SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

FACULTY OF SCIENCE AND HUMANITIES

DEPARTMENT OF COMPUTER APPLICATIONS

PRACTICAL RECORD NOTE

STUDENT NAME :

REGISTER NUMBER :

CLASS : **Section:**

YEAR & SEMESTER :

SUBJECT CODE :

SUBJECT TITLE :

NOVERMBER2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

FACULTY OF SCIENCE AND HUMANITIES

DEPARTMENT OF COMPUTER APPLICATIONS

SRM Nagar, Kattankulathur- 603 203

CERTIFICATE

Certified to be the bonafide record of practical work done by

---

Register No: _____ of _____ Degree course

for_____ in the computer lab

in SRM Institute of Science and Technology during the academic year 2024-2025.

Staff In-charge ***Head of the Department***

Submitted for Semester Practical Examination held on_____.

Internal Examiner-1 ***Internal Examiner-2***

INDEX

Date: 24-06-2024

Lab 1: Getting started with Jupyter Note Book

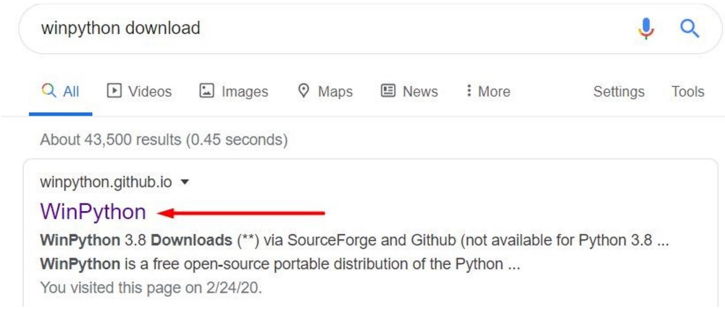
Objective:

In this lab we are going to install softwares used for python programming. WinPython is a free open-source portable distribution of the Python programming language for Windows 8/10 and scientific and educational usage. Project Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages.

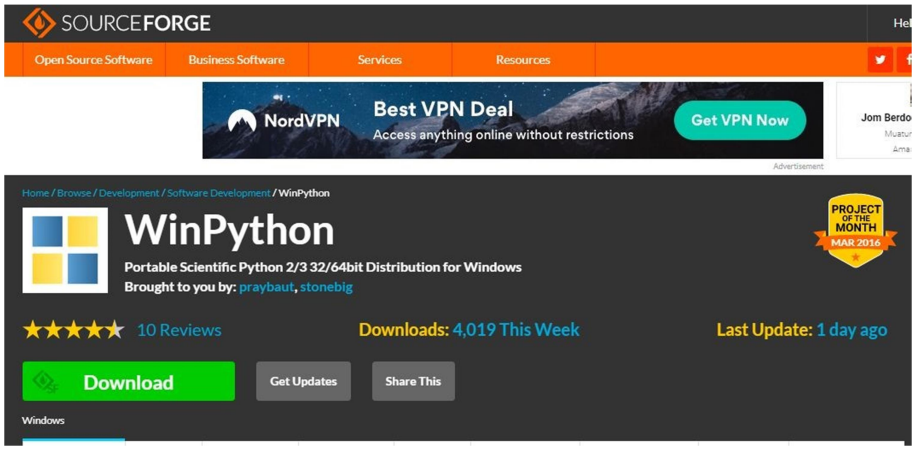
Steps:

Install WinPython

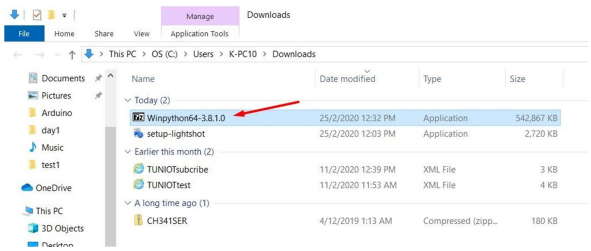
1. **Search "WinPython download" in your browser**
   https://sourceforge.net/projects/winpython/

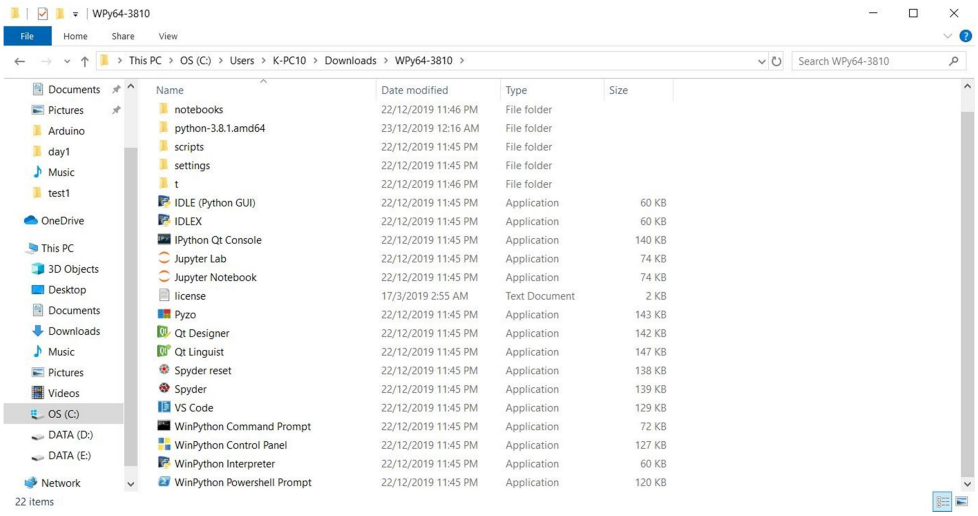


2. **Click at downloads, and it will starts shortly**



3. **Extract the downloaded file**

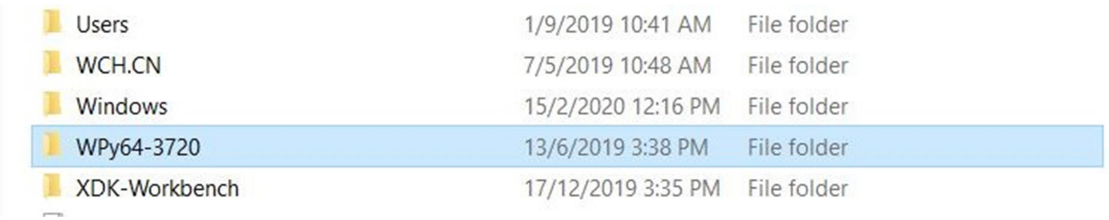




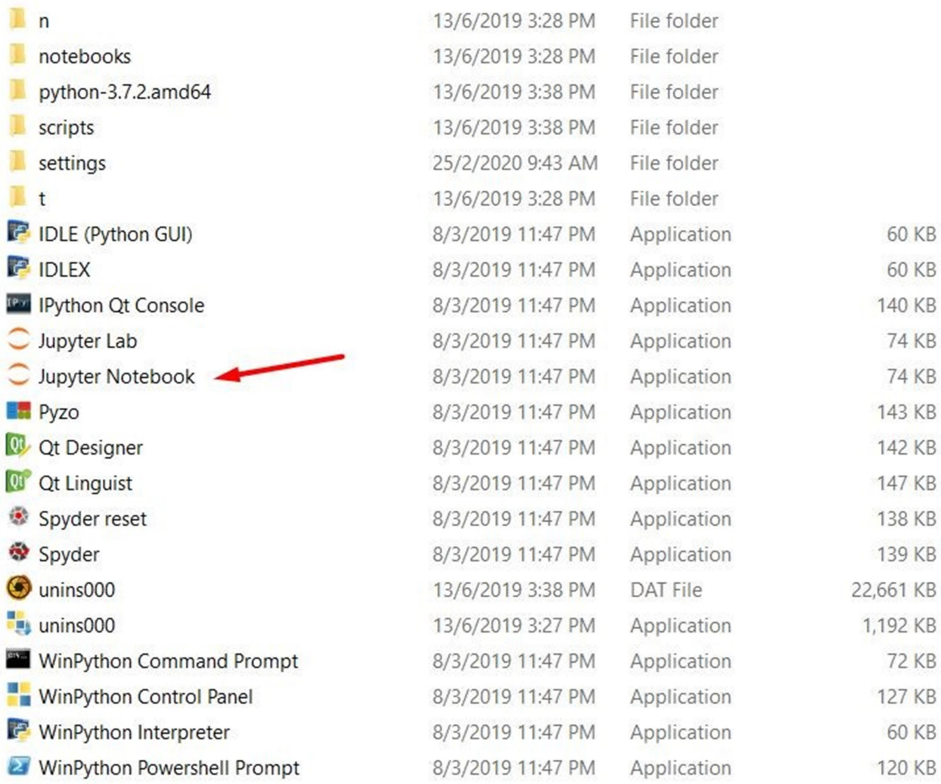
4. **Application inside the folder**

Use Jupyter Notebook

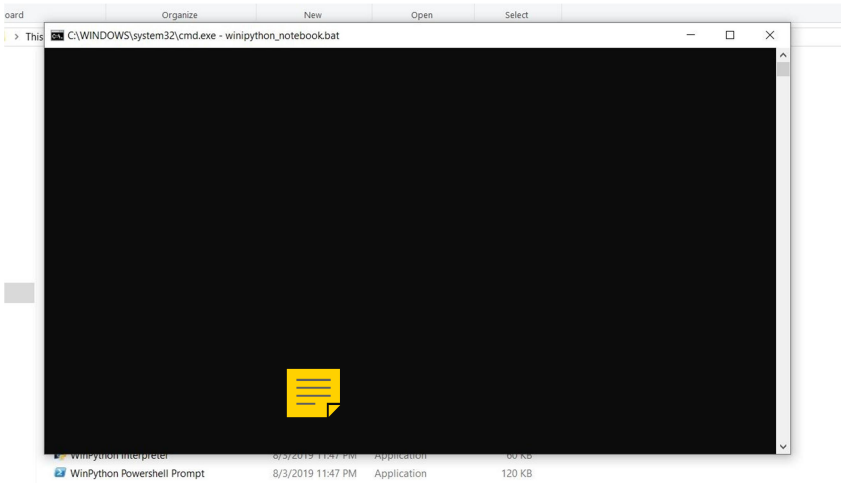
1. **Open winPython folder**



2. **Start jupyter notebook**



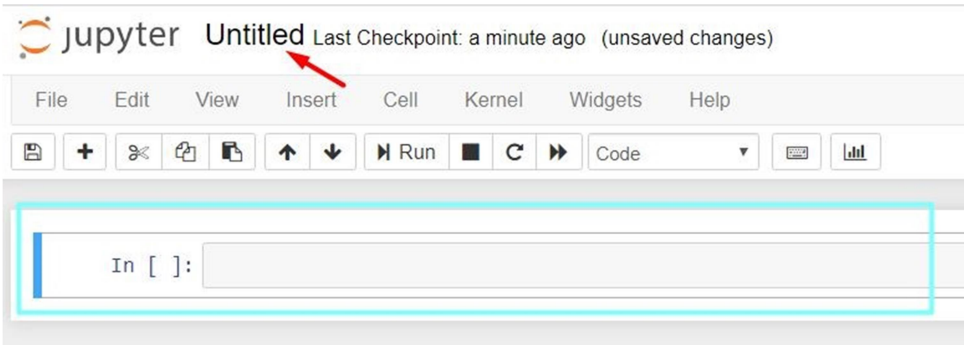
n	13/6/2019 3:28 PM	File folder	
notebooks	13/6/2019 3:28 PM	File folder	
python-3.7.2.amd64	13/6/2019 3:38 PM	File folder	
scripts	13/6/2019 3:38 PM	File folder	
settings	25/2/2020 9:43 AM	File folder	
t	13/6/2019 3:28 PM	File folder	
IDLE (Python GUI)	8/3/2019 11:47 PM	Application	60 KB
IDLEX	8/3/2019 11:47 PM	Application	60 KB
IPython Qt Console	8/3/2019 11:47 PM	Application	140 KB
Jupyter Lab	8/3/2019 11:47 PM	Application	74 KB
Jupyter Notebook	8/3/2019 11:47 PM	Application	74 KB
Pyzo	8/3/2019 11:47 PM	Application	143 KB
Qt Designer	8/3/2019 11:47 PM	Application	142 KB
Qt Linguist	8/3/2019 11:47 PM	Application	147 KB
Spyder reset	8/3/2019 11:47 PM	Application	138 KB
Spyder	8/3/2019 11:47 PM	Application	139 KB
unins000	13/6/2019 3:38 PM	DAT File	22,661 KB
unins000	13/6/2019 3:27 PM	Application	1,192 KB
WinPython Command Prompt	8/3/2019 11:47 PM	Application	72 KB
WinPython Control Panel	8/3/2019 11:47 PM	Application	127 KB
WinPython Interpreter	8/3/2019 11:47 PM	Application	60 KB
WinPython Powershell Prompt	8/3/2019 11:47 PM	Application	120 KB



3. **Open web browser and search localhost:8888/tree**

4. **Create new python 3**

5. **Change the title and start code**



Date: 01-07-2024

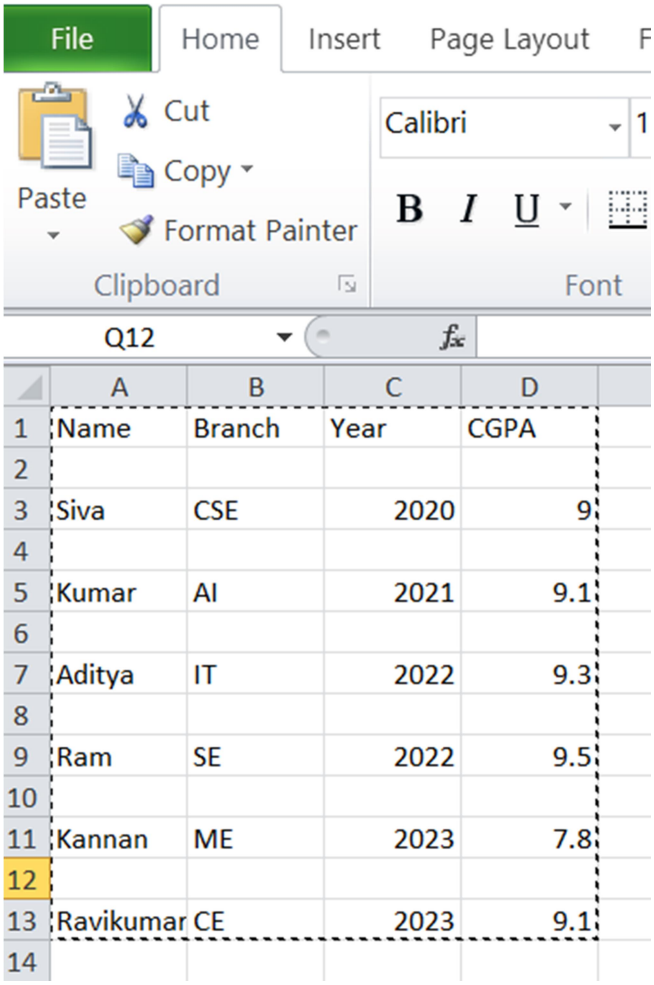
Lab: 2 write a python program to read and write student information with CSV files.

Aim: To read and write student information with CSV file.

Source code: Write the student information in BCA.csv file.

```
import csv

# field names

fields = ['Name', 'Branch', 'Year', 'CGPA']

# data rows of csv file

rows = [ ['Siva', 'CSE', '2020', '9.0'],

          ['Kumar', 'AI', '2021', '9.1'],

          ['Aditya', 'IT', '2022', '9.3'],

          ['Ram', 'SE', '2022', '9.5'],

          ['Kannan', 'ME', '2023', '7.8'],

          ['Ravikumar', 'CE', '2023', '9.1']]

    # name of csv file

    filename = "BCA.csv"

    # writing to csv file

    with open(filename, 'w') as csvfile:

    # creating a csv writer object

         csvwriter = csv.writer(csvfile)

         # writing the fields

         csvwriter.writerow(fields)

         # writing the data rows

         csvwriter.writerows(rows)
```

Output:



	A	B	C	D
1	Name	Branch	Year	CGPA
2				
3	Siva	CSE	2020	9
4				
5	Kumar	AI	2021	9.1
6				
7	Aditya	IT	2022	9.3
8				
9	Ram	SE	2022	9.5
10				
11	Kannan	ME	2023	7.8
12				
13	Ravikumar	CE	2023	9.1
14				

```
print(filename)
```

Out put:

```
BCA.csv   // csv file has created in the directory.
```

Source code: Read the student information in BCA.csv file.

```
with open('BCA.csv', mode ='r')as file:
    csvFile = csv.reader(file)
    for lines in csvFile:
    print(lines)
```

Output:

In [24]:

```
with open('BCA.csv', mode ='r')as file:
  csvFile = csv.reader(file)
  for lines in csvFile:
        print(lines)
```

```
['Name', 'Branch', 'Year', 'CGPA']
[]
['Siva', 'CSE', '2020', '9.0']
[]
['Kumar', 'AI', '2021', '9.1']
[]
['Aditya', 'IT', '2022', '9.3']
[]
['Ram', 'SE', '2022', '9.5']
[]
['Kannan', 'ME', '2023', '7.8']
[]
['Ravikumar', 'CE', '2023', '9.1']
[]
```

Date: 08-07-2024

Lab 3.a: Write a python to load the data into a dataframe, and then we visualize the data in a tabular format

Aim:

To load the data into a dataframe, and then we visualize the data in a tabular format using python program.

Source Code:

```
# Import pandas package

import pandas as pd

data = {'Name': ['Jai', 'Princi', 'Gaurav',

                 'Anuj', 'Ravi', 'Natasha', 'Riya'],

        'Age': [17, 17, 18, 17, 18, 17, 17],

        'Gender': ['M', 'F', 'M', 'M', 'M', 'F', 'F'],

        'Marks': [90, 76, 'NaN', 74, 65, 'NaN', 71]}

df = pd.DataFrame(data)

print(df)
```

Output:

	Name	Age	Gender	Marks
0	Jai	17	M	90
1	Princi	17	F	76
2	Gaurav	18	M	NaN
3	Anuj	17	M	74
4	Ravi	18	M	65
5	Natasha	17	F	NaN
6	Riya	17	F	71

Date: 08-07-2024

Lab 3.b: Write a python to Dealing with missing values in Python

Aim:

As we can see from the previous output, there is NaN values present in the MARKS column which is a missing value in the dataframe that is going to be taken care of in data wrangling by replacing them with the column mean.

Source Code:

```
# Compute average
c = avg = 0
for ele in df['Marks']:
   if str(ele).isnumeric():
   c += 1
   avg += ele
avg /= c

# Replace missing values
df = df.replace(to_replace="NaN",value=avg)
print(df)
```

Output:

	Name	Age	Gender	Marks
0	Jai	17	M	90.0
1	Princi	17	F	76.0
2	Gaurav	18	M	75.2
3	Anuj	17	M	74.0
4	Ravi	18	M	65.0
5	Natasha	17	F	75.2
6	Riya	17	F	71.0

Date: 08-07-2024

Lab 3.c: Write a python to Filtering data in Data Wrangling.

Aim:

To find the details of regarding name, gender, and marks of the top-scoring and need to remove some using the pandas slicing method in data wrangling from unwanted data.

Source Code:

```
# Filter top scoring students
df = df[df['Marks'] >= 75].copy()

# Remove age column from filtered DataFrame
df.drop('Age', axis=1, inplace=True)

# Display data
Print(df)
```

Output:

	Name	Gender	Marks
0	Jai	0.0	90.0
1	Princi	1.0	76.0
2	Gaurav	0.0	75.2
5	Natasha	1.0	75.2

Date: 15-07-2024

Lab 4.a: Write a python to Creating the First Dataframe to Perform Merge Operation using Data Wrangling.

Aim:

To create First Dataframe to Perform Merge Operation using python with Data Wrangling.

Source Code:

```
import pandas as pd
# creating DataFrame for Student Details
details = pd.DataFrame({
    'ID': [101, 102, 103, 104, 105, 106,
         107, 108, 109, 110],
  'NAME': ['Jagroop', 'Praveen', 'Harjot',
           'Pooja', 'Rahul', 'Nikita',
           'Saurabh', 'Ayush', 'Dolly', "Mohit"],
'BRANCH': ['CSE', 'CSE', 'CSE', 'CSE', 'CSE',
         'CSE', 'CSE', 'CSE', 'CSE', 'CSE']})
# printing details
print(details)
```

Output:

	ID	NAME	BRANCH
0	101	Jagroop	CSE
1	102	Praveen	CSE
2	103	Harjot	CSE
3	104	Pooja	CSE
4	105	Rahul	CSE
5	106	Nikita	CSE
6	107	Saurabh	CSE
7	108	Ayush	CSE
8	109	Dolly	CSE
9	110	Mohit	CSE

Date: 15-07-2024

Lab 4.b: Write a python to Creating the Second Dataframe to Perform Merge Operation using Data Wrangling.

Aim:

To create Second Dataframe to Perform Merge Operation using python with Data Wrangling.

Source Code:

```
import pandas as pd

# Creating Dataframe for Fees_Status

fees_status = pd.DataFrame(
      {'ID': [101, 102, 103, 104, 105,
          106, 107, 108, 109, 110],
     'PENDING': ['5000', '250', 'NIL',
            '9000', '15000', 'NIL',
            '4500', '1800', '250', 'NIL']})
# Printing fees_status
print(fees_status)
```

Output:

	ID	PENDING
0	101	5000
1	102	250
2	103	NIL
3	104	9000
4	105	15000
5	106	NIL
6	107	4500
7	108	1800
8	109	250
9	110	NIL

Date: 15-07-2024

Lab 4.c: Write a python to Merge Operation with first and Second Dataframe using Data Wrangling.

Aim:

To merge operation with first and Second Dataframe using Data Wrangling.

Source Code:

```
import pandas as pd
 # First Creating Dataframe
details = pd.DataFrame({
    'ID': [101, 102, 103, 104, 105,
           106, 107, 108, 109, 110],
    'NAME': ['Jagroop', 'Praveen', 'Harjot',
             'Pooja', 'Rahul', 'Nikita',
             'Saurabh', 'Ayush', 'Dolly', "Mohit"],
    'BRANCH': ['CSE', 'CSE', 'CSE', 'CSE', 'CSE',
               'CSE', 'CSE', 'CSE', 'CSE', 'CSE']})
 # Second Creating Dataframe
fees_status = pd.DataFrame(
    {'ID': [101, 102, 103, 104, 105,
            106, 107, 108, 109, 110],
     'PENDING': ['5000', '250', 'NIL',
                 '9000', '15000', 'NIL',
                 '4500', '1800', '250', 'NIL']})
# Merging Dataframe
print(pd.merge(details, fees_status, on='ID'))
```

Output:

	ID	NAME	BRANCH	PENDING
0	101	Jagroop	CSE	5000
1	102	Praveen	CSE	250
2	103	Harjot	CSE	NIL
3	104	Pooja	CSE	9000
4	105	Rahul	CSE	15000
5	106	Nikita	CSE	NIL
6	107	Saurabh	CSE	4500
7	108	Ayush	CSE	1800
8	109	Dolly	CSE	250
9	110	Mohit	CSE	NIL

Date: 23-07-2024

Lab 5.a: Write a python program File formats using Regular Expressions with Data Wrangling.

Aim:

To Validating Programming File formats using Regular Expressions.

Source Code:

```
import re
def isValidProgrammingFileFormat(str):
       regex=  "^.*\.(java|py|c|PHP|cs|cake|cshtml|csx|cpp|cp|cc|
       cobol|cob|ccp|css|flux|fx|go|jsp|kt|ktm|kts|numpy|numpyw|n
       umsc|pls|pck|pkb|pks|plbplsql|sql|perl|r|rd|rsx|sh|bash|st
       |cs |swift|vb|class)$"
       p = re.compile(regex)
       if (str == None):
          return "false"
       if(re.search(p, str)):
          return "true"
       else:
          return "false"
if __name__ == '__main__':
       str1 = "file_name.java"
       print(isValidProgrammingFileFormat(str1))
       str2 = "file_name01.py"
       print(isValidProgrammingFileFormat(str2))
       str3 = "Java123"
       print(isValidProgrammingFileFormat(str3))
       str4 = "file_name.class"
       print(isValidProgrammingFileFormat(str4))
       str5 = "file_name.css"
       print(isValidProgrammingFileFormat(str5))
```

```
str6 = ".PHP_file_name"
print(isValidProgrammingFileFormat(str6))
```

Output

```
true
true
false
true
true
false
```

Date: 23-07-2024

Lab 5.b: Write a Python program to find all Regular Expressions functions in Data Wrangling.

Aim:

To Validating Python program to find all the Regular Expressions functions in Data Wrangling.

Source Code:

(i) The findall( ) Function

```
import re
txt = "The rain in Spain"
x = re.findall("ai", txt)
print(x)
```

Output:

['ai', 'ai']

The findall( ) Function using Conditional statements

```
import re
txt = "The rain in Spain"
x = re.findall("rain", txt)
print(x)
if (x):
  print("Yes, there is at least one match!")
else:
  print("No match")
```

Output:

['rain']

Yes, there is at least one match!

(ii) The search() Function

```
import re
txt = "The rain in Spain"
x = re.search("\s", txt)
```

```
print("The first white-space character is located in position:",
x.start())
y = re.search("Portugal", txt)
print(y)
```

Output:

The first character is located in position: 3

None

(iii) The split( ) Function

```
import re
#Split the string at every white-space character:
txt = "The rain in Spain"
x = re.split("\s", txt)
print(x)
```

Output:

['The', 'rain', 'in', 'Spain']

(iv) The sub( ) Function

```
import re
#Replace all white-space characters with the digit "9":
txt = "The rain in Spain"
x = re.sub("\s", "9", txt)
y = re.sub("\s", "9", txt, 2)
#Replace the first two occurrences of a white-space
character with the digit 9:
print(x)
print(y)
```

Output:

```
The9rain9in9Spain
The9rain9in Spain
```

Date: 30-07-2024

Lab 6.a: Write a Python program to find web scraping, the process of extracting data from websites using request library.

Aim:

To Validating Python program to find web scraping, the process of extracting data from websites using request library .

Source Code:

```
import requests

# Making a GET request

r=requests.get('https://www.geeksforgeeks.org/python-
programming-language/')

# check status code for response received

# success code - 200

print(r)

# print content of request

print(r.content)
```

Output:

Date: 30-07-2024

Lab 6.b: Write a Python program to find web scraping, the process of extracting data from websites using BeautifulSoup Library.

Aim:

To Validating Python program to find web scraping, the process of extracting data from websites using BeautifulSoup Library .

Source Code:

```
import requests
from bs4 import BeautifulSoup
# Making a GET request
r = requests.get('https://www.geeksforgeeks.org/python-
programming-language/')
# check status code for response received
# success code - 200
print(r)
# Parsing the HTML
soup = BeautifulSoup(r.content, 'html.parser')
print(soup.prettify())
```

Output:

```
<Response [200]>
<!DOCTYPE html>
<!--[if IE 7]>
<html class="ie ie7" lang="en-US" prefix="og: http://ogp.me/ns#">
<![endif]-->
<!--[if IE 8]>
<html class="ie ie8" lang="en-US" prefix="og: http://ogp.me/ns#">
<![endif]-->
<!--[if !(IE 7) | !(IE 8)  ]><!-->
<html lang="en-US" prefix="og: http://ogp.me/ns#">
 <!--<![endif]-->
 <head>
  <meta charset="utf-8"/>
  <meta content="Data Structures, Algorithms, Python, Java, C, C++, JavaScript, Android Development, SQL, Data Science, Machi
ne Learning, PHP, Web Development, System Design, Tutorial, Technical Blogs, Interview Experience, Interview Preparation, Pro
gramming, Competitive Programming, Jobs, Coding Contests, GATE CSE, HTML, CSS, React, NodeJS, Placement, Aptitude, Quiz, Comp
uter Science, Programming Examples, GeeksforGeeks Courses, Puzzles, SSC, Banking, UPSC, Commerce, Finance, CBSE, School, k12,
General Knowledge, News, Mathematics, Exams" name="keywords"/>
  <meta content="width=device-width, initial-scale=1.0, minimum-scale=0.5, maximum-scale=3.0" name="viewport"/>
```

Date: 06-08-2024

Lab 7.a: Write a Python program to apply Conditional Selection involves selecting rows from a DataFrame or Series based on a condition or set of conditions.

Aim:

To Validating Python program to to apply Conditional Selection involves selecting rows from a DataFrame or Series based on a condition or set of conditions.

Source Code:

```
import pandas as pd

# Sample DataFrame

df = pd.DataFrame({

    'Name': ['Alice', 'Bob', 'Charlie', 'David'],

    'Age': [25, 30, 35, 40],

    'Salary': [50000, 60000, 70000, 80000]})

# Condition: Select rows where Age is greater than 30

condition = df['Age'] > 30

filtered_df = df[condition] # Apply condition to DataFrame

print(filtered_df)
```

Output:

```
      Name  Age  Salary
2  Charlie   35   70000
3    David   40   80000
```

Date:06-08-2024

7.b. Multiple Conditions:

```
import pandas as pd
# Sample DataFrame
df = pd.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie', 'David'],
    'Age': [25, 30, 35, 40],
    'Salary': [50000, 60000, 70000, 80000]
})
# Multiple conditions: Age > 30 AND Salary > 60000
filtered_df = df[(df['Age'] > 30) & (df['Salary'] > 60000)]
print(filtered_df)
```

Output:

```
      Name  Age  Salary
2  Charlie   35   70000
3    David   40   80000
```

Date: 06-08-2024

Lab 7.c: Write a Python program to use replacing values in Pandas.

Aim:

To Validating Python program to use replacing values in Pandas.

Source Code:

```
import pandas as pd
# Sample DataFrame
df = pd.DataFrame({
    'A': [1, 2, 2, 3, 4],
    'B': [5, 6, 7, 8, 9]
})
# Replace specific value in column 'A'
df['A'].replace(2, 20, inplace=True)
print(df)
```

Output:

```
    A  B
0   1  5
1  20  6
2  20  7
3   3  8
4   4  9
```

Replacing Multiple Values

```
import pandas as pd
# Sample DataFrame
df = pd.DataFrame({
    'A': [1, 2, 2, 3, 4],
    'B': [5, 6, 7, 8, 9]
})
replacements = {1: 'One', 2: 'Two', 3: 'Three'}
df['A'].replace(replacements, inplace=True)
print(df)
```

Output:

```
       A  B
0    One  5
1    Two  6
2    Two  7
3  Three  8
4      4  9
```

Date: 13-08-2024

Lab 8: Write a Python program to find outlier detection using a simple statistical test detect and remove outliners in Z-score treatment.

Aim:

To Validating Python program to find outlier detection using a simple statistical test detects and remove outliners in Z-score treatment.

Source Code:

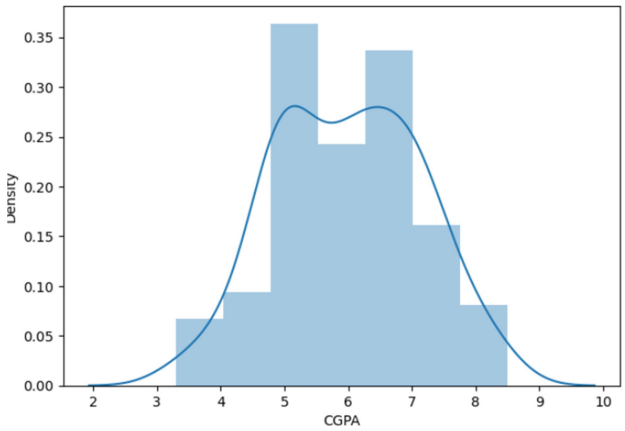
```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read_csv('Placement.csv')

df.sample(5)
```

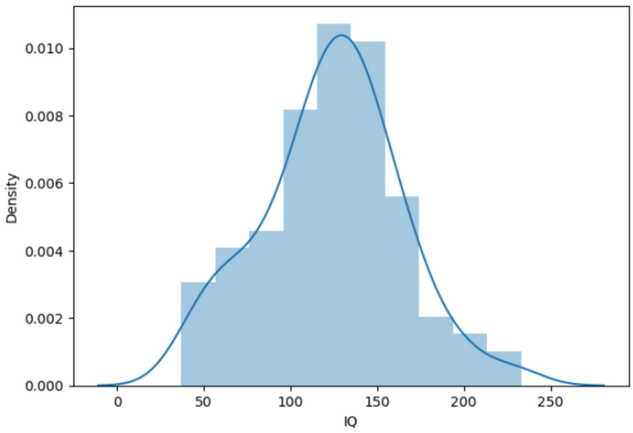
Out[4]:

	Student_ID	CGPA	IQ	Placement
90	91	7.3	86	1
81	82	5.4	107	0
60	61	6.9	139	1
27	28	6.0	124	1
24	25	4.7	121	0

#Plot the distribution plots for the features

```
import warnings

warnings.filterwarnings('ignore')

plt.figure(figsize=(16,5))

plt.subplot(1,2,1)

sns.distplot(df['CGPA'])

plt.subplot(1,2,2)

sns.distplot(df['IQ'])

plt.show()
```





Finding the boundary values

```
print("Highest allowed",df['CGPA'].mean() + 3*df['CGPA'].std())
print("Lowest allowed",df['CGPA'].mean() - 3*df['CGPA'].std())
```

```
Highest allowed 9.421901021332708
Lowest allowed 2.560098978667293
```

Finding the outliers

```
df[(df['CGPA'] > 8.80) | (df['CGPA'] < 5.11)]
```

Trimming of outliers

```
new_df = df[(df['CGPA'] < 8.80) & (df['CGPA'] > 5.11)]
print(new_df)
```

Capping on outliers

```
upper_limit = df['CGPA'].mean() + 3*df['CGPA'].std()
lower_limit = df['CGPA'].mean() - 3*df['CGPA'].std()
```

Now, apply the capping

```
df['CGPA'] = np.where(
df['CGPA']>upper_limit,
upper_limit,
np.where( df['CGPA']<lower_limit,
lower_limit,
df['CGPA']))
```

Now, see the statistics using the “Describe” function

```
df['CGPA'].describe()
```

Output :

```
Out[30]: count    100.000000
         mean       5.991000
         std        1.143634
         min        3.300000
         25%        5.075000
         50%        6.000000
         75%        6.900000
         max        8.500000
         Name: CGPA, dtype: float64
```

Date: 21-08-2024

Lab 9: Write a Python program to read any tabular dataset and perform data cleaning

Aim:

To Validating Python program to read any tabular dataset and performs data cleaning empty cells and removes rows

Source Code: Return a new Data Frame with no empty cells:

```
import pandas as pd
df = pd.read_csv('data.csv')
new_df = df.dropna()
print(new_df.to_string())
```

out put:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2

Remove all rows with NULL values:

```
import pandas as pd
df = pd.read_csv('data.csv')
df.dropna(inplace = True)
print(df.to_string())
```

Output:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	2020/12/17'	100	120	300.0
18	60	2020/12/19'	103	123	323.0

Replace Empty Values

Another way of dealing with empty cells is to insert a *new* value instead.

The fillna( ) method allows us to replace empty cells with a value:

```
import pandas as pd
df = pd.read_csv('data.csv')
df.fillna(130, inplace = True)
#Notice in the result: empty cells got the value 130 (in row 18, 22 and 28).
```

Output:

11	60	'2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	130.0
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
22	45	130	100	119	282.0
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5
26	60	2020/12/26	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	130.0

Data of Wrong Format

Cells with data of wrong format can make it difficult, or even impossible, to analyze data.

To fix it, you have two options: remove the rows, or convert all cells in the columns into the same format.

```
import pandas as pd
df = pd.read_csv('data.csv')
df['Date'] = pd.to_datetime(df['Date'])
print(df.to_string())
```

Output:

In our Data Frame, we have two cells with the wrong format. Check out row 22 and 26, the 'Date' column should be a string that represents a date:

10	60	2020/12/11	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	NaN
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
22	45	NaN	100	119	282.0
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5
26	60	20201226	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	NaN

Replacing Values

One way to fix wrong values is to replace them with something else.

In our example, it is most likely a typo, and the value should be "45" instead of "450", and we could just insert "45" in row 7:

```
df.loc[7, 'Duration'] = 45
```

Output:

```
6          60  '2020/12/07'    110         136      374.0
7          45  '2020/12/08'    104         134      253.3
```

Date: 29-08-2024

Lab 10: Write a Python program to implement Combine and merge of data in Pandas Object in joins operation.

Aim:

To Validating python program to implement combine and merge of data in Pandas object in joins operation.

Source Code : Pandas Inner Join

```
a = pd.DataFrame()              # Creating Dictionary
d = {'id': [1, 2, 10, 12],'val1': ['a', 'b', 'c', 'd']}
a = pd.DataFrame(d)             # Creating dataframe b
b = pd.DataFrame()              # Creating dictionary
d = {'id': [1, 2, 9, 8],'val1': ['p', 'q', 'r', 's']}
b = pd.DataFrame(d)
df = pd.merge(a, b, on='id', how='inner')     # inner join
df                          # display dataframe
```

Output:

	id	val1_x	val1_y
0	1	a	p
1	2	b	q

Source Code : Pandas Left Join

```
df = pd.merge(a, b, on='id', how='left')
# display dataframe
Df
```

Output:

	id	val1_x	val1_y
0	1	a	p
1	2	b	q
2	10	c	NaN
3	12	d	NaN

Source Code : Pandas Right Join

```
df = pd.merge(a, b, on='id', how='right')
# display dataframe
df
```

Output:

	id	val1_x	val1_y
0	1	a	p
1	2	b	q
2	9	NaN	r
3	8	NaN	s

Source Code : Pandas Full Outer Join

```
df = pd.merge(a, b, on='id', how='outer')
# display dataframe
df
```

Output:

	id	val1_x	val1_y
0	1	a	p
1	2	b	q
2	10	c	NaN
3	12	d	NaN
4	9	NaN	r
5	8	NaN	s

Source Code : Pandas Index Join

```
df = pd.merge(a, b, left_index=True, right_index=True)
# display dataframe
df
```

	id_x	val1_x	id_y	val1_y
0	1	a	1	p
1	2	b	2	q
2	10	c	9	r
3	12	d	8	s

Date: 05-09-2024

Lab 11: Write a Python program to implement reshaping and pivoting using Pandas object with set hierarchical index.

Aim:

To Validating python program to implement reshaping and pivoting using Pandas object with set hierarchical index.

Source Code : Set Hierarchical Index

```
data = {
    'City': ['New York', 'New York', 'Los Angeles', 'Los Angeles'],
    'Date': ['2024-01-01', '2024-01-02', '2024-01-01', '2024-01-
       02'],
    'Temperature': [32, 31, 75, 74]
}
# Create DataFrame
df = pd.DataFrame(data)
# Set hierarchical index
df.set_index(['City', 'Date'], inplace=True)
print(df)
```

Output:

```
                     Temperature
City        Date
New York    2024-01-01           32
            2024-01-02           31
Los Angeles 2024-01-01           75
            2024-01-02           74
```

Source Code : swaplevel()and Sort by index

```
df_swapped = df.swaplevel()
print(df_swapped)
# Sort by index
df_sorted = df.sort_index(level=['Date', 'City'])
print(df_sorted)
```

Output:

```
                     Temperature
Date       City
2024-01-01 New York            32
2024-01-02 New York            31
2024-01-01 Los Angeles         75
2024-01-02 Los Angeles         74
                     Temperature
City        Date
Los Angeles 2024-01-01         75
New York    2024-01-01         32
Los Angeles 2024-01-02         74
New York    2024-01-02         31
```

Source Code : reshaping data, such as pivoting or unstacking data.

```
# Unstack the DataFrame, Hierarchical indexes are particularly
      useful for reshaping data, such as pivoting or unstacking
      data.
df_unstacked = df.unstack(level='Date')
print(df_unstacked)
# Stack the DataFrame (inverse of unstack)
df_stacked = df_unstacked.stack()
print(df_stacked)
```

Output:

```
            Temperature
Date         2024-01-01 2024-01-02
City
Los Angeles          75         74
New York             32         31
                        Temperature
City        Date
Los Angeles 2024-01-01           75
            2024-01-02           74
New York    2024-01-01           32
            2024-01-02           31
```

Date: 12-09-2024

Lab 12: Write a Python program to implement concat() function and merge of data in Pandas Object in joins operation.

Aim:

To Validating python program to implement concat() function and merge of data in Pandas Object in joins operation.

Source Code: Pandas concat() function

```
import pandas as pd
# Sample DataFrames
df1 = pd.DataFrame({'key': ['A', 'B', 'C'], 'value1': [1, 2, 3]})
df4 = pd.DataFrame( { "B": ["B2", "B3", "B6", "B7"],
"D": ["D2", "D3", "D6", "D7"],
"F": ["F2", "F3", "F6", "F7"],
},
index=[2, 3, 6, 7],)
result = pd.concat([df1, df4], axis=1)
result
```

Output:

	key	value1	B	D	F
0	A	1.0	NaN	NaN	NaN
1	B	2.0	NaN	NaN	NaN
2	C	3.0	B2	D2	F2
3	NaN	NaN	B3	D3	F3
6	NaN	NaN	B6	D6	F6
7	NaN	NaN	B7	D7	F7

Source Code: Pandas concat() function with inner join

```
result = pd.concat([df1, df4], axis=1, join="inner")
result
```

Output:

	key	value1	B	D	F
2	1	3	B2	D2	F2

Source Code: Pandas ignore_index ignores overlapping indexes

```
result = pd.concat([df1, df4], ignore_index=True, sort=False)
result
```

Output:

	key	value1	B	D	F
0	A	1.0	NaN	NaN	NaN
1	B	2.0	NaN	NaN	NaN
2	C	3.0	NaN	NaN	NaN
3	NaN	NaN	B2	D2	F2
4	NaN	NaN	B3	D3	F3
5	NaN	NaN	B6	D6	F6
6	NaN	NaN	B7	D7	F7

Source Code: Pandas result can be reindexed

```
result = pd.concat([df1, df4], axis=1).reindex(df1.index)
result
```

Output

	key	value1	B	D	F
0	1.0	1.0	NaN	NaN	NaN
1	1.0	2.0	NaN	NaN	NaN
2	1.0	3.0	B2	D2	F2

Date: 20-09-2024

Lab 13: Write a Python program to implement data visualization with Pandas is the presentation of data in a graphical format.

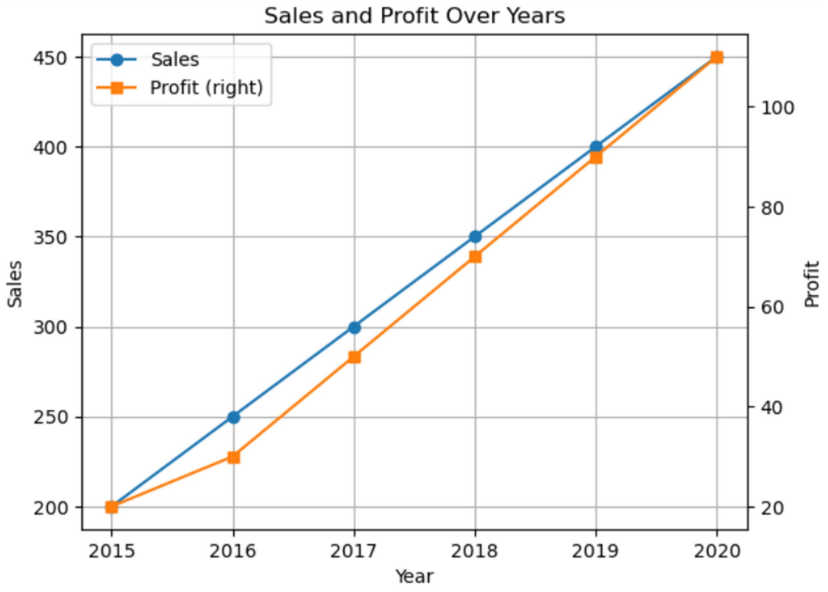
Aim:

To Validating python program to implement data visualization with Pandas is the presentation of data in a graphical format.

Source Code: A line plot is a graph that shows the frequency of data along a number line

```
import pandas as pd
import matplotlib.pyplot as plt
data = { 'Year': [2015, 2016, 2017, 2018, 2019, 2020],
    'Sales': [200, 250, 300, 350, 400, 450],
    'Profit': [20, 30, 50, 70, 90, 110]
}
df = pd.DataFrame(data)
# Plotting both Sales and Profit over the years
ax = df.plot(x='Year', y='Sales', kind='line', marker='o',
      title='Sales and Profit Over Years')
df.plot(x='Year', y='Profit', kind='line', marker='s', ax=ax,
      secondary_y=True)
# Adding labels and grid
ax.set_ylabel('Sales')
ax.right_ax.set_ylabel('Profit')
ax.grid(True)
plt.show()
```

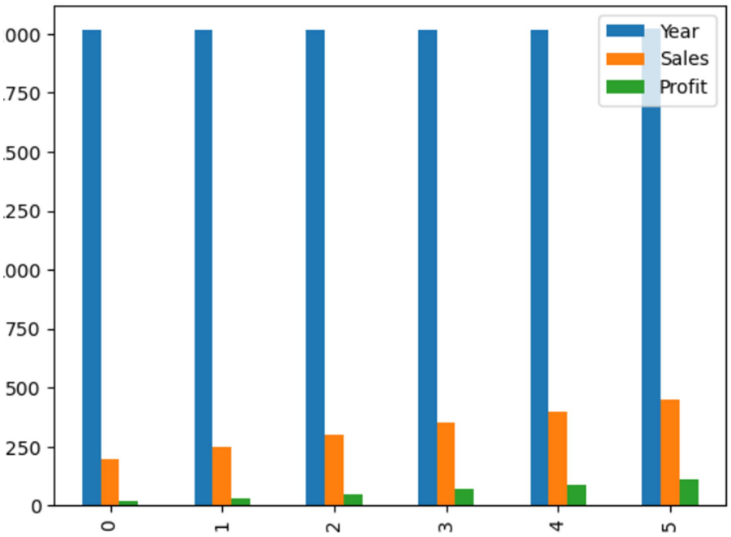
Output:



Source code: A bar chart or bar graph is a chart

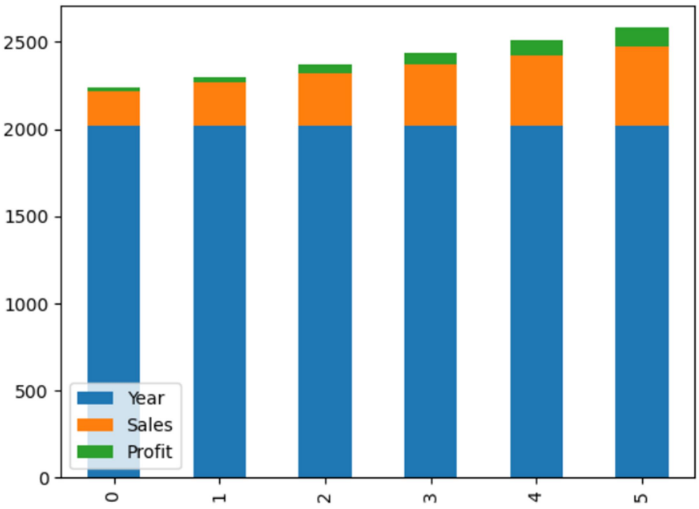
```
df.plot.bar()
```

Output:



Source code: stacked bar chart or Stacked bar plots

```
df.plot.bar(stacked=True)
```



Date: 27-09-2024

Lab 14: Write a Python program to implement Use Pandas groupby function to group the data based on the "Team".

Aim:

To Validating python program to implement Use Pandas groupby function to group the data based on the "Team".

Source Code:

```
import pandas as pd
# Creating the dataframe
df = pd.read_csv("nba.csv")
# Print the dataframe
print(df.head())
```

Output

```
              Name            Team  Number Position   Age Height  Weight
0  Avery Bradley  Boston Celtics     0.0       PG  25.0    6-2   180.0
1    Jae Crowder  Boston Celtics    99.0       SF  25.0    6-6   235.0
2   John Holland  Boston Celtics    30.0       SG  27.0    6-5   205.0
3    R.J. Hunter  Boston Celtics    28.0       SG  22.0    6-5   185.0
4  Jonas Jerebko  Boston Celtics     8.0       PF  29.0   6-10   231.0

              College     Salary
0              Texas  7730337.0
1           Marquette  6796117.0
2  Boston University        NaN
3      Georgia State  1148640.0
4                NaN  5000000.0
```

Source Code: applying groupby() function to group the data on team value.

```
# applying groupby() function to group the data on team value.
gk = df.groupby('Team')
# Let's print the first entries in all the groups formed.
gk.first()
```

Output

Team	Name	Number	Position	Age	Height	Weight	College	Salary
Atlanta Hawks	Kent Bazemore	24.0	SF	26.0	6-5	201.0	Old Dominion	2000000.0
Boston Celtics	Avery Bradley	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
Brooklyn Nets	Bojan Bogdanovic	44.0	SG	27.0	6-8	216.0	Oklahoma State	3425510.0
Charlotte Hornets	Nicolas Batum	5.0	SG	27.0	6-8	200.0	Virginia Commonwealth	13125306.0
Chicago Bulls	Cameron Bairstow	41.0	PF	25.0	6-9	250.0	New Mexico	845059.0
Cleveland Cavaliers	Matthew Dellavedova	8.0	PG	25.0	6-4	198.0	Saint Mary's	1147276.0
Dallas Mavericks	Justin Anderson	1.0	SG	22.0	6-6	228.0	Virginia	1449000.0
Denver Nuggets	Darrell Arthur	0.0	PF	28.0	6-9	235.0	Kansas	2814000.0
Detroit Pistons	Joel Anthony	50.0	C	33.0	6-9	245.0	UNLV	2500000.0
Golden State Warriors	Leandro Barbosa	19.0	SG	33.0	6-3	194.0	North Carolina	2500000.0
Houston Rockets	Trevor Ariza	1.0	SF	30.0	6-8	215.0	UCLA	8193030.0
Indiana Pacers	Lavoy Allen	5.0	PF	27.0	6-9	255.0	Temple	4050000.0
Los Angeles Clippers	Cole Aldrich	45.0	C	27.0	6-11	250.0	Kansas	1100602.0
Los Angeles Lakers	Brandon Bass	2.0	PF	31.0	6-8	250.0	LSU	3000000.0
Memphis Grizzlies	Jordan Adams	3.0	SG	21.0	6-5	209.0	UCLA	1404600.0
Miami Heat	Chris Bosh	1.0	PF	32.0	6-11	235.0	Georgia Tech	22192730.0
Milwaukee Bucks	Giannis Antetokounmpo	34.0	SF	21.0	6-11	222.0	Arizona	1953960.0
Minnesota Timberwolves	Nemanja Bjelica	88.0	PF	28.0	6-10	240.0	Louisville	3950001.0
New Orleans Pelicans	Alexis Ajinca	42.0	C	28.0	7-2	248.0	California	4389607.0
New York Knicks	Arron Afflalo	4.0	SG	30.0	6-5	210.0	UCLA	8000000.0

Source Code: get_group() to find the entries contained in any of the groups.

```
gk.get_group('Boston Celtics')
```

Output

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
5	Amir Johnson	Boston Celtics	90.0	PF	29.0	6-9	240.0	NaN	12000000.0
6	Jordan Mickey	Boston Celtics	55.0	PF	21.0	6-8	235.0	LSU	1170960.0
7	Kelly Olynyk	Boston Celtics	41.0	C	25.0	7-0	238.0	Gonzaga	2165160.0
8	Terry Rozier	Boston Celtics	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0
9	Marcus Smart	Boston Celtics	36.0	PG	22.0	6-4	220.0	Oklahoma State	3431040.0
10	Jared Sullinger	Boston Celtics	7.0	C	24.0	6-9	260.0	Ohio State	2569260.0
11	Isaiah Thomas	Boston Celtics	4.0	PG	27.0	5-9	185.0	Washington	6912869.0
12	Evan Turner	Boston Celtics	11.0	SG	27.0	6-7	220.0	Ohio State	3425510.0
13	James Young	Boston Celtics	13.0	SG	20.0	6-6	215.0	Kentucky	1749840.0

Source Code: First grouping based on "Team" within each team we are grouping based on "Position"

```
import pandas as pd
# Creating the dataframe
df = pd.read_csv("nba.csv")
# First grouping based on "Team"
# Within each team we are grouping based on "Position"
gkk = df.groupby(['Team', 'Position'])
# Print the first value in each group
gkk.first()
```

Output:

Team	Position	Name	Number	Age	Height	Weight	College	Salary
Atlanta Hawks	**C**	Al Horford	15.0	30.0	6-10	245.0	Florida	12000000.0
	PF	Kris Humphries	43.0	31.0	6-9	235.0	Minnesota	1000000.0
	PG	Dennis Schroder	17.0	22.0	6-1	172.0	Wake Forest	1763400.0
	SF	Kent Bazemore	24.0	26.0	6-5	201.0	Old Dominion	2000000.0
	SG	Tim Hardaway Jr.	10.0	24.0	6-6	205.0	Michigan	1304520.0
...	**...**	...	...	...	...	...	...	...
Washington Wizards	**C**	Marcin Gortat	13.0	32.0	6-11	240.0	North Carolina State	11217391.0
	PF	Drew Gooden	90.0	34.0	6-10	250.0	Kansas	3300000.0
	PG	Ramon Sessions	7.0	30.0	6-3	190.0	Nevada	2170465.0
	SF	Jared Dudley	1.0	30.0	6-7	225.0	Boston College	4375000.0
	SG	Alan Anderson	6.0	33.0	6-6	220.0	Michigan State	4000000.0

149 rows × 7 columns

Date: 27-09-2024

Lab 15: Write a Python program to implement Use Pandas groupby function to group the data based on the "Team".

Aim:

To Validating python program to implement Use Pandas groupby function to group the data based on the "Team".

Source Code: printing the first 10 rows of the dataframe

```
import pandas as pd
# Creating our dataset
df = pd.DataFrame([[9, 4, 8, 9],
 [8, 10, 7, 6],
[7, 6, 8, 5]],
columns=['Maths', 'English','Science', 'History'])
# display dataset
print(df)
```

Output

	Maths	English	Science	History
0	9	4	8	9
1	8	10	7	6
2	7	6	8	5

Source Code:

```
a = df.groupby('Maths')
a.first()
```

Output

Maths	English	Science	History
7	6	8	5
8	10	7	6
9	4	8	9

Source Code:

```
b = df.groupby(['Maths', 'Science'])
b.first()
```

Output

		English	History
Maths	**Science**		
7	**8**	6	5
8	**7**	10	6
9	**8**	4	9

Source code: Mechanics of Group By Operation

```
import pandas as pd
data1 = {'Name':['Jai', 'Anuj', 'Jai', 'Princi', 'Gaurav',
       'Anuj', 'Princi', 'Abhi'], 'Age':[27, 24, 22, 32, 33, 36,
       27, 32], 'Address':['Nagpur', 'Kanpur', 'Allahabad',
       'Kannuaj', 'Jaunpur', 'Kanpur', 'Allahabad', 'Aligarh'],
       'Qualification':['Msc', 'MA', 'MCA', 'Phd', 'B.Tech',
       'B.com', 'Msc', 'MA']}
df = pd.DataFrame(data1)
print(df)
```

Output:

	Name	Age	Address	Qualification
0	Jai	27	Nagpur	Msc
1	Anuj	24	Kanpur	MA
2	Jai	22	Allahabad	MCA
3	Princi	32	Kannuaj	Phd
4	Gaurav	33	Jaunpur	B.Tech
5	Anuj	36	Kanpur	B.com
6	Princi	27	Allahabad	Msc
7	Abhi	32	Aligarh	MA

```
# with one key

df.groupby('Name')

print(df.groupby('Name').groups)
```

Output:

```
{'Abhi': [7], 'Anuj': [1, 5], 'Gaurav': [4], 'Jai': [0, 2], 'Princi': [3,
6]}
```