ANNAMALAI UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

B.E. Computer Science and Engineering (Data Science)

Semester IV

22DSCP410 – Data Science Lab

Name :

Reg.No.:

ANNAMALAI UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

B.E. Computer Science and Engineering (Data Science)

Semester IV

22DSCP410 – Data Science Lab

Certified that this is the bona fide record of work done by

Mr./Ms.____

Reg. No.____ of B.E. Computer Science and Engineering (Data Science) in the 22DSCP410 – Data Science Lab during the even semester of the academic year 2023–24.

Staff in-charge   Internal Examiner

Annamalainagar   External Examiner

Date:..…/…./2024

22DSCP410	**Data Science Lab**	**L**	**T**	**P**	**C**
		0	**0**	**3**	**1.5**

Course Objectives:

- To learn to implement the concepts of data science through Python programs.

- To load various kinds of data and display them in various formats for better understanding.
- To learn to collect, explore, clean, munge and manipulate data.
- To understand how statistics and probability is used in data science applications.

LIST OF EXERCISES:

1. Study of Python Data Science Environment (NumPy, SciPy, matplotLib, Pandas, Scikit-learn).
2. Operations on Python Data Structures.
3. Reading data from various sources (Text files, CSV files, Excel files, HTML/XML files, JSON files).
4. Exploring data through simple visualization tools like charts and graphs using matplotlib.
5. Data cleansing operations for handling missing data.
6. Data Wrangling (Filtering, Pivoting dataset, Melting Shifted Datasets, Merging Melted data, Concatenating data, Exporting Data).
7. Data Aggregation (Grouping, Group wise operations and transformations).
8. Data Transformations (Rescaling and Dimensionality Reduction).
9. Measuring Central Tendency, Variability and Correlation.
10. Creating, Plotting and Understanding Probability Distributions.
11. Hypothesis Testing.
12. Creating and Displaying Geographic Maps.
13. Handling Graph Data.
14. Creating and Displaying Heat Maps.

Course Outcomes:

At the end of this course the, students will be able to

1. Experiment the various data structures and libraries in Python for data science programming.
2. Conduct and present statistical measurements, hypothesis and tests on data.
3. Develop practical applications covering the concepts of Data Science.

Mapping of Course Outcomes with Programme Outcomes												
	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**
CO1	2	2	3	-	-	-	-	-	-	-	-	-
CO2	1	2	-	2	-	-	-	-	-	-	-	-
CO3	2	2	-	-	-	-	-	-	-	2	-	2

Vision-Mission of Faculty of Engineering and Technology

Vision

Providing world class quality education with strong ethical values to nurture and develop outstanding professionals fit for globally competitive environment.

Mission

- Provide quality technical education with a sound footing on basic engineering principles, technical and managerial skills, and innovative research capabilities.
- Transform the students into outstanding professionals and technocrats with strong ethical values capable of creating, developing and managing global engineering enterprises.
- Develop a Global Knowledge Hub, striving continuously in pursuit of excellence in Education, Research, Entrepreneurship and Technological services to the Industry and Society.
- Inculcate the importance and methodology of life-long learning to move forward with updated knowledge to face the challenges of tomorrow.

Vision-Mission of the Department of Computer Science and Engineering

Vision

To provide a congenial ambience for individuals to develop and blossom as academically superior, socially conscious and nationally responsible citizens.

Mission

- Impart high quality computer knowledge to the students through a dynamic scholastic environment wherein they learn to develop technical, communication and leadership skills to bloom as a versatile professional.
- Develop life-long learning ability that allows them to be adaptive and responsive to the changes in career, society, technology, and environment.
- Build student community with high ethical standards to undertake innovative research and development in thrust areas of national and international needs.
- Expose the students to the emerging technological advancements for meeting the demands of the industry.

Program Educational Objectives (PEOs)

PEO	PEO Statements
PEO1	To prepare the graduates with the potential to get employed in the right role and/or become entrepreneurs to contribute to the society.

PEO2	To provide the graduates with the requisite knowledge to pursue higher education and carry out research in the field of Computer Science.
PEO3	To equip the graduates with the skills required to stay motivated and adapt to the dynamically changing world so as to remain successful in their career.
PEO4	To train the graduates to communicate effectively, work collaboratively and exhibit high levels of professionalism and ethical responsibility.

Program Outcomes (POs)

S. NO.	Program Outcomes
PO1	**Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**Problem Analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.
PO3	**Design/Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	**Conduct Investigations of Complex Problems:** Use research-based knowledge andresearch methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	**Modern Tool Usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complexengineering activities with an understanding of the limitations.
PO6	**The Engineer and Society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	**Environment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate theknowledge of, and need for sustainable development.
PO8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and Team Work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project Management and Finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12	**Life-long Learning:** Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

Program Specific Outcomes (PSOs)

S.NO	Program Specific Outcomes
PSO1	Acquire the ability to understand basic sciences, humanity sciences, basic engineering sciences and fundamental core courses in Computer Science and Engineering to realize and appreciate real life problems in diverse fields for proficient design of computer- based systems of varying complexity.
PSO2	Learn specialized courses in Computer Science and Engineering to build up the aptitude for applying typical practices and approaches to deliver quality products intended for business and industry requirements.
PSO3	Apply technical and programming skills in Computer Science and Engineering essential for employing current techniques in software development crucial in industries, to create pioneering career paths for pursuing higher studies, research and to be an entrepreneur.

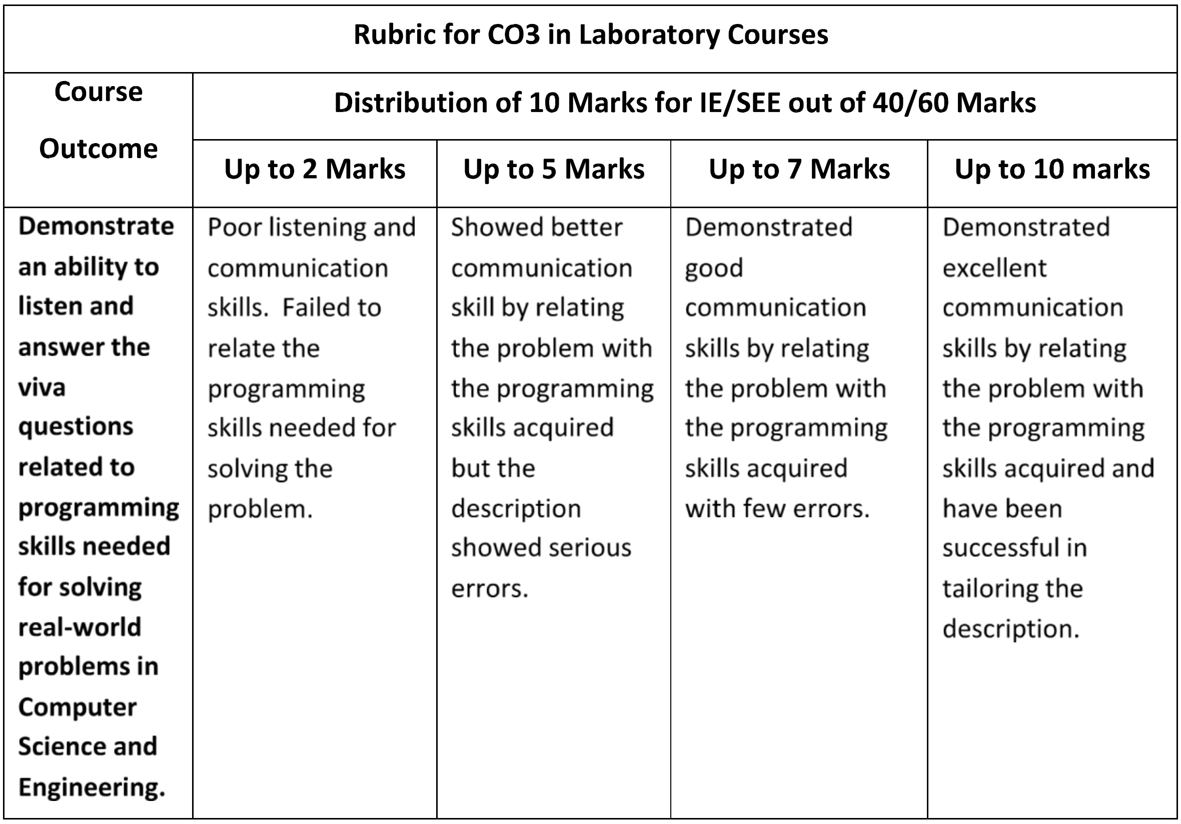
Rubrics for Laboratory Examination (Internal/External)

(Internal: Two tests - 15 marks each, **External:** Two questions - 25 marks each)

Rubric	Poor Up to (1/2)	Average Up to (2/4)	Good Up to (3/6)	Excellent Up to (5/8*)
Syntax and Logic **Ability to understand, specify the data structures appropriate for the problem domain**	Program does not compile with typographical errors and incorrect logic leading to infinite loops.	Program compiles that signals major syntactic errors and logic shows severe errors.	Program compiles with minor syntactic errors and logic is mostly correct with occasional errors.	Program compiles with evidence of good syntactic understanding of the syntax and logic used.
Modularity **Ability to decompose a problem into coherent and reusable functions, files, classes, or objects (as appropriate for the programming language and platform).**	Program is one big Function or is decomposed in ways that make little/no sense.	Program is decomposed into units of appropriate size, but they lack coherence or reusability. Program contains unnecessary repetition.	Program is decomposed into coherent units, but may still contain some unnecessary repetition.	Program is decomposed into coherent and reusable units, and unnecessary repetition are eliminated.
Clarity and Completeness **Ability to code formulae and algorithms that produce appropriate results. Ability to apply rigorous test case analysis to the problem domain.**	Program does not produce appropriate results for most inputs. Program shows little/no ability to apply different test cases.	Program approaches appropriate results for most inputs, but contain some miscalculations. Program shows evidence of test case analysis, but missing significant test cases or mistaken some test cases.	Program produces appropriate results for most inputs. Program shows evidence of test case analysis that is mostly complete, but missed to handle all possible test cases.	Program produces appropriate results for all inputs tested. Program shows evidence of excellent test case analysis, and all possible cases are handled appropriately.

* 8 marks for syntax and logic, 8 marks for modularity, and 9 marks for Clarity and Completeness.

Rubric for CO3 in Laboratory Course



Rubric for CO3 in Laboratory Courses				
Course Outcome	**Distribution of 10 Marks for IE/SEE out of 40/60 Marks**			
	Up to 2 Marks	**Up to 5 Marks**	**Up to 7 Marks**	**Up to 10 marks**
Demonstrate an ability to listen and answer the viva questions related to programming skills needed for solving real-world problems in Computer Science and Engineering.	Poor listening and communication skills. Failed to relate the programming skills needed for solving the problem.	Showed better communication skill by relating the problem with the programming skills acquired but the description showed serious errors.	Demonstrated good communication skills by relating the problem with the programming skills acquired with few errors.	Demonstrated excellent communication skills by relating the problem with the programming skills acquired and have been successful in tailoring the description.

CONTENTS

Average:

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 where we study 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 lot of skills and expertise in several domains such as statistics, machine learning, and programming. Data Science projects require a knowledge of Python Programming and packages such as NumPy, SciPy, Pandas and matplotlib.

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.

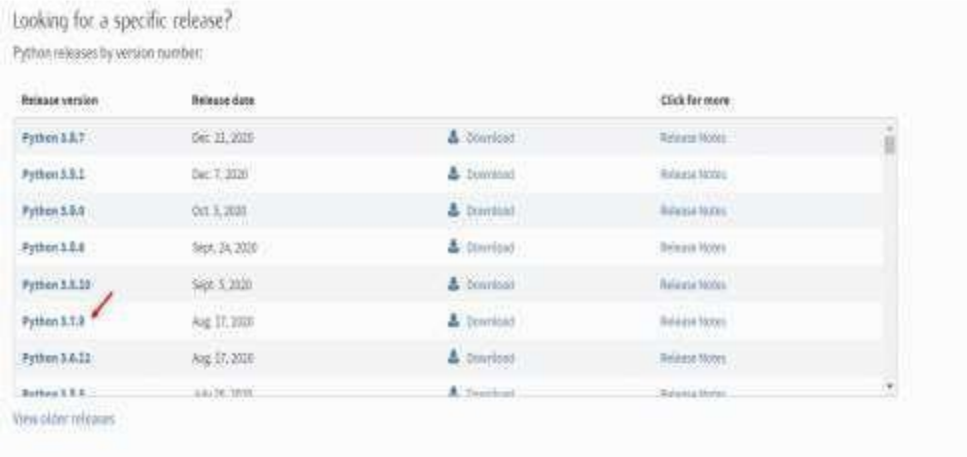
Sections that are indicated as optional are marked with **[Optional]**. Though optional, students are strongly encouraged to try out these sections.

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:
    a. Navigate to https://www.python.org/downloads/
    b. Scroll down to "Looking for a specific release?" section and click on Python 3.7.9 as shown below:



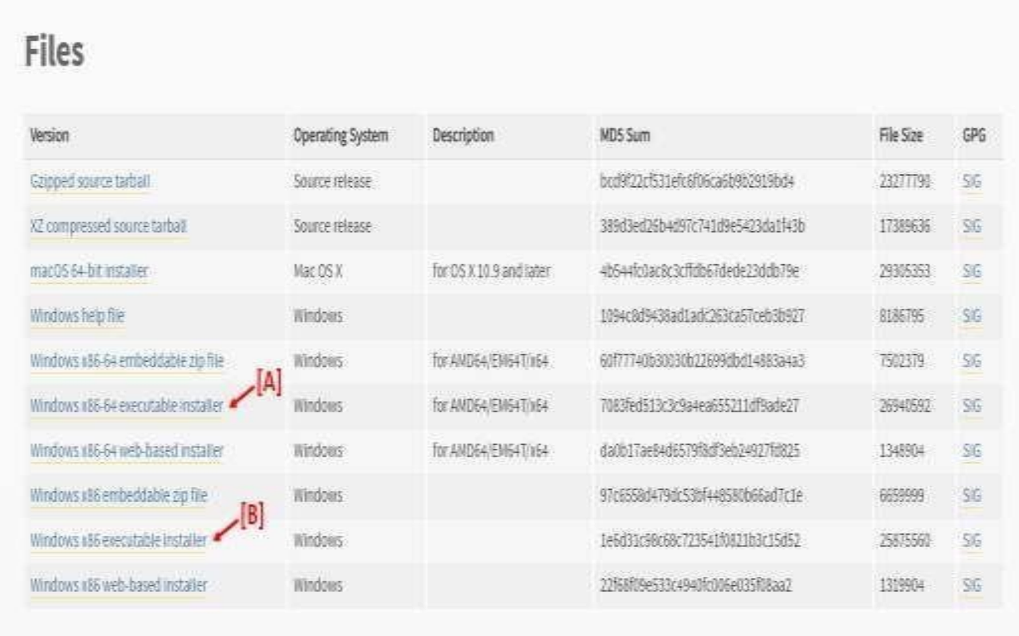
Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.8.7	Dec. 21, 2020	Download	Release Notes
Python 3.9.1	Dec. 7, 2020	Download	Release Notes
Python 3.9.0	Oct. 5, 2020	Download	Release Notes
Python 3.8.6	Sept. 24, 2020	Download	Release Notes
Python 3.5.10	Sept. 5, 2020	Download	Release Notes
Python 3.7.9	Aug. 17, 2020	Download	Release Notes
Python 3.6.12	Aug. 17, 2020	Download	Release Notes

View older releases

    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.



Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		bcd9f22cf531efc6f06ca6b9b2919bd4	23277790	SIG
XZ compressed source tarball	Source release		389d3ed26b4d97c741d9e5423da1f43b	17389636	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	4b544fc0ac8c3cffdb67dede23ddb79e	29305353	SIG
Windows help file	Windows		1094c8d9438ad1adc263ca57ceb3b927	8186795	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	60f77740b30030b22699dbd14883a4a3	7502379	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	7083fed513c3c9a4ea655211df9ade27	26940592	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	da0b17ae84d6579f8df3eb24927fd825	1348904	SIG
Windows x86 embeddable zip file	Windows		97c6558d479dc53bf448580b66ad7c1e	6659999	SIG
Windows x86 executable installer	Windows		1e6d31c98c68c723541f0821b3c15d52	25875560	SIG
Windows x86 web-based installer	Windows		22f68f09e533c4940fc006e035f08aa2	1319904	SIG

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:

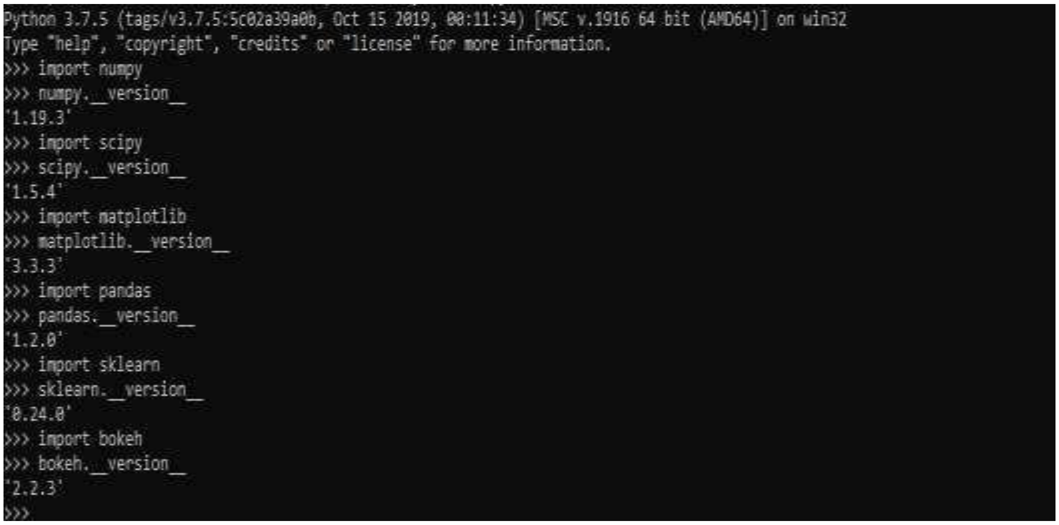
1. In Command Prompt, type “jupyter notebook” and hit Enter key.
2. By default a web browser tab with jupyter notebook will open. If not, type in the following URL to open - http://localhost:8888/tree
3. Do not close this Command Prompt opened in Step 1.
4. Click on New -> Python 3 (right top) to open 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:



```
Python 3.7.5 (tags/v3.7.5:5c02a39a0b, Oct 15 2019, 00:11:34) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> numpy.__version__
'1.19.3'
>>> import scipy
>>> scipy.__version__
'1.5.4'
>>> import matplotlib
>>> matplotlib.__version__
'3.3.3'
>>> import pandas
>>> pandas.__version__
'1.2.0'
>>> import sklearn
>>> sklearn.__version__
'0.24.0'
>>> import bokeh
>>> bokeh.__version__
'2.2.3'
>>>
```

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 anagram(s1, s2):
    str1 = sorted(s1.lower())
    str2 = sorted(s2.lower())
    if str1 == str2:
        return True
    else:
        return False

if __name__ == "__main__":
    s1 = "Binary"
    s2 = "Brainy"
    print(anagram(s1, s2))
```

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 clean_str(s, rem_list):
    s = s.lower()
    for ch in rem_list:
        s = s.replace(ch, "")
    return s

def word_freq(s):
    words = s.split()
    counts = []
    for w in words:
        counts.append(words.count(w))
    return dict(zip(words, counts))

def show_count(w, freq_dict):
    w = w.lower()
    if w in freq_dict:
        return freq_dict[w]
    else:
        return 0

if __name__ == "__main__":
    inp = "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."
    rem = [".", ",", "?"]
    clean = clean_str(inp, rem)
    freq = word_freq(clean)
    test = "Shells"
    print(show_count(test, freq))
```

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 r, o: round(r/o, 2)


def get_data():
    data = [
        ("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


def check(p, data):
    w, eco = None, None
    for d in data:
        if p in d:
            w = d[4]
            eco = E(d[3], d[1])
    if w is not None:
        return f"{p} picked up {w} wickets at an Economy of {eco} RPO"
    else:
        return f"{p} did not bowl in this match"


if __name__ == "__main__":
    d = get_data()
    p = "Yuvraj Singh"
    res = check(p, d)
    print(res)
```

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 uniq(x):
   return list(set(x))

def comm(x, y):
   return list(set(x) & set(y))

if __name__ == "__main__":
   l1 = [11, 22, 33, 44, 33, 22, 1]
   l2 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
   print(uniq(l1))
   print(comm(l1, l2))
```

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 mat_sum(a, b):

   if a.shape == b.shape:

      return a + b

   else:

      return None


def mat_diff(a, b):

   if a.shape == b.shape:

      return a - b

   else:

      return None


def mat_mul(a, b):

   if a.shape[1] == b.shape[0]:

      return np.dot(a, b)

   else:

      return None
```

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


if __name__ == "__main__":
    np.random.seed(3)
    a = np.random.randint(1, 20, (3, 3)); print("Matrix A:\n", a)
    b = np.random.randint(1, 20, (3, 3)); print("Matrix B:\n", b)
    c = np.random.randint(1, 20, (5, 5)); print("Matrix C:\n", c)


    print("Sum:\n", mat_sum(a, b))
    print("Difference:\n", mat_diff(a, b))
    print("Multiplication:\n", mat_mul(a, b))
    print("Subset of C:\n", subset(c, 1, 1, 3, 3))
```

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, numpy as np

d = {'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.','Pr
iya.N','Mani.R.','Ravi.S'],
    'A8':[92,100,100,100,76,96,100,92,68,52,72,80],
    'A9':[84,95,90,100,42,84,95,100,53,16,53,np.nan],
    'A10':[100,100,94,100,31,81,100,100,94,13,88,6],
    'Asg':[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,'-']}

df = pd.DataFrame(d)

print('Missing values:\n', df.isnull().sum())
```

```
df['A9'] = df['A9'].fillna(0)
df['Asg'] = df['Asg'].fillna(df['Asg'].min())
df = df.replace('-',0)
print(df)

r1 = df[df['A8']>=80]
r2 = df[(df['A8']<80)&(df['A8']>=70)]
r3 = df[df['A8']<70]
print('Above 80:\n',r1)
print('70 to 80:\n',r2)
print('Below 70:\n',r3)

srt = df.sort_values(by='A9', ascending=False)
print('Sorted Sept Attendance:\n')
display(srt[['RollNo.','Name','A9']])

df['ConsAttend'] = (df['A8']+df['A9']+df['A10'])/3
print('Consolidated Attendance:\n',df)

print('Students with max Test marks:\n')
display(df[df['Test']==df['Test'].max()])

mean_asg = df['Asg'].mean()
print('Students with Asg < Avg:\n')
display(df[df['Asg']<mean_asg])

df['Res'] = df['Asg'] + df['Test']
df['Res'] = df['Res'].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	Res lt
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.

AIM:

To

5

DATA CLEANING AND PROCESSING IN CSV FILES

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:

```
import pandas as pd


def calc_gf(df):
    if df.isnull().values.any():
        print("Detected NaN, replacing with 0")
        df.fillna(0)
    else:
        df.drop(columns=['G3'], inplace=True)
        df.insert(len(df.columns), 'Gfinal', '')
        df['Gfinal'] = (df['G1'] + df['G2']) * 100 / 40
    pass_50 = df[df['Gfinal'] >= 50]
    below_50 = df[df['Gfinal'] < 50]
    return pass_50, below_50


if __name__ == "__main__":
    df = pd.read_csv("student-mat.csv", delimiter=";")
    pass_50_df, below_50_df = calc_gf(df)
```

Ex. No.

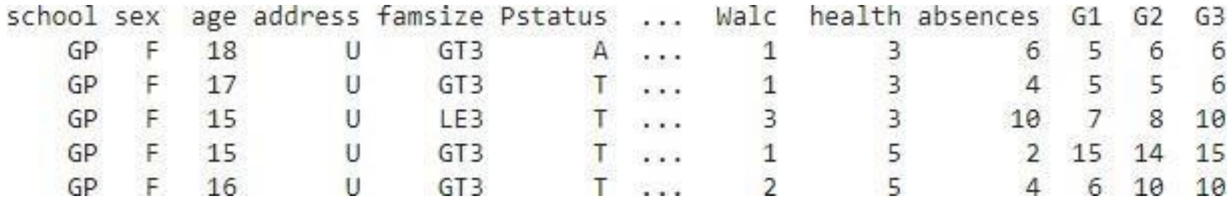
AIM:

To

```
pass_50_df.to_csv("result_50plus.csv",sep=',',
index=False)

below_50_df.to_csv("result_below50.csv",sep=',',
index=False)
```

TEST CASE:

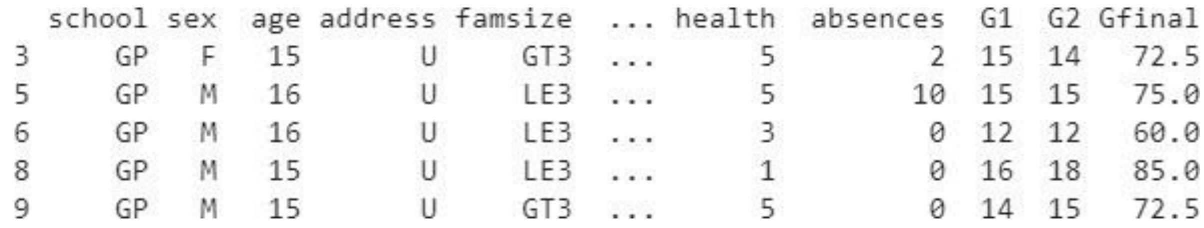
INPUT:



```
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
```

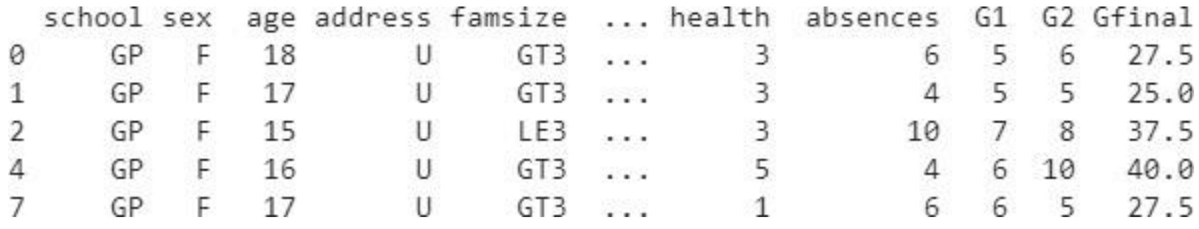
OUTPUT:

Gfinal >= 50



```
  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



```
  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.

6

HANDLING CSV FILES

read from and write onto CSV files using Pandas package.

PROBLEM DEFINITION:

Perform 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 as dt

def get_hl(df):
   df.drop(df.columns[-1], axis=1, inplace=True)
   df["Date"] = pd.to_datetime(df["Date"], format='%d-%B-%Y')
   df.fillna(0, inplace=True)
   s = dt.datetime.strptime('2018-03-31', '%Y-%m-%d')
   e = dt.datetime.strptime('2019-04-01', '%Y-%m-%d')
   df_fy = df[(df["Date"] > s) & (df["Date"] < e)]
   hi = df_fy["High"].max()
   lo = df_fy["Low"].min()
   return hi, lo, df_fy

if __name__ == "__main__":
   df = pd.read_csv("csv_base_sensex_2018to2020.csv")
   hi, lo, df_fy = get_hl(df)
   df_fy.to_csv("sensex_fy2019-20.csv", index=False)
   print("SENSEX High & Low in FY2019-20:", hi, "&", lo)
```

TEST CASE:

INPUT: -- (S&P BSE Sensex data – daily data for the period 2018 to 2020)

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.

AIM:

To

AIM:

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 information available in IRNSS Wikipedia webpage.

CODE:

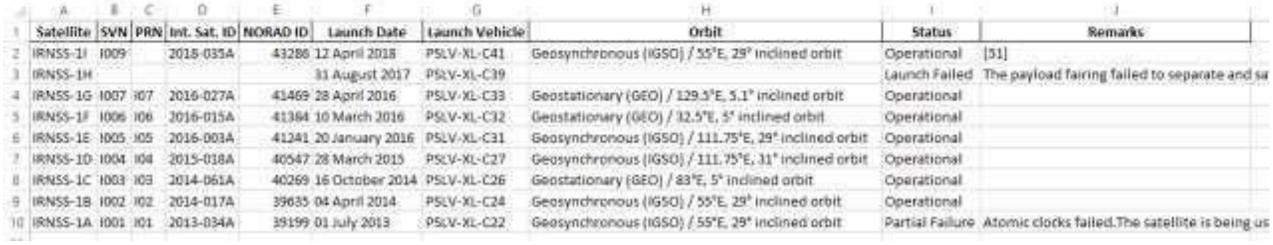
```
import pandas as pd


def get_irnss_data(url, table):
   data = pd.read_html(url, match=table)
   df = data[0]
   df_sub = df[~df['Status'].str.contains('Planned')]
   df_sub['Launch Date'] = pd.to_datetime(df_sub['Launch Date'], format='%d %B %Y')
   df_sub = df_sub.sort_values(by='Launch Date', ascending=False)
   df_sub['Launch Date'] = df_sub['Launch Date'].apply(lambda x: x.strftime('%d %B %Y'))
   return df_sub


if __name__ == "__main__":
   url = "https://en.wikipedia.org/wiki/Indian_Regional_Navigation_Satellite_System"
   table = "IRNSS-1 series satellites"
   result_df = get_irnss_data(url, table)
   result_df.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:



	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	[51]
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 thus handled.

Ex. No.

AIM:

To
8

PROCESSING TEXT FILES

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:

```
def read_process(file_name):
  words = []
  with open(file_name, "rt") as f:
    words = [word for line in f for word in line.split()]

  words = [w.lower() for w in words]
  char_clean = '''!;:'"\, ./?@#$%^&*_~'''
  clean_words = []

  for word in words:
    for char in word:
      if char in char_clean:
        word = word.replace(char, "")
    clean_words.append(word)

  return clean_words

def count_freq(words, word_to_count):
  return words.count(word_to_count.lower())


if __name__ == "__main__":
  word_list = read_process("TxtSample.txt")
  print(count_freq(word_list, "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.

AIM:

To

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:

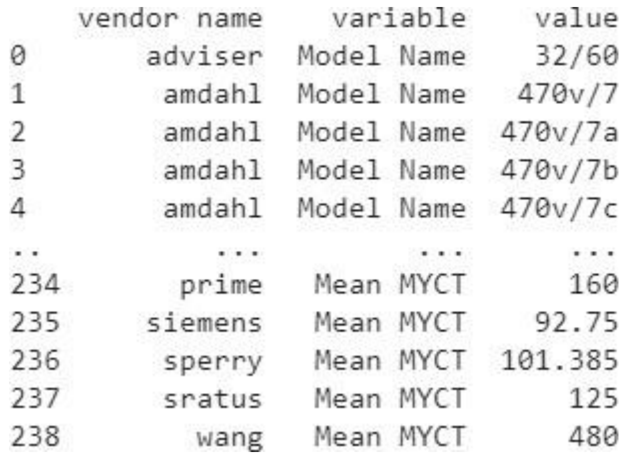
```
# Data Source
# Title: Computer Hardware Data Set
# Hosted Link : https://archive.ics.uci.edu/ml/datasets/Computer+Hardware
# Download Link: https://archive.ics.uci.edu/ml/machine-learning-databases/cpu-performance/

# 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 get_model(df):

    m_mean = pd.pivot_table(df, values=["MYCT", "MMIN", "MMAX", "CACH", "CHMIN", "CHMAX", "PRP"], columns="vendor name", aggfunc=np.mean)

    m_median = pd.pivot_table(df, values=["MYCT", "MMIN", "MMAX", "CACH", "CHMIN", "CHMAX", "PRP"], columns="vendor name", aggfunc=np.median)


    df_mean_myct = pd.DataFrame({"vendor name": list(m_mean.columns), "Mean MYCT": m_mean.values.tolist()[5]})
    m_models = pd.melt(df, id_vars=["vendor name"], value_vars=["Model Name"])
    m_myct = pd.melt(df_mean_myct, id_vars=["vendor name"], value_vars=["Mean MYCT"])


    result = pd.concat([m_models, m_myct], ignore_index=True)
    return result


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

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.

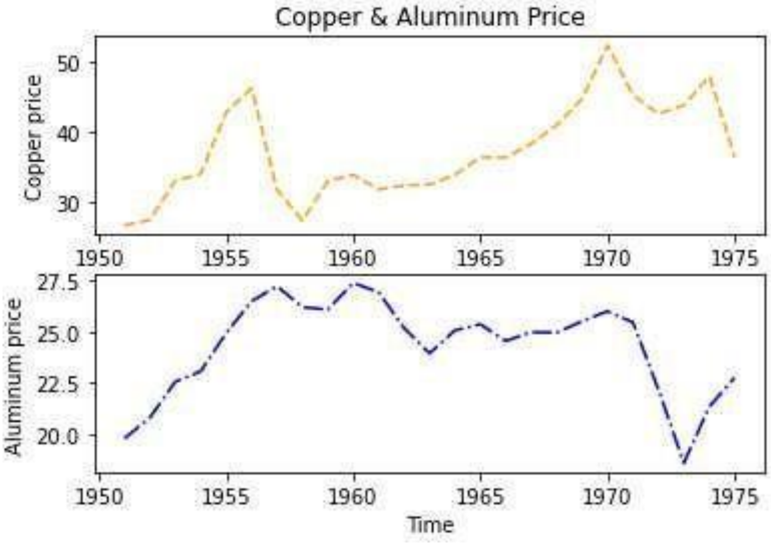
CODE:

```
# https://www.statsmodels.org/devel/datasets/index.html

# https://github.com/statsmodels/statsmodels/tree/master/statsmodels/datasets
import statsmodels.api as sm
import matplotlib.pyplot as plt

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

fig = plt.figure(figsize=(10, 5))
ax1 = plt.subplot(2, 1, 1)
ax2 = plt.subplot(2, 1, 2)

x = range(1951, 1975 + 1)
ax1.plot(x, df["COPPERPRICE"], color='orange', ls='--')
ax1.set(xlabel='Time', ylabel='Copper price', title="Copper & Aluminum Price")

ax2.plot(x, df["ALUMPRICE"], color='blue', ls='-.')
ax2.set(xlabel='Time', ylabel='Aluminum price')

plt.show()
```

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:

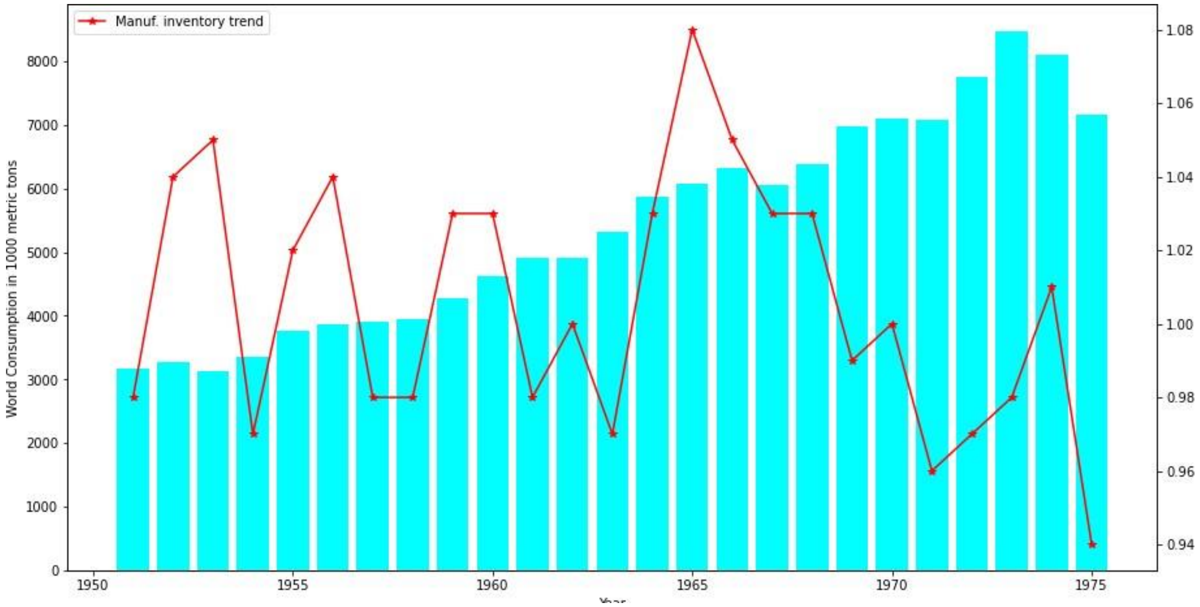
```
import statsmodels.api as sm

import matplotlib.pyplot as plt


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

x = range(1951, 1976)

y1 = df["WORLDCONSUMPTION"].values

y2 = df["INVENTORYINDEX"].values


fig, 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 (1000 metric tons)')
```

```
ax2.plot(x, y2, 'r-*', label="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.

11

DISPLAY DATA IN GEOGRAPHICAL MAP

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

PROBLEM DEFINITION:

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

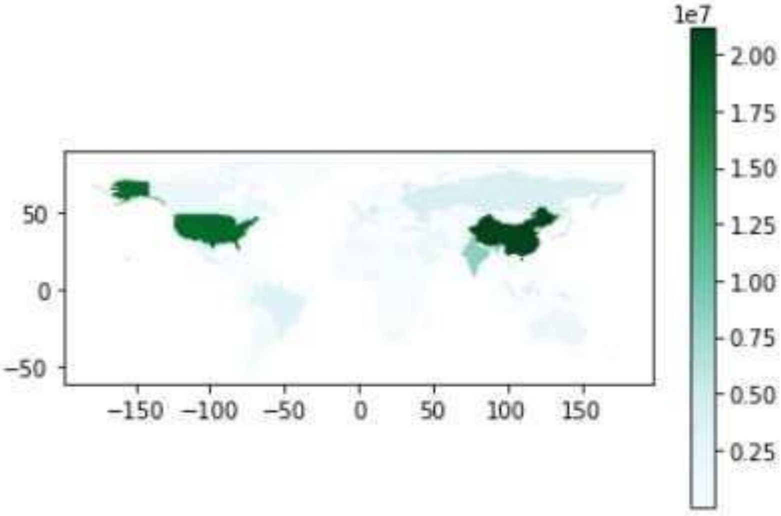
CODE:

```
# 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.

12

DISPLAY DATA IN HEATMAP

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.

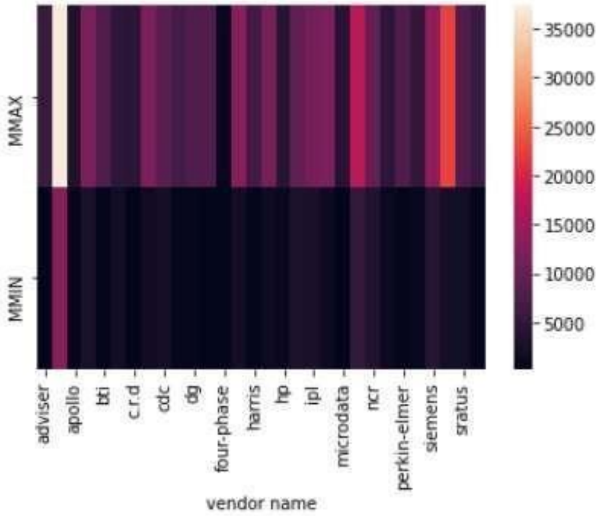
CODE:

```
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: --

OUTPUT:



RESULT:

Data was displayed in the form of heatmap.

13

NORMAL AND CUMULATIVE DISTRIBUTION

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 as plt
from scipy.stats import
norm

def fn_pdf():
    h = np.linspace(150,
180, 100)
    plt.hist(h, 12)
    plt.show()
    mean_h = np.mean(h)
    stdev_h = np.std(h)
    pdf_h = norm.pdf(h,
mean_h, stdev_h)
    fig, ax = plt.subplots()
    ax.set_xlabel('Adult
Height')

ax.set_ylabel('Probabilitie
s')
    plt.plot(h, pdf_h)
    plt.show()
    return [mean_h,
stdev_h, pdf_h]

def fn_test(td, params):
    mean_h = params[0]
    stdev_h = params[1]
    pdf_h = params[2]
    pdf_td = norm.pdf(td,
mean_h, stdev_h)
    min_pdf_h = min(pdf_h)
    max_pdf_h =
max(pdf_h)
    if pdf_td >= min_pdf_h
and pdf_td <= max_pdf_h:
        return 'Adult height'
    else:
        return 'Not adult
height'

if __name__ ==
"__main__":
    params = fn_pdf()
    result = fn_test(170,
params)
    print(result):
```

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 as plt
from scipy.stats import norm

# Function to create a normal distribution model
def create_pdf():
   h = np.linspace(150, 180, 100)
   mean = np.mean(h)
   std = np.std(h)
   pdf = norm.pdf(h, mean, std)
   params = [mean, std]
   return params
```

```
def test_prob(x1, x2, params):
    mean, std = params
    p1 = norm(loc=mean, scale=std).cdf(x1)
    p2 = norm(loc=mean, scale=std).cdf(x2) - norm(loc=mean, scale=std).cdf(x1)
    p3 = 1 - norm(loc=mean, scale=std).cdf(x2)
    return [p1, p2, p3]


if __name__ == "__main__":
    params = create_pdf()
    x1 = 160
    x2 = 170
    res = test_prob(x1, x2, params)
    print('Probability of height under 160cm =', res[0])
    print('Probability height between 160 and 170cm =', res[1])
    print('Probability height above 170cm =', res[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 hypothesised population mean.

# The One Sample t Test is a parametric test.

```
from scipy.stats import ttest_1samp

import numpy as np


def t_test(val):

   h = np.array([165, 170, 160, 154, 175, 155, 167, 177, 158, 178])

   print(h)

   mean_h = np.mean(h)

   print('Mean Height =', mean_h)

   t_stat, p = ttest_1samp(h, val)

   print('p-value =', p)


   if p < 0.05:

      res = 'Reject null hypothesis'

   else:

      res = 'Accept null hypothesis'

   return res


if __name__ == "__main__":

   val = 170

   res = t_test(val)
```

```
print(res)
```

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.