Count Number of Words in a Column using Python

Most data science professionals use the pandas library for data handling and preparation. The pandas library doesn't have any method to count the number of words in a piece of text. One way to solve this problem is by finding the length of the text by splitting the complete text.

Important Pandas Functions for Data Science

Pandas is a very fast and efficient DataFrame object for working with data. It provides highly efficient functions, ranging from reading and writing data to manipulate and preparing data for any kind of data science task. Although you need to learn all the functions of the Pandas library, if you want to know the most important functions that it provides for data science, this article is for you. In this article, I'm going to introduce you to some of the most important Pandas functions for data science that you need to know.

1. Intelligent data alignment
2. Integrated handling of missing data
3. Flexible data reshaping
4. Easy insertion and deletion of columns
5. data aggregation and transformation
6. Merging and joining of datasets
7. Time series functionality
8. Academic and Commercial usage

Reading a Dataset:

In [10]:
```
import pandas as pd
data = pd.read_csv("GOOG.csv")
```

In [11]:
```
print(data.head())
```

```
         Date         Open         High          Low        Close  \
0  2019-08-09  1197.989990  1203.880005  1183.603027  1188.010010
1  2019-08-12  1179.209961  1184.959961  1167.671997  1174.709961
2  2019-08-13  1171.459961  1204.780029  1171.459961  1197.270020
3  2019-08-14  1176.310059  1182.300049  1160.540039  1164.290039
4  2019-08-15  1163.500000  1175.839966  1162.109985  1167.260010

     Adj Close   Volume
0  1188.010010  1065700
1  1174.709961  1003000
2  1197.270020  1294400
3  1164.290039  1578700
4  1167.260010  1218700
```

Checking Null Values:

Having missing values in a dataset affects the analysis of the data, so it is very important to remove missing values or fill them in. But before you move on to fill in or delete your data, you need to know how many missing values you have. So here's how to find missing values in a dataset:

In [12]:
```
print(data.isnull().sum())
```

```
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

In [13]:
```
data.dropna()
```

Out[13]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2019-08-09	1197.989990	1203.880005	1183.603027	1188.010010	1188.010010	1065700
1	2019-08-12	1179.209961	1184.959961	1167.671997	1174.709961	1174.709961	1003000
2	2019-08-13	1171.459961	1204.780029	1171.459961	1197.270020	1197.270020	1294400
3	2019-08-14	1176.310059	1182.300049	1160.540039	1164.290039	1164.290039	1578700
4	2019-08-15	1163.500000	1175.839966	1162.109985	1167.260010	1167.260010	1218700
...	...	...	...	...	...	...	...
247	2020-08-03	1486.640015	1490.469971	1465.640015	1474.449951	1474.449951	2330200
248	2020-08-04	1476.569946	1485.560059	1458.650024	1464.969971	1464.969971	1903500
249	2020-08-05	1469.300049	1482.410034	1463.459961	1473.609985	1473.609985	1979500
250	2020-08-06	1471.750000	1502.390015	1466.000000	1500.099976	1500.099976	1995400
251	2020-08-07	1500.000000	1516.844971	1481.640015	1494.489990	1494.489990	1576600

252 rows × 7 columns

Filling Missing Values:

In [14]:
```
data.fillna(0)
```

Out[14]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2019-08-09	1197.989990	1203.880005	1183.603027	1188.010010	1188.010010	1065700
1	2019-08-12	1179.209961	1184.959961	1167.671997	1174.709961	1174.709961	1003000
2	2019-08-13	1171.459961	1204.780029	1171.459961	1197.270020	1197.270020	1294400
3	2019-08-14	1176.310059	1182.300049	1160.540039	1164.290039	1164.290039	1578700
4	2019-08-15	1163.500000	1175.839966	1162.109985	1167.260010	1167.260010	1218700
...	...	...	...	...	...	...	...
247	2020-08-03	1486.640015	1490.469971	1465.640015	1474.449951	1474.449951	2330200
248	2020-08-04	1476.569946	1485.560059	1458.650024	1464.969971	1464.969971	1903500
249	2020-08-05	1469.300049	1482.410034	1463.459961	1473.609985	1473.609985	1979500
250	2020-08-06	1471.750000	1502.390015	1466.000000	1500.099976	1500.099976	1995400
251	2020-08-07	1500.000000	1516.844971	1481.640015	1494.489990	1494.489990	1576600

252 rows × 7 columns

Query Data:

In [15]:
```
print(data.query("Close > 1500"))
```

```
           Date         Open         High          Low        Close  \
126  2020-02-10  1474.319946  1509.500000  1474.319946  1508.680054
127  2020-02-11  1511.810059  1529.630005  1505.637939  1508.790039
128  2020-02-12  1514.479980  1520.694946  1508.109985  1518.270020
129  2020-02-13  1512.689941  1527.180054  1504.599976  1514.660034
130  2020-02-14  1515.599976  1520.739990  1507.339966  1520.739990
131  2020-02-18  1515.000000  1531.630005  1512.589966  1519.670044
132  2020-02-19  1525.069946  1532.105957  1521.400024  1526.689941
133  2020-02-20  1522.000000  1529.640015  1506.819946  1518.150024
230  2020-07-09  1506.449951  1522.719971  1488.084961  1510.989990
231  2020-07-10  1506.150024  1543.829956  1496.540039  1541.739990
232  2020-07-13  1550.000000  1577.131958  1505.243042  1511.339966
233  2020-07-14  1490.310059  1522.949951  1483.500000  1520.579956
234  2020-07-15  1523.130005  1535.329956  1498.000000  1513.640015
235  2020-07-16  1500.000000  1518.689941  1486.310059  1518.000000
236  2020-07-17  1521.619995  1523.439941  1498.420044  1515.550049
237  2020-07-20  1515.260010  1570.290039  1503.599976  1565.719971
238  2020-07-21  1586.989990  1586.989990  1554.280029  1558.420044
239  2020-07-22  1560.500000  1570.000000  1546.099976  1568.489990
240  2020-07-23  1566.969971  1571.869995  1507.391968  1515.680054
241  2020-07-24  1498.930054  1517.635986  1488.400024  1511.869995
242  2020-07-27  1515.599976  1540.969971  1515.209961  1530.199951
243  2020-07-28  1525.180054  1526.479980  1497.660034  1500.339966
244  2020-07-29  1506.319946  1531.251953  1501.329956  1522.020020
245  2020-07-30  1497.000000  1537.869995  1492.219971  1531.449951
250  2020-08-06  1471.750000  1502.390015  1466.000000  1500.099976

       Adj Close   Volume
126  1508.680054  1419900
127  1508.790039  1344600
128  1518.270020  1167600
129  1514.660034   929500
130  1520.739990  1197800
131  1519.670044  1120700
132  1526.689941   949300
133  1518.150024  1096600
230  1510.989990  1423300
231  1541.739990  1856300
232  1511.339966  1846400
233  1520.579956  1585000
234  1513.640015  1610700
235  1518.000000  1519300
236  1515.550049  1456700
237  1565.719971  1557300
238  1558.420044  1218600
239  1568.489990   932000
240  1515.680054  1627600
241  1511.869995  1544000
242  1530.199951  1246000
243  1500.339966  1702200
244  1522.020020  1106500
245  1531.449951  1671400
250  1500.099976  1995400
```

Sorting Values:

In [16]:
```
print(data.sort_values(by="Close"))
```

```
           Date         Open         High          Low        Close  \
155  2020-03-23  1061.319946  1071.319946  1013.536011  1056.619995
154  2020-03-20  1135.719971  1143.989990  1065.489990  1072.319946
150  2020-03-16  1096.000000  1152.266968  1074.439941  1084.329956
152  2020-03-18  1056.510010  1106.500000  1037.280029  1096.800049
164  2020-04-03  1119.015015  1123.540039  1079.810059  1097.880005
..          ...          ...          ...          ...          ...
245  2020-07-30  1497.000000  1537.869995  1492.219971  1531.449951
231  2020-07-10  1506.150024  1543.829956  1496.540039  1541.739990
238  2020-07-21  1586.989990  1586.989990  1554.280029  1558.420044
237  2020-07-20  1515.260010  1570.290039  1503.599976  1565.719971
239  2020-07-22  1560.500000  1570.000000  1546.099976  1568.489990

       Adj Close   Volume
155  1056.619995  4044100
154  1072.319946  3601800
150  1084.329956  4252400
152  1096.800049  4233400
164  1097.880005  2313400
..           ...      ...
245  1531.449951  1671400
231  1541.739990  1856300
238  1558.420044  1218600
237  1565.719971  1557300
239  1568.489990   932000

[252 rows x 7 columns]
```

In [17]:
```
print(data.sort_values(by="Close", ascending=False))
```

```
           Date         Open         High          Low        Close  \
239  2020-07-22  1560.500000  1570.000000  1546.099976  1568.489990
237  2020-07-20  1515.260010  1570.290039  1503.599976  1565.719971
238  2020-07-21  1586.989990  1586.989990  1554.280029  1558.420044
231  2020-07-10  1506.150024  1543.829956  1496.540039  1541.739990
245  2020-07-30  1497.000000  1537.869995  1492.219971  1531.449951
..          ...          ...          ...          ...          ...
164  2020-04-03  1119.015015  1123.540039  1079.810059  1097.880005
152  2020-03-18  1056.510010  1106.500000  1037.280029  1096.800049
150  2020-03-16  1096.000000  1152.266968  1074.439941  1084.329956
154  2020-03-20  1135.719971  1143.989990  1065.489990  1072.319946
155  2020-03-23  1061.319946  1071.319946  1013.536011  1056.619995

       Adj Close   Volume
239  1568.489990   932000
237  1565.719971  1557300
238  1558.420044  1218600
231  1541.739990  1856300
245  1531.449951  1671400
..           ...      ...
164  1097.880005  2313400
152  1096.800049  4233400
150  1084.329956  4252400
154  1072.319946  3601800
155  1056.619995  4044100

[252 rows x 7 columns]
```

Descriptive Statistics:

To get the descriptive statistical information about your data, Pandas provides the describe() function that returns:

1. the total of all the columns
2. mean value of all the columns
3. the standard deviation of all the columns
4. minimum and maximum values of all the columns
5. 1st, 2nd, and 3rd quartile of all the columns

In [18]:
```
print(data .describe())
```

```
              Open         High          Low        Close    Adj Close  \
count   252.000000   252.000000   252.000000   252.000000   252.000000
mean   1330.245284  1345.712141  1316.713037  1332.321488  1332.321488
std     121.453125   120.306284   122.092082   121.333070   121.333070
min    1056.510010  1071.319946  1013.536011  1056.619995  1056.619995
25%    1230.180023  1243.845001  1218.210022  1230.957550  1230.957550
50%    1334.229981  1350.729981  1323.200012  1338.174988  1338.174988
75%    1433.782501  1443.512512  1421.597473  1436.064972  1436.064972
max    1586.989990  1586.989990  1554.280029  1568.489990  1568.489990

             Volume
count  2.520000e+02
mean   1.708384e+06
std    7.665229e+05
min    3.475000e+05
25%    1.218225e+06
50%    1.515100e+06
75%    1.905950e+06
max    4.267700e+06
```

Correlation:

In [19]:
```
print(data.corr())
```

```
               Open      High       Low     Close  Adj Close    Volume
Open       1.000000  0.993979  0.992965  0.986880   0.986880 -0.184352
High       0.993979  1.000000  0.989503  0.992714   0.992714 -0.139278
Low        0.992965  0.989503  1.000000  0.992617   0.992617 -0.248279
Close      0.986880  0.992714  0.992617  1.000000   1.000000 -0.195943
Adj Close  0.986880  0.992714  0.992617  1.000000   1.000000 -0.195943
Volume    -0.184352 -0.139278 -0.248279 -0.195943  -0.195943  1.000000
```

```
/tmp/ipykernel_9822/3359323643.py:1: FutureWarning: The default value of numeric_only
in DataFrame.corr is deprecated. In a future version, it will default to False. Selec
t only valid columns or specify the value of numeric_only to silence this warning.
  print(data.corr())
```

In [ ]:

Count Number of Words in a Column using Python

In [20]:
```
import pandas as pd
data = pd.read_csv("https://raw.githubusercontent.com/amankharwal/Website-data/master/
print(data.head())
```

```
                                                Article  \
0  Data analysis is the process of inspecting and...
1  The performance of a machine learning algorith...
2  You must have seen the news divided into categ...
3  When there are only two classes in a classific...
4  The Multinomial Naive Bayes is one of the vari...

                                                 Title
0                  Best Books to Learn Data Analysis
1         Assumptions of Machine Learning Algorithms
2          News Classification with Machine Learning
3  Multiclass Classification Algorithms in Machin...
4        Multinomial Naive Bayes in Machine Learning
```

In [21]:

```
data["Number of Words"] = data["Article"].apply(lambda n : len(n.split()))
print(data.head())
```

```
                                                Article  \
0  Data analysis is the process of inspecting and...
1  The performance of a machine learning algorith...
2  You must have seen the news divided into categ...
3  When there are only two classes in a classific...
4  The Multinomial Naive Bayes is one of the vari...

                                                 Title  Number of Words
0                  Best Books to Learn Data Analysis               76
1         Assumptions of Machine Learning Algorithms               56
2          News Classification with Machine Learning               70
3  Multiclass Classification Algorithms in Machin...               66
4        Multinomial Naive Bayes in Machine Learning               96
```

In [ ]:

Summary

The pandas library doesn't have any method to count the number of words in a piece of text. One way to solve this problem is by finding the length of the text by splitting the complete text. So, this is how you can count the number of words in any column while working on a textual dataset. I hope you liked this concept using Python

In [ ]: