Working with Pandas

	Function	Syntax	Purpose	Additional Remarks
Creating a DataFrame	read_csv	df = pd.read_csv('Iris_data_sample.csv',index_col=0)	To read a CSV file	Index_col = 0, will prevent additional index column to be introduced and first column becomes index column
	read_excel	df=pd.read_excel("Iris_data_sample.xlsx","Iris_data",index_col=0)	To read a Excel file	Same as above
	read_table	df = pd.read_table('Iris_data_sample.txt',sep=' ')	To read a txt file	Separator is mentioned as space
	read_csv/ read_excel/ read_table	df = pd.read_csv('Iris_data_sample.csv',missing_values=['??','###']) df = pd.read_csv('Iris_data_sample.xlsx',missing_values=['??','###']) df =	To read a csv, excel or txt file with junk values	Junk values such as '??' and '###' may be read as na values

		pd.read_csv('Iris_data_sample.txt',missing_values=[‘??’,’###’],,sep=' ‘)		
Reading records of a data frame	head()	df.head()	return top n (5 by default) rows of a data frame or series	df.head(7) - will return top 7 records ‘n’ may be used as an argument for head function when top ‘n’ records are required
	tail()	df.tail()	return bottom n (5 by default) rows of a data frame or series	df.head(10) - will return top 10 records ‘n’ may be used as an argument for tail function when last/bottom ‘n’ records are required
Row and Column names	index	df.index	Attribute that returns a sequence of row names of a dataframe	These row names or indices are only used to refer to a row of a dataframe
	columns	df.columns	Attribute that returns a sequence of column names of a dataframe	These column names are used to refer to a particular column of a dataframe
Dimensions of dataframe	size	df.size	Return the number of data in the	

			dataframe	
	Shape	df.shape	Return a tuple of the number of records(rows) and number of attributes (columns)	
	ndim	df.ndim	Return number of dimensions of the dataframe	Generally 2
Access Columns	Square brackets	df[‘attr1’]	Get reference to a column of a dataframe	Enclose attribute name in either single or double quotes
	Dot operator	df.attr1	Get reference to a column of a dataframe	Use column name like an attribute name of the dataframe. No need of quotes around the column name
	Square brackets and multiple columns	df[‘attr1’,’attr2’]	Get reference to multiple columns of a dataframe	Enclose each attribute name in either single or double quotes and separate each attribute by a comma

Memory Usage	memory_usage	df.memory_usage()	Returns the number of bytes consumed by dataframe df	
	nbytes	df[‘attr1’].nbytes (or) df.attr1.nbytes	Returns the number of bytes required for a particular column	
Summary about dataframe	info	df.info()	used to print a concise summary of a DataFrame	Give information about a DataFrame including the index dtype and column dtypes, non-null values and memory usage
Unique	Unique in Pandas	df.SepalLengthCm.unique() (or) df['SepalWidthCm'].unique()	used to get unique values of Series object	Values returned are in order same as they appear in dataframe
	Unique in NumPy	np.unique(df[‘attr1’])	used to get unique values of a column	Values returned are in ascending order same
Reteriving row(s) with row names/indices	loc - for refering all rows with particular value in a column	df = pd.read_csv("Iris_data_sample.csv",index_col='Species') cat = df.loc["Iris-setosa"]	Used to get all rows with species column as “Iris-setosa”	Indexing by column name species is required

	loc - for refering all rows with start value and end value in a column	df = pd.read_csv("Iris_data_sample.csv",index_col=0,na_values=['??','###']) rows = df.loc[2:5]	Used to get all rows that has index values from 2 to 5 (inclusive)	First column in the dataset is considered as index value in the dataframe
Reterieving row elements of a particular column	loc with two arguments	rows = df.loc[:,'SepalLengthCm']	All row values of a column 'SepalLengthCm'	Reterieves all rows
	Slicing	rows = df.loc[3:10,'SepalLengthCm']	All row values with index 3 to 10 (inclusive) of a column 'SepalLengthCm'	Reterives specific rows
Reterieving row elements of a multiple columns		rows = df.loc[2:10,['SepalLengthCm','SepalWidthCm']]	Get two columns 'SepalLengthCm' and 'SepalWidthCm' of rows from index value 2 to 10	
Fetching multiple columns of random row indices		rows = df.loc[[10,11,15,21,23,25,26,29,41],'SepalLengthCm':'PetalLengthCm']	Get all column values from 'SepalLengthCm' and 'PetalLengthCm' with rows indices value	

			10,11,15,21,23,25,26,29,41	
Internal indexing of Pandas	Iloc - similar to indexing in list Fetching a row by index internally assigned by Pandas	row_3 = df.iloc[3]	Get a row with index 3	Internal indexing of rows and columns of a dataframe starts with 0
	Iloc - similiar to slicing Fetching a group of rows by index internally assigned by Pandas	row_3 = df.iloc[3:10]	Get rows with index 3 to 9	End index is not included in the fetched data
Internal reverse indexing of Pandas	Iloc - similar to reverse indexing in list Fetching a row by reverse index internally	rows = df.iloc[-2,1:3]	Get the row which is at index last but one	Reverse indexing is -1 for last record and -n for first record. Where n is the total number of rows in a file

	assigned by Pandas			
	Iloc - similiar to slicing with reverse indices Fetching a group of rows by reverse index internally assigned by Pandas	rows = df.iloc[-2:,1:3]	Get all rows from last but one	
Datatypes in Pandas	Check datatype of columns in dataframe	df.dtypes	Give datatype of all fields in the dataframe	Possible datatypes are int64, float64, object and category Object is equivalent in Python
	Find count of each datatype in column	df.dtypes.value_counts()	Give count of datatypes	
	Create a new dataframe from existing one with few datatypes	data_fil = df.select_dtypes(include=[float,int])	Create a new dataframe data_fil with only integer and float datatype	

	Create a new dataframe from existing one without few datatypes	data_fil = df.select_dtypes(exclude=[object])	Create a new dataframe data_fil all datatypes without object datatype	
Statistical measures of numerical data	count	df.count()	Returns number of non-NA/null observations of all columns	
	max	df.max()	Returns maximum of the values in the dataframe of each column	
	min	df.min()	Minimum of the values in the dataframe of each column	
	mean	df.mean()	Mean of the values of each column	
	std	df.std()	Standard deviation of each column can be found	
Describe	Give statistical description about the	df.describe()	Display only descrpition about numerical data Three percentiles	Return the following statitical measures for numerical data only: 1. Count

	dataframe		25%, 50% and 75% are also displayed. If 25% is 1.0000 then it means that 25% of your data have the value 1.0000 or below. If 25% is 1.0000 then it means that 50% of your data have the value 1.5000 or below. If 75% is 2.2500 then it means that 75% of your data have the value 2.2500 or below.	2. Max 3. Min 4. 25% 5. 50% 6. 75%
Arguments for describe	Two arguments may be given Include and exclude	df.describe(include=[float,int]) df.describe(include=[object]) df.describe(exclude=[object]) df.describe(include='all')	Include numerical datatypes only Includes object datatype only Include all datatype except object Include all datatypes	Statistical information for category and object datatype are Unique - Number of unique values in the column Top - most common value Freq - most

				common value's frequency.
Changing datatype	astype	df['SepalLengthCm'] = df['SepalLengthCm'].astype(float) df['Species'] = df['Species'].astype('category')	Change datatype of columns	Column to be assigned to itself after changes to see changes We have to put quotes for category datatype alone
Count null values of columns	isnull	df.isnull().sum()	Returns null values of a column	
Drop null values	dropna	df.dropna(inplace=True)	Drop null values	Changes will get reflected in current dataframe only if inplace = True is given Otherwise a new dataframe will be created and given
Replace values in a column	replace	df['SepalLengthCm'].replace('five',5,inplace=True)	Replace five by 5 in SepalLengthCm column	Changes will get reflected in current dataframe only if inplace = True is

				given Otherwise a new dataframe will be created and given
Conditional indexing	Works like a filter operation	df1 = df[df.SepalLengthCm>5]	Creates a new dataframe df1 with all records having SepalLengthCm>5 in df	
Copy	copy	df1= df.copy()	Create a copy of df	A separate memory is allocated
Mean, median of a column	mean/median	mean_val = df1['SepalLengthCm'].mean() mean_val = df1['SepalLengthCm'].median()	Return mean of a numerical column Returns median of a numerical column	
Mode of a column	mode	mean_val = df1['SepalLengthCm'].mode()	Since there can be more than one mode, a pandas series is returned	
Count unique values of all columns or a particular column	value_counts	val = df.value_counts() val = df['SepalLengthCm'].value_coun	Return unique combination of values of all columns and their counts Return unique	

		ts()	values of a specific column and their counts	
Replace missing values	fillna	mean_val = df['SepalLengthCm'].mean() df['SepalLengthCm'] = df['SepalLengthCm'].fillna(mean_val)	Null values will be replaced by mean value of the corressponding column	
Insert a new column		df.insert(2,'length_Converted',0)	Inserts a new column at index 2, with all values initialized to 0	
Change value of a column by broadcasting userdefined functions		def convert(val): return val*10 df['length_Converted']=convert(df['SepalLengthCm'])	User defined function is defined that takes a number and returns by multiplying it by 10	Broadcasting is done by Pandas for calling user defined functions also

To convert a column with null values to numeric

df['set_of_numbers'] = pd.to_numeric(df['set_of_numbers'], errors='coerce')