Exploratory Data Analysis Using Python - Lab Manual

Prepared By : Vasudevan T V

List of Programs

0 - Get the Largest Number from a List

Write a python program to get the largest number from a list.

```
Program - Version 1
-------------------
list=[10,20,30]
print("List:", list)
print("The largest element is:", max(list))

Program - Version 2
-------------------
list1=[]
while True :
  print("Enter an integer to the list:")
  element = int(input())
  list1.append(element)
  print("Is it end of list?(y/n)")
  flag = input()
  if flag == "y" :
    break
print("The List =", list1)
print("Maximum Value in this list:", max(list1))
```

0 - Remove Duplicates from a List

Write a python program to remove duplicates from a list.

```
Program - Version 1
-------------------
list1=[20,20,10,30,30]
print("Duplicated List:", list1)
list2=[]
for i in list1:
    if i not in list2:
        list2.append(i)
print("Deduplicated List:", list2)

Program - Version 2
-------------------
list1 = [9,9,5,6,3,4,1,9,2,4,2]
```

```
print("Original List:", list1)
list2 = list(set(list1))
print("Duplicates Removed List : ", list2)
```

0 - Conversion of Tuple to Dictionary

Write a python program to convert a tuple to a dictionary.

```
Program
-------
tuple = (('Rajesh', 75),('Vasudevan', 80), ('Thomas', 85))
print("Tuple=",tuple)
dictionary = dict(tuple)
print("Dictionary=",dictionary)
```

0 - Merging Dictionaries

Write a python program to merge two python dictionaries.
Hint - Use update() method or ** operator or | operator

```
Program - Version 1
-------------------
dict1 = {'a': 1, 'b': 2}
print("Dictionary 1=",dict1)
dict2 = {'c': 3, 'd': 4}
print("Dictionary 2=",dict2)

# Merging dict2 into dict1
dict1.update(dict2)

print(dict1)
```

```
Program - Version 2
-------------------
dict1 = {'a': 1, 'b': 2}
print("Dictionary 1=",dict1)
dict2 = {'c': 3, 'd': 4}
print("Dictionary 2=",dict2)
```

```
# Merging dictionaries
merged_dictionary = {**dict1, **dict2}

print("Merged Dictionary=", merged_dictionary)

Program - Version 3
-------------------
dict1 = {'a': 1, 'b': 2}
print("Dictionary 1=",dict1)
dict2 = {'c': 3, 'd': 4}
print("Dictionary 2=",dict2)

# Merging dictionaries
merged_dictionary = dict1|dict2

print("Merged Dictionary=", merged_dictionary)
```

0 - Checking Common Member in Lists

Write a python program that takes two lists and returns true if they have at least one common member.

```
Program
-------
list1=[10,20,30]
# list is converted to set to make the elements unique
# and to perform set operations
set1=set(list1)
list2=[40,50,60]
set2=set(list2)
result=set1 & set2
# If there is no common element, then result = set()
if (result== set()):
    print("False")
else :
    print("True")
```

0 - Find Earlier Date

Write a python program to determine which one is the earlier date from the two given dates.

```
Program
-------
date1="2025-01-26"
date2="2025-08-15"
if date1<date2 :
    print("Earlier date is", date1)
elif date1>date2 :
    print("Earlier date is", date2)
else:
    print("Two dates are equal")
```

0 - Subtract 5 Days from Current Date

Write a python program to subtract 5 days from current date.

```
Program
-------
import datetime
today=datetime.date.today()
no_of_days=datetime.timedelta(days=5)
print(today-no_of_days)
```

0 - File Copy Operation

Write a Python program to open a file and copy the contents to another file.

```
Program
-------
source_file=open("file1.txt","r")
contents=source_file.read()
source_file.close()
destination_file=open("file2.txt","w")
destination_file.write(contents)
destination_file.close()
```

0 - Capitalise Each Word in a File

Write a python program to capitalise each word in a file.
Hint - Use str.upper() method

```
Program
-------
file=open("file1.txt","r")
contents=file.read()
new_contents=contents.upper()
file.close()
file=open("file1.txt","w")
file.write(new_contents)
file.close()
```

0 - Search and Replace in a File

Write a Python program to search for a word in a file and replace it with another word.
Hint - Use str.replace("old text","new text") method

```
Program
-------
file=open("file1.txt","r")
contents=file.read()
new_contents=contents.replace("Python","C")
file.close()
file=open("file1.txt","w")
file.write(new_contents)
file.close()
```

0 - Count Number of Lines in a File

Write a python program to count number of lines in a file.

```
Program
-------
count = 0
file = open("file1.txt", "r")
for line in file:
 count = count + 1
print("Number of Lines:", count)
```

0 - Retrieve Lines Having Two Consecutive 1's

Write a python program to retrieve lines having two consecutive 1's.

```
Program
-------
lines_with_consecutive_ones = []
file = open("file1.txt", "r")
for line in file:
    if '11' in line:  # Check for two consecutive '1's
        lines_with_consecutive_ones.append(line.strip()) # remove leading
                                                         # and trailing
                                                         # white space
print(lines_with_consecutive_ones)
print("Lines containing two consecutive '1's:")
for line in lines_with_consecutive_ones:
    print(line)
```

0 - Matrix Operations Using Vectorisation

Write python programs to perform the following matrix operations using vectorisation. (Use matrices of order 3 X 3)

1. addition
2. subtraction
3. multiplication
4. scalar multiplication
5. transpose

```
Program
-------
import numpy
matrix1=numpy.matrix([[1,2,3],[1,1,1],[1,1,1]])
matrix2=numpy.matrix([[1,0,0],[0,1,0],[0,0,1]])
print("Matrix 1=\n",matrix1)
print("Matrix 2=\n",matrix2)
```

```
matrix3 = numpy.add(matrix1,matrix2)
matrix4 = numpy.subtract(matrix1,matrix2)
matrix5 = numpy.matmul(matrix1,matrix2)
matrix6 = numpy.transpose(matrix1)
print("Matrix 1 + Matrix 2=\n", matrix3)
print("Matrix 1 - Matrix 2=\n", matrix4)
print("Matrix 1 * Matrix 2=\n", matrix5)
print("2 * Matrix 1=\n", 2*matrix1)
print("Transpose of Matrix 1=\n", matrix6)
```

1 - Introduction to Python Libraries for Data Analysis(Pandas,NumPy,Matplotlib)

You can use the Titanic dataset for practice:

https://raw.githubusercontent.com/datasciencedojo/datasets /master/titanic.csv

This contains details of passengers on the British ship Titanic that sank on 15 April,1912.

PassengerID: A unique identifier for each passenger.
Survived: Survival Status (1 = Survived, 0 = Not Survived)
Pclass: Ticket class (1 = Upper, 2 = Middle, 3 = Lower).
Name: Name of the passenger.
Sex: Gender of the passenger.
Age: Age of the passenger.
SibSp: Number of siblings or spouses aboard.
Parch: Number of parents or children aboard.
Ticket: Ticket number.
Fare: Passenger fare.
Cabin: Cabin number (if available).
Embarked: Port of embarkation (C = Cherbourg, Q = Queenstown, S = Southampton).

1. Create a NumPy array with values from 1 to 50.

   Find its: Mean, Maximum, Minimum, Standard deviation

2. Load the Titanic dataset using NumPy Find: Number of passengers, Number of survivors, Average age of passengers

3. Using Matplotlib:

   (i) Plot a histogram of the Fare column in Titanic dataset.

(ii) Plot a bar chart showing the number of passengers by Pclass.

(iii) Plot a line chart showing numbers from 1 to 10 and their squares.

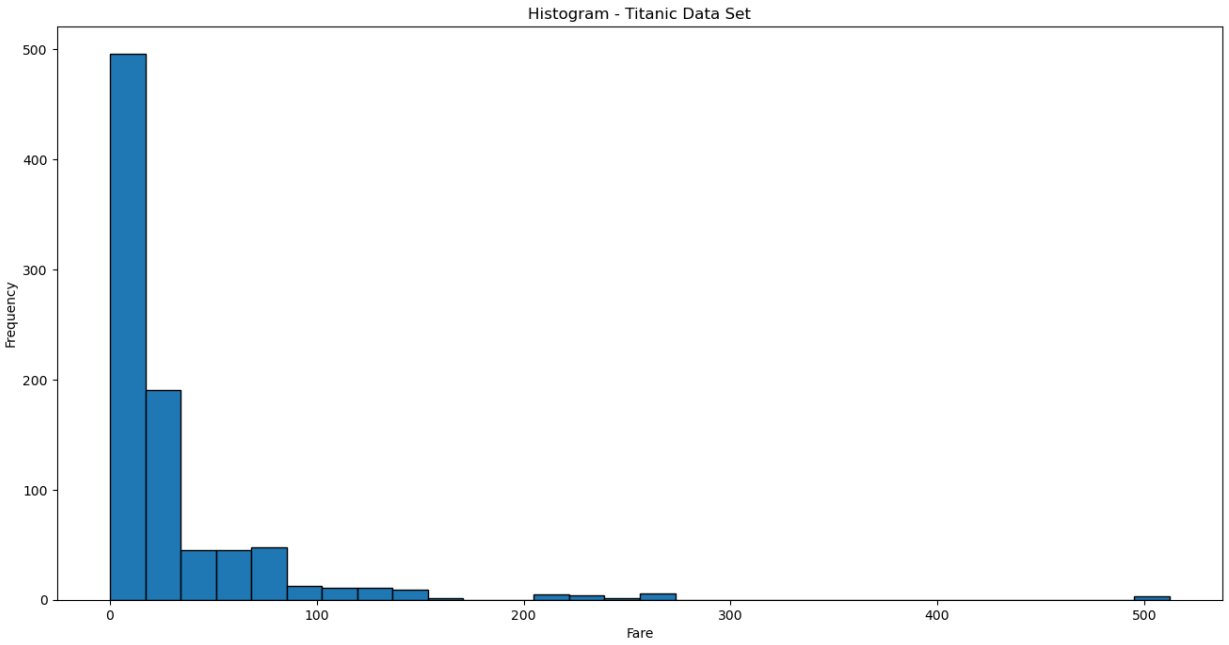
```
Program 1
---------
# Imports the numpy module, used for numeric computing
import numpy

# Create a NumPy array with values from 1 to 50
# Generates an array with values starting from 1 up to,but not including,51
array = numpy.arange(1, 51)

# Calculate the mean
mean_value = numpy.mean(array)

# Calculate the maximum
max_value = numpy.max(array)

# Calculate the minimum
min_value = numpy.min(array)

# Calculate the standard deviation
std_deviation = numpy.std(array)

# Output the results
print("Array:", array)
print("Mean:", mean_value)
print("Maximum:",max_value)
print("Minimum:",min_value)
print("Standard Deviation:","%.2f" % std_deviation)

Output
------
Mean: 25.5
Maximum: 50
Minimum: 1
Standard Deviation: 14.43

Program 2
---------
# Imports the numpy module, used for numeric computing
import numpy
```

```
# Imports the pandas module, used for data manipulation and analysis
import pandas

# Load Titanic dataset
titanic_data = pandas.read_csv('titanic.csv')

# Converting DataFrame to NumPy array for analysis
data_array = titanic_data.to_numpy()

# data_array.shape[0] returns the number of rows in a numpy array
# data_array.shape[1] returns the number of columns in a numpy array
# data_array.shape returns the number of rows and columns in a numpy array
# Number of passengers
number_of_passengers = data_array.shape[0]

# Number of survivors (assuming 'Survived' column indicates 0 for No
# and 1 for Yes)
# Assuming 'Survived' is in the second column
number_of_survivors = numpy.sum(data_array[:, 1])

# Retrieve the passenger ages from sixth column of the numpy array
ages = data_array[:, 5]

# Convert ages to float
ages = ages.astype(float)

# Calculate the average age while ignoring NaN values
average_age = numpy.nanmean(ages)

# Output results
print("Number of passengers:",number_of_passengers)
print("Number of survivors:", number_of_survivors)
# Prints with precision 2 after decimal point
print("Average age of passengers:","%.2f" % average_age)

Output
------
Number of passengers: 891
Number of survivors: 342
Average age of passengers: 29.70
```

Program 3(i)

```
# import pandas module used for data manipulation and analysis
import pandas
# imports pyplot, a module used in the package matplotlib to plot various
# figures
from matplotlib import pyplot
# load the Titanic dataset
titanic_data = pandas.read_csv('titanic.csv')
# plots the histogram of fare attribute
# edge colour is set as black
# number of bins is set as 30 ( By default bins = 10 )
titanic_data['Fare'].plot(kind='hist', edgecolor="black", bins=30)
# sets the title
pyplot.title("Histogram - Titanic Data Set")
# sets the x-axis label
pyplot.xlabel('Fare')
# shows the histogram
pyplot.show()
```

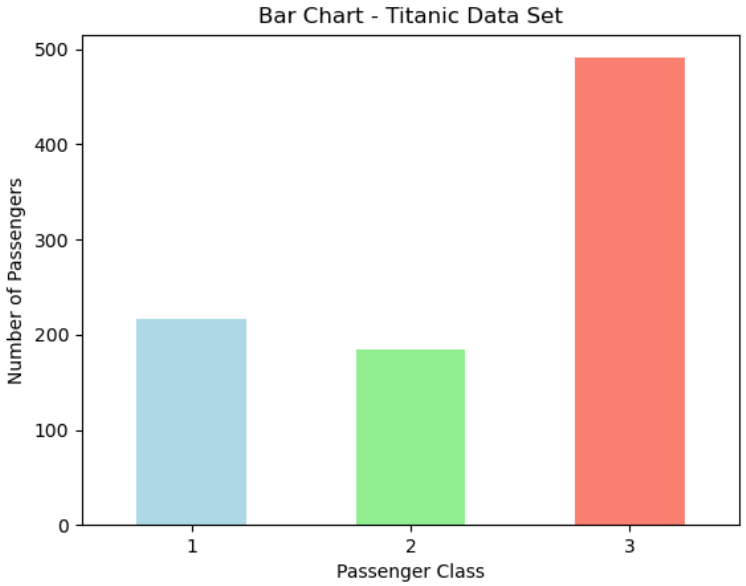
Output



Program 3(ii)

```
# import pandas module used for data manipulation and analysis
import pandas
# imports pyplot, a module used in the package matplotlib to plot various
# figures
from matplotlib import pyplot
# Load the Titanic dataset
titanic_data = pandas.read_csv('titanic.csv')
# Count the number of passengers by Pclass sorted by Pclass order
pclass_counts = titanic_data['Pclass'].value_counts().reindex([1,2,3])
# plots the bar chart with different colours for each Pclass
pclass_counts.plot(kind='bar', color=['lightblue', 'lightgreen', 'salmon'])
# sets the title
pyplot.title('Bar Chart - Titanic Data Set')
# sets the x-axis label
pyplot.xlabel('Passenger Class')
# sets the y-axis label
pyplot.ylabel('Number of Passengers')
# displays x-axis tick labels horizontally( default behaviour is vertical )
pyplot.xticks(rotation='horizontal')
# shows the bar chart
pyplot.show()
```

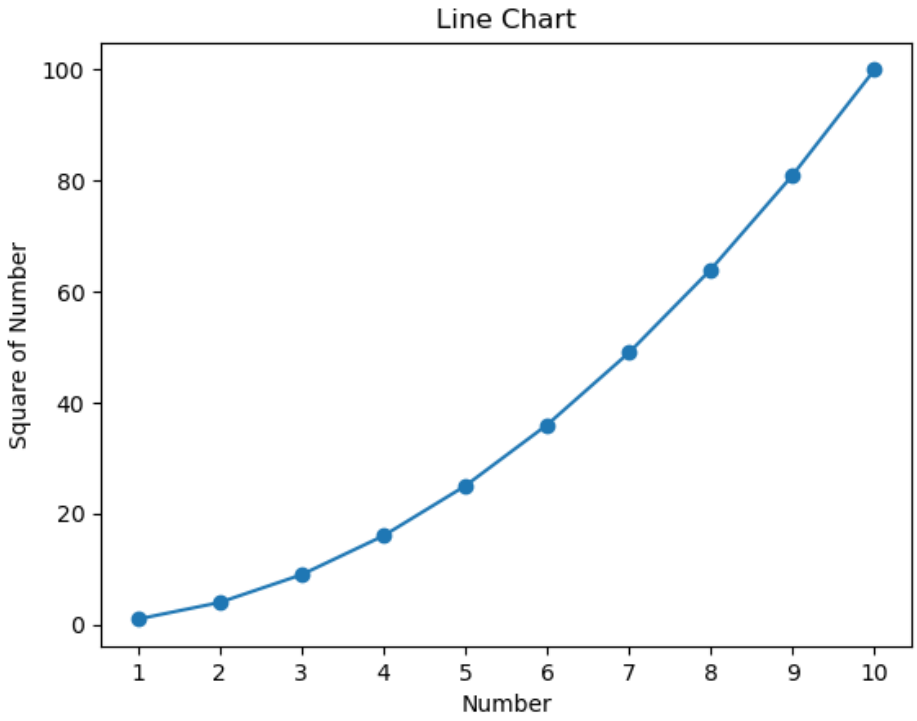
Output



Program 3(iii)

```
# imports pyplot, a module used in the package matplotlib to plot various
# figures
from matplotlib import pyplot
# Data: numbers from 1 to 10 stored in a list x
x = list(range(1, 11))
# Calculate their squares stored in a list y
y = [i ** 2 for i in x]
# plots the line chart, adding circular markers at each data point
pyplot.plot(x, y,marker='o')
# Adding title and labels
pyplot.title('Line Chart')
pyplot.xlabel('Number')
pyplot.ylabel('Square of Number')
# Set x-ticks to the numbers
pyplot.xticks(x)
# Show the plot
pyplot.show()
```

Output



2 - Data Importing and Basic Exploration Using Pandas

Load the Titanic dataset.

1. Print (i) Number of rows and columns (ii) Column data types (iii) Number of missing values in each column (iv) Basic descriptive statistics
2. Find (i) The number of passengers in each class (ii) The average age of passengers (iii) The number of survivors

```
Program 1
---------
# Imports the pandas module, used for data manipulation and analysis
import pandas

# Load Titanic dataset into a data frame
titanic_data = pandas.read_csv('titanic.csv')

# titanic_data.shape returns the number of rows and columns in the data
# frame
number_of_rows, number_of_columns = titanic_data.shape

# Print number of rows and columns
print("Titanic data set")
print("----------------")
print("Number of rows:",number_of_rows)
print("Number of columns:", number_of_columns)

# Print column data types
print("Column data types")
print("-----------------")
print(titanic_data.dtypes)

# Calculate and print the number of missing values in each column
# titanic_data.isnull()creates a DataFrame of the same shape as
# titanic_data, but with True for missing values and False for non-missing
# values
# .sum() counts the number of True values (i.e., missing values) for each
# column
missing_values = titanic_data.isnull().sum()
print("Number of missing values")
print("------------------------")
```

```
print(missing_values)

# Print basic descriptive statistics
print("Basic Descriptive Statistics")
print("----------------------------")
descriptive_stats = titanic_data.describe()
print(descriptive_stats)
```

Output

```
Titanic data set
----------------
Number of rows: 891
Number of columns: 12
Column data types
-----------------
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
Number of missing values
------------------------
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
Basic Descriptive Statistics
------------------------
       PassengerId    Survived      Pclass  ...       SibSp       Parch        Fare
count   891.000000  891.000000  891.000000  ...  891.000000  891.000000  891.000000
mean    446.000000    0.383838    2.308642  ...    0.523008    0.381594   32.204208
std     257.353842    0.486592    0.836071  ...    1.102743    0.806057   49.693429
min       1.000000    0.000000    1.000000  ...    0.000000    0.000000    0.000000
25%     223.500000    0.000000    2.000000  ...    0.000000    0.000000    7.910400
50%     446.000000    0.000000    3.000000  ...    0.000000    0.000000   14.454200
75%     668.500000    1.000000    3.000000  ...    1.000000    0.000000   31.000000
max     891.000000    1.000000    3.000000  ...    8.000000    6.000000  512.329200
```