**ANNAMALAI UNIVERSITY DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

# B. E. COMPUTER SCIENCE & ENGINEERING (DATA SCIENCE) SEMESTER – IV

**21DSCP409. DATA SCIENCE LAB**

**LABORATORY MANUAL**

**(JANUARY 2023 – APRIL 2023)**

**LAB INCHARGE:**

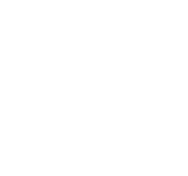
**Dr. R. RAGUPATHY, Associate Professor, Dept. of CSE, A.U**

## 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

1. PROCESSING TEXT FILES
2. DATA WRANGLING (PIVOT TABLE, MELT, CONCAT)
3. GENERATING LINE CHART AND BAR GRAPH USING MATPLOTLIB
4. DISPLAY DATA IN GEOGRAPHICAL MAP
5. DISPLAY DATA IN HEATMAP
6. NORMAL AND CUMULATIVE DISTRIBUTION
7. 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

**AIM:**

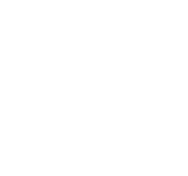
## STUDY OF PYTHON DATA SCIENCE ENVIRONMENT

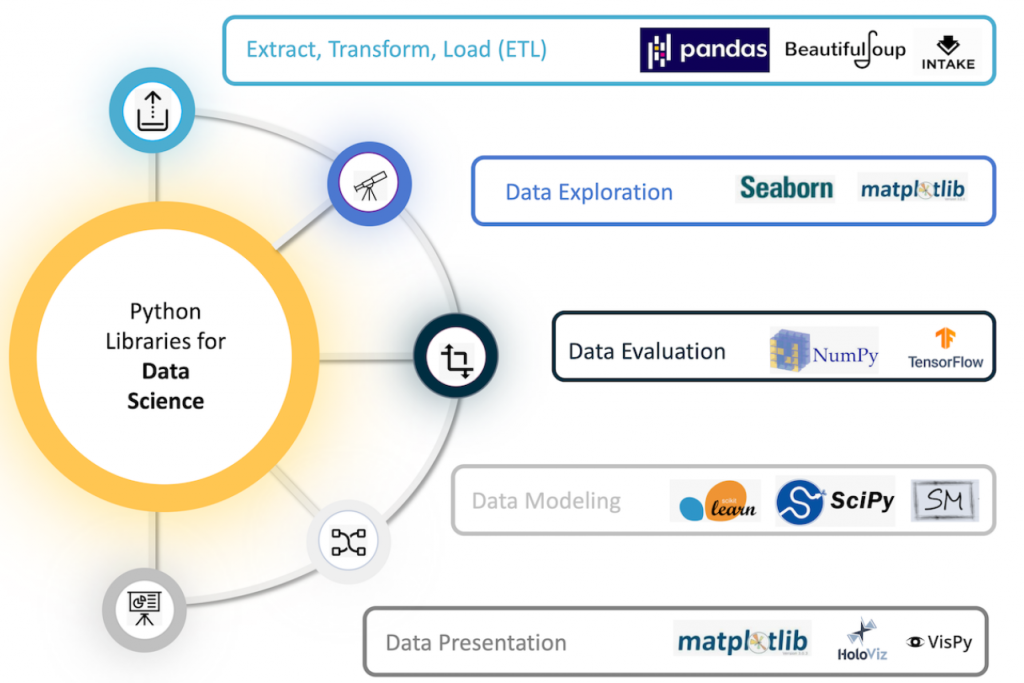
To study the Python Data Science Environment (NumPy, SciPy, Pandas, Matplotlib).

## PROBLEM DEFINITION:

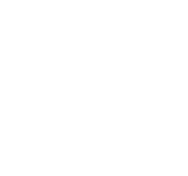
Study the features of Python, packages required for data science operations and their installation procedure required for Data Science programming.

1. **PYTHON FOR DATA SCIENCE**

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 knowledge of the following software:

.

## 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:
3. In Command Prompt, type “jupyter notebook” and hit Enter key.
4. By default a web browser tab with jupyter notebook will open. If not, type in the following URL to open - http://localhost:8888/tree
5. Do not close this Command Prompt opened in Step 1.
6. Click on New -> Python 3 (right top) to open a new Notebook.
7. 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

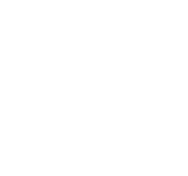
We will install the following packages: numpy, scipy, matplotlib, pandas, scikit-learn (sklearn), bokeh.

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

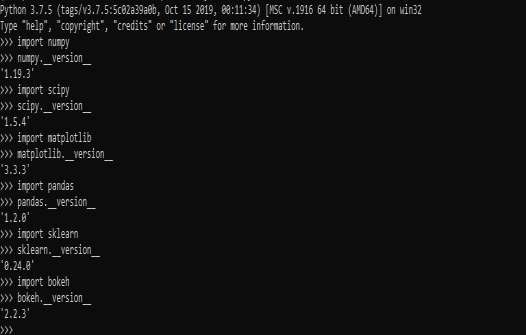
\*\*If one encounters issue with installing/using numpy, try “pip install numpy==1.19.3”

1. Type in “pip install scipy matplotlib pandas sklearn bokeh” and hit Enter key.
2. To verify installation:
   1. Type in “python”, hit enter.
   2. Type in

import <package\_name>

<package\_name>. version

* 1. 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

**AIM:**

## OPERATIONS ON PYTHON DATA STRUCTURES

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

## 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

## 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 |

## 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-](http://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

* 1. How many wickets did the bowler pick?
  2. 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

## SET, LIST PROBLEM DEFINITION:

Generate a python program to do the following using SET operations:

1. To return a list without duplicates
2. 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:

1. Duplicate Removal

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

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

1. 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

**AIM:**

## ARRAY OPERATIONS USING NUMPY

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

**AIM:**

## OPERATIONS ON PANDAS DATAFRAME

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:

1. Check for missing values,
2. Fill missing values in Attend9 with 0,
3. Fill missing values with minimum value in Assignment,
4. Replace by 0 in Test,
5. Select rows based on conditions >=80, <80 and >=70, <70 for August Attendance,
6. Arrange and display students in decreasing order of September attendance,
7. Find students with 100% attendance for all three months together and include/display consolidated attendance as last column,
8. Display the details of students who scored maximum marks in test,
9. Display the details of students whose Assignment marks is less than Average of Assignment marks, and
10. 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  dtype: int64 | | 0 |
| 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  RollNo. | | Attendance  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 |

**lt**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **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 Resu** | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | **Attendance** |  |
| **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

**AIM:**

## DATA CLEANING AND PROCESSING IN CSV FILES

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

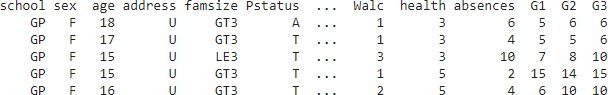
if name == " main ":

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)

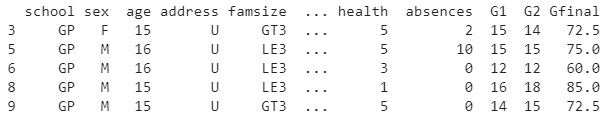
## TEST CASE:

**INPUT:** *student-mat.csv*

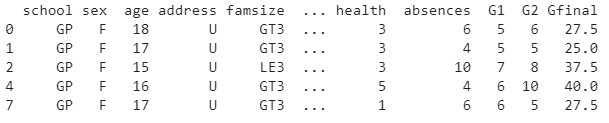


## OUTPUT:

**Gfinal >= 50 (***result\_50plus.csv***)**



**Gfinal < 50 (***result\_below50.csv***)**



## RESULT:

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

## Ex. No. 6

**AIM**:

## HANDLING CSV FILES

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](http://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

**AIM:**

## HANDLING HTML AND EXCEL FILES

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"

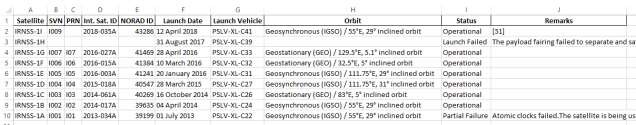
df\_out = fn\_irnss\_df(target\_URL, target\_table) df\_out.to\_excel(r'result.xlsx', sheet\_name='IRNSS Launch', index = False)

## 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***)**



## RESULT:

HTML and Excel files were handled using Pandas package..

## Ex. No. 8

**AIM:**

## PROCESSING TEXT FILES

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

**AIM:**

## DATA WRANGLING (PIVOT TABLE, MELT, CONCAT)

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 ":

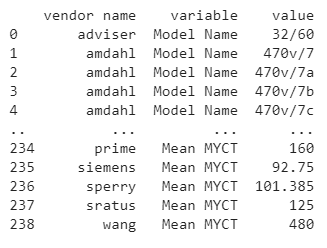
data\_frame\_ip = pd.read\_csv("machine.data", index\_col=None, header=None, names=["vendor name", "Model Name", "MYCT", "MMIN", "MMAX", "CACH", "CHMIN", "CHMAX", "PRP", "ERP"])

data\_model\_myct = fn\_get\_model\_myct(data\_frame\_ip) print(data\_model\_myct)

## TEST CASE:

**INPUT:** -- (preloaded machine dataset)

## OUTPUT:



**RESULT:**

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

## Ex. No. 10

**AIM:**

## GENERATING LINE CHART AND BAR GRAPH USING MATPLOTLIB

To use Matplotlib for plotting line chart and bar graph.

## 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](http://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

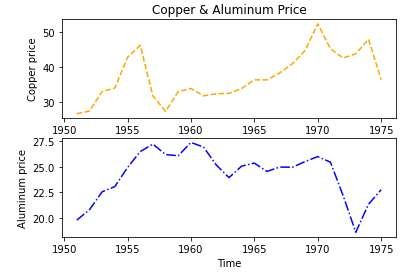
ax2.plot(ax2\_x, ax2\_y, color='blue', ls='-.')

# 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:



1. **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')

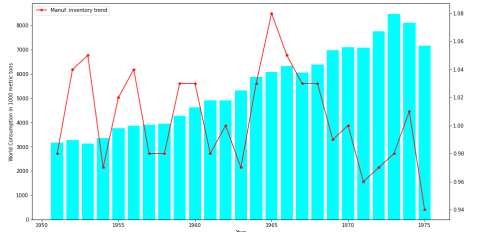
ax1.set\_ylabel('World Consumption in 1000 metric tons') ax2.plot(x, y2, 'r-\*', label = "Manuf. inventory trend", zorder=1) ax2.legend(loc="upper left")

plt.show()

## TEST CASE:

**INPUT:** -- (built-in dataset)

## OUTPUT:



**RESULT:**

Line Chart and Bar Graph was generated using Matplotlib.

## Ex. No. 11

**AIM**:

## 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:

# 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")]

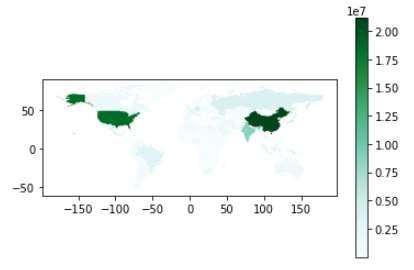
fig, ax = plt.subplots(1, 1)

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

**AIM**:

## 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

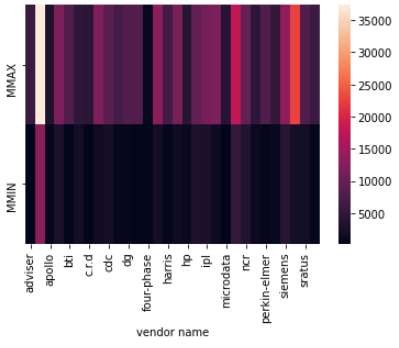
= np.mean)

h\_map = sns.heatmap(df\_mean\_sub, annot=False) plt.show()

## TEST CASE:

**INPUT**: (*machine.data*)

## OUTPUT:



**RESULT:**

Data was displayed in the form of heatmap.

## Ex. No. 13

**AIM**:

## NORMAL AND CUMULATIVE DISTRIBUTION

To implement normal and cumulative distribution models using SciPy package.

## 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

## 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

* 1. less than 160 cm,
  2. between 160 and 170 cm, and
  3. 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

**AIM**:

## HYPOTHESIS TESTING

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.

## Ex. No. 1

**AIM:**

# ADDITIONAL EXERCISES

## GENERATION OF FACTOR PAIRS OF A GIVEN INTEGER

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

**AIM:**

## AVERAGE POOLING ON A GIVEN NXN MATRIX WITH A MXM KERNEL

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

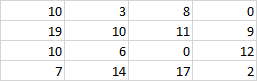
if name == " main ": np.random.seed(3);

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



## OUTPUT:



**RESULT:**

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