Working With CSV Files in Python:

Simple Data Analysis Examples

CSV (Comma-Separated Values) files are widely used for storing and exchanging tabular data. Python, with its powerful libraries like pandas, provides a convenient and efficient way to work with CSV files. In this article, we will explore how to work with CSV files in Python and demonstrate simple data analysis examples using pandas.

Why Use CSV Files in Python?

CSV files are popular because they are easy to create, read, and share across different platforms and software. Python, with its robust data analysis capabilities, is an excellent choice for processing and analyzing CSV data. By leveraging Python and its libraries, you can perform a wide range of data analysis tasks quickly and effectively.

Steps for Working with CSV Files in Python

1- **Copy the file** 'data.csv' from the teacher shared drive into a new folder Lab16 on the P: drive of the lab computer. Note that you can use your laptop if you have downloaded the required modules (pandas and matplotlib) but it is highly recommended to use the lab computer to practice for the upcoming lab exam.

2- **Importing Libraries:** When importing modules with long names, we often use an alias name to make our code shorter.

```
import pandas as pd
import matplotlib.pyplot as plt
from math import radians,sin,cos
```

3- **Reading CSV Files**

```
df = pd.read_csv('data.csv', header=0)
```

```
            header = 0 sets the first row as the header of the dataframe.
It is the default so you can also simply use:
```

```
df = pd.read_csv('data.csv')
```

4- **Basic Data Analysis**

- **Viewing the Data:** Use the `head()` method to display the first few rows of the data frame and get an overview of the data structure.
  print(data.head())

You will note that the file 'data.csv' has column labels for all the columns. This will always be the case in .csv data files given to you.

You can use the column labels to extract data:

```
x = df['x1']
y = df['y1']
```

print('length of x is:',len(x))
print('length of y is:',len(y))

- **Summary Statistics:** Use the `statistics module` to obtain summary statistics, such as mean, standard deviation, minimum, maximum for numerical columns in the DataFrame.

```
Import statistics as st
mean = st.mean(y)
std_dev = st.stdev(y)
print(mean, std_dev)
```

`Now use f strings to display both values with the formats xxx.xx`

```
print(f'mean is : {mean:4.2f} and standard deviation is:
{std_dev:4.2f}')
```

5- **Data Visualization**

Python provides various libraries for data visualization, such as matplotlib. These libraries allow you to create visually appealing plots and charts to explore and present your data. By visualizing your data, you can gain valuable insights and communicate your findings effectively.

#Plot your data in a scatter plot

```
plt.scatter(x, y)
plt.xlabel('x1')
plt.ylabel('x1')
plt.ylim([0,13])
plt.title("My First Plot")
plt.show()
```

6- **You can Generate Theoretical Plot Data based on a given simple function (similar to manual process in high school)**

# y1=sin(x) + cos(x)
# y2=0.5(sin(x) + cos(x))

```
# Define 2 empty lists to hold the x and y coordinates
 X1 = []
 Y1 = []
```

```
for d in range(0,370,20): #d is angle in degrees, steps of 20
    angle=radians(d)
    x.append(d)
    y1.append(sin(angle)+cos(angle))
    y2.append(0.5*sin(angle)+cos(angle))
```