ANNAMALAI UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

B. E. COMPUTER SCIENCE & ENGINEERING (DATA SCIENCE)

SEMESTER – IV

19DSCP409. DATA SCIENCE LAB

LABORATORY MANUAL

(JANUARY 2021 – APRIL 2021)

LAB INCHARGE:

Dr. AN. SIGAPPI, Professor, Dept. of CSE, A.U

ANNAMALAI UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

19DSCP 409. DATA SCIENCE LAB (PRACTICAL)

COURSE TEACHER: Dr. AN. SIGAPPI, Professor, Dept. of CSE, AU

LIST OF EXPERIMENTS

CYCLE - I

1. STUDY OF PYTHON DATA SCIENCE ENVIRONMENT
2. OPERATIONS ON PYTHON DATA STRUCTURES
3. ARRAY OPERATIONS USING NUMPY
4. OPERATIONS ON PANDAS DATAFRAME
5. DATA CLEANING AND PROCESSING IN CSV FILES
6. HANDLING CSV FILES
7. HANDLING HTML AND EXCEL FILES

CYCLE - II

8. PROCESSING TEXT FILES
9. DATA WRANGLING (PIVOT TABLE, MELT, CONCAT)
10. GENERATING LINE CHART AND BAR GRAPH USING MATPLOTLIB
11. DISPLAY DATA IN GEOGRAPHICAL MAP
12. DISPLAY DATA IN HEATMAP
13. NORMAL AND CUMULATIVE DISTRIBUTION
14. HYPOTHESIS TESTING

ADDITIONAL EXERCISES

1. GENERATION OF FACTOR PAIRS OF A GIVEN INTEGER
2. AVERAGE POOLING ON A GIVEN *n x n* MATRIX WITH A *m x m* KERNEL

Ex. No. 1

STUDY OF PYTHON DATA SCIENCE ENVIRONMENT

AIM:

To study the Python Data Science Environment (NumPy, SciPy, Pandas, Matplotlib).

PROBLEM DEFINITION:

Study the features of Python, packages required for data science operations and their installation procedure required for Data Science programming.

a) PYTHON DATA SCIENCE ENVIRONMENT

Data Science is a branch of computer science that deals with how to store, use and analyze data for deriving information from it. Analyzing the data involves examining it in ways that reveal the relationships, patterns, trends, etc. that can be found within it. The applications of data science range from Internet search to recommendation systems to customer services and Stock market analysis. The data science application development pipeline has the following elements: Obtain the data, wrangle the data, explore the data, model the data and generate the report. Each element requires skills and expertise in several domains such as statistics, machine learning, and programming. Data Science projects require a knowledge of the following software:

PYTHON: Python is a high-level, interpreted, interactive and object-oriented scripting language that provides very high-level dynamic data types and supports dynamic type checking. It is most suited for developing data science projects.

NUMPY: NumPy provides n-dimensional array object and several mathematical functions which can be used in numeric computations.

SCIPY: SciPy is a collection of scientific computing functions and provides advanced linear algebra routines, mathematical function optimization, signal processing, special mathematical functions, and statistical distributions.

PANDAS: Pandas is used for data analysis and can take multi-dimensional arrays as input and produce charts/graphs. Pandas can also take a table with columns of different datatypes and may input data from various data files and database like SQL, Excel, CSV.

MATPLOTLIB: Matplotlib is scientific plotting library used for data visualization by plotting line charts, bar graphs, scatter plots.

b) INSTALLATION OF PYTHON AND DATA SCIENCE PACKAGES

The following documentation includes setting up the environment and executing programming exercises targeted for users using Windows 10 with Python 3.7 or later version. Steps should work on most machines running Windows 7 or 8 as well.

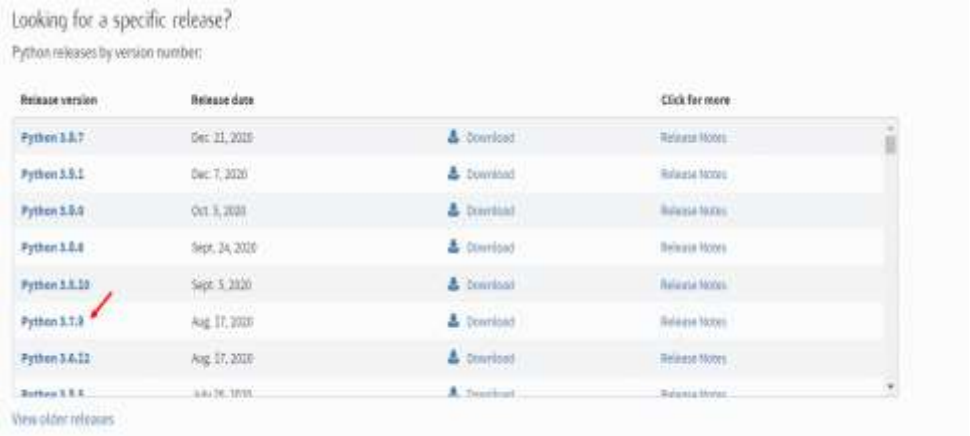
Sections that are indicated as optional are marked with **[Optional]**. Though optional, students are strongly encouraged to try out these sections.

We use the default python package management system - pip to install packages through one may prefer to install using conda.

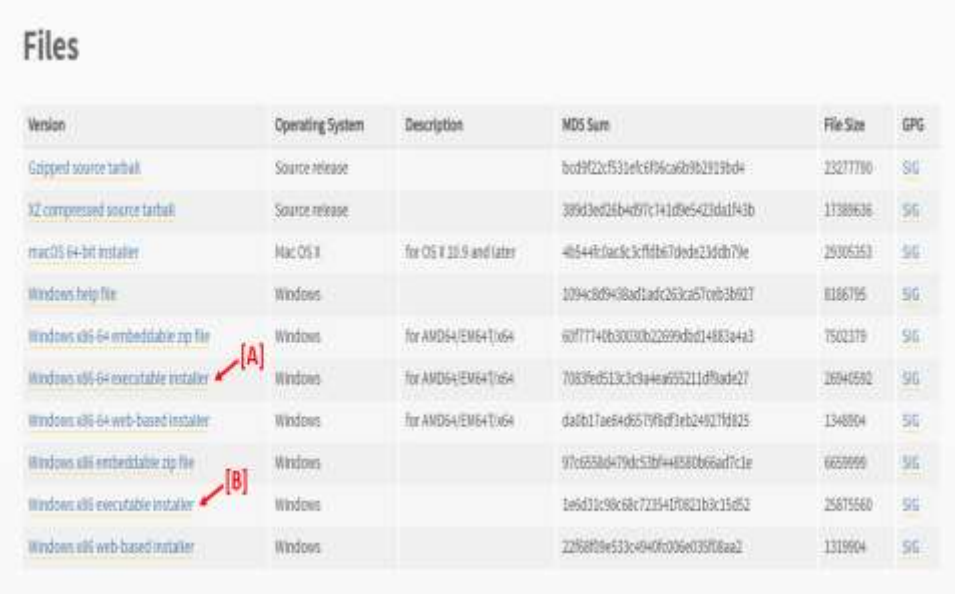
Setting up Environment:

Python:

1. To install Python 3 on Windows, navigate to https://www.python.org/downloads/ on your web browser, download and install the desired version.
2. For example to install Python 3.7.9:
    1. Navigate to https://www.python.org/downloads/
    2. Scroll down to “Looking for a specific release?” section and click on Python 3.7.9 as shown below:



    c. Scroll down to “Files” section and click on “Windows x86-64 executable installer” (Indicated [A]) if running a 32 bit machine or “Windows x86 executable installer” (indicated [B]) if running a 64 bit machine. If not sure if your machine is 32 or 64 bit, we recommend installing the 32 bit version.



    d. Double click the downloaded exe to run the installer. Follow the prompts on the screen and install with default options.

3. To verify installation, go to Start->Command Prompt. Type in “python --version” and hit Enter key. This will display “Python 3.7.9” or similar in the next line. If you do not see this or see any other error, please revisit the above steps.
4. Advanced Windows users or users facing issues can refer to https://docs.python.org/3/using/windows.html
5. To install Python on other distributions refer to:
a. Macintosh OS: https://docs.python.org/3/using/mac.html
b. Unix distros: https://docs.python.org/3/using/unix.html

Additional Resource:
https://docs.python.org/3/installing/index.html#basic-usage

pip

Python installation comes with a default package management/install system (pip - “pip installs Package”). Make sure to verify this by:

1. Start->Command Prompt.
2. Type in “pip --version” and hit Enter key.
3. This will display “pip 20.0.2 from “c:\users\DELL\appdata\local\programs\python\python37\lib\site-packages\pip (python 3.7)” or similar in the next line.

Virtual Environment (venv) [Optional]

Follows steps from here to install/use virtual environment:
https://docs.python.org/3/tutorial/venv.html#creating-virtual-environments

Jupyter Notebook [Optional]

Jupyter Notebook is a web based interactive development environment, usually preferred for quick prototyping.

To install:

1. Start->Command Prompt.
2. Type in “pip install jupyter” and hit Enter key.

To use:

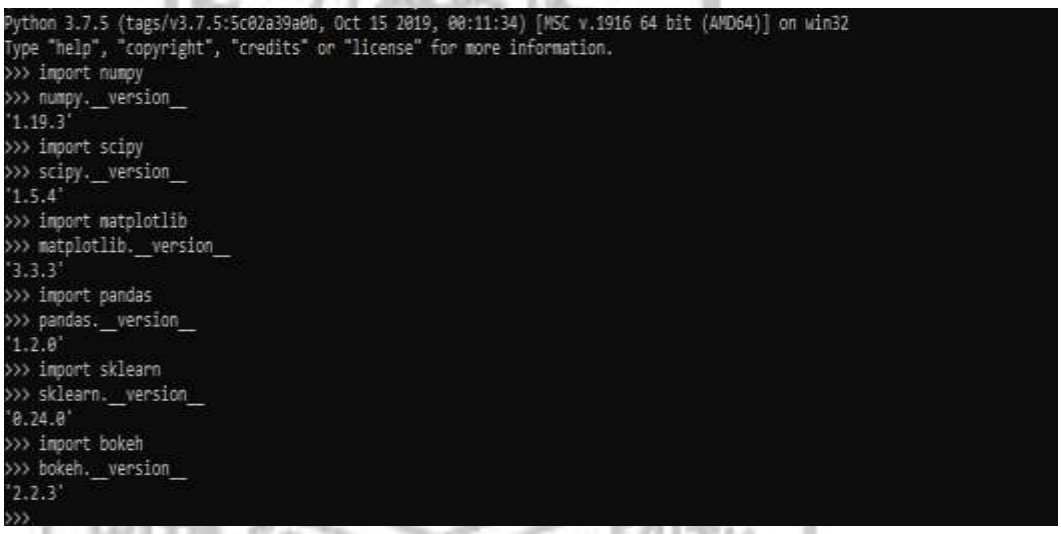
1. In Command Prompt, type “jupyter notebook” and hit Enter key.
2. By default a web browser tab with jupyter notebook will open. If not, type in the following URL to open - http://localhost:8888/tree
3. Do not close this Command Prompt opened in Step 1.
4. Click on New -> Python 3 (right top) to open a new Notebook.
5. To close (also called as “Shut down Jupyter”), close all newly created notebook tabs and click on “Quit”.

More on Jupyter Notebooks at https://jupyter.org/

Packages

We will install the following packages: numpy, scipy, matplotlib, pandas, scikit-learn (sklearn), bokeh.

1. Start->Command Prompt.
2. Type in “pip install numpy” and hit Enter key**.
   **If one encounters issue with installing/using numpy, try “pip install numpy==1.19.3”
3. Type in “pip install scipy matplotlib pandas sklearn bokeh” and hit Enter key.
4. To verify installation:
   a. Type in “python”, hit enter.
   b. Type in
      import <package_name>
      <package_name>.__version__
   c. This will display the desired package with it’s version number if properly installed as indicated below:



RESULT:

A study on the Python Data Science environment was carried out to understand and install the software packages required for Data Science experiments.

Ex. No. 2

OPERATIONS ON PYTHON DATA STRUCTURES

AIM:

To develop Python programs to perform operations on Python Data Structures such as String, List, Tuple, Dictionary, and Set.

(a) STRINGS

PROBLEM DEFINITION:

Check if the given pair of words are anagram using sorted() function. Print “True” if it is an anagram and “False” if not.

CODE:

```
def fn_test_anagram(string1, string2):
    string1_sorted = sorted(string1.lower())
    string2_sorted = sorted(string2.lower())
    if(string1_sorted == string2_sorted):
        return True
    else:
        return False


if __name__ == "__main__":
    input1 = "Binary"
    input2 = "Brainy"
    print(fn_test_anagram(input1, input2))
```

TEST CASE:

CASE 1: INPUT: Listen, Silent OUTPUT: True

CASE 2: INPUT: Chin, Inch OUTPUT: True

CASE 3: INPUT: Binary, Brainy OUTPUT: True

CASE 4: INPUT: About, Other OUTPUT: False

(b) DICTIONARY, LIST

PROBLEM DEFINITION:

Generate a dictionary of words and the corresponding number of times it occurred in a given sentence. Print the occurrence when the user enters a word and 0 if a word is not found. (Ignore ‘,’, ‘.’ and ‘?’)

CODE:

```
def fn_clean_string(test_string, list_to_remove):
    test_string = test_string.lower()
    for item in list_to_remove:
        test_string = test_string.replace(item, "")
    return test_string

def fn_word_frequency(test_string):
    word_list = test_string.split()
    word_count = []
    for word in word_list:
        word_count.append(word_list.count(word))
    word_freq_dict = dict(list(zip(word_list, word_count)))
    return word_freq_dict

def fn_display_count(test_word, word_freq_dict):
    test_word = test_word.lower()
    if test_word in word_freq_dict.keys():
        return word_freq_dict[test_word]
    else:
        return 0

if __name__ == "__main__":
    input_string = "She sells seashells on the sea shore. The shells she sells are seashells, I'm
sure. And if she sells seashells on the sea shore, Then I'm sure she sells seashore shells."
    list_to_remove = [".", ",", "?"]
    clean_string = fn_clean_string(input_string, list_to_remove)
    word_freq_dict = fn_word_frequency(clean_string)
    test_word = "Shells"
    print(fn_display_count(test_word, word_freq_dict))
```

TEST CASE:

CASE 1: INPUT: Shells OUTPUT: 2

CASE 2: INPUT: The OUTPUT: 3

CASE 3: INPUT: Sea shell OUTPUT: 0

CASE 4: INPUT: Shore. OUTPUT: 0

(c) TUPLES, LIST

PROBLEM DEFINITION:

Table given below is the Bowling scorecard from ICC Cricket World Cup Final, Apr 1 2011 - India vs Sri Lanka:

Bowler	Overs	Maidens	Runs	Wickets	Economy
Zaheer Khan	**10**	**3**	**60**	**2**	??
Sreesanth	**8**	**0**	**52**	**0**	??
Munaf Patel	**9**	**0**	**41**	**0**	??
Harbhajan Singh	**10**	**0**	**50**	**1**	??
Yuvraj Singh	**10**	**0**	**49**	**2**	??
Sachin Tendulkar	**2**	**0**	**12**	**0**	??
Virat Kohli	**1**	**0**	**6**	**0**	??

*(Source: ESPN cricinfo, https://www.espncricinfo.com/series/icc-cricket-world-cup-2010-11-381449/india-vs-sri-lanka-final-433606/full-scorecard)

Generate a list of tuples to store this data and perform the following operations. When user enters a player name, display

(i)How many wickets did the bowler pick?

(ii)What was the bowler's economy? (Economy = Runs/Overs)

CODE:

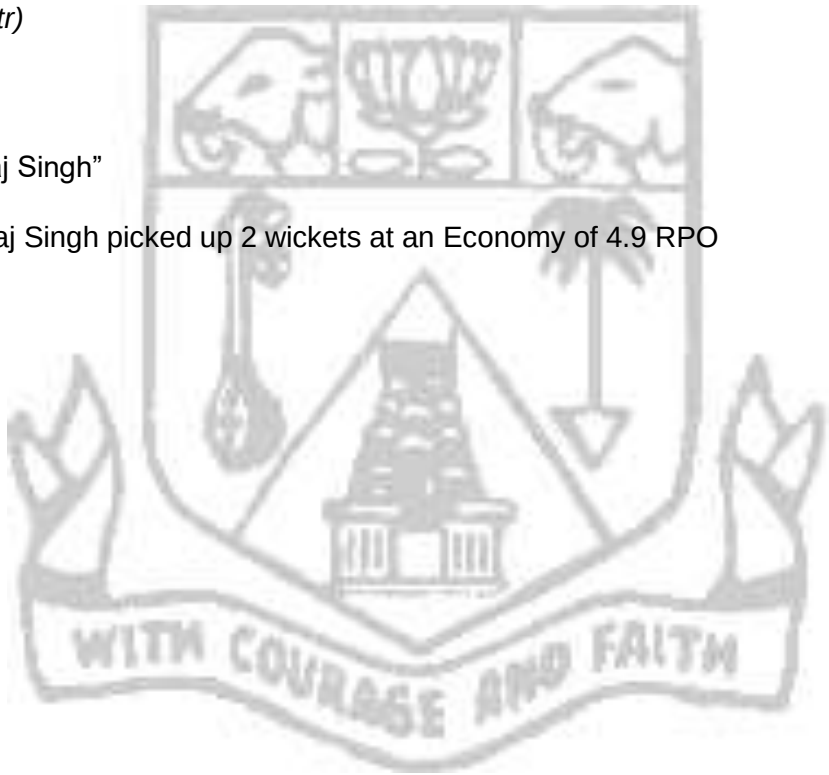
```
E = lambda a, b : round(a/b, 2)

def fn_create_tuple():

    data_list = [
    ( "Zaheer Khan", 10, 3, 60, 2),
    ( "Sreesanth", 8, 0, 52, 0),
    ( "Munaf Patel", 9, 0, 41, 0),
    ( "Harbhajan Singh", 10, 0, 50, 1),
    ( "Yuvraj Singh", 10, 0, 49, 2),
    ( "Sachin Tendulkar", 2, 0, 12, 0),
    ( "Virat Kohli", 1, 0, 6, 0)
    ]
    return data_list

def fn_inspect(player_name, data_list):
    wickets, economy = None, None
    for data_tuple in data_list:
```

```
        if player_name in data_tuple:
            wickets = data_tuple[4]
            economy = E(data_tuple[3], data_tuple[1])
    if wickets != None:
        result_str = player_name + " picked up " + str(wickets) +" wickets at an Economy of " +
str(economy) + " RPO"
    else:
        result_str = player_name + " did not bowl in this match"
    return result_str

if __name__ == "__main__":
    data_list = fn_create_tuple()
    player_name = "Yuvraj Singh"
    result_str = fn_inspect(player_name, data_list)
    print(result_str)
```

TEST CASE:

INPUT: “Yuvaraj Singh”

OUTPUT: Yuvraj Singh picked up 2 wickets at an Economy of 4.9 RPO



(d) SET, LIST

PROBLEM DEFINITION:

Generate a python program to do the following using SET operations:
a) To return a list without duplicates
b) To return a list that contains only the elements that are common between the lists

CODE:

```
def fn_dedup(x):
    return(list(set(x)))

def fn_find_common(x, y):
    return(list(set(x).intersection(set(y))))

if __name__ == "__main__":
    inp_list1 = [11, 22, 33, 44, 33, 22, 1]
    inp_list2 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
    print(fn_dedup(inp_list1))
    print(fn_find_common(inp_list1, inp_list2))
```

TEST CASE:

a) Duplicate Removal

INPUT: [11, 22, 33, 44, 33, 22, 1]

OUTPUT: [33, 1, 11, 44, 22]

b) Finding Common Elements

INPUT: [11, 22, 33, 44, 33, 22, 1] and [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

OUTPUT: [1, 11]

RESULT:

Python programs were developed to perform the desired operations on various data structures in Python.

Ex. No. 3

ARRAY OPERATIONS USING NUMPY

AIM:

To write Python program to perform simple arithmetic operations on 2D arrays using NumPy package.

PROBLEM DEFINITION:

Perform various matrix operations on 2D numpy matrices - Addition, Subtraction & Multiplication and generate a subset matrix using the concept of matrix slicing.

CODE:

```
import numpy as np

def fn_mat_sum(mat_a, mat_b):
    if mat_a.shape == mat_b.shape:
        mat_sum = mat_a + mat_b
    else:
        mat_sum = None
    return mat_sum

def fn_mat_diff(mat_a, mat_b):
    if mat_a.shape == mat_b.shape:
        mat_diff = mat_a - mat_b
    else:
        mat_diff = None
    return mat_diff

def fn_mat_mul(mat_a, mat_b):
    if mat_a.shape[1] == mat_b.shape[0]:
        mat_mul = np.dot(mat_a, mat_b)
    else:
        mat_mul = None
    return mat_mul

def fn_subset_mat(mat, r1, c1, r2, c2):
    if (r1>-1) and (c1>-1) and (r1<r2) and (c1<c2) and r2<mat.shape[0] and c2<mat.shape[1]:
        res = mat[r1:r2, c1:c2]
    else:
        res = None
    return res

if __name__ == "__main__":
    np.random.seed(3);
    ip_mat_a = np.random.randint(1, 20, size=(3, 3)); print(ip_mat_a)
    ip_mat_b = np.random.randint(1, 20, size=(3, 3)); print(ip_mat_b)
```

```
ip_mat_c = np.random.randint(1, 20, size=(5, 5)); print(ip_mat_c)
res_sum = fn_mat_sum(ip_mat_a, ip_mat_b)
res_diff = fn_mat_diff(ip_mat_a, ip_mat_b)
res_mul = fn_mat_mul(ip_mat_a, ip_mat_b)
res_subset_mat = fn_subset_mat(ip_mat_c, r1=1, c1=1, r2=3, c2=3)
print("Sum:\n", res_sum)
print("Diff:\n", res_diff)
print("Mult:\n", res_mul)
print("Subset:\n",res_subset_mat)
```

TEST CASE:

INPUT: -- (random number generation)

OUTPUT:

```
[[11  4  9]
 [ 1 11 12]
 [10 11  7]]
[[ 1 13  8]
 [15 18  3]
 [ 3  2  6]]
[[ 9 15  2 11  8]
 [12  2 16 17  6]
 [18 15  1  1 10]
 [19  6  8  6 15]
 [ 2 18  2 11 12]]
Sum:
 [[12 17 17]
 [16 29 15]
 [13 13 13]]
Diff:
 [[ 10  -9   1]
 [-14  -7   9]
 [  7   9   1]]
Mult:
 [[ 98 233 154]
 [202 235 113]
 [196 342 155]]
Subset:
 [[ 2 16]
 [15  1]]
```

RESULT:

Matrix operations on 2D arrays was carried out using NumPy.

Ex. No. 4

OPERATIONS ON PANDAS DATAFRAME

AIM:

To perform operations on Pandas DataFrame.

PROBLEM DEFINITION:

Create a Pandas dataframe from a dictionary of student details and perform the following operations on the data frame:

(i) Check for missing values,

(ii) Fill missing values in Attend9 with 0,

(iii) Fill missing values with minimum value in Assignment,

(iv) Replace by 0 in Test,

(v) Select rows based on conditions >=80, <80 and >=70, <70 for August Attendance,

(vi) Arrange and display students in decreasing order of September attendance,

(vii) Find students with 100% attendance for all three months together and include/display consolidated attendance as last column,

(viii) Display the details of students who scored maximum marks in test,

(ix) Display the details of students whose Assignment marks is less than Average of Assignment marks, and

(x) Display Result='Pass' if the student has scored more than 20 marks in Assignment+Test put together.

CODE:

```
import pandas as pd
import numpy as np

dictionary = {'RollNo.': [501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512],
        'Name': ['Ram.N.K', 'Kumar.A', 'Kavi.S', 'Malar.M', 'Seetha.P.', 'Kishore.L', 'Amit.M ',
'Daniel.R', 'Shyam.M.', 'Priya.N', 'Mani.R.', 'Ravi.S'],
        'Attend8': [92, 100, 100, 100, 76, 96, 100, 92, 68, 52, 72, 80],
        'Attend9' : [84, 95, 90, 100, 42, 84, 95, 100, 53, 16, 53, np.nan],
        'Attend10': [100, 100, 94, 100, 31, 81, 100, 100, 94, 13, 88, 6],
        'Assignment' : [15, 13, 14, 14, 13, 14, 14, 14, 5, np.nan, np.nan, np.nan],
        'Test' : [19, 14, 19, 18, 17, 19, 19, 19, 18, '-', 18, '-' ]
       }
#convert dictionary to pandas dataframe
df = pd.DataFrame(dictionary)
# print(df)
```

```
# Check for missing values
print('Count of missing values: \n' , df.isnull().sum())

# Fill missing values in Attend9 with 0
df['Attend9'] = df['Attend9'].fillna(0)

# Fill missing values with minimum value in Assignment
df['Assignment'] = df['Assignment'].fillna(df['Assignment'].min())

# Replace by 0 in Test
df = df.replace(['-'], 0)
print(df)

# Select rows based on conditions >=80, <80 and >=70, <70 for August Attendance
result80above_df = df[(df['Attend8']>=80)]
result70to80_df = df[(df['Attend8']<80) & (df['Attend8']>=70)]
result70below_df = df.loc[df['Attend8']<70]
print('Attendance above 80 \n', result80above_df)
print('Attendance between 70 and 80 \n', result70to80_df)
print('Attendance below 70 \n', result70below_df)

# Arrange and display students in decreasing order of September attendance
Attend9sorted_df = df.sort_values(by='Attend9', ascending=False)
print('Sorted September Attendance \n')
display(Attend9sorted_df.loc[:,['RollNo.','Name','Attend9']])

# Find students with 100% attendance for all three months together
# and include/display consolidated attendance as last column
sum_df = df['Attend8'] + df['Attend9'] + df['Attend10']
finalattend_df = sum_df/3
df['Consolidated Attendance'] = finalattend_df
print('Consolidated Attendance = \n', df)

# Display the details of students who scored maximum marks in test
Test_max = df['Test'].max()
Assign_max = df['Assignment'].max()
# using logical indexing display details of all students who scored maximum marks in Test
print('Details of students who scored maximum marks in Test = \n')
display(df.loc[df['Test']==df['Test'].max()])

# Display details of students whose Assignment marks is < than average of Assignment marks
Assign_mean = df['Assignment'].mean()
print('Details of students whose Assignment marks is less than Average of Assignment marks:
\n')
display(df[(df['Assignment']< Assign_mean)])

# Display Result='Pass' if the student has scored > than 20 in assignment+test put together
df['Result'] = df['Assignment']+ df['Test']
df['Result'] = df['Result'].apply(lambda x: 'Pass' if x >= 20 else 'Fail')
display(df)
```

TEST CASE:

INPUT: --

OUTPUT:

```
Count of missing values:
 RollNo.       0
Name           0
Attend8        0
Attend9        1
Attend10       0
Assignment     3
Test           0
dtype: int64
    RollNo.        Name  Attend8  Attend9  Attend10  Assignment  Test
0        501     Ram.N.K       92     84.0       100        15.0    19
1        502     Kumar.A      100     95.0       100        13.0    14
2        503      Kavi.S      100     90.0        94        14.0    19
3        504     Malar.M      100    100.0       100        14.0    18
4        505   Seetha.P.       76     42.0        31        13.0    17
5        506   Kishore.L       96     84.0        81        14.0    19
6        507     Amit.M       100     95.0       100        14.0    19
7        508    Daniel.R       92    100.0       100        14.0    19
8        509    Shyam.M.       68     53.0        94         5.0    18
9        510     Priya.N       52     16.0        13         5.0     0
10       511     Mani.R.       72     53.0        88         5.0    18
11       512      Ravi.S       80      0.0         6         5.0     0
Attendance above 80
     RollNo.        Name  Attend8  Attend9  Attend10  Assignment  Test
0        501     Ram.N.K       92     84.0       100        15.0    19
1        502     Kumar.A      100     95.0       100        13.0    14
2        503      Kavi.S      100     90.0        94        14.0    19
3        504     Malar.M      100    100.0       100        14.0    18
5        506   Kishore.L       96     84.0        81        14.0    19
6        507     Amit.M       100     95.0       100        14.0    19
7        508    Daniel.R       92    100.0       100        14.0    19
11       512      Ravi.S       80      0.0         6         5.0     0
Attendance between 70 and 80
     RollNo.        Name  Attend8  Attend9  Attend10  Assignment  Test
4        505   Seetha.P.       76     42.0        31        13.0    17
10       511     Mani.R.       72     53.0        88         5.0    18
Attendance below 70
    RollNo.       Name  Attend8  Attend9  Attend10  Assignment  Test
8       509  Shyam.M.       68     53.0        94         5.0    18
```

```
9       510    Priya.N       52     16.0        13         5.0     0
Sorted September Attendance
```

	RollNo.	Name	Attend9
3	504	Malar.M	100.0
7	508	Daniel.R	100.0
1	502	Kumar.A	95.0
6	507	Amit.M	95.0
2	503	Kavi.S	90.0
0	501	Ram.N.K	84.0
5	506	Kishore.L	84.0
8	509	Shyam.M.	53.0
10	511	Mani.R.	53.0
4	505	Seetha.P.	42.0
9	510	Priya.N	16.0
11	512	Ravi.S	0.0

```
Consolidated Attendance =
     RollNo.       Name  Attend8  Attend9  Attend10  Assignment  Test  \
0        501    Ram.N.K       92     84.0       100        15.0    19
1        502    Kumar.A      100     95.0       100        13.0    14
2        503     Kavi.S      100     90.0        94        14.0    19
3        504    Malar.M      100    100.0       100        14.0    18
4        505  Seetha.P.       76     42.0        31        13.0    17
5        506  Kishore.L       96     84.0        81        14.0    19
6        507    Amit.M       100     95.0       100        14.0    19
7        508   Daniel.R       92    100.0       100        14.0    19
```

```
8       509    Shyam.M.       68      53.0        94         5.0     18
9       510     Priya.N       52      16.0        13         5.0      0
10      511     Mani.R.       72      53.0        88         5.0     18
11      512      Ravi.S       80       0.0         6         5.0      0

    Consolidated Attendance
0                 92.000000
1                 98.333333
2                 94.666667
3                100.000000
4                 49.666667
5                 87.000000
6                 98.333333
7                 97.333333
8                 71.666667
9                 27.000000
10                71.000000
11                28.666667
Details of students who scored maximum marks in Test =
```

	RollNo.	Name	Attend8	Attend9	Attend10	Assignment	Test	Consolidated Attendance
0	501	Ram.N.K	92	84.0	100	15.0	19	92.000000
2	503	Kavi.S	100	90.0	94	14.0	19	94.666667
5	506	Kishore.L	96	84.0	81	14.0	19	87.000000
6	507	Amit.M	100	95.0	100	14.0	19	98.333333
7	508	Daniel.R	92	100.0	100	14.0	19	97.333333

```
Details of students whose Assignment marks is less than Average of Assignment
marks:
```

	RollNo.	Name	Attend8	Attend9	Attend10	Assignment	Test	Consolidated Attendance
8	509	Shyam.M.	68	53.0	94	5.0	18	71.666667
9	510	Priya.N	52	16.0	13	5.0	0	27.000000

	RollNo.	Name	Attend8	Attend9	Attend10	Assignment	Test	Consolidated Attendance
10	511	Mani.R.	72	53.0	88	5.0	18	71.000000
11	512	Ravi.S	80	0.0	6	5.0	0	28.666667

	RollNo.	Name	Attend8	Attend9	Attend10	Assignment	Test	Consolidated Attendance	Result
0	501	Ram.N.K	92	84.0	100	15.0	19	92.000000	Pass
1	502	Kumar.A	100	95.0	100	13.0	14	98.333333	Pass
2	503	Kavi.S	100	90.0	94	14.0	19	94.666667	Pass
3	504	Malar.M	100	100.0	100	14.0	18	100.000000	Pass
4	505	Seetha.P.	76	42.0	31	13.0	17	49.666667	Pass
5	506	Kishore.L	96	84.0	81	14.0	19	87.000000	Pass
6	507	Amit.M	100	95.0	100	14.0	19	98.333333	Pass
7	508	Daniel.R	92	100.0	100	14.0	19	97.333333	Pass
8	509	Shyam.M.	68	53.0	94	5.0	18	71.666667	Pass
9	510	Priya.N	52	16.0	13	5.0	0	27.000000	Fail
10	511	Mani.R.	72	53.0	88	5.0	18	71.000000	Pass
11	512	Ravi.S	80	0.0	6	5.0	0	28.666667	Fail

RESULT:

The given operations were performed on Pandas DataFrame.

Ex. No. 5

DATA CLEANING AND PROCESSING IN CSV FILES

AIM:

To perform reading, data cleaning, processing and writing operations in CSV files using Pandas package.

PROBLEM DEFINITION:

Compute the final student grade based on two intermediate grades, such that Gfinal = (G1 + G2)*100/40 and save as two separate csv files based on Gfinal score (50+ and below 50) . Data is to be read from a csv file and stored back in a new csv (Use , as separator).

CODE:

```
# Data Source
# Title: Student Performance Data Set
# Hosted Link : https://archive.ics.uci.edu/ml/datasets/Student+Performance
# Download Link: https://archive.ics.uci.edu/ml/machine-learning-databases/00320/student.zip

# Note: For the following program download the dataset on your local machine and name it as "student-mat.csv" in the current folder.

import pandas

def fn_compute_gfinal(data_frame):
    # Check if there are any missing values in the data
    if data_frame.isnull().values.any():
        # Replace all NaN with zeros
        print("Detected NaN, replacing with 0")
        data_frame.fillna(0)
    else:
        # G1 & G2 indicates scores by students in first & second internal exams resp.
        # Delete the attribute G3
        data_frame.drop(columns=['G3'], inplace=True);
        # Create a new attribute named "Gfinal" (last attribute), Gfinal = (G1 + G2)*100/40
        data_frame.insert(len(data_frame.columns), 'Gfinal', '');
        data_frame['Gfinal']=(data_frame['G1'] + data_frame['G2'])*100/40;
        df_50plus = data_frame[data_frame['Gfinal'] >= 50]
        df_below50 = data_frame[data_frame['Gfinal'] < 50]
        return df_50plus, df_below50

if __name__ == "__main__":
    data_frame_ip = pandas.read_csv("student-mat.csv", delimiter=";")
    df_50plus_op, df_below50_op = fn_compute_gfinal(data_frame_ip)
    # Use the following statement to display a sample of data frames
    # print(df_50plus_op.head(), df_below50_op.head())
    df_50plus_op.to_csv("result_50plus.csv", sep=',', index=False)
    df_below50_op.to_csv("result_below50.csv", sep=',', index=False)
```

TEST CASE:

INPUT: *student-mat.csv*

school	sex	age	address	famsize	Pstatus	...	Walc	health	absences	G1	G2	G3
GP	F	18	U	GT3	A	...	1	3	6	5	6	6
GP	F	17	U	GT3	T	...	1	3	4	5	5	6
GP	F	15	U	LE3	T	...	3	3	10	7	8	10
GP	F	15	U	GT3	T	...	1	5	2	15	14	15
GP	F	16	U	GT3	T	...	2	5	4	6	10	10

OUTPUT:

Gfinal >= 50 (*result_50plus.csv***)**

	school	sex	age	address	famsize	...	health	absences	G1	G2	Gfinal
3	GP	F	15	U	GT3	...	5	2	15	14	72.5
5	GP	M	16	U	LE3	...	5	10	15	15	75.0
6	GP	M	16	U	LE3	...	3	0	12	12	60.0
8	GP	M	15	U	LE3	...	1	0	16	18	85.0
9	GP	M	15	U	GT3	...	5	0	14	15	72.5

Gfinal < 50 (*result_below50.csv***)**

	school	sex	age	address	famsize	...	health	absences	G1	G2	Gfinal
0	GP	F	18	U	GT3	...	3	6	5	6	27.5
1	GP	F	17	U	GT3	...	3	4	5	5	25.0
2	GP	F	15	U	LE3	...	3	10	7	8	37.5
4	GP	F	16	U	GT3	...	5	4	6	10	40.0
7	GP	F	17	U	GT3	...	1	6	6	5	27.5

RESULT:

Reading, data cleaning, processing and writing operations in CSV files was carried out using Pandas package.

Ex. No. 6

HANDLING CSV FILES

AIM:

To read from and write onto CSV files using Pandas package.

PROBLEM DEFINITION:

Perform data analysis on historical BSE SENSEX data from 2018 to 2020.

CODE:

```
# Data: Indices - S&P BSE SENSEX
# Source: https://www.bseindia.com/indices/IndexArchiveData.html
# Note: Make sure to name the data file "csv_base_sensex_2018to2020.csv" and is located in
the current folder.


import pandas as pd
import datetime
import numpy as np

def fn_extract_high_low(data_frame):
    # Data Cleanup
    data_frame.drop(data_frame.columns[-1], axis=1, inplace=True)
    data_frame["Date"] = pd.to_datetime(data_frame["Date"], format='%d-%B-%Y')
    # Write your code here to ensure all nan/empty cells are taken care of
    # Filter data for FY 2018-19
    start_date = datetime.datetime.strptime('2018-03-31', '%Y-%m-%d')
    end_date = datetime.datetime.strptime('2019-04-01', '%Y-%m-%d')
    df_fy = data_frame[(data_frame["Date"] > start_date) & (data_frame["Date"] < end_date)]
    # Other way: df_fy = data_frame[(data_frame["Date"] > '2018-03-31') & (data_frame["Date"] <
'2019-04-01')]

    fy_high = df_fy["High"].max()
    fy_low = df_fy["Low"].min()

    # print(df_fy_mean.head()); print(df_fy_median.head())

    return fy_high, fy_low, df_fy

if __name__ == "__main__":
    data_frame_ip = pd.read_csv("csv_base_sensex_2018to2020.csv", index_col=None)
    fy_high, fy_low, df_fy = fn_extract_high_low(data_frame_ip)
    df_fy.to_csv("sensex_fy2019-20.csv", sep=',', index=False)
    print("S&P BSE SENSEX High & Low in FY2019-20: ", fy_high, " & ", fy_low)
```

TEST CASE:

INPUT: *csv_base_sensex_2018to2020.csv*

OUTPUT:

S&P BSE SENSEX High & Low in FY2019-20: 38989.65 & 32972.56

RESULT:

Reading from and writing to CSV files was done using Pandas package.

Ex. No. 7

HANDLING HTML AND EXCEL FILES

AIM:

To write Python program to handle HTML and EXCEL files.

PROBLEM DEFINITION:

Find the list of Indian Regional Navigation Satellite System IRNSS-1 series satellites launched so far into Space using the information available in IRNSS Wikipedia webpage.

CODE:

```
# Title: Wikipedia - Indian Regional Navigation Satellite System
# Link: https://en.wikipedia.org/wiki/Indian_Regional_Navigation_Satellite_System

# Note: Your computer should have an active internet connection and must be able to access the above link

import pandas
def fn_irnss_df(target_URL, target_table):
    irnss_data = pd.read_html(target_URL, match=target_table)
    irnss_df = irnss_data[0]
    # Create a dataframe without Planned Satellite Launch
    irnss_df_sub = irnss_df[~irnss_df['Status'].str.contains('Planned')]
    # Sort the dataframe in order of date with latest at first
    irnss_df_sub['Launch Date'] =  pd.to_datetime(irnss_df_sub['Launch Date'], format='%d %B %Y')
    irnss_df_sub = irnss_df_sub.sort_values(by='Launch Date', ascending=False)
    # Store the data in the same format (as in original dataframe) to an Excel file
    irnss_df_sub['Launch Date'] = irnss_df_sub['Launch Date'].apply(lambda x: x.strftime('%d %B %Y'))
    return irnss_df_sub

if __name__ == "__main__":
    target_URL = "https://en.wikipedia.org/wiki/Indian_Regional_Navigation_Satellite_System"
    target_table = "IRNSS-1 series satellites"
    df_out =  fn_irnss_df(target_URL, target_table)
    df_out.to_excel(r'result.xlsx', sheet_name='IRNSS Launch', index = False)
```

TEST CASE:

INPUT: -- (given in program)

target_URL = "https://en.wikipedia.org/wiki/Indian_Regional_Navigation_Satellite_System"
target_table = "IRNSS-1 series satellites"

OUTPUT: (*'result.xlsx***)**

	A	B	C	D	E	F	G	H	I	J
1	Satellite	SVN	PRN	Int. Sat. ID	NORAD ID	Launch Date	Launch Vehicle	Orbit	Status	Remarks
2	IRNSS-1I	I009		2018-035A	43286	12 April 2018	PSLV-XL-C41	Geosynchronous (IGSO) / 55°E, 29° inclined orbit	Operational	[31]
3	IRNSS-1H					31 August 2017	PSLV-XL-C39		Launch Failed	The payload fairing failed to separate and sa
4	IRNSS-1G	I007	I07	2016-027A	41469	28 April 2016	PSLV-XL-C33	Geostationary (GEO) / 129.5°E, 5.1° inclined orbit	Operational	
5	IRNSS-1F	I006	I06	2016-015A	41384	10 March 2016	PSLV-XL-C32	Geostationary (GEO) / 32.5°E, 5° inclined orbit	Operational	
6	IRNSS-1E	I005	I05	2016-003A	41241	20 January 2016	PSLV-XL-C31	Geosynchronous (IGSO) / 111.75°E, 29° inclined orbit	Operational	
7	IRNSS-1D	I004	I04	2015-018A	40547	28 March 2015	PSLV-XL-C27	Geosynchronous (IGSO) / 111.75°E, 31° inclined orbit	Operational	
8	IRNSS-1C	I003	I03	2014-061A	40269	16 October 2014	PSLV-XL-C26	Geostationary (GEO) / 83°E, 5° inclined orbit	Operational	
9	IRNSS-1B	I002	I02	2014-017A	39635	04 April 2014	PSLV-XL-C24	Geosynchronous (IGSO) / 55°E, 29° inclined orbit	Operational	
10	IRNSS-1A	I001	I01	2013-034A	39199	01 July 2013	PSLV-XL-C22	Geosynchronous (IGSO) / 55°E, 29° inclined orbit	Partial Failure	Atomic clocks failed.The satellite is being us

RESULT:

HTML and Excel files were handled using Pandas package..

Ex. No. 8

PROCESSING TEXT FILES

AIM:

To write a Python program to read and process text file.

PROBLEM DEFINITION:

Find the frequency of occurrence of a given word in a given text file.

CODE:

```
# Note: To execute this code, keep the text data file "TxtSample.txt" in the current folder.
def fn_read_process(f_name):
    doc_as_word = []
    with open(f_name, "rt") as f_obj:
        doc_as_words =[word for line in f_obj for word in line.split()]

    doc_as_words = [elem.lower() for elem in doc_as_words]
    # Data Cleanup - Removing Punc
    char_to_clean = '"!;:"\, ./?@#$%^&*_~'"
    doc_as_words_clean = []
    for list_entry in doc_as_words:
        flag = False
        for entry in list_entry:
            if entry in char_to_clean:
                flag = True
                list_entry = list_entry.replace(entry, "")
                doc_as_words_clean.append(list_entry)
        if flag == False:
            doc_as_words_clean.append(list_entry)
    return doc_as_words_clean

def fn_count_freq(words, test_word):
    return words.count(test_word.lower())

if __name__ == "__main__":
    words_list = fn_read_process(f_name='TxtSample.txt')
    print(fn_count_freq(words_list, test_word="test"))
```

TEST CASE:

CASE1: INPUT: Text OUTPUT: 6

CASE 2: INPUT: data OUTPUT: 1

CASE 3: INPUT: INDIA OUTPUT: 0

RESULT:

A given text file was processed using Python program.

Ex. No. 9

DATA WRANGLING (PIVOT TABLE, MELT, CONCAT)

AIM:

To perform data wrangling using Pandas.

PROBLEM STATEMENT:

Perform analysis on Computer hardware dataset to extract available vendor names, their models & machine cycle times (MYCT).

CODE:

```
# Data Source
# Title: Computer Hardware Data Set
# Hosted Link : https://archive.ics.uci.edu/ml/datasets/Computer+Hardware
# Download Link: https://archive.ics.uci.edu/ml/machine-learning-databases/cpu-performance/

# Note: In the following program the dataset be named "machine.data" (a csv file) and located in the current folder.

import pandas as pd
import numpy as np

def fn_get_model_myct(df):
    # Perform statistical summary - Mean and Median using Pivot table function
    df_mean = pd.pivot_table(df, values=["MYCT", "MMIN", "MMAX", "CACH", "CHMIN", "CHMAX", "PRP"], columns="vendor name", aggfunc = np.mean)
    df_median = pd.pivot_table(df, values=["MYCT", "MMIN", "MMAX", "CACH", "CHMIN", "CHMAX", "PRP"], columns="vendor name", aggfunc = np.mean)
    # Create a new dataframe from df_mean such that it has the folowing columns: ["vendor name", "Mean MYCT"]
    df_myct_mean = pd.DataFrame({"vendor name" : list(df_mean.columns),
                "Mean MYCT":df_mean.values.tolist()[5]})

    # Use pandas.melt() function to extract "Model Name"
    df_melt_models = pd.melt(df, id_vars =["vendor name"], value_vars =["Model Name"])
    # Use pandas.melt() function to convert df_myct_mean to long format
    df_melt_myct_mean = pd.melt(df_myct_mean, id_vars =["vendor name"], value_vars =["Mean MYCT"])
    # Stack df_melt_models and df_melt_myct_mean vertically
    data_model_myct = pd.concat([df_melt_models, df_melt_myct_mean], ignore_index=True)
    return data_model_myct

if __name__ == "__main__":
    data_frame_ip = pd.read_csv("machine.data", index_col=None, header=None, names=["vendor name", "Model Name", "MYCT", "MMIN", "MMAX", "CACH", "CHMIN", "CHMAX", "PRP", "ERP"])
    data_model_myct = fn_get_model_myct(data_frame_ip)
    print(data_model_myct)
```

TEST CASE:

INPUT: -- (preloaded machine dataset)

OUTPUT:

	vendor name	variable	value
0	adviser	Model Name	32/60
1	amdahl	Model Name	470v/7
2	amdahl	Model Name	470v/7a
3	amdahl	Model Name	470v/7b
4	amdahl	Model Name	470v/7c
..	...	...	...
234	prime	Mean MYCT	160
235	siemens	Mean MYCT	92.75
236	sperry	Mean MYCT	101.385
237	sratus	Mean MYCT	125
238	wang	Mean MYCT	480

RESULT:

Data Wrangling including pivoting, melting and concatenating the data loaded in data frames was done using Pandas.

Ex. No. 10

GENERATING LINE CHART AND BAR GRAPH USING MATPLOTLIB

AIM:

To use Matplotlib for plotting line chart and bar graph.

(a) LINE CHART

PROBLEM STATEMENT:

Create a figure with two subplots using Matplotlib package to display copper and aluminium prices during 1951-1975.

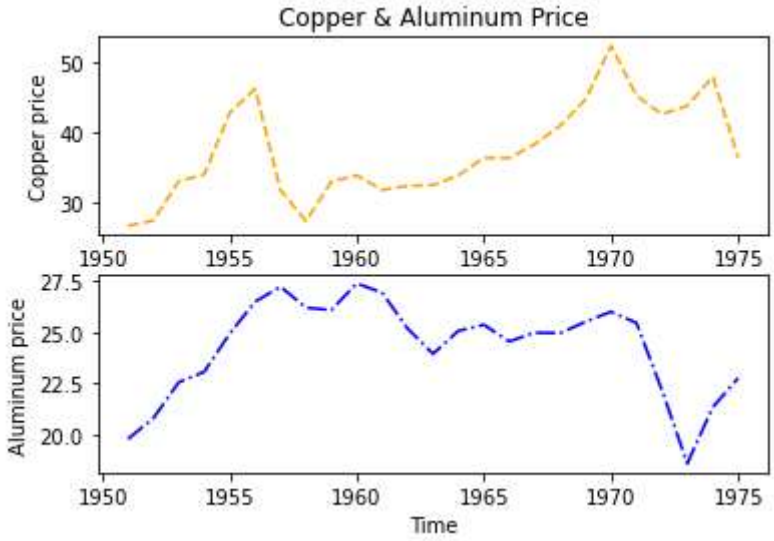
CODE:

```
# https://www.statsmodels.org/devel/datasets/index.html
# https://github.com/statsmodels/statsmodels/tree/master/statsmodels/datasets

# Brief Info on Dataset: sm.datasets.<data_set_name>.NOTE
# Extract pandas data_frame from Dataset: sm.datasets.<data_set_name>.load_pandas().data

import statsmodels.api as sm
# Color List: https://matplotlib.org/tutorials/colors/colors.html
import matplotlib.pyplot as plt

# Loading "World Copper Market 1951-1975 Dataset"
#print(sm.datasets.copper.NOTE)
df = sm.datasets.copper.load_pandas().data

# Create a figure with two subplots - ax1, ax2
fig1 = plt.figure() #Use argument figsize=(10,5) to create fig of specific size
ax1 = plt.subplot(2,1,1)
ax2 = plt.subplot(2,1,2)

ax1_x = range(1951,1975+1)
ax1_y = df["COPPERPRICE"].values
ax1.plot(ax1_x, ax1_y, color='orange', ls='--')

ax2_x = range(1951,1975+1)
ax2_y = df["ALUMPRICE"].values
ax2.plot(ax2_x, ax2_y, color='blue', ls='-.')

# Syntax for label/title: ax.set(xlabel='x', ylabel='y', title='t')
ax1.set(xlabel='Time', ylabel='Copper price', title = "Copper & Aluminum Price")
ax2.set(xlabel='Time', ylabel='Aluminum price')
```

TEST CASE:

INPUT: -- (built-in dataset)

OUTPUT:



(b) BAR GRAPH

PROBLEM DEFINTION:

Create a visualization using bar plot and line chart in the same figure to depict the world consumption and manufacturing inventory trend of copper.

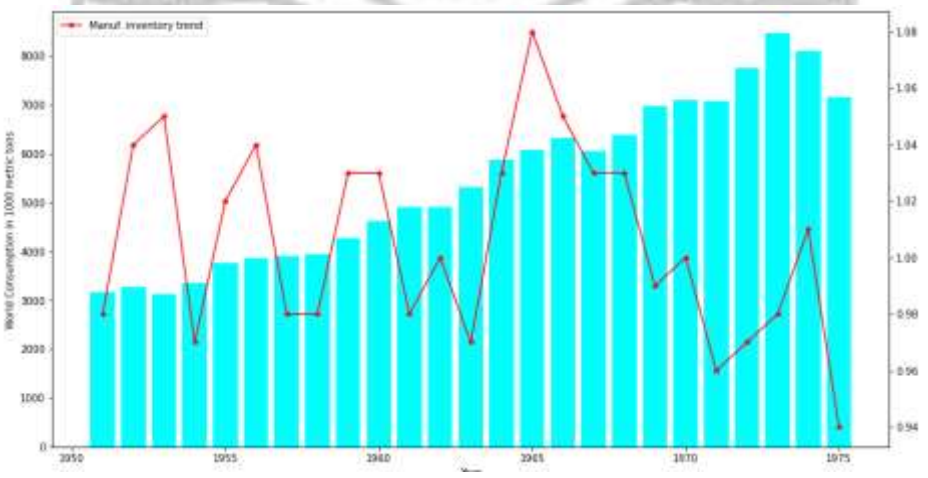
CODE:

```
import statsmodels.api as sm
import matplotlib.pyplot as plt

df = sm.datasets.copper.load_pandas().data

x = range(1951,1975+1)
y1 = df["WORLDCONSUMPTION"].values
y2 = df["INVENTORYINDEX"].values

fig2, ax1 = plt.subplots(figsize=(15,8))
ax2 = ax1.twinx()
ax1.bar(x, y1, color = 'cyan', zorder=2)
ax1.set_xlabel('Year')
ax1.set_ylabel('World Consumption in 1000 metric tons')
ax2.plot(x, y2, 'r-*', label = "Manuf. inventory trend", zorder=1)
ax2.legend(loc="upper left")
plt.show()
```

TEST CASE:

INPUT: -- (built-in dataset)

OUTPUT:



RESULT:

Line Chart and Bar Graph was generated using Matplotlib.

Ex. No. 11

DISPLAY DATA IN GEOGRAPHICAL MAP

AIM:

To use the GeoPandas package to plot data in geographical map.

PROBLEM DEFINITION:

Plot GDP estimates on the world map using the GeoPandas package.

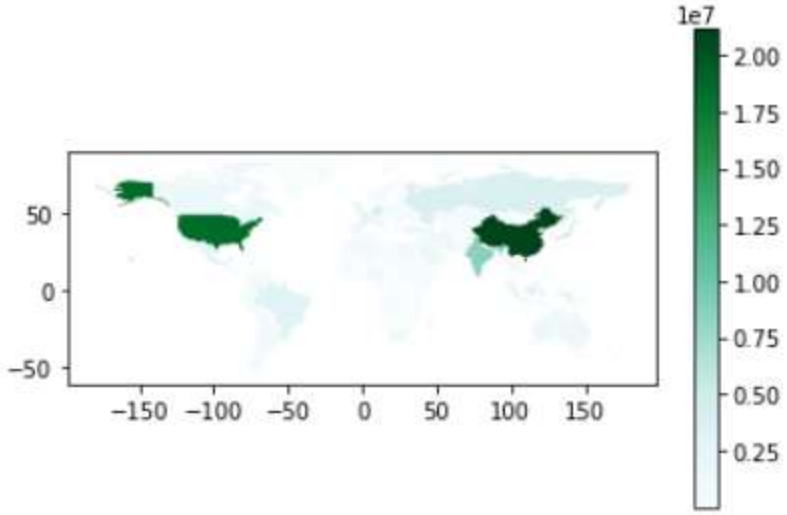
CODE:

```
# Reference: https://geopandas.org/mapping.html
# Make sure to install GeoPandas package
# Run “pip install geopandas” on command window and invoke jupyter notebook again to run code

import geopandas
import matplotlib.pyplot as plt
world = geopandas.read_file(geopandas.datasets.get_path('naturalearth_lowres'))
world = world[(world.name!="Antarctica")]
fig, ax = plt.subplots(1, 1)
world.plot(column='gdp_md_est', ax=ax, legend=True, cmap='BuGn')
```

TEST CASE:

INPUT: --

OUTPUT:



RESULT:

Data was displayed on geographical map using GeoPandas package.

Ex. No. 12

DISPLAY DATA IN HEATMAP

AIM:

To display data in the form of Heatmap.

PROBLEM DEFINITION:

Plot the minimum and maximum values against the vendor names from the machine data (used in Ex. No. 9) in the form of heatmap.

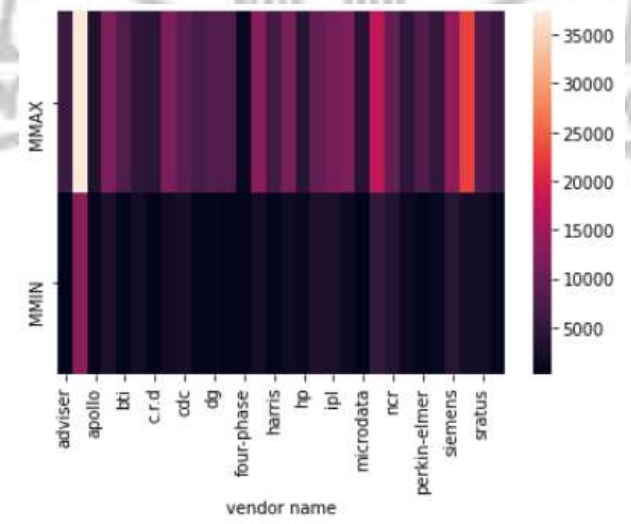
CODE:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv("machine.data", index_col=None, header=None, names=["vendor name", "Model Name", "MYCT", "MMIN", "MMAX", "CACH", "CHMIN", "CHMAX", "PRP", "ERP"])
df_mean_sub = pd.pivot_table(df, values=["MMIN", "MMAX"], columns="vendor name", aggfunc = np.mean)
h_map = sns.heatmap(df_mean_sub, annot=False)
plt.show()
```

TEST CASE:

INPUT: (*machine.data*)

OUTPUT:



RESULT:

Data was displayed in the form of heatmap.

Ex. No. 13

NORMAL AND CUMULATIVE DISTRIBUTION

AIM:

To implement normal and cumulative distribution models using SciPy package.

(a) **NORMAL DISTRIBUTION**

PROBLEM DEFINITION:

Create a normal distribution model for adult height in the range of values 150 to 180 and test whether a given height is adult or not.

CODE:

```
import numpy as np
from matplotlib import pyplot
from scipy.stats import norm

# Function to create a normal distribution model to model adult height
def fn_create_normalpdf():
    # create a height array to store height values from 150 to 180
    height = np.linspace(150,180,100)
    # plot a histogram of geight values
    pyplot.hist(height,12)
    pyplot.show()
    # find the parameters required to compute normal distribution
    mean_height = np.mean(height)
    stdev_height = np.std(height)
    # calculate the Normal Distribution pdf for height data
    pdf_height = norm.pdf(height, mean_height, stdev_height)
    # plot Normal Distribution curve to show Adult Height Model
    figure,ax = pyplot.subplots()
    ax.set_xlabel('Adult Height')
    ax.set_ylabel('Probabilities of Adult Height')
    pyplot.plot(height, pdf_height)
    pyplot.show()
    # create a list of values to be returned to main function
    pdf_params = [mean_height, stdev_height, pdf_height]
    return pdf_params

# Function to test whether a given height is adult or not
def fn_test(test_data, pdf_params):
    mean_height = pdf_params[0]
    stdev_height = pdf_params[1]
    pdf_height = pdf_params[2]
    pdf_test_data = norm.pdf(test_data, mean_height, stdev_height)
    print(pdf_test_data)
    min_pdf_height = min(pdf_height)
    max_pdf_height = max(pdf_height)
    if pdf_test_data >= min_pdf_height and pdf_test_data <= max_pdf_height:
```

```
        result = 'test data is adult height '
    else:
        result = 'test data is not adult height '
    return result

if _name__ == "__main__":
    pdf_params = fn_create_normalpdf()
    test_data = 170
    result = fn_test(test_data, pdf_params)
    print(result)
```

TEST CASE:

CASE 1: INPUT: 100 OUTPUT: test data is not adult height

CASE 2: INPUT: 170 OUTPUT: test data is adult height

(b) **CUMULATIVE DISTRIBUTION**

PROBLEM DEFINITION:

Using Cumulative distribution, find the probability that the height of the person (randomly picked from the distribution that models adult height in the range 150 to 180) will be

(i) less than 160 cm,
(ii) between 160 and 170 cm, and
(iii) greater than 170 cm.

CODE:

```
import numpy as np
from matplotlib import pyplot
from scipy.stats import norm

# Function to create a normal distribution to model adult height
def fn_create_normalpdf():
    # Create the distribution
    height = np.linspace(150,180,100)

    mean_height = np.mean(height)
    stdev_height = np.std(height)

    # calculate the Normal Distribution pdf for height data
    pdf_height = norm.pdf(height, mean_height, stdev_height)
    pdf_params = [mean_height, stdev_height]
    return(pdf_params)

def fn_test(test_data1, test_data2, pdf_params):
    # Probability of height to be under 160cm.
    mean_height = pdf_params[0]
    stdev_height = pdf_params[1]
    prob_1 = norm(loc = mean_height , scale = stdev_height).cdf(test_data1)

    # probability that the height of the person will be between 160 and 170 cm.
    cdf_upper_limit = norm(loc = mean_height , scale = stdev_height).cdf(test_data2)
    cdf_lower_limit = norm(loc = mean_height , scale = stdev_height).cdf(test_data1)
    prob_2 = cdf_upper_limit - cdf_lower_limit

    # probability that the height of a person chosen randomly will be above 170 cm.
    cdf_value = norm(loc = mean_height , scale = stdev_height).cdf(test_data2)
    prob_3 = 1- cdf_value

    result = [prob_1, prob_2, prob_3]
    return(result)

if __name__ == "__main__":
    pdf_params = fn_create_normalpdf()
    test_data1 = 160
    test_data2 = 170
```

```
    result = fn_test(test_data1, test_data2, pdf_params)
    print('Probability of height to be under 160cm is = ', result[0])
    print('probability that the height of the person will be between 160 and 170 cm = ', result[1])
    print('probability that the height of a person chosen randomly will be above 170 cm = ',
result[2])
```

TEST CASE:

INPUT: 160, 170 (given in code)

OUTPUT:

```
Probability of height to be under 160cm is =  0.28379468592429447
probability that the height of the person will be between 160 and 170 cm =
0.43241062815141107
probability that the height of a person chosen randomly will be above 170
cm =  0.28379468592429447
```

RESULT:

Normal and Cumulative distribution models were implemented using SciPy package.

Ex. No. 14

HYPOTHESIS TESTING

AIM:

To use the SciPy package to conduct hypothesis testing.

PROBLEM DEFINITION:

Create a data array with 10 height values and check whether a given test height (example: 170 or 165 or 70 or 120) is the average height or not using One Sample t Test as hypothesis testing tool.

CODE:

```
# One Sample t Test determines whether the sample mean is statistically different from a known or hypothesized population mean.
# The One Sample t Test is a parametric test.

from scipy.stats import ttest_1samp
import numpy as np

def one_sample_t_test(test_data):
    height = np.array([165,170,160,154,175,155,167,177,158,178])
    print(height)
    height_mean = np.mean(height)
    print('Mean Height = ', height_mean)
    tset, pval = ttest_1samp(height, test_data)
    print('p-values are: ', pval)
    if pval < 0.05:    # alpha value is 0.05 or 5%
       result = 'we are rejecting null hypothesis '
    else:
      result = 'we are accepting null hypothesis '
    return result

if __name__ == "__main__":
    test_data = 170
    result = one_sample_t_test(test_data)
    print(result)
```

TEST CASE:

CASE 1: INPUT: 170 OUTPUT: we are accepting null hypothesis

CASE 2: INPUT: 90 OUTPUT: we are rejecting null hypothesis

RESULT:

Hypothesis testing was accomplished using SciPy package.

ADDITIONAL EXERCISES

Ex. No. 1

GENERATION OF FACTOR PAIRS OF A GIVEN INTEGER

AIM:

To write a Python program to generate the factor pairs of a given integer.

PROBLEM DEFINITION:

Find the factor pairs of the given integer and store them as a list of tuples.
Factor Pair: Pairs of numbers that multiply to generate the original number are called as factor pair
Example: Factor pair of 12 are: 1 x 12 = 12, 2 x 6 = 12, 3 x 4 = 12

CODE:

```
def fn_factor_pair(test_num):
    factor_pair_list = []
    factor_list = []
    for num in range(1,test_num+1):
        if test_num % num == 0:
            factor_list.append(num)

    len_factor_list = len(factor_list)
    for iter_var1 in range(0, len_factor_list-1):
        for iter_var2 in range(iter_var1, len_factor_list):
            if factor_list[iter_var1]*factor_list[iter_var2] == test_num:
                factor_pair_list.append((factor_list[iter_var1], factor_list[iter_var2]))
    return factor_pair_list

if __name__ == "__main__":
    input_num = 36
    print(fn_factor_pair(input_num))
```

TEST CASE:

CASE 1: INPUT: 60 OUTPUT: [(1, 60), (2, 30), (3, 20), (4, 15), (5, 12), (6, 10)]

CASE 2: INPUT: 47 OUTPUT: [(1, 47)]

CASE 3: INPUT: 36 OUTPUT: [(1, 36), (2, 18), (3, 12), (4, 9), (6, 6)]

RESULT:

The factor pairs for a given integer were generated.

Ex. No. 2

AVERAGE POOLING ON A GIVEN NXN MATRIX WITH A MXM KERNEL

AIM:

To perform “average pooling” on a given *n x n* matrix with a *m x m* kernel.

PROBLEM DEFINITION:

Perform an “average pooling” on a given *n x n* matrix with a *m x m* kernel using Numpy package.

CODE:

```
import numpy as np

def fn_create_avg_pool(data_array, k_size):
    avg_pool_matrix = np.zeros((len(data_array)-k_size+1, len(data_array)-k_size+1));
    for ix_r in range(0, len(data_array)-k_size+1):
        for ix_c in range(0, len(data_array)-k_size+1):
            temp_np = np.array([])
            for k_ix_r in range(ix_r, ix_r+k_size):
                for k_ix_c in range(ix_c, ix_c+k_size):
                    temp_np = np.append(temp_np, [data_array[k_ix_r, k_ix_c]])
            avg_pool_matrix[ix_r, ix_c] = np.average(temp_np)

    return avg_pool_matrix

if __name__ == "__main__":
    np.random.seed(3);
    input_data = np.random.randint(20, size=(4, 4)); print(input_data)
    input_k_size = 2; #Kernel size
    result_mat = fn_create_avg_pool(input_data, input_k_size)
    print(result_mat)
```

TEST CASE:

INPUT: 4x4 matrix, kernel size = 2x2

10	3	8	0
19	10	11	9
10	6	0	12
7	14	17	2

OUTPUT:

10.5	8	7
11.25	6.75	8
9.25	9.25	7.75

RESULT:

Average pooling was done on a given *n x n* matrix with a *m x m* kernel.