

### install and import

```
installing pandas
pip install pandas

pip install pandas

import pandas as pd
```

### Reading and describing

```
pd -> pandas
df-> dataframe

to read a file into a dataframe
df= pd.read_csv('filename')
```

```
look at the first 5 lines
df.head()
```

```
to describe df
df.describe()
```

```
df.info()
```

```
to print all the column names
telecom_data.columns
```

```
to get the dimension of df
df.shape
```

### Sorting and filtering

#### sort

```
sorting can be done column wise - default is ascending
df.sort_values(by='Total day charge')
```

```
df.sort_val-      Sort values by col1 in
ues(col1)         ascending order (use
                  ascending =False for
                  descending sort)
```

```
df.sort_val-      Sort values by col1 in
ues([col1,co-     ascending order then
l2],ascendin-     col2 in descending
g=[True,F-       order
alse])
```

#### Filtering

```
df[condition]     #eg:
df[df['col']>5]
```

```
df[df['col']      Rows where the
> 0.5]           column col is greater
                than 0.5
```

```
df[(df[col] >   Rows where 0.7 > col
0.5) &         > 0.5
(df[col] <
0.7)]
```

### Inplace

### Rows and columns

```
to delete a row - [axis=0 means rows]
new_df = df.drop([2,3],axis = 0)
#this drops the row with index
2,3
```

```
to delete a column- [axis=1 means columns]
new_df = df.drop(['col1','c-
ol2'],axis = 0) #this drops the
column with name col1 and col2
```

### Df manipulation

#### create or edit a new column

```
df['new_colname'] = 5 #this
creates a new new column with
all values as 5
```

#### create a new column

```
df['new_colname'] = [list of
values] #this creates a new
column with list of values
assigned to each corresponding
row
```

```
NOTE: df['new_colname'] = [list
of values] throws an error if the no of
items in [list of values] doesn't match no of
rows
```

#### create or edit a new row

```
df.loc[index_of_row] = [list of
items]
```

```
NOTE: df.loc[index_of_row] =
[list of items] throws an error if the
no of items doesn't match no of rows
```

### Selection

### Data Cleaning

```
df.set_index('column_-   Change the
one')                   index with a new
                        column
```

```
df.columns = ['new_col-   Rename
_name1','new_col_n-     columns
ame2','new_col_-
name3']
```

```
pd.isnull()             Checks for null
                        Values, Returns
                        Boolean Array
```

```
pd.notnull()            Opposite of
                        pd.isnull()
```

```
df.dropna()             Drop all rows
                        that contain null
                        values
```

```
df.dropna(axis=1)       Drop all
                        columns that
                        contain null
                        values
```

```
df.dropna(axis=1,thre-   Drop all rows
sh=n)                   have have less
                        than n non null
```

```
df.fillna(x)            Replace all null
                        values with x
```

### JOIN/COMBINE

```
df1.append(df2)         Adds the rows in df1
                        to the end of
                        df2 (columns should
                        be identical)
```

```
pd.concat([df1,        Adds the columns in
df2],axis=1)           df1 to the end
                        of df2 (rows should
                        be
                        identical)
```

```
df1.join(df2, o-        joins the columns in
n=col1,how='inner')    df1 with the columns
                        on
                        df2 where the rows
                        for col have identical
                        values. how can be
                        one of 'left',
                        'right', 'outer', 'inner'
```

left = takes the index of left df

right =takes the index of left

outer = union of both keys

inner = intersection of both keys

## NOTE

`df.merge(df2)` gives you a copy of `df` merged with `df2`. you may save it to a new variable. ex `df3=df.merge(df2)`

if you want to merge `df2` to `df` right away use `inplace`. `df.merge(df2,inplace=True)`

`df[col]` Returns column with label `col` as Series

`df[[col1, col2]]` Returns multiple columns as a new DataFrame

	Country	Capital	Population
1	Belgium	Brussels	11190846
2	India	New Delhi	1303171035
3	Brazil	Brasilia	207847528

`df.iloc([0], [0])` --> 'Belgium' |  
`s.iloc[0]` | Selection by position (0th position on row and column)

`df.loc([0], ['Country'])` --> 'Belgium'

`df.ix[2]` -->

Country	Brazil
Capital	Brasilia
Population	207847528

`df.ix[1, 'Capital']` --> 'New Delhi'

`df.iloc[0,:]` | select First row

C

By **sanjeev95**  
[cheatography.com/sanjeev95/](https://cheatography.com/sanjeev95/)

Published 20th January, 2020.  
Last updated 22nd January, 2020.  
Page 1 of 2.

Sponsored by **Readable.com**  
Measure your website readability!  
<https://readable.com>