = 0.

While num !=0

num // = 10

Count + =1

Print ("Number of digits: " + str(count))
```

RESULT:

Thus the program is executed successfully.

EXERCISE NO:20	INSTALLING & USING PACKAGES IN PYTHON
DATE :	

AIM:

To learn how to install and use external packages in Python for performing specific tasks.

PROCEDURE:

1) Open the command prompt (or terminal)

i) Use pip to install a package. Example

pip install numpy.

iv) Import the installed package using the impart statement

v) Use function from the package in your program.

PROGRAM:

```
#Step 1: Install numpy (done in terminal)
pip install numpy

# Step 2: Import and use numpy import numpy as np
# Create an array
arr = np. array ([1,2,3,4,5])
Print ("Array", arr)
Print ("Mean of Array: ", np.mean (arr))
```

RESULT:

We successfully installed the Numbly package using pip and used its functions in Python to Perform mathematical operation.

EXERCISE NO:21	FILE HANDLING IN PYTHON
DATE :	

EXERCISE NO: 21

AIM:

To study and implement basic file handling Operations such as creating, writing, reding and appending data in a file using Python

PROCEDURE:

1. Start Python (IDLE / Jupyter Notebook / VS Code).

2. Create or open a file using the built-in open() function with different modes:

o "w" -> Write mode (creates a new file or overwrites an existing one).

o "r"- Read mode (reads the content of a file).

o "a" -> Append mode (adds new data without deleting existing content).

3. Use file object methods like:

o write() -> To write content into the file.

o read() / readline() / readlines() - To read content from the file.

o close() -> To close the file after operations.

4. Save the program and run it.

5. Verify the output by checking the file contents in your system.

Program & Output:

```
# File Handling Example

# Step 1: Create and write to a file

file = open("example.txt", "w")

file.write("Hello, this is the first line.\n")

file.write("Python File Handling Example.\n")

file.close()

# Step 2: Append new data to the file

file = open("example.txt", "a")

file.wr t ("This line is appended to the file.\n")

file.close()
```

```
# Step 3: Read data from the file
file = open("example.txt", "r")
content = file.read()
print("File Content:\n", content)
file.close()
```

Result:

File handling in Python was successfully implemented. A text file was created, written with content, appended with new data, and read back using Python.

EXERCISE NO:22	LIST IN PYTHON
DATE :	

AIM:

To understand how to create, access, modify, and perform operations on list in Python.

Procedure:

1. Open Python IDE (Jupyter Notebook, PyCharm, VS Code, or IDLE).

2. Create a new Python file or notebook.

3. Define a list using square brackets [].

4. Perform different list operations:

o Access elements using index.

o Modify elements by assigning new values.

Use built-in functions like append(), insert(), remove(), pop(), sort(), reverse(), etc.

5. Print the results to observe how lists work.

6. Run the program.

Program (Example Code)

```
# Creating and working with lists in Python
# Creating a list
fruits = ["apple", "banana", "cherry"]
# Displaying the list
print("Original List:", fruits)
# Accessing elements
print("First fruit:", fruits[0])
print("Last fruit:", fruits[-1])
# Modifying elements
fruits[ ]= "mango"
print("After modification:", fruits)

# Adding elements
```

```
fruits.append("orange")
print("After appending:", fruits)
# Inserting at specific position
fruits.insert(1, "grapes")
print("After inserting grapes:", fruits)
# Removing elements
fruits.remove("cherry")
print("After removing cherry:", fruits)
```

Result

Thus, we have successfully created and performed different operations on lists in Python, including creation, modification, insertion, deletion, sorting, and reversing.

EXERCISE NO:23	TUPLE IN PYTHON
DATE :	

Aim:

To understand and implement tuples in Python, and learn their characteristics such as immutability, indexing, and usage.

Procedure:

```
# Creating a tuple
my_tuple =(10, 20, 30, 40,50)
# Display the tuple
print("Tuple Elements:", my_tuple)
print("First Element:", my_tuple[0])
print("Last Element:", my tuple [-1])
print("Slice from 1 to 3:", my_tuple[1:4])
try:
my_tuple[0]=100
except TypeError as e:
print("Error:", e)
# Tuple with mixed data types
mixed_tuple =(1,"Python", 3.14, True)
print("Mixed Tuple:", mixed_tuple)
# Nested tuple
nested_tuple =(1,2,(3,4,5))
# Nested tuple
nested_tuple=(1, 2, (3,4,5))
print("Nested Tuple:", nested_tuple)
```

Result:

Tuples in Python are immutable ordered collections that can hold elements of different data types.

EXERCISE NO:24	DICTIONARY IN PYTHON
DATE :	

Aim

To study and implement dictionaries in Python, and understand their characteristics such as key-value pairs, mutability, and common operations.

Procedure

1. Open your Python IDE (IDLE, Jupyter Notebook, PyCharm, or VS Code).

2. Create a new Python file named dictionary_example.py.

3. Write the following Python code to demonstrate dictionary creation, accessing, updating, deleting, and iterating through key-value pairs.

```
# Creating a dictionary
student = {
"name": "Rahul",
"age": 21,
"course": "BBA",
"marks": 88
}
# Display the dictionary
print("Student Dictionary:", student)
# Accessing values using keys
print("Name:", student["name"])
print("Age:", student.get("'age"))
# Adding a new key-value pair
student["semester"] = 5
print("After Adding Semester:", student)
# Updating a value

student["marks"]=92
print("After Updating Marks:", student)
```

```
# Deleting a key-value pair
del student["course"]
print("After Deleting Course:", student)
# Iterating through dictionary
print("\nIterating through dictionary:")
for key, value in student.items():
print(key, ":", value)
# Checking if a key exists
print("\nls 'age' present?", "age" in student)
print("Is 'course' present?", "course" in student)
```

4. Save and run the file.

5. Observe the output to analyze dictionary operations.

Result

Dictionaries in Python are mutable, unordered collections of key-value pairs. They allow efficient data retrieval using keys, support dynamic updates, and provide useful methods for adding, updating, deleting, and iterating over elements.

EXERCISE NO:25	**SETS IN PYTHON**
DATE :	

Aim:

To understand the concept of **sets in Python**, and perform basic set operations such as creation,

adding elements, removing elements, and performing union, intersection, and difference.

Procedure:

1. Open a Python environment (IDLE, Jupyter Notebook, or any IDE).
2. Create a set using curly braces {} or the set() function.
3. Add elements to the set using the add() method.
4. Remove elements using remove() or discard().
5. Perform common set operations such as **union (| or union()), intersection (& or intersection()), and difference (- or difference())**.
6. Print the results to verify the operations.

Program / Code:

```
fruits = {"apple", "banana", "cherry"}
print("Initial Set:", fruits)
fruits.add("orange")
print("After Adding Orange:", fruits)
# Removing an element
fruits.remove("banana")
print("After Removing Banana:", fruits)
set1 = {1, 2, 3, 4}
set2 = {3, 4, 5, 6}
print("Union:", set1 | set2)
print("Intersection:", set1 & set2)
print("Difference:", set1 - set2)
```

Result:

The concept of sets in Python was successfully implemented. We learned how to create sets,add/remove elements, and perform set operations like union, intersection, and difference.

EXERCISE NO:26	DATA SLICING IN PYTHON
DATE :	

Aim:

To understand and demonstrate data slicing in Python using the pandas Data Frame, which allows us to extract specific rows, columns, or subsets of data for analysis.

Procedure:

1. Import the pandas library.

2. Create a DataFrame with sample data (e.g., employee details).

3. Perform different slicing operations:

Select a single column.

Select multiple columns.

Slice rows by index.

Use loc for label-based slicing.

Use iloc for integer position-based slicing.

4. Display the sliced outputs to understand how slicing works.

Code

```
import pandas as pd
# Sample DataFrame
data = {
'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
'Age': [24, 27, 22, 32, 29],
'Department': ['HR', 'IT', 'Finance', 'Marketing', 'IT'],
'Salary': [40000, 50000, 45000, 60000, 52000]
Df = pd.DataFrame(data)
# 1. Selecting a single column
print("Single Column (Name):\n", df ['Name'],"\n")
# 2. Selecting multiple columns
print("Multiple Columns (Name & Salary):\n", df [['Name', 'Salary']],"\n")
# 3. Selecting rows by index (slicing rows)
```

```
print("Row Slicing (rows 1 to 3):\n", df [1:4],"\n")
# 4. Using loc for label-based slicing
Print ("loc Slicing (rows 1 to 3, Name & Department):\n", df.loc[1:3, ['Name', 'Department']],
"\n")
# 5. Using iloc for integer-based slicing
Print ("iloc Slicing (rows 0 to 2, columns 1 to 2):\n", df.iloc[0:3, 1:3],"n")
```

Result

We successfully performed data slicing operations in Python using pandas.

· Extracted single and multiple columns.

. Sliced rows using index ranges.

. Used loc (label-based) and iloc (integer-based) indexing to select specific rows and columns.

EXERCISE NO:27	IMPORTING CSV DATA IN PANDAS
DATE :	

Aim:

To learn how to import data from a CSV file using Pandas in Python.

Procedure:

1. Install Pandas (if not already installed):
2. pip install pandas
3. Import the pandas library in Python.
4. Use the read_csv() function to load data from a CSV file.
5. Display the data using print() or head().

Program:

```
import pandas as pd
# Import CSV file
data= pd.read_csv("D:/students.csv")
# Display first 5 rows
print(data.head())
```

Result:

We successfully imported data from a CSV file into a Pandas DataFrame and displayed it.

EXERCISE NO:28	IMPORTING EXCEL DATA IN PANDAS
DATE :	

Aim:

To learn how to import data from an Excel file using Pandas in Python.

Procedure

1. Install Pandas and openpyxl (Excel engine):

```
pip install pandas openpyxl
```

2. Import the pandas library.

3. Use the read_excel() function to load data from an Excel file.

4. Display the data using print() or head().

Program:

```
import pandas as pd
# Import Excel file
data= pd.read_excel("students.xlsx")
# Display first 5 rows
print(data.head())
```

Result:

We successfully imported data from an Excel file into a Pandas DataFrame and displayed it.

EXERCISE NO:29	INTRODUCING TO R PROGRAMMING
DATE :	

INTRODUCTION TOR

R is a and programming language and software environment used for statistical analysis, graphical representation, and reporting. It is open-source and available under the GNU General Public License. Compatible with windows, Linux, and Max systems Developed by robert Gentlemen and Ross Ihaka support integration with C, CH, Python, net and FORTRAN for enhanced performance

FEATURES OF R:

Simple, effective language with conditionals, lops and functions Strong data handling and storage facilities Operators fear arrays, lists, nectars and matrices tools and graphical Comprehensive data analysis tools and Capabilities

DATA TYPES:

Data types:

The variables are assigned with R-Objects and the data type of the R-object becomes the data type of the variable. There are many types of R-objects. The frequently used ones are -

· Vectors
. Lists
. Matrices
. Arrays
. Factors
· Data Frames

OPERATORS

Types of Operators

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. R language is rich in built-in operators and provides following types of

· Arithmetic Operators

. Relational Operators

· Logical Operators

. Assignment Operators

. Miscellaneous Operators

FUNCTIONS:

An R function is created by using the keyword function.

EXERCISE NO:30	HOW TO INSTALL R PROGRAMMING
DATE :	

Aim:

To learn to install R programming

Procedure: To install R and RStudio on windows, go through the following steps: Install R on windows

Step – 1: Go to CRAN R project website.

Step – 2: Click on the Download R for Windows link.

Step – 3: Click on the base subdirectory link or install R for the first time link.

Step – 4: Click Download R X.X.X for Windows (X.X.X stand for the latest version of R. eg: 3.6.1) and save the executable .exe file.

Step – 5: Run the .exe file and follow the installation instructions.

Select the desired language and then click Next.

Read the license agreement and click Next.

Select the components you wish to install (it is recommended to install all the components). Click Next.

Enter/browse the folder/path you wish to install R into and then confirm by clicking Next.

Select additional tasks like creating desktop shortcuts etc. then click Next.

Wait for the installation process to complete.

Click on Finish to complete the installation.

Install RStudio on Windows

Step – 1: With R-base installed, let's move on to installing RStudio. To begin, go to download RStudio and click on the download button for **RStudio desktop**.
Step – 2: Click on the link for the windows version of RStudio and save the .exe file.
Step – 3: Run the .exe and follow the installation instructions.
3.a. Click **Next** on the welcome window.
3.b. Enter/browse the path to the installation folder and click **Next** to proceed.
3.c. Select the folder for the start menu shortcut or click on do not create shortcuts and then click
Next.

3.d. Wait for the installation process to complete.
3.e. Click **Finish** to end the installation.

EXERCISE NO:31	UNDERSTANDING BASIC COMMANDS IN R
DATE :	

AIM:

To understand the basic commands in R programming.

PROCEDURE:

R as a calculator

```
> 1 + 2
[1] 3
```

Variable Assignment:

Values can be assigned using ‘=’ or ‘<-‘

Eg: ‘x=1’

Functions

Function are called by name followed by parentheses eg: ‘c (1, 2, 3)’ combines number into a water.

```
[1] 1 2 3
```

Comments

Use "#" for comment example : ‘1+1# comment’.

R Data Type

Numeric
Integer
Complex
Logical
Character

Vectors

Created using ‘c()’
Can be numerical, logical or characts arithmetic is done.

Lists

It can stare different types of elements
Access with ‘x [[2]] ‘ for direct member reference.

Data frames

Stare taluclar data (like excel)
Built – in dataset

INFERENCE

R is a versatile tool for statistical analysis, data manipulation and visualization. It supports Various data types & structures like vectors, lists, and data frames for effective Computation.

EXERCISE NO:32	IMPORTING & EXPORTING DATA IN R
DATE :	

AIM:

To importing and exporting the data using R Programming

PROCEDURE:

The sample data can also be in comma separated values (CSV) format. Each cell inside such data file is separated by a special character, which usually is a comma, although other characters can be used as well.

The first row of the data file should contain the column names instead of the actual data. Here is a sample of the expected format.

```
Col1, Col2, Col3
100, a1, b1
200, a2, b2
300, a3, b3
```

After we copy and paste the data above in a file named "mydata.csv" with a text editor, we can read the data with the function read.csv.

```
> mydata = read.csv("mydata.csv") # read csv file
> mydata
  Col1 Col2 Col3
 1 100 a1 b1
 2 200 a2 b2
 3 300 a3 b3
```

```
# Write CSV in R
>write.csv (My Data, file = "MyData.csv",row.names=FALSE)
```

RESULT:

Thus importing and exporting the data using R Programming was executed successfully

EXERCISE NO:33	EXPLORING THE DATA USING R STUDIO
DATE :	

AIM:

To write a R program to explore the data using R studio.

PROCEDURE:

Step 1: Create a sample dataset.
Step 2: Install and load the ggplot2 package for data visualization.
Step 3: Create a histogram of ages.
Step 4: Create a box plot of income.

SOURCE CODE:

```
# Create a sample dataset data
    Data <- data.frame
    ID = 1:10,
    Age = c(25, 30, 22, 40, 35, 28, 55, 29, 37, 45),
   Income = c(40000, 50000, 32000, 60000, 55000, 45000, 75000, 52000, 68000, 80000),
   Education = c("High School", "Bachelor's", "High School", "Master's", "PhD",
"Bachelor's",  "PhD", "Master's", "PhD", "Bachelor's"),
   Gender = c("Male", "Female", "Male", "Female", "Male", "Male", "Female", "Female",
"Male", "Male")
)
head(data)
summary(data$Age)
summary(data$Income)
table(data$Education)
table(data$Gender)
# Install and load the ggplot2 package for data visualization install.packages("ggplot2")
library(ggplot2)
# Create a histogram of ages
ggplot(data, aes(x = Age)) + geom_histogram(binwidth = 5, fill = "blue") + labs(title = "Age
Distribution")
# Create a box plot of income
ggplot(data, aes(x = "", y = Income)) + geom_boxplot(fill = "green") + labs(title = "Income
Distribution")
```

RESULT:

Thus the data has been explored in various ways using R studio successfully.

EXERCISE NO:34	EXPLORING THE DATA USING R STUDIO
DATE :	

AIM:

To creating a Pie Chart in R Programming using Command

PROCEDURE:

A pie chart of a qualitative data sample consists of pizza wedges that show the frequency distribution graphically.

The following script creates a pie chart.

1. This starts a pie chart function. The “x” parameter is the data that needs to be charted. In this line, the feed variable in the dick cuts data frame is extracted to a table since the pie chart

2.These lines define the main title and colors used for the pie chart. These parameters are the same as was seen in other graphs in this lab.

3. This tells R to use the labels used in the feed variable as the labels on the pie chart. # Count of Chicks by Feed pie(x =table(chick uts $ feed),
main = "Count of Chicks by Feed", col = rainbow(6),
labels = c(levels(chick wts $ feed))
)

RESULT :

Thus Pie Chart in R Programming was executed successfully .

EXERCISE NO:35	CREATING HISTOGRAM IN R
DATE :	

AIM:

To Create a Histogram in R Programming Command.

PROCEDURE:

A histogram is a graph that shows the distribution of data that are continuous in nature, for example, age or height. A histogram resembles a bar chart but there is an important difference: a histogram is used for continuous data while a bar chart is used for categorical data. To emphasize that difference, histograms are normally drawn with no space between the bars (the data are continuous along the entire x-axis) while bar charts are normally drawn with a small space between bars (the data are categorical along the x-axis).

```
# Histogram

hist(morley$Speed,

main = "Morley's Experiment", xlab = "Speed",

ylab = "Frequency", breaks
= 8,
col = cm.colors(10)

)
```

Result:

Thus histogram was created successfully using R

EXERCISE NO:36	CREATING BARPLOT IN R
DATE :	

AIM:

Creating a Bar Plot using R Programming Commands.

PROCEDURE:

A bar graph of a qualitative data sample consists of vertical parallel bars that show the frequency distribution graphically. A bar plot is used to display the frequency count for categorical data.
The following figure is a bar plot showing the number of automobiles with three, four, and five gears according to the mtcars data frame.

SOURCE CODE:

```
      # Clustered Bar Plot With Gradient Colors colpal <

colorRampPalette(c("blue", "white")) barplot(height =

table(mtcars$cyl, mtcars$gear), main = "Cars by Gears and Cylinders",

xlab

=" gears", ylab =

" count",   legend =

  TRUE,        beside  = TRUE,

  args.legend = list(title = "Cylinders"), col = colpal(3)

)
```

RESULT

Thus Bar Plot using R Programming Commands was created successfully

EXERCISE NO:37 DATE :	CREATING BARPLOT IN R

AIM:

To give a demonstration about handling ggplot package in R studio.

PROCEDURE:

Step 1: Load the ggplot2 library
Step 2: Supply sample data.
Step 3: Create a scatter plot Step 4: Print the results.

SOURCE CODE :

```
# Load the ggplot2 library library(ggplot2) # Sample data data <- data.frame
( X = c(1, 2, 3, 4, 5),
Y = c(2, 3, 1, 4, 2)
)
# Create a scatter plot ggplot(data, aes(x = X, y = Y)) + geom_point() +
labs(title = "Scatter Plot Example", x = "X-axis", y = "Y-axis")
```

RESULT

Thus the ggplot package has been handled and demonstrated successfully

EXERCISE NO:38	CREATING FREQUENCY DISTRIBUTION IN R
DATE :	

AIM:

To calculate the Frequency distribution in R programming.

PROCEDURE:

A data sample is called qualitative, also known as categorical, if its values belong to a collection of known defined non-overlapping classes. Common examples include student letter grade (A, B, C, D or F), commercial bond rating (AAA, AAB,) and consumer clothing shoe sizes (1, 2, 3,).

The tutorials in this section are based on an R built-in data frame named painters. It is a compilation of technical information of a few eighteenth century classical painters. The data set belongs to the MASS package, and has to be pre-loaded into the R workspace prior to its use.

The last School column contains the information of school classification of the painters. The schools are named as A, B, ...,*etc*, and the School variable is

For further details of the paintersdata set, please consult the R documentation.

```
help(painters)
```

The frequency distribution of a data variable is a summary of the data occurrence in a collection of non-overlapping categories.

Result :

Thus frequency distribution in R programming was conducted successfully.

EXERCISE NO:39	DESCRIPTIVE STATISTICS USING R
DATE :	

AIM:

To create a Descriptive Statistics using R Programming Commands.

PROCEDURE:

```
>require("datasets") # Load Dataset Package
>data(cars)
>summary (cars $ speed) # Summary for one variable Min.
```

```
Min. : 4.0  Min.  : 2.00 1st
 Qu.:12.0 1st Qu.: 26.00
Median :15.0  Median : 36.00 Mean
 :15.4 Mean :
 42.98 3rd
 Qu.:19.0 3rd
 Qu.: 56.00 Max.
 :25.0  Max.  :120.00
```

Result :

Thus Descriptive Statistics using R Programming Commands was executed successfully

EXERCISE NO:40	DESCRIPTIVE STATISTICS USING R
DATE :	

AIM:

To analyze and interpret correlation using Pearson’s r, Spearman’s Rho and Kendall’s Tau Measures.

PROCEDURE:

Correlation is a method used to describe a relationship between the independent (or x-axis) and dependent (or y-axis) variables in some research project. A correlation is a number between -1.0 and +1.0, where 0.0 means there is no correlation between the two variables and either +1.0 or -

1.0 means there is a perfect correlation. A positive correlation means that as one variable increases the other also increases.

```
Spearman's rank correlation rho
      data: as.numeric(esoph$agegp) and esoph$ncases S = 57515, p-value = 1.029e- 06
alternative hypothesis: true rho is not equal to 0 sample estimates:
rho 0.49354 37
```

Result :

Thus correlation was analyzed and interpreted using Pearson’s r, Spearman’s Rho and Kendall’s Tau