Advanced Data Science

Introduction to Numpy, Pandas and Matplotlib

Session 2

Kiran Waghmare
Program Manager
C-DAC Mumbai

Agenda

- **Numpy**
- **Pandas**
- **Matplotlib**

Unit II:
Introduction to NumPy

Brief Python Data Types;
NumPy arrays;
Data Types;
Array Functions;
Universal Functions;
Aggregations;
Broadcasting;
Fancy Functions;
Sorting arrays:
Partial Sort;
Structures arrays;

Data Manipulation using Pandas;

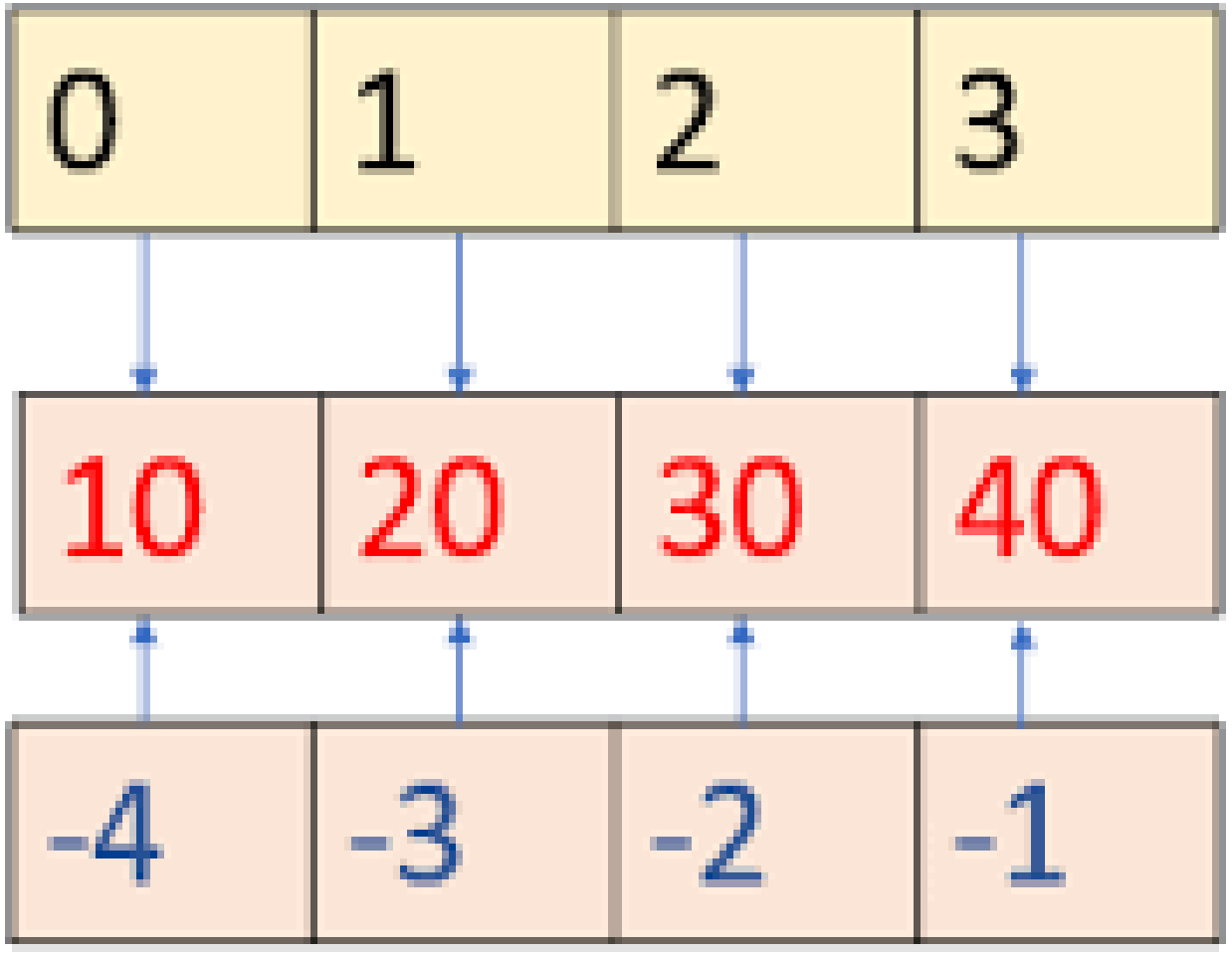
Handling Missing Data;
Hierarchical Indexing;

Visualization Using Matplotlib.

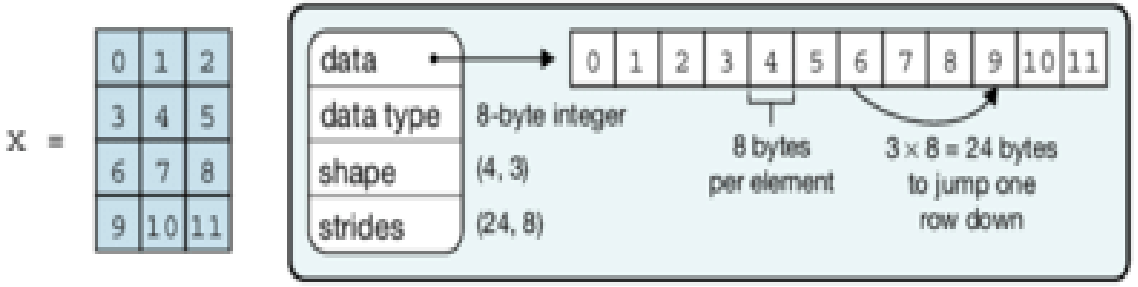
Positive Indexing

One_d

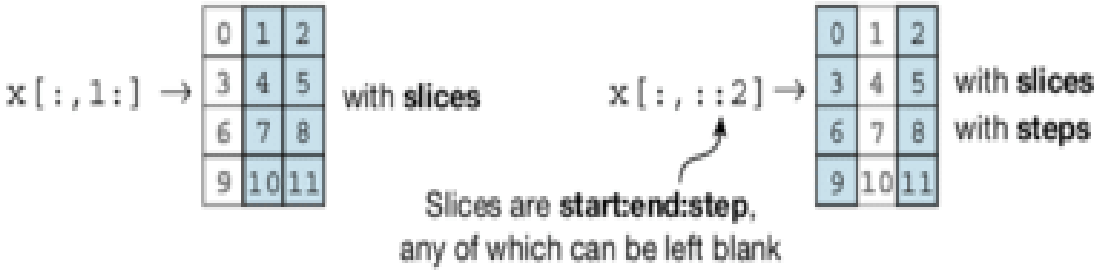
Negative Indexing



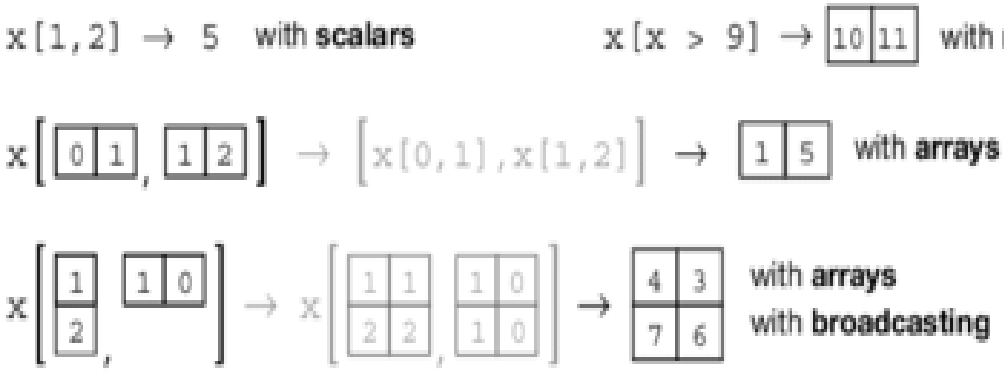
a Data structure



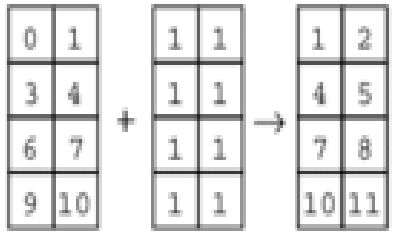
b Indexing (view)



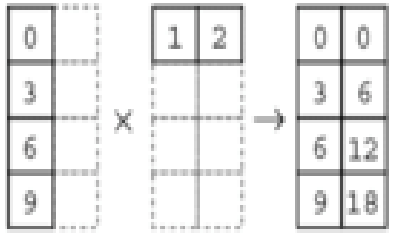
c Indexing (copy)



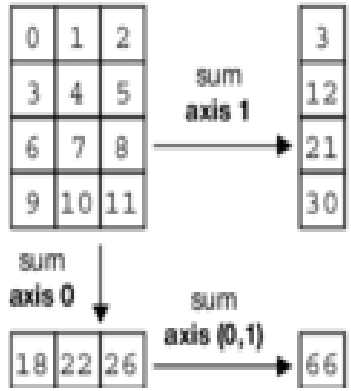
d Vectorization



e Broadcasting

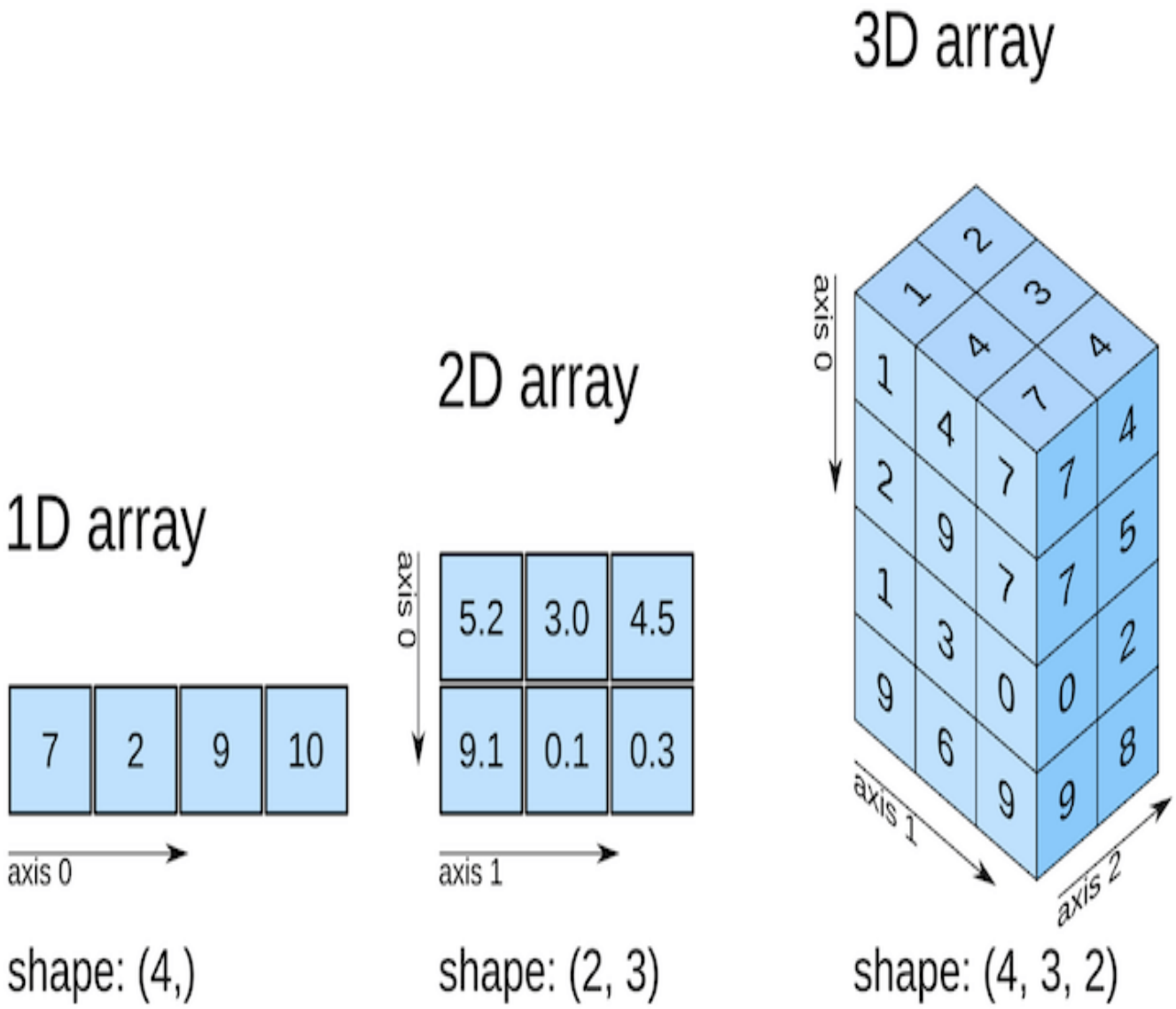


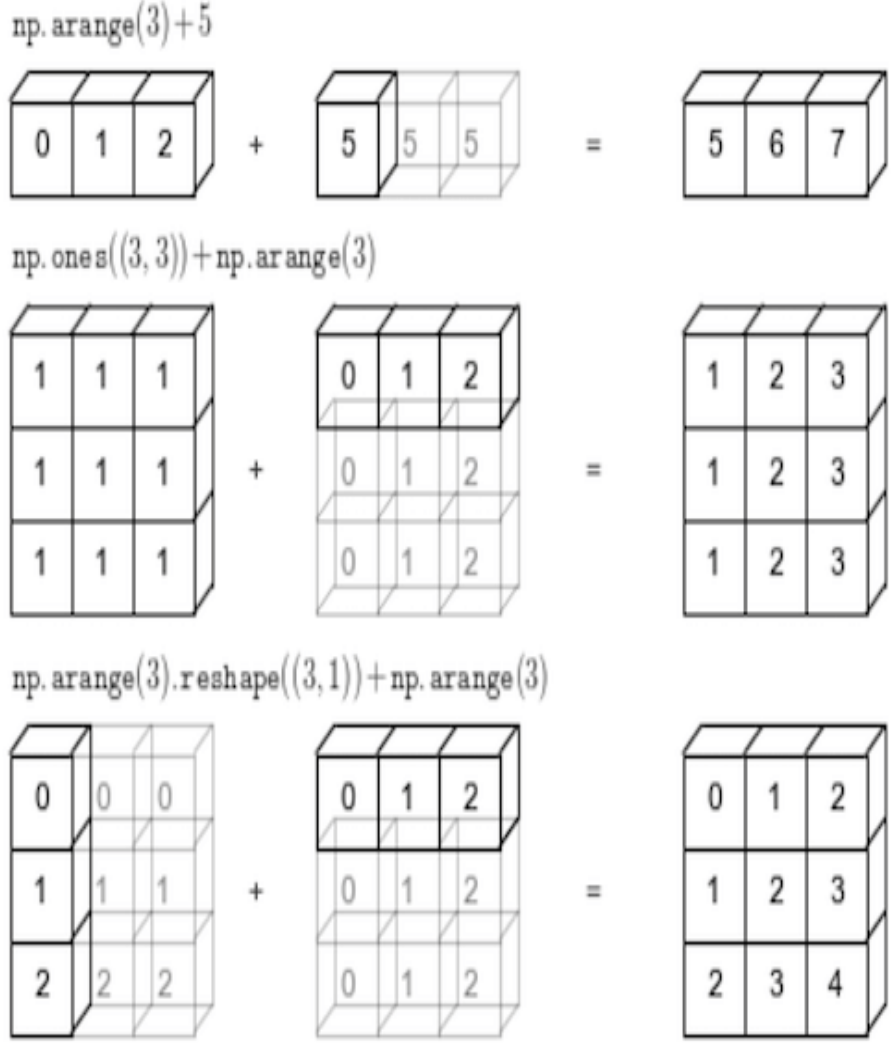
f Reduction



g Example

```
In [1]: import numpy as np

In [2]: x = np.arange(12)

In [3]: x = x.reshape(4, 3)

In [4]: x
Out[4]:
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])

In [5]: np.mean(x, axis=0)
Out[5]: array([4.5, 5.5, 6.5])

In [6]: x = x - np.mean(x, axis=0)

In [7]: x
Out[7]:
array([[-4.5, -4.5, -4.5],
       [-1.5, -1.5, -1.5],
       [ 1.5,  1.5,  1.5],
       [ 4.5,  4.5,  4.5]])
```


1D array
7 2 9 10
axis 0
shape: (4,)
2D array
axis 0
5.2 3.0 4.5
9.1 0.1 0.3
axis 1
shape: (2, 3)
3D array
axis 0
axis 1
axis 2
shape: (4, 3, 2)


np.arange(3)+5
0 1 2 + 5 5 5 = 5 6 7
np.ones((3,3))+np.arange(3)
1 1 1 / 1 1 1 / 1 1 1 + 0 1 2 / 0 1 2 / 0 1 2 = 1 2 3 / 1 2 3 / 1 2 3
np.arange(3).reshape((3,1))+np.arange(3)
0 / 1 / 2 + 0 1 2 / 0 1 2 / 0 1 2 = 0 1 2 / 1 2 3 / 2 3 4

NumPy Basics

NumPy Basics

Operator	Description
`np.array([1,2,3])`	1d array
`np.array([(1,2,3),(4,5,6)])`	2d array
`np.arange(start,stop,step)`	range array

Placeholders

Operator	Description
`np.linspace(0,2,9)`	Add evenly spaced values btw interval to array of length
`np.zeros((1,2))`	Create and array filled with zeros
`np.ones((1,2))`	Creates an array filled with ones
`np.random.random((5,5))`	Creates random array
`np.empty((2,2))`	Creates an empty array

Array

Operator	Description
`array.shape`	Dimensions (Rows,Columns)
`len(array)`	Length of Array
`array.ndim`	Number of Array Dimensions
`array.dtype`	Data Type
`array.astype(type)`	Converts to Data Type
`type(array)`	Type of Array

Copying/Sorting

Operator	Description
`np.copy(array)`	Creates copy of array
`other = array.copy()`	Creates deep copy of array
`array.sort()`	Sorts an array
`array.sort(axis=0)`	Sorts axis of array

Array Manipulation

Adding or Removing Elements

Operator	Description
`np.append(a,b)`	Append items to array
`np.insert(array, 1, 2, axis)`	Insert items into array at axis 0 or 1
`np.resize((2,4))`	Resize array to shape(2,4)
`np.delete(array,1,axis)`	Deletes items from array

Combining Arrays

Operator	Description
`np.concatenate((a,b),axis=0)`	Split an array into multiple sub-arrays.
`np.vstack((a,b))`	Split an array in sub-arrays of (nearly) identical size
`np.hstack((a,b))`	Split the array horizontally at 3rd index

More

Operator	Description
`other = ndarray.flatten()`	Flattens a 2d array to 1d
`array = np.transpose(other)`	Transpose array
`array.T`	Transpose array
`inverse = np.linalg.inv(matrix)`	Inverse of a given matrix

Slicing and Subsetting

Operator	Description
`array[i]`	1d array at index i
`array[i,j]`	2d array at index[i][j]
`array[i<4]`	Boolean Indexing, see Tricks
`array[0:3]`	Select items of index 0, 1 and 2
`array[0:2,1]`	Select items of rows 0 and 1 at column 1
`array[:1]`	Select items of row 0 (equals array[0:1, :])
`array[1:2, :]`	Select items of row 1
`[comment]: <> (`	array[1,...]
`array[ : :-1]`	Reverses array

Mathematics

Operations

Operator	Description
`np.add(x,y)`	Addition
`np.substract(x,y)`	Subtraction
`np.divide(x,y)`	Division
`np.multiply(x,y)`	Multiplication
`np.sqrt(x)`	Square Root
`np.sin(x)`	Element-wise sine
`np.cos(x)`	Element-wise cosine
`np.log(x)`	Element-wise natural log
`np.dot(x,y)`	Dot product
`np.roots([1,0,-4])`	Roots of a given polynomial coefficients

Comparison

Operator	Description
`==`	Equal
`!=`	Not equal
`<`	Smaller than
`>`	Greater than
`<=`	Smaller than or equal
`>=`	Greater than or equal
`np.array_equal(x,y)`	Array-wise comparison

Basic Statistics

Operator	Description
`np.mean(array)`	Mean
`np.median(array)`	Median
`array.corrcoef()`	Correlation Coefficient
`np.std(array)`	Standard Deviation

More

Operator	Description
`array.sum()`	Array-wise sum
`array.min()`	Array-wise minimum value
`array.max(axis=0)`	Maximum value of specified axis
`array.cumsum(axis=0)`	Cumulative sum of specified axis

Create Test Objects

Operator	Description
`pd.DataFrame(np.random.rand(20,5))`	5 columns and 20 rows of random floats
`pd.Series(my_list)`	Create a series from an iterable my_list
`df.index = pd.date_range('1900/1/30', periods=df.shape[0])`	Add a date index

Viewing/Inspecting Data

Operator	Description
`df.head(n)`	First n rows of the DataFrame
`df.tail(n)`	Last n rows of the DataFrame
`df.shape`	Number of rows and columns
`df.info()`	Index, Datatype and Memory information
`df.describe()`	Summary statistics for numerical columns
`s.value_counts(dropna=False)`	View unique values and counts
`df.apply(pd.Series.value_counts)`	Unique values and counts for all columns

Selection

Operator	Description
`df[col]`	Returns column with label col as Series
`df[[col1, col2]]`	Returns columns as a new DataFrame
`s.iloc[0]`	Selection by position
`s.loc['index_one']`	Selection by index
`df.iloc[0,:]`	First row
`df.iloc[0,0]`	First element of first column

Data Cleaning

Operator	Description
`df.columns = ['a','b','c']`	Rename columns
`pd.isnull()`	Checks for null Values, Returns Boolean Arrray
`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,thresh=n)`	Drop all rows have have less than n non null values
`df.fillna(x)`	Replace all null values with x
`s.fillna(s.mean())`	Replace all null values with the mean
`s.astype(float)`	Convert the datatype of the series to float
`s.replace(1,'one')`	Replace all values equal to 1 with 'one'
`s.replace([2,3],['two', 'three'])`	Replace all 2 with 'two' and 3 with 'three'
`df.rename(columns=lambda x: x + 1)`	Mass renaming of columns
`df.rename(columns={'old_name': 'new_ name'})`	Selective renaming
`df.set_index('column_one')`	Change the index
`df.rename(index=lambda x: x + 1)`	Mass renaming of index

Filter, Sort, and Groupby

Operator	Description
`df[df[col] > 0.6]`	Rows where the column col is greater than 0.6
`df[(df[col] > 0.6) & (df[col] < 0.8)]`	Rows where 0.8 > col > 0.6
`df.sort_values(col1)`	Sort values by col1 in ascending order
`df.sort_values(col2,ascending=False)`	Sort values by col2 in descending order.5
`df.sort_values([col1,col2],ascending=[True,False])`	Sort values by col1 in ascending order then col2 in descending order
`df.groupby(col)`	Returns a groupby object for values from one column
`df.groupby([col1,col2])`	Returns groupby object for values from multiple columns
`df.groupby(col1)[col2]`	Returns the mean of the values in col2, grouped by the values in col1
`df.pivot_table(index=col1,values=[col2,col3],aggfunc=mean)`	Create a pivot table that groups by col1 and calculates the mean of col2 and col3
`df.groupby(col1).agg(np.mean)`	Find the average across all columns for every unique col1 group
`df.apply(np.mean)`	Apply the function np.mean() across each column
`nf.apply(np.max,axis=1)`	Apply the function np.max() across each row

Join/Combine

Operator	Description
`df1.append(df2)`	Add the rows in df1 to the end of df2 (columns should be identical)
`pd.concat([df1, df2],axis=1)`	Add the columns in df1 to the end of df2 (rows should be identical)
`df1.join(df2,on=col1, how='inner')`	SQL-style join the columns in df1 with the columns on df2 where the rows for col have identical values. The 'how' can be 'left', 'right', 'outer' or 'inner'

Statistics

Operator	Description
`df.describe()`	Summary statistics for numerical columns
`df.mean()`	Returns the mean of all columns
`df.corr()`	Returns the correlation between columns in a DataFrame
`df.count()`	Returns the number of non-null values in each DataFrame column
`df.max()`	Returns the highest value in each column
`df.min()`	Returns the lowest value in each column
`df.median()`	Returns the median of each column
`df.std()`	Returns the standard deviation of each column

Importing Data

Operator	Description
`pd.read_csv(filename)`	From a CSV file
`pd.read_table(filename)`	From a delimited text file (like TSV)
`pd.read_excel(filename)`	From an Excel file
`pd.read_sql(query, connection_object)`	Read from a SQL table/database
`pd.read_json(json_string)`	Read from a JSON formatted string, URL or file.
`pd.read_html(url)`	Parses an html URL, string or file and extracts tables to a list of dataframes
`pd.read_clipboard()`	Takes the contents of your clipboard and passes it to read_table()
`pd.DataFrame(dict)`	From a dict, keys for columns names, values for data as lists

Exporting Data

Operator	Description
`df.to_csv(filename)`	Write to a CSV file
`df.to_excel(filename)`	Write to an Excel file
`df.to_sql(table_name, connection_object)`	Write to a SQL table
`df.to_json(filename)`	Write to a file in JSON format

- **Unit II:**
- **Introduction to NumPy**
- **Brief Python Data Types;**
- **NumPy arrays;**
- **Data Types;**
- **Array Functions;**
- **Universal Functions;**
- **Aggregations;**
- **Broadcasting;**
- **Fancy Functions;**
- **Sorting arrays:**
- **Partial Sort;**
- **Structures arrays;**

- **Data Manipulation using Pandas;**
- **Handling Missing Data;**
- **Hierarchical Indexing;**
-
- **Visualization Using Matplotlib.**