Pandas - A Brief Introduction

An Outlook

Pandas is a library in python which is used mostly for Data Visualisation. Pandas stands for **Python for Data Analaysis**. Before we head in to this, there are some points to be noted down:

- CSV file stands for *comma separated value*. This is a text file in which values are separated by commas.
- **Data Structure** means to manage and organize the data. A variable that can contain lists or dictionaries is called a data structure
- Anything that can hold multiple values is known to be a data structure.
- Pandas primarily deals with two main data structures: Series and DataFrame.

Pandas

Pandas are libraries which are imported through the syntax import pandas as pd. Here, pd is a wordly recognized convention. We can use other words such as "p" or "ps" etc. But its the best practice to use pd.

Data Frames

1. A Pandas **DataFrame** is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.
2. A DataFrame is a data structure that organizes data into a 2-dimensional table of rows and columns, much like a spreadsheet.

As we dig deeper, we will see how this all works. Lets get started!

Open a CSV File

- To open a csv file, import the pandas library then read the file
- read is an attribute and csv is a type of file that is going to be read. pd iis the pandas library.
- To see what type to variable **who** holds, we can use the type function

[1]:

```
import pandas as pd
who = pd.read_csv("WHO.csv")
```

[2]:

```
type(who) #Holds the DataFrame type of Data
```

[2]:

```
pandas.core.frame.DataFrame
```

We can also see how many*dimensions* are held by the variable, who. To see that we use the .ndim method

[3]:

```
who.ndim # 2 Dimensional
```

[3]:

```
2
```

[4]:

```
who
```

[4]:

	Date_reported	Country	New_cases	Cumulative_cases	New_deaths	Cumulative_deaths
0	2020-01-04	China	1	1	0	0
1	2020-01-05	China	0	1	0	0
2	2020-01-06	China	3	4	0	0
3	2020-01-07	China	0	4	0	0
4	2020-01-08	China	0	4	0	0
...	...	...	...	...	...	...
31871	2020-07-31	Panama	1046	63269	25	1374
31872	2020-07-31	Timor-Leste	0	24	0	0
31873	2020-07-31	Guatemala	1221	48826	32	1867
31874	2020-07-31	Saint Vincent and the Grenadines	0	52	0	0

	Date_reported	Country	New_cases	Cumulative_cases	New_deaths	Cumulative_deaths
31875	2020-07-31	Democratic Republic of the Congo	79	9009	2	214

31876 rows × 6 columns

In the above cell, we used who to display the table. In this:

- There are ellipses which are a built-in singleton object representing an infinite or unspecified number of arguments.
- Ellipses are three dots (...)

Head method

The head() method gives us the number of rows specified. The default is 5 where as we can specify our own too. If we do this: head() this will display default 5 rows and if we specify number head(20), it will display 20 rows

[5]:

```
who.head(20)
```

[5]:

	Date_reported	Country	New_cases	Cumulative_cases	New_deaths	Cumulative_deaths
0	2020-01-04	China	1	1	0	0
1	2020-01-05	China	0	1	0	0
2	2020-01-06	China	3	4	0	0
3	2020-01-07	China	0	4	0	0
4	2020-01-08	China	0	4	0	0
5	2020-01-09	China	0	4	0	0
6	2020-01-10	China	0	4	0	0
7	2020-01-11	China	41	45	1	1
8	2020-01-12	China	0	45	0	1
9	2020-01-13	Thailand	5	5	0	0
10	2020-01-13	China	0	45	0	1
11	2020-01-14	China	0	45	0	1
12	2020-01-14	Japan	5	5	0	0
13	2020-01-14	Thailand	0	5	0	0
14	2020-01-15	Thailand	0	5	0	0
15	2020-01-15	Japan	0	5	0	0
16	2020-01-15	China	0	45	0	1
17	2020-01-16	Thailand	0	5	0	0
18	2020-01-16	Japan	0	5	0	0

	Date_reported	Country	New_cases	Cumulative_cases	New_deaths	Cumulative_deaths
19	2020-01-16	China	0	45	0	1

Time Series
* Time series analysis helps organizations understand the underlying causes of trends or systemic patterns over time
* A Series is a one-dimensional array with a time label for each row.
* If we call a single column, that will be known to be a one dimensional series.

How to Call Single Column?
* To call any column from the data set, we can simply use the column name inside `[" "]` or name of column after `.`.
* To call multiple columns, we specify ranges

[6]:

```
# Single Column by Inverted Commas - Method 1
who["Date_reported"]
```

[6]:

```
0      2020-01-04
1      2020-01-05
2      2020-01-06
3      2020-01-07
4      2020-01-08
         ...
31871   2020-07-31
31872   2020-07-31
31873   2020-07-31
31874   2020-07-31
31875   2020-07-31
Name: Date_reported, Length: 31876, dtype: object
```

[7]:

```
# Single Column using .ColumnName - Method 2
who.Date_reported
```

[7]:

```
0      2020-01-04
1      2020-01-05
2      2020-01-06
3      2020-01-07
4      2020-01-08
         ...
31871   2020-07-31
31872   2020-07-31
31873   2020-07-31
31874   2020-07-31
31875   2020-07-31
Name: Date_reported, Length: 31876, dtype: object
```

In the above examples, we can use any method. But .ColumnName should be used if the column name is not separated by Space.

Info Method

The info() method provides **granular information** about the whole data. It specifies everything such as the data type, number of rows and columns, amount of storage etc.

[8]:

```
who.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31876 entries, 0 to 31875
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Date_reported    31876 non-null  object
 1   Country          31876 non-null  object
 2   New_cases        31876 non-null  int64
 3   Cumulative_cases  31876 non-null  int64
 4   New_deaths       31876 non-null  int64
 5   Cumulative_deaths  31876 non-null  int64
dtypes: int64(4), object(2)
memory usage: 1.5+ MB
```

Now in the above table, we can see that the **Data Type** for *Date_reported* is **object** which should not be the case. The rest of all of them is fine. To fix this, we can do the following:

1. use .to_datetime syntax
2. Then specify the name of column

[10]:

```
pd.to_datetime(who["Date_reported"])
```

[10]:

```
0      2020-01-04
1      2020-01-05
2      2020-01-06
3      2020-01-07
4      2020-01-08
         ...
31871   2020-07-31
31872   2020-07-31
31873   2020-07-31
```

```
31874   2020-07-31
31875   2020-07-31
Name: Date_reported, Length: 31876, dtype: datetime64[ns]
```

Still if we use .info() it will show us the **object** DType for Date_reported. In order to fix this, simply use variable assignment. The syntax can be of: who["Date_reported"] = pd.to_datetime(who["Date_reported"])

[15]:

```
who["Date_reported"] = pd.to_datetime(who["Date_reported"])
who.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31876 entries, 0 to 31875
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype         
---  ------           --------------  -----         
 0   Date_reported    31876 non-null  datetime64[ns]
 1   Country          31876 non-null  object        
 2   New_cases        31876 non-null  int64         
 3   Cumulative_cases  31876 non-null  int64         
 4   New_deaths       31876 non-null  int64         
 5   Cumulative_deaths  31876 non-null  int64         
dtypes: datetime64[ns](1), int64(4), object(1)
memory usage: 1.5+ MB
```

Multiple Columns

To get multiple columns we can pass a list with the column names inside []. Now when I say list, I mean [["Column A", "Column B"]]

[16]:

```
who[["Date_reported", "Cumulative_cases"]]
```

[16]:

	Date_reported	Cumulative_cases
0	2020-01-04	1
1	2020-01-05	1
2	2020-01-06	4
3	2020-01-07	4
4	2020-01-08	4
...	...	...
31871	2020-07-31	63269
31872	2020-07-31	24
31873	2020-07-31	48826
31874	2020-07-31	52
31875	2020-07-31	9009

31876 rows × 2 columns

Unique() Method

The unique() method is from the time series. The unique() function is used to get unique values of Series object. Uniques are returned in order of appearance. In the below example, we will use Country attribute with unique() method that will display all the countries in the data.

[17]:

```
who.Country.unique()
```

[17]:

```
array(['China', 'Thailand', 'Japan', 'Republic of Korea',
       'United States of America', 'Singapore', 'Nepal', 'France',
       'Viet Nam', 'Malaysia', 'Australia', 'Canada', 'Sri Lanka',
       'Cambodia', 'Germany', 'Italy', 'Finland', 'United Arab Emirates',
       'India', 'Philippines', 'Sweden', 'Russian Federation',
       'The United Kingdom', 'Spain', 'Belgium', 'Other', 'Egypt',
       'Iran (Islamic Republic of)', 'Israel', 'Lebanon', 'Oman',
       'Bahrain', 'Switzerland', 'Iraq', 'Afghanistan', 'Kuwait',
       'Algeria', 'Austria', 'Romania', 'Norway', 'North Macedonia',
```

'Croatia', 'Georgia', 'Pakistan', 'Brazil', 'Denmark', 'Greece',
'Estonia', 'Nigeria', 'Lithuania', 'New Zealand', 'Mexico',
'Netherlands', 'San Marino', 'Ecuador', 'Monaco', 'Qatar',
'Azerbaijan', 'Belarus', 'Armenia', 'Czechia', 'Ireland',
'Iceland', 'Luxembourg', 'Morocco', 'Saint Barthélemy', 'Portugal',
'Tunisia', 'Saint Martin', 'Indonesia', 'Jordan',
'Dominican Republic', 'Latvia', 'Saudi Arabia', 'Senegal',
'Andorra', 'Argentina', 'Ukraine', 'Gibraltar', 'Chile', 'Poland',
'Slovenia', 'South Africa', 'Bosnia and Herzegovina',
'occupied Palestinian territory, including east Jerusalem',
'Liechtenstein', 'Hungary', 'Faroe Islands', 'Holy See', 'Serbia',
'Cameroon', 'Slovakia', 'Colombia', 'Bhutan', 'Martinique', 'Togo',
'French Guiana', 'Maldives', 'Malta', 'Peru',
'Republic of Moldova', 'Costa Rica', 'Paraguay', 'Bangladesh',
'Bulgaria', 'Albania', 'Guernsey', 'Cyprus', 'Panama',
'Brunei Darussalam', 'Mongolia', 'Jamaica', 'Turkey',
'Democratic Republic of the Congo', 'Burkina Faso',
'Bolivia (Plurinational State of)', 'Côte d'Ivoire', 'Jersey',
'French Polynesia', 'Honduras', 'Guyana',
'Saint Vincent and the Grenadines', 'Cayman Islands', 'Cuba',
'United States Virgin Islands', 'Curaçao', 'Réunion',
'Antigua and Barbuda', 'Trinidad and Tobago', 'Kosovo[1]',
'Central African Republic', 'Ghana', 'Guadeloupe', 'Kazakhstan',
'Gabon', 'Guinea', 'Kenya', 'Sudan', 'Namibia', 'Puerto Rico',
'Venezuela (Bolivarian Republic of)', 'Mayotte',
'Equatorial Guinea', 'Eswatini', 'Ethiopia', 'Mauritania',
'Uruguay', 'Congo', 'Seychelles', 'Suriname', 'Guatemala',
'Saint Lucia', 'Rwanda', 'Somalia', 'Uzbekistan',
'United Republic of Tanzania', 'Benin', 'Guam', 'Aruba', 'Liberia',
'Montenegro', 'Bahamas', 'Sint Maarten', 'Bermuda', 'Montserrat',
'Djibouti', 'Mauritius', 'Kyrgyzstan', 'Gambia', 'Barbados',
'Greenland', 'Zambia', 'Nicaragua', 'New Caledonia', 'Niger',
'Chad', 'Fiji', 'El Salvador', 'Haiti', 'Cabo Verde',
'Papua New Guinea', 'Zimbabwe', 'Isle of Man', 'Timor-Leste',
'Uganda', 'Eritrea', 'Angola', 'Madagascar', 'Grenada',
'Syrian Arab Republic', 'Mozambique', 'Dominica', 'Belize',
"Lao People's Democratic Republic", 'Myanmar',
'Turks and Caicos Islands', 'Libya', 'British Virgin Islands',
'Anguilla', 'Mali', 'Guinea-Bissau', 'Saint Kitts and Nevis',
'Northern Mariana Islands (Commonwealth of the)', 'Sierra Leone',
'Burundi', 'Botswana', 'Malawi',
'Bonaire, Sint Eustatius and Saba', 'Falkland Islands (Malvinas)',
'Sao Tome and Principe', 'South Sudan',
'Saint Pierre and Miquelon', 'Yemen', 'Comoros', 'Tajikistan',
'Lesotho'], dtype=object)

To find any specific country we can do the below:

[18]:

```
Pakistan = who.Country == "Pakistan".title()
Pakistan
```

[18]:

```
0      False
1      False
2      False
3      False
4      False
       ...
31871   False
31872   False
31873   False
31874   False
31875   False
Name: Country, Length: 31876, dtype: bool
```

Notice it, there is Boolean data type (False) because we used == operator. This operator shows either True or False. To see the country as we desire we will use it inside the actual variable of who

[19]:

```
who[Pakistan]
```

[19]:

	Date_reported	Country	New_cases	Cumulative_cases	New_deaths	Cumulative_deaths
898	2020-02-26	Pakistan	6	6	0	0
910	2020-02-27	Pakistan	0	6	0	0
995	2020-02-28	Pakistan	0	6	0	0
1022	2020-02-29	Pakistan	2	8	0	0
1113	2020-03-01	Pakistan	0	8	0	0

	Date_reported	Country	New_cases	Cumulative_cases	New_deaths	Cumulative_deaths
...	...	...	...	...	...	...
30882	2020-07-27	Pakistan	1176	274289	20	5842
31051	2020-07-28	Pakistan	936	275225	23	5865
31392	2020-07-29	Pakistan	1063	276288	27	5892
31586	2020-07-30	Pakistan	1114	277402	32	5924
31831	2020-07-31	Pakistan	903	278305	27	5951

157 rows × 6 columns

[20]:

```
Italy = who.Country == "Italy"
who[Italy]
```

[20]:

	Date_reported	Country	New_cases	Cumulative_cases	New_deaths	Cumulative_deaths
123	2020-01-29	Italy	6	6	0	0
144	2020-01-30	Italy	0	6	0	0
158	2020-01-31	Italy	0	6	0	0
187	2020-02-01	Italy	0	6	0	0
215	2020-02-02	Italy	0	6	0	0
...	...	...	...	...	...	...
30870	2020-07-27	Italy	254	246118	5	35107
31097	2020-07-28	Italy	168	246286	5	35112
31333	2020-07-29	Italy	202	246488	11	35123
31629	2020-07-30	Italy	288	246776	6	35129
31686	2020-07-31	Italy	382	247158	3	35132

185 rows × 6 columns

Other Methods/Attributes and Dimensionality

Shape

1. shape attribute in Pandas enables us to obtain the shape of a DataFrame.
2. For example, if a DataFrame has a shape of (80, 10) , this implies that the DataFrame is made up of 80 rows and 10 columns of data.

[24]:

```
who.shape # This means Rows of 31876 and Columns of 6
```

[24]:

```
(31876, 6)
```

Dimensionality

Remember:

1. DataFrame is of 2 Dimensions
2. Time series is of 1 Dimension
3. Images are 3 Dimensional
4. A table is 2 dimensional