Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Experiment No. 11
Program to demonstrate data frame creation and Manipulation using Pandas
Date of Performance:
Date of Submission:

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Experiment No. 11

Title: Program to demonstrate data frame creation and Manipulation using Pandas

Aim: To study and implement data frame creation and Manipulation using Pandas

Objective: To introducc Pandas packagc for python

Theory:

Pandas is an open-source library that is built on top of NumPy library. It is a Python package that offers various data structures and operations for manipulating numerical data and time series. It is mainly popular for importing and analyzing data much easier. Pandas is fast and it has high-performance & productivity for users.

Installation of Pandas

If you have Python and PIP already installed on a system, then installation of Pandas is very easy.

Install it using this command:

```
C:\Users\Your Name>pip install pandas
```

If this command fails, then use a python distribution that already has Pandas installed like, Anaconda, Spyder etc.

Import Pandas

Once Pandas is installed, import it in your applications by adding the import keyword:

```
import pandas
```

Now Pandas is imported and ready to use.

Example
Code:

```
import pandas
mydataset = {
```

```
 'Fruit': ["Orange", "apple", "mango"],
 'passings': [8 , 3 , 2]
}
myvar = pandas.DataFramc(mydatasct)
print(myvar)
```

OUTPUT:

```
    Fruit  passings
0  Orange         8
1   apple         3
2   mango         2
```

Pandas is usually imported under the pd alias.

alias: In Python alias are an alternate name for referring to the same thing.

Create an alias with the as keyword while importing:

```
import pandas as pd
```

Now the Pandas package can be referred to as pd instead of pandas.

Example

```
import pandas as pd

mydataset = {
 'Fruit': ["Orange", "apple", "mango"],
 'passings': [8 , 3 , 2]
}

myvar = pd.DataFrame(mydataset)

print(myvar)
```

Checking Pandas Version

The version string is stored under __version__ attribute.

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Example

```
import pandas as pd

print(pd.__version__)
```

Output:

```
2.0.3
```

What is a Series?

A Pandas Series is like a column in a table.

It is a one-dimensional array holding data of any type.

Example

Create a simple Pandas Series from a list:

```
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a)

print(myvar)
```

```
0    1
1    7
2    2
dtype: int64
```

Labels

If nothing else is specified, the values are labeled with their index number. First value has index 0, second value has index 1 etc.

This label can be used to access a specified value.

Example

Return the first value of the Series:

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

```
print(myvar[0])
```

```
1
```

Create Labels

With the index argument, you can name your own labels.

Example

Create your own labels:

```
import pandas as pd
a = [7, 5, 8]
myvar = pd.Series(a, index = ["a", "b", "c"])
print(myvar)
```

```
a    7
b    5
c    8
dtype: int64
```

When you have created labels, you can access an item by referring to the label.

Example

Return the value of "y":

```
print(myvar["y"])
```

Key/Value Objects as Series

You can also use a key/value object, like a dictionary, when creating a Series.

Example

Create a simple Pandas Series from a dictionary:

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

```
import pandas as pd

calories = {"day1": 420, "day2": 380, "day3": 390}

myvar = pd.Series(calories)

print(myvar)
```

```
day1    420
day2    380
day3    390
dtype: int64
```

To select only some of the items in the dictionary, use the index argument and specify only the items you want to include in the Series.

Example

Create a Series using only data from "day1" and "day2":

```
import pandas as pd

calories = {"day1": 420, "day2": 380, "day3": 390}

myvar = pd.Series(calories, index = ["day1", "day2"])

print(myvar)
```

```
day1    420
day2    380
dtype: int64
```

DataFrames

Data sets in Pandas are usually multi-dimensional tables, called DataFrames.

Series is like a column, a DataFrame is the whole table.

Example

Create a DataFrame from two Series:

```
import pandas as pd

data = {
  "calories": [420, 380, 390],
```

```
  "duration": [50, 40, 45]
}

myvar = pd.DataFrame(data)

print(myvar)
```

```
   calories  duration
0       420        50
1       380        40
2       390        45
```

Read CSV Files

A simple way to store big data sets is to use CSV files (comma separated files).

CSV files contains plain text and is a well know format that can be read by everyone including Pandas.

In our examples we will be using a CSV file called 'data.csv'.

Example

Load the CSV into a DataFrame:

```
import pandas as pd

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

print(df.to_string())
```

Example:

Print the DataFrame without the to_string() method:

```
import pandas as pd

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

print(df)
```

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

max_rows

The number of rows returned is defined in Pandas option settings.

You can check your system's maximum rows with the pd.options.display.max_rows statement.

Example

Check the number of maximum returned rows:

```
import pandas as pd

print(pd.options.display.max_rows)
```

```
import pandas as pd

print(pd.options.display.max_rows)
```

```
60
```

Example

Increase the maximum number of rows to display the entire DataFrame:

```
import pandas as pd

pd.options.display.max_rows = 9999

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

print(df)
```

Code:

```
import pandas as pd
df = pd.read_csv('data.csv')
print("Original DataFramc:")
print(df)
print()
df['Average'] = df[['quiz1-marks', 'quiz2-marks', 'quiz3-marks']].mean(axis=1)
```