Numpy, Pandas Basics – Learning made easy

(import numpy as np)

Press tab after ' .' for method options.
Shift+tab for more info on a method.

1. Creating Numpy Arrays

np.arange() – sequence of numbers
np.random.rand() – random numbers
(several other np.random. modules available)
np.linspace() – to create equally spaced numbers
np.array(list or tuple) – to convert list or tuple into array.

2. Attributes of Arrays

my_arr.ndim – for number of dimensions of array
my_arr.shape – for the shape of the array (dimensions)
my_arr.reshape() – to reshape an array

3. Indexing/Slicing

my_arr[start:stop:step] – for 1D array
my_arr[Start:stop:step , start:stop:step] – for 2D array

4. Boolean Masking

my_arr[conditional] – for 1D array
my_arr[conditional, :] – for 2D array

5. Modifying an array

my_arr[row, column] = value
my_arr[:,column] = [values]
np.vstack(my_arr1, my_arr2) – stacking array in rows
np.hstack(my_arr1, my_arr2) – stacking array in columns
np.vsplit(my_arr, [indices]) – vertical split at indices
np.hsplit(my_arr,[indices]) – horizontal split at indices

6.Ufuncs

my_arr*n – multiplies each element of array with n
(similarly operators like +, -, /, **, can be directly applied)
np.sum(), np.mean(), np.max(), np.argmax(), np.abs(), etc
np.sort() - sorts the values in an array
np.argsort() – sorts the array, and returns the indices.
np.where(value) – returns indices where value is present
(For all available methods **np.** Tab**)**

(import pandas as pd)

- **Built on top of Numpy, Contains labelled data**

Series

1. Creating Pandas Series

pd.Series(list or tuple or array or dictionary)
my_series.shape – for the dimensions
my_series.index – for the index names

2. Indexing/Slicing

my_series.loc[index name] - indexing by names
my_series.iloc[start:stop:step] – by index position
Masking, modifying similar to dataframes.

DataFrame

1. Creating a Pandas DataFrame

pd.DataFrame(2D np array or dictionary of (lists or series or dictionaries) or csv files)
df.shape – for the dimensions
df.columns – get column names
df.index – get row or index names
df.info() – for brief info on the data frame
df.describe() – for brief statistics on numeric columns
df.reset_index() – resets index
df.set_index(index) – sets new index. Index can be a col

2. Indexing/Slicing/masking

df.loc[index names, col names] - indexing by names
(for multiple row names or col names, pass in a list)
df.iloc[start:stop:step, star:stop:step] – by position
df[col names] - for indexing on column names
df.loc[conditional, :] - for Boolean masking

3. Modification of array

df[col name] = [values]
df.assign(values) – to add new columns
df.drop(col or row name, axis)- to drop columns or rows
df.pop(col name) - will drop a column
pd.concat(df1,df2,axis) – concatenate rows or columns
pd.merge(df1,df2) – to merge on a common column
(several more options for merge) **pd.merge** SHIFT+Tab

DataFrame (continued…)

4. Few Important DataFrame Methods

df.max(axis) – max of each column or row
df.idxmax(axis) - index where value is max
df.min(), df.idxmin(), df.sum(), df.count() etc..
df.apply(func, axis) – apply a func over each col or row

df[col name].unique() – to get unique values in a column
df[col name].nunique() – # of unique values in a column
df[col name].value_counts() -unique value and frequency

df.sort_values(col name,axis)- sort df based on given col
df[col name].sort_values() – will sort values in a column

df.grouby(col name) – will create a groupby object on which we can use several other aggregation methods.
df.groupby(col name).mean() – will group by a col values and obtains mean of each group for all columns
df.groupby(col1)[col2].mean() – will group by a col1 values and obtains mean of each group for col2

5. Missing Values

df.isna(), df.isnull() – returns a Boolean data frame
df.notna(), df.notnull() – returns a Boolean data frame
df.isnull().sum(0) – returns a count of all missing values in each column

df.dropna(axis) – drops NaN values in col or rows
df.fillna(value, method, axis) – fill missing values

6. Numerical operations

Similar to numpy operations. Any Ufunc will also work on data frames or series.

7. Reading, writing files

pd.read_csv() – to read a 'csv' file
pd.read_excel() – to read excel file
df.to_csv() – to write a file

Dr.Palle, spalle@emory.edu , Big Data Analytics Program, ECE, Emory University, Atlanta, GA