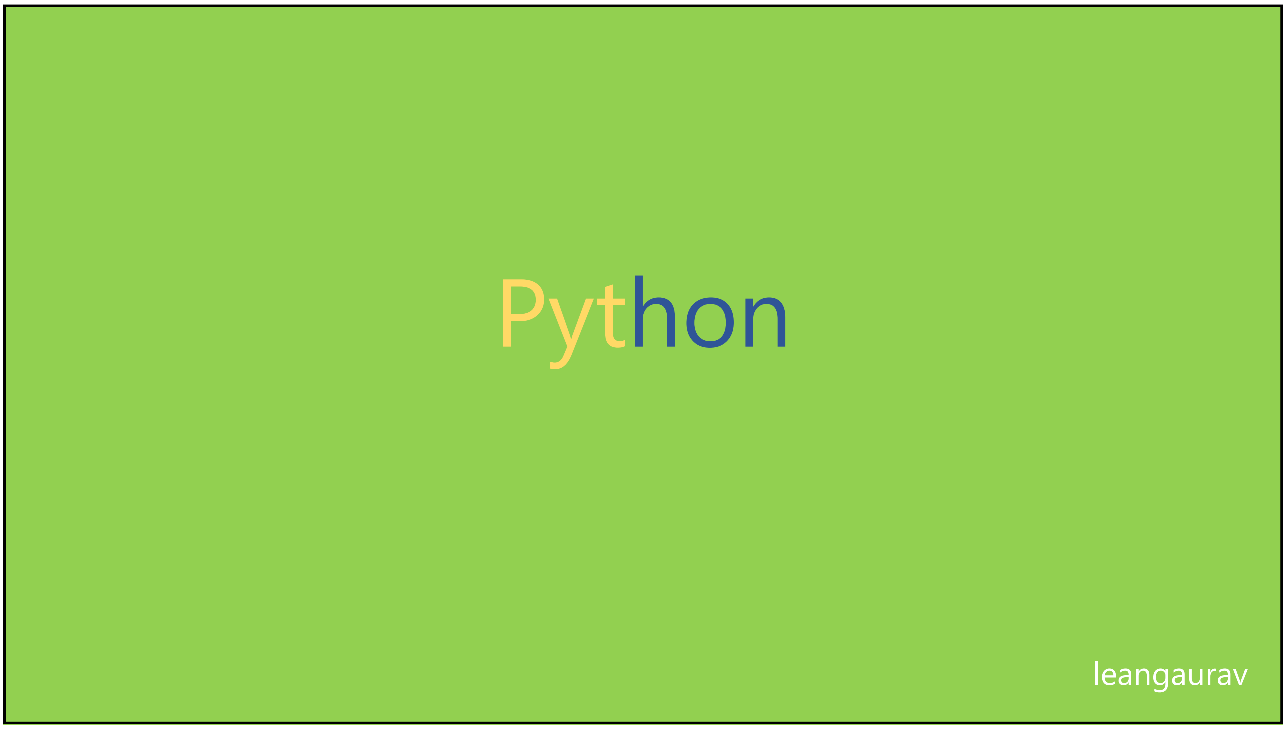
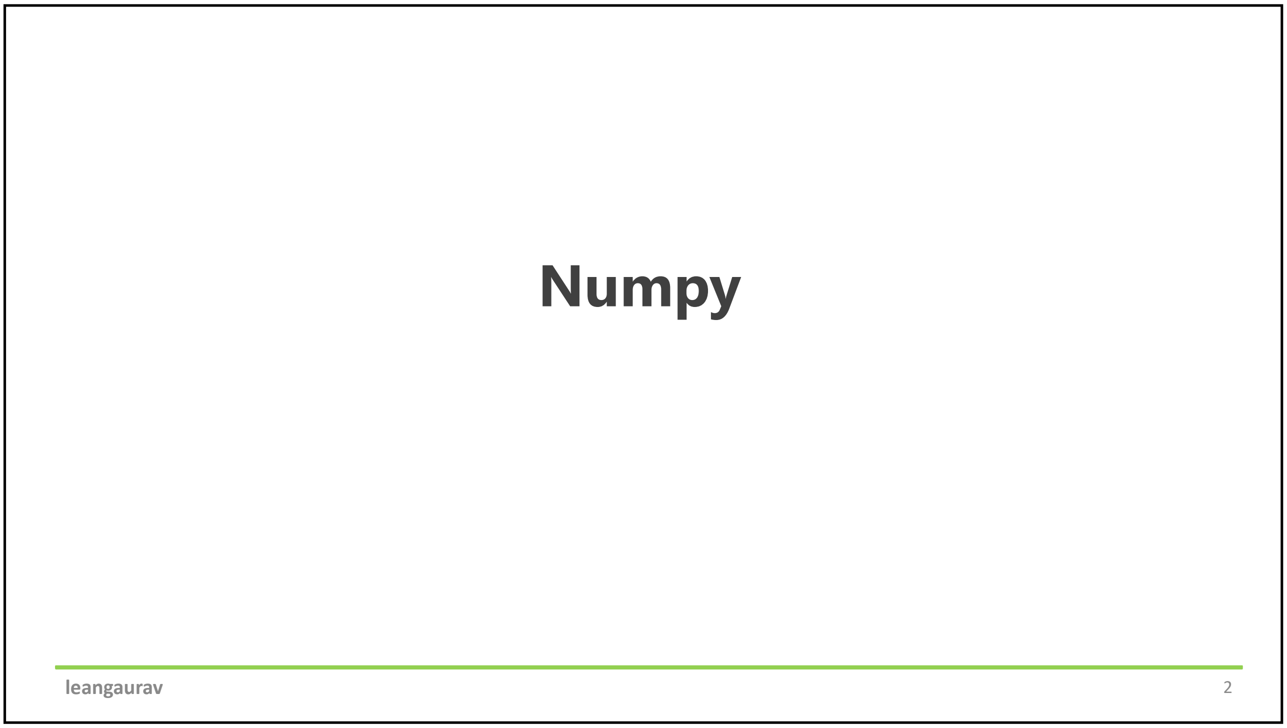


Python

leangaurav

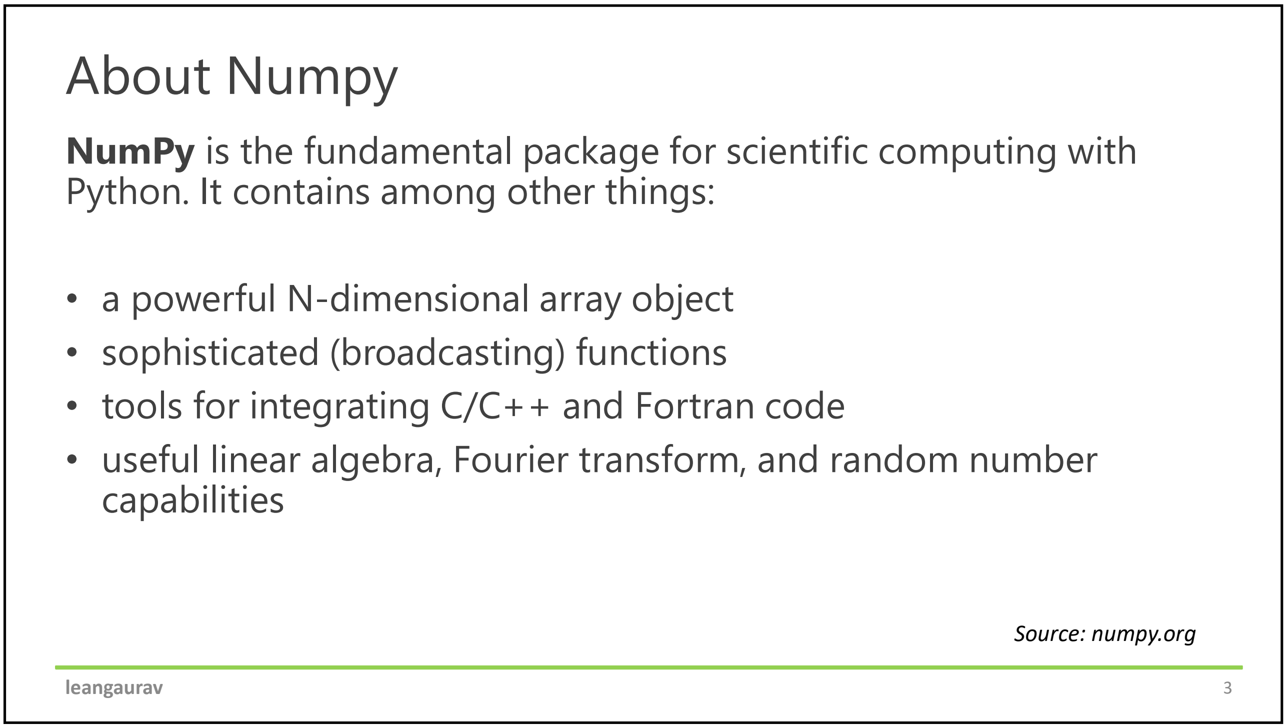
1



Numpy

leangaurav 2

2



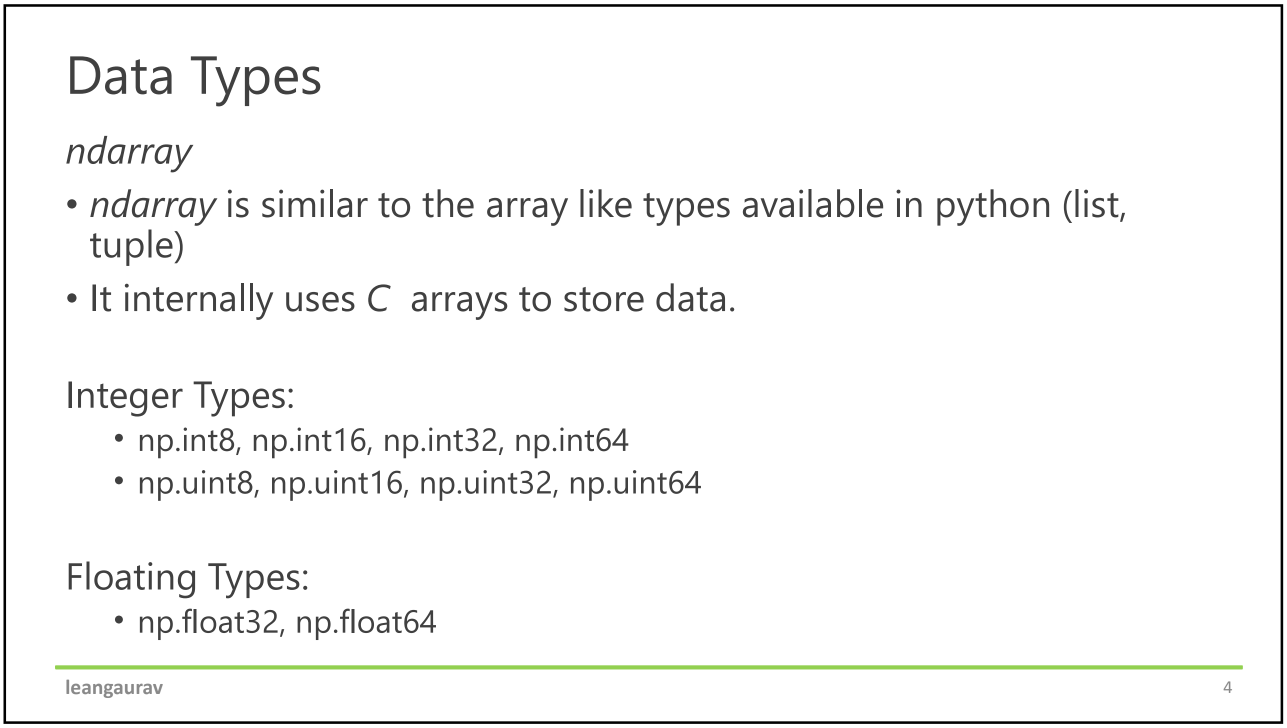
About Numpy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Source: numpy.org

3



Data Types

ndarray

- *ndarray* is similar to the array like types available in python (list, tuple)
- It internally uses *C* arrays to store data.

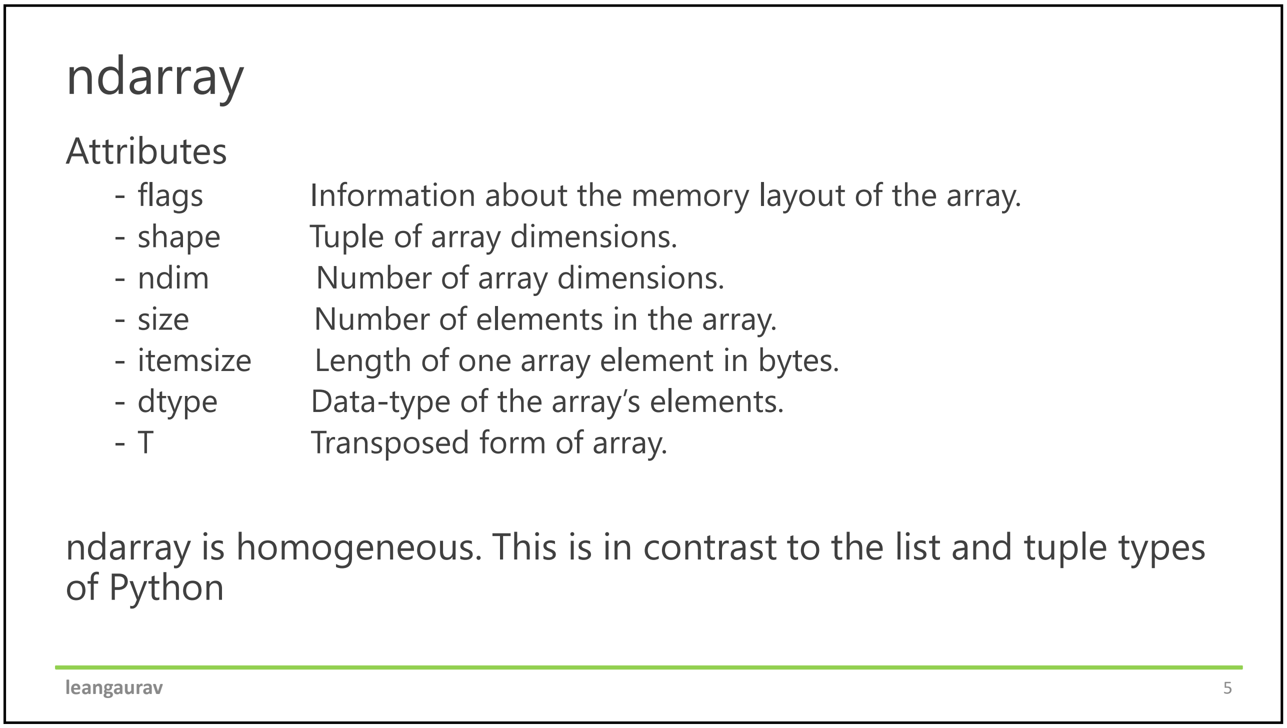
Integer Types:

- np.int8, np.int16, np.int32, np.int64
- np.uint8, np.uint16, np.uint32, np.uint64

Floating Types:

- np.float32, np.float64

4



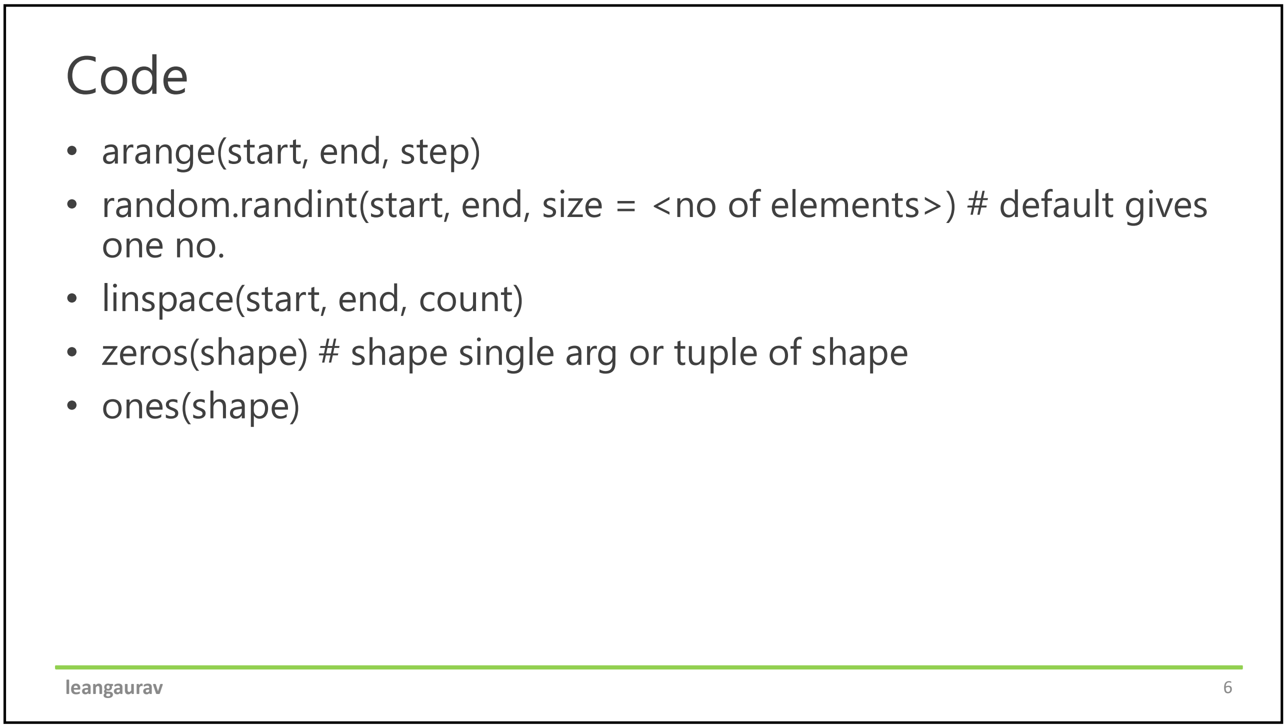
ndarray

Attributes

- flags	Information about the memory layout of the array.
- shape	Tuple of array dimensions.
- ndim	Number of array dimensions.
- size	Number of elements in the array.
- itemsize	Length of one array element in bytes.
- dtype	Data-type of the array’s elements.
- T	Transposed form of array.

ndarray is homogeneous. This is in contrast to the list and tuple types of Python

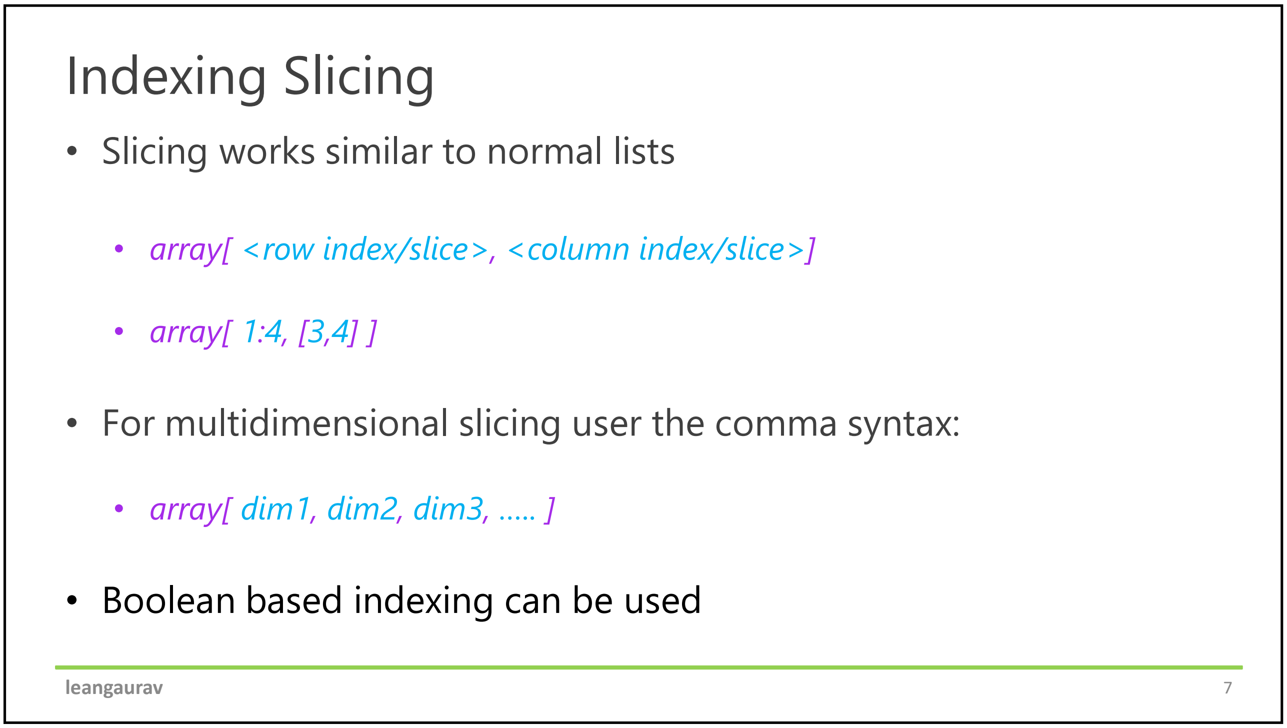
5



Code

- arange(start, end, step)
- random.randint(start, end, size = <no of elements>) # default gives one no.
- linspace(start, end, count)
- zeros(shape) # shape single arg or tuple of shape
- ones(shape)

6

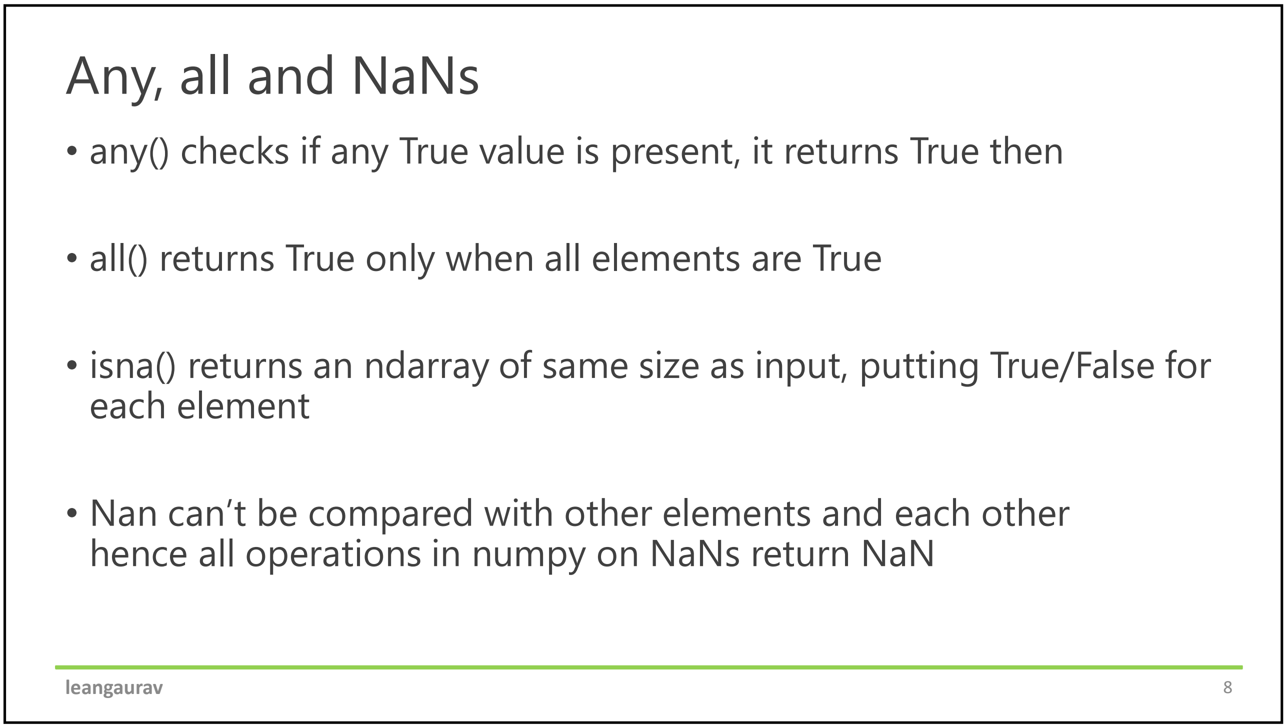


Indexing Slicing

- Slicing works similar to normal lists
  - *array[<row index/slice>, <column index/slice>]*
  - *array[1:4, [3,4]]*
- For multidimensional slicing user the comma syntax:
  - *array[dim1, dim2, dim3, …..]*
- Boolean based indexing can be used

leangaurav 7

7

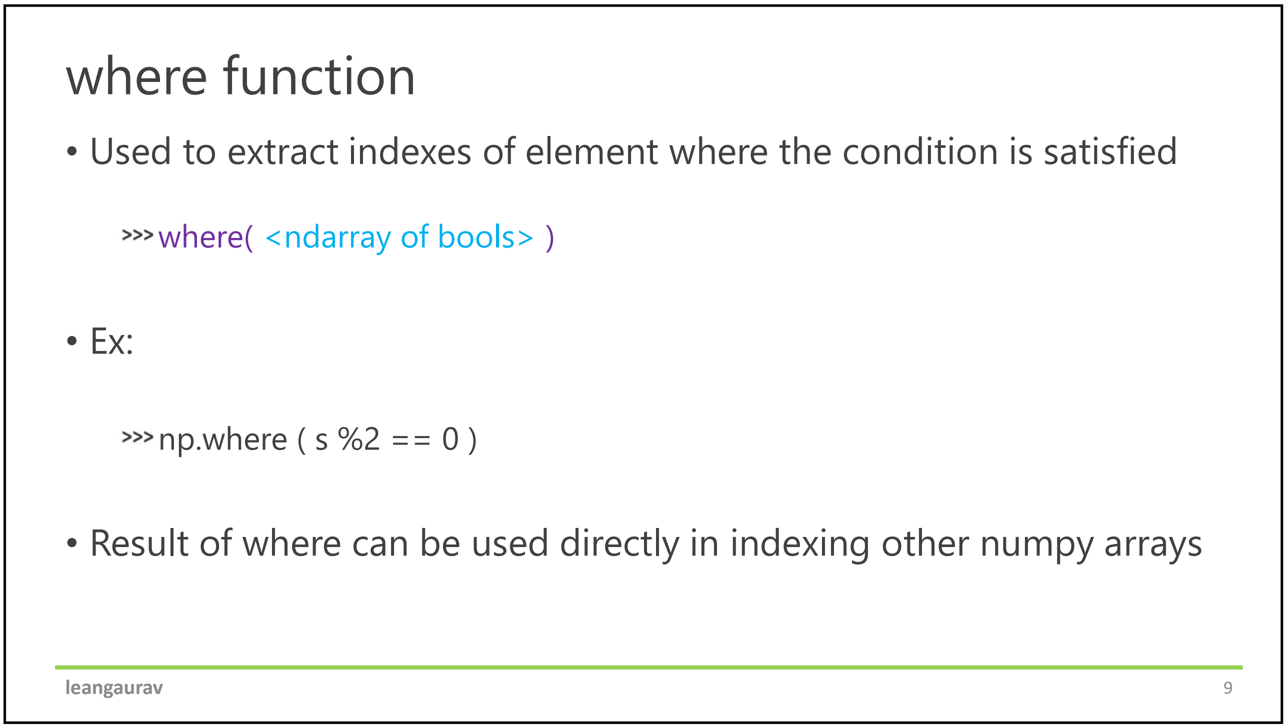


Any, all and NaNs

- any() checks if any True value is present, it returns True then
- all() returns True only when all elements are True
- isna() returns an ndarray of same size as input, putting True/False for each element
- Nan can’t be compared with other elements and each other hence all operations in numpy on NaNs return NaN

leangaurav 8

8



where function

- Used to extract indexes of element where the condition is satisfied

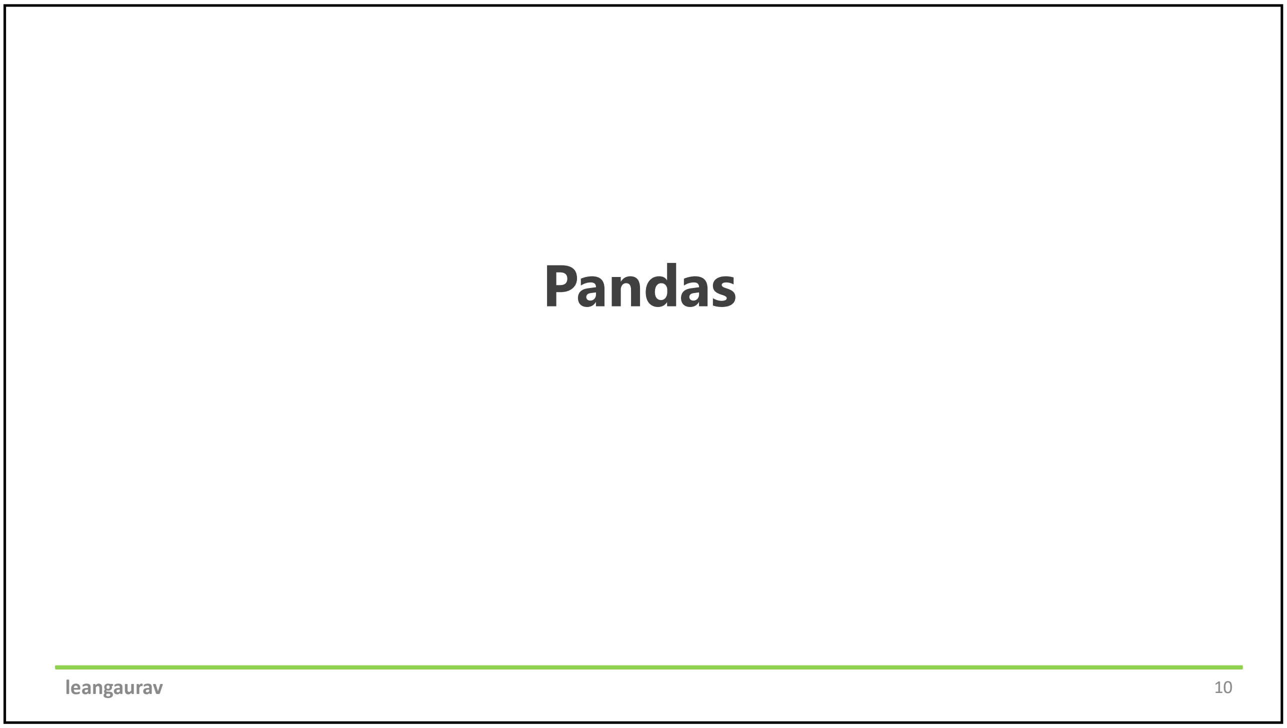
```
>>>where( <ndarray of bools> )
```

- Ex:

```
>>>np.where ( s %2 == 0 )
```

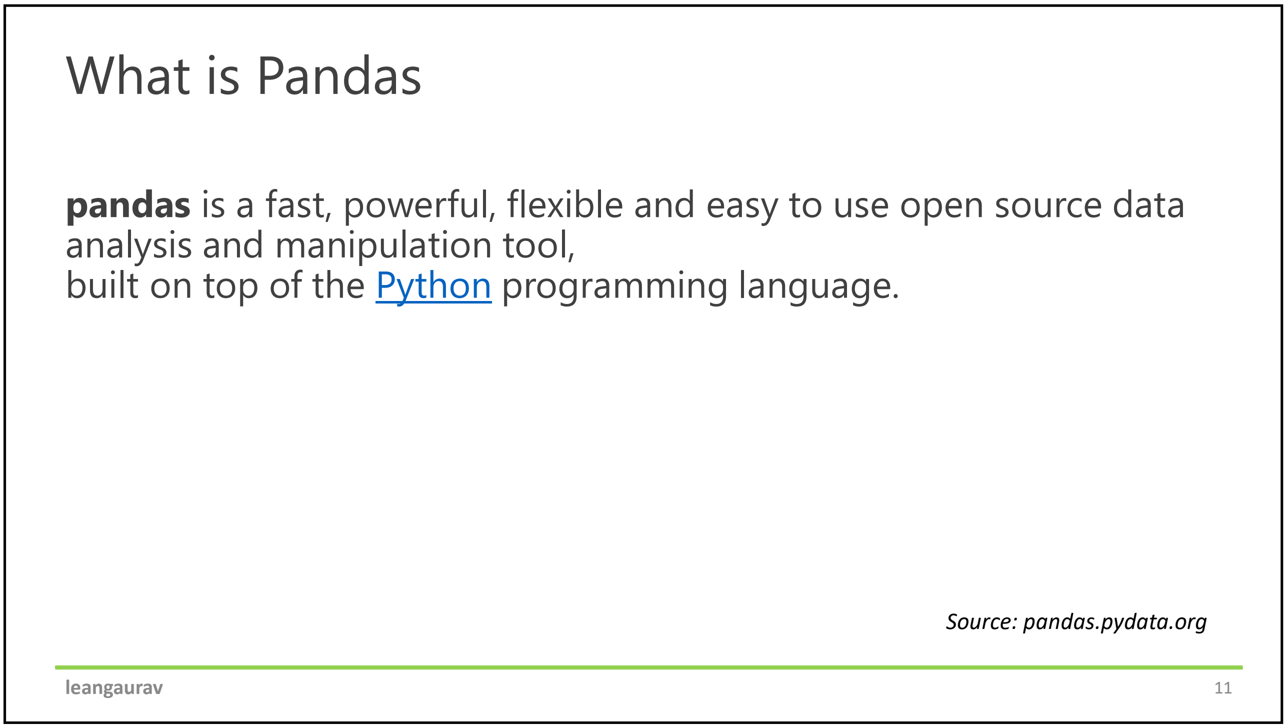
- Result of where can be used directly in indexing other numpy arrays

9



Pandas

10

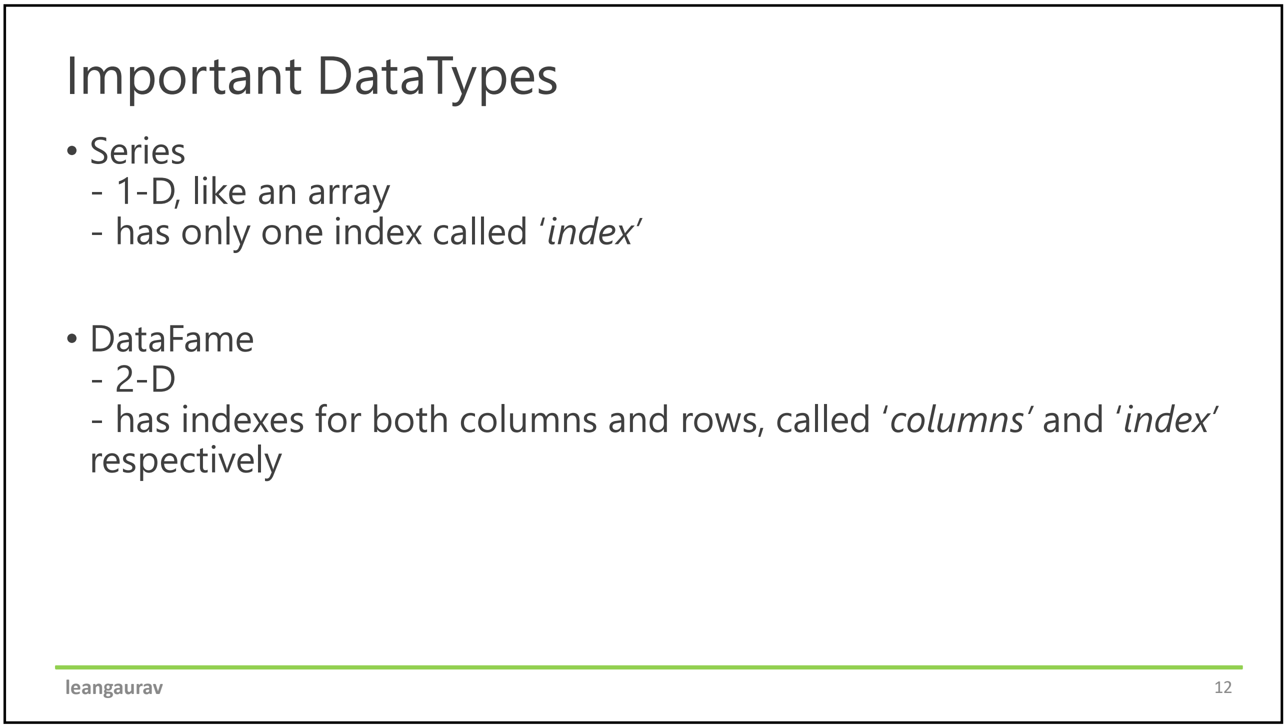


What is Pandas

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool,
built on top of the Python programming language.

Source: pandas.pydata.org

11

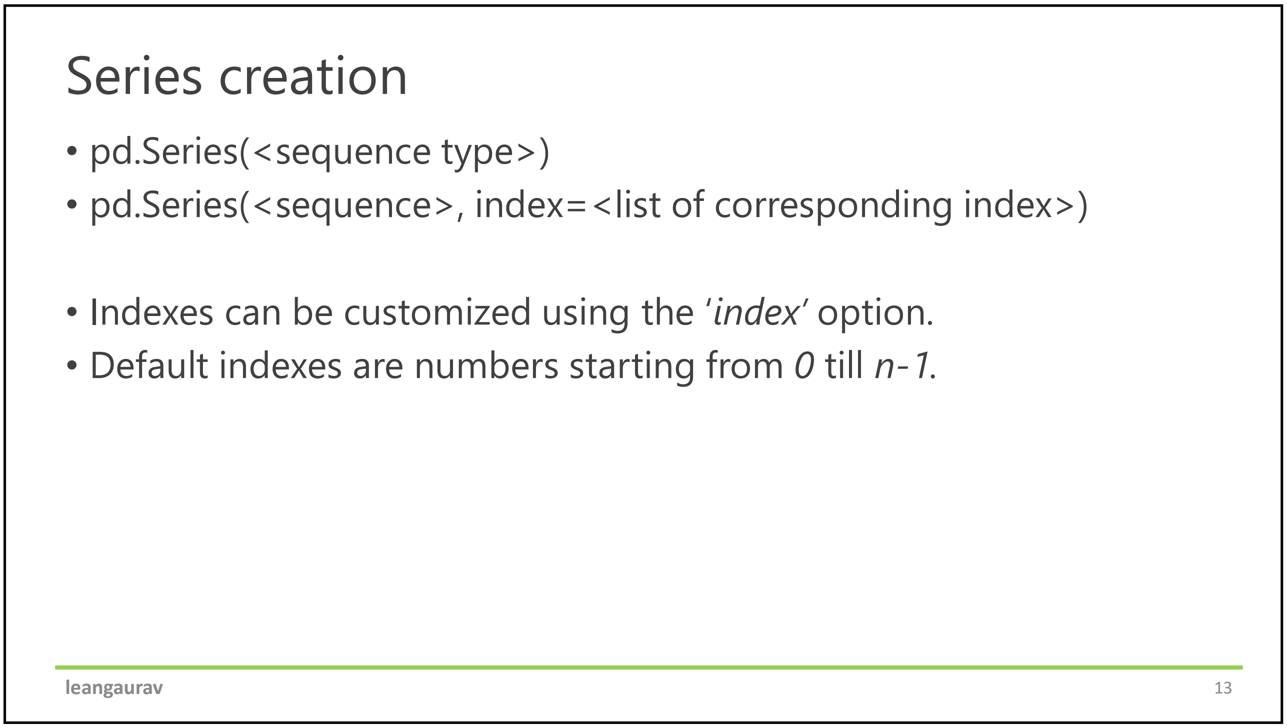


Important DataTypes

- Series
  - 1-D, like an array
  - has only one index called '*index*'

- DataFame
  - 2-D
  - has indexes for both columns and rows, called '*columns*' and '*index*' respectively

12



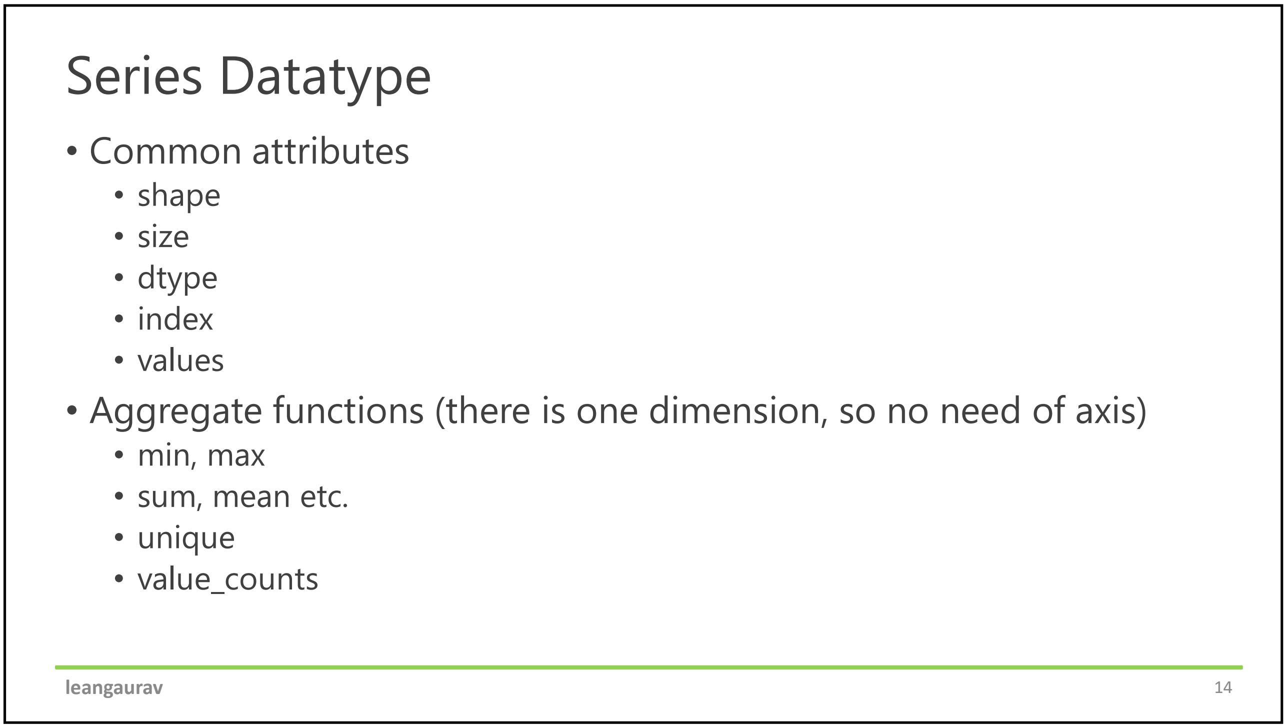
Series creation

- pd.Series(<sequence type>)
- pd.Series(<sequence>, index=<list of corresponding index>)

- Indexes can be customized using the '*index*' option.
- Default indexes are numbers starting from *0* till *n-1*.

leangaurav 13

13

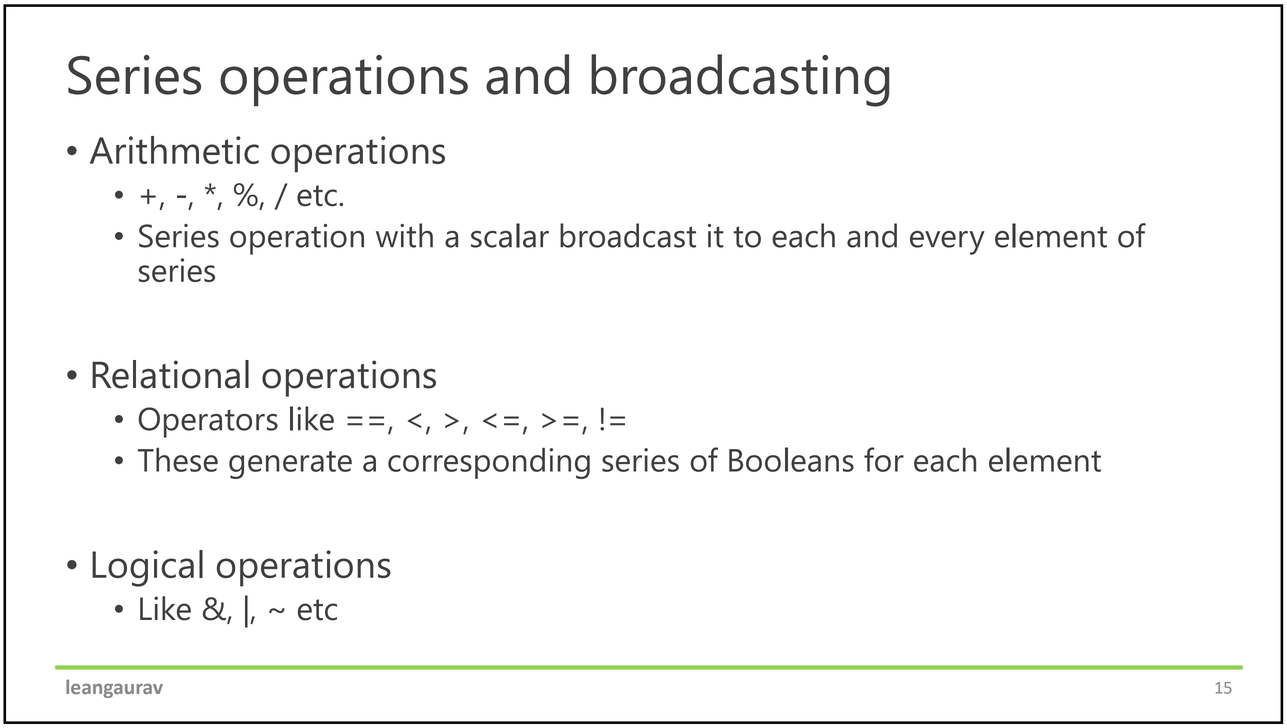


Series Datatype

- Common attributes
  - shape
  - size
  - dtype
  - index
  - values
- Aggregate functions (there is one dimension, so no need of axis)
  - min, max
  - sum, mean etc.
  - unique
  - value_counts

leangaurav 14

14

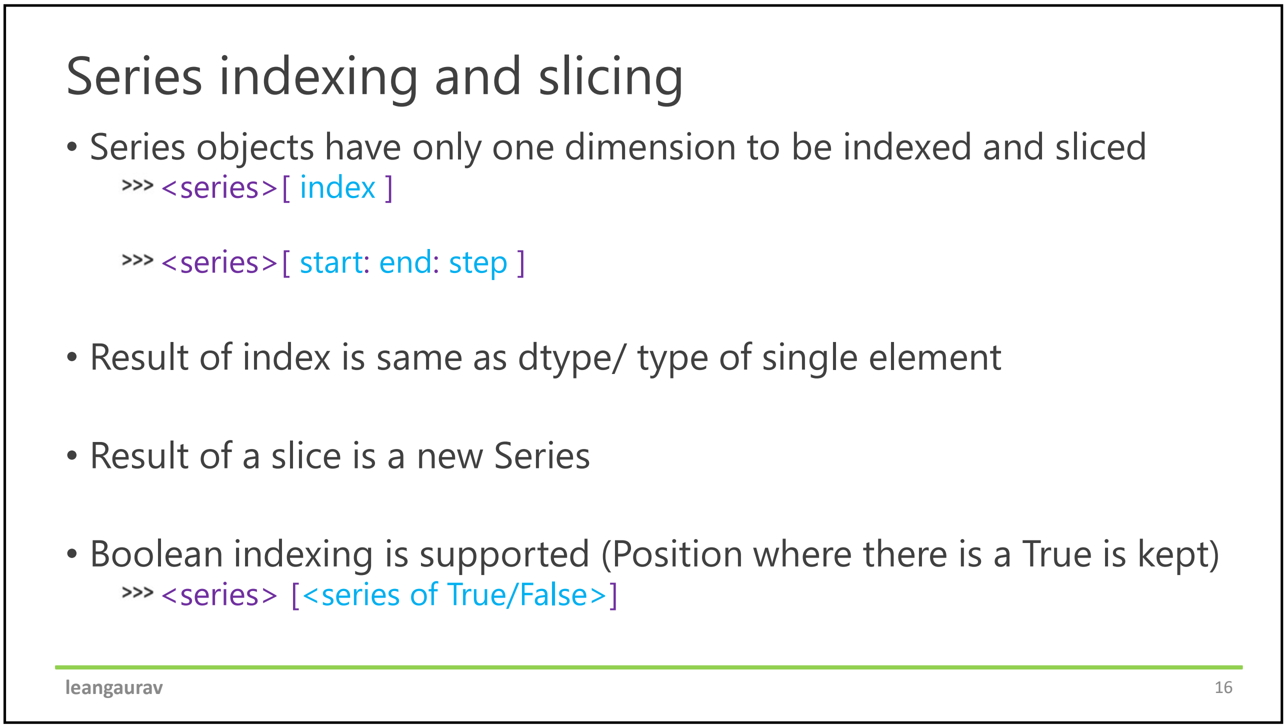


Series operations and broadcasting

- Arithmetic operations
  - +, -, *, %, / etc.
  - Series operation with a scalar broadcast it to each and every element of series
- Relational operations
  - Operators like ==, <, >, <=, >=, !=
  - These generate a corresponding series of Booleans for each element
- Logical operations
  - Like &, |, ~ etc

leangaurav 15

15



Series indexing and slicing

- Series objects have only one dimension to be indexed and sliced

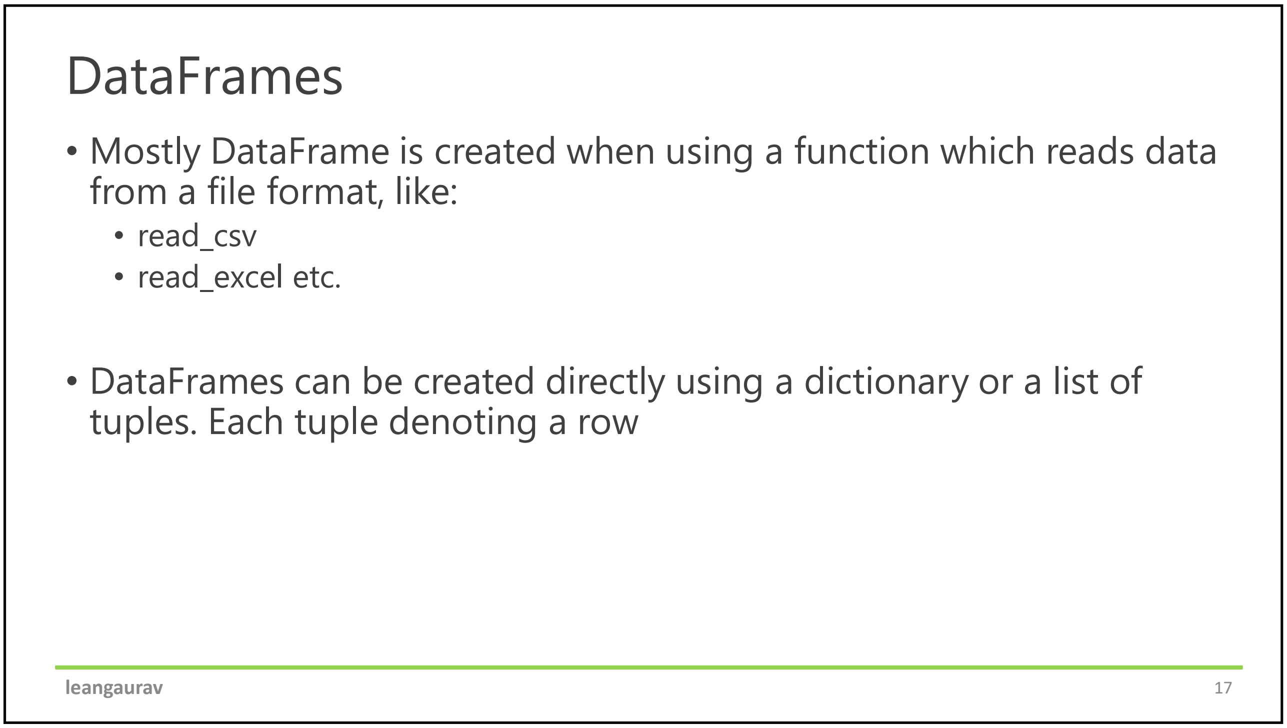
```
>>> <series>[ index ]

>>> <series>[ start: end: step ]
```

- Result of index is same as dtype/ type of single element
- Result of a slice is a new Series
- Boolean indexing is supported (Position where there is a True is kept)

```
>>> <series> [<series of True/False>]
```

leangaurav 16

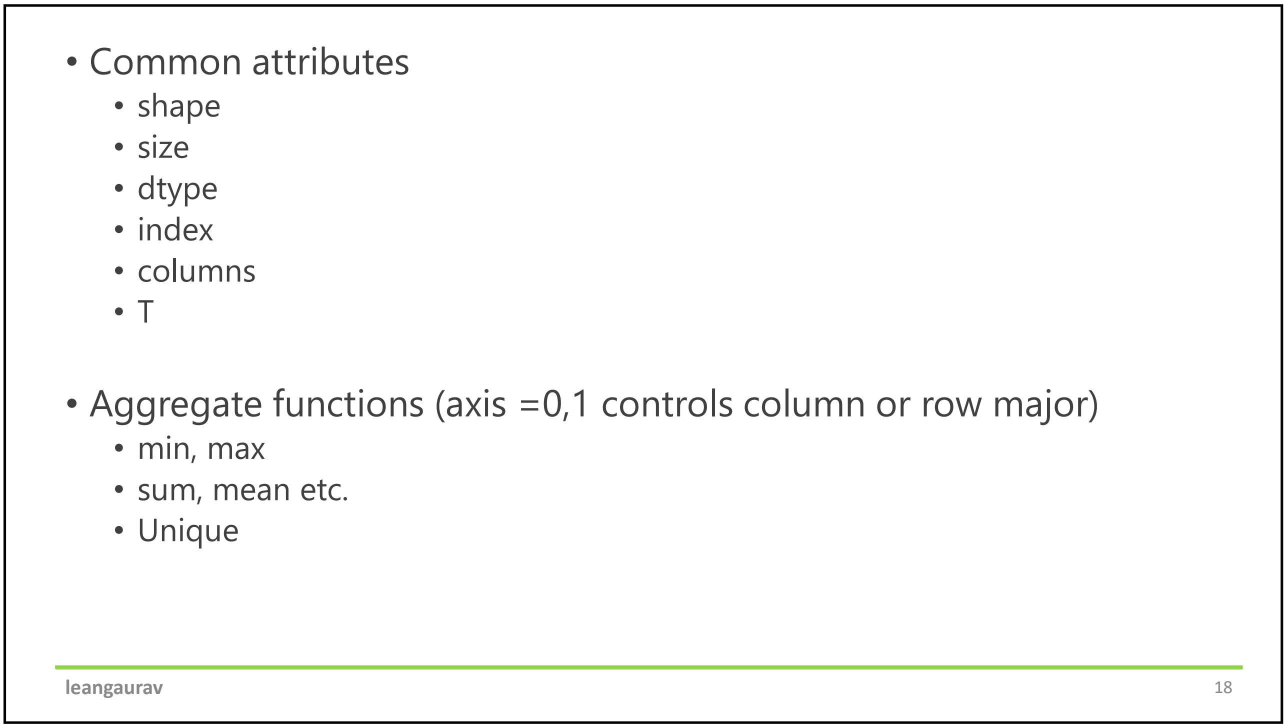
16



DataFrames

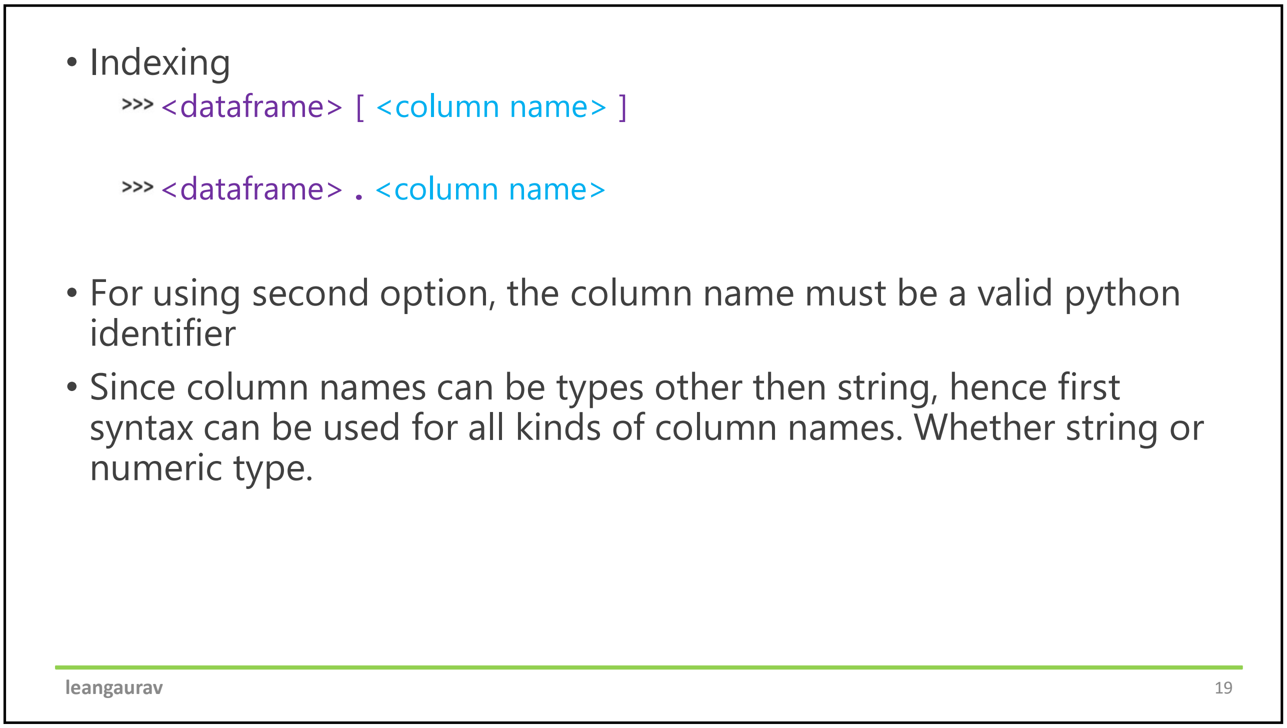
- Mostly DataFrame is created when using a function which reads data from a file format, like:
  - read_csv
  - read_excel etc.

- DataFrames can be created directly using a dictionary or a list of tuples. Each tuple denoting a row



- Common attributes
  - shape
  - size
  - dtype
  - index
  - columns
  - T

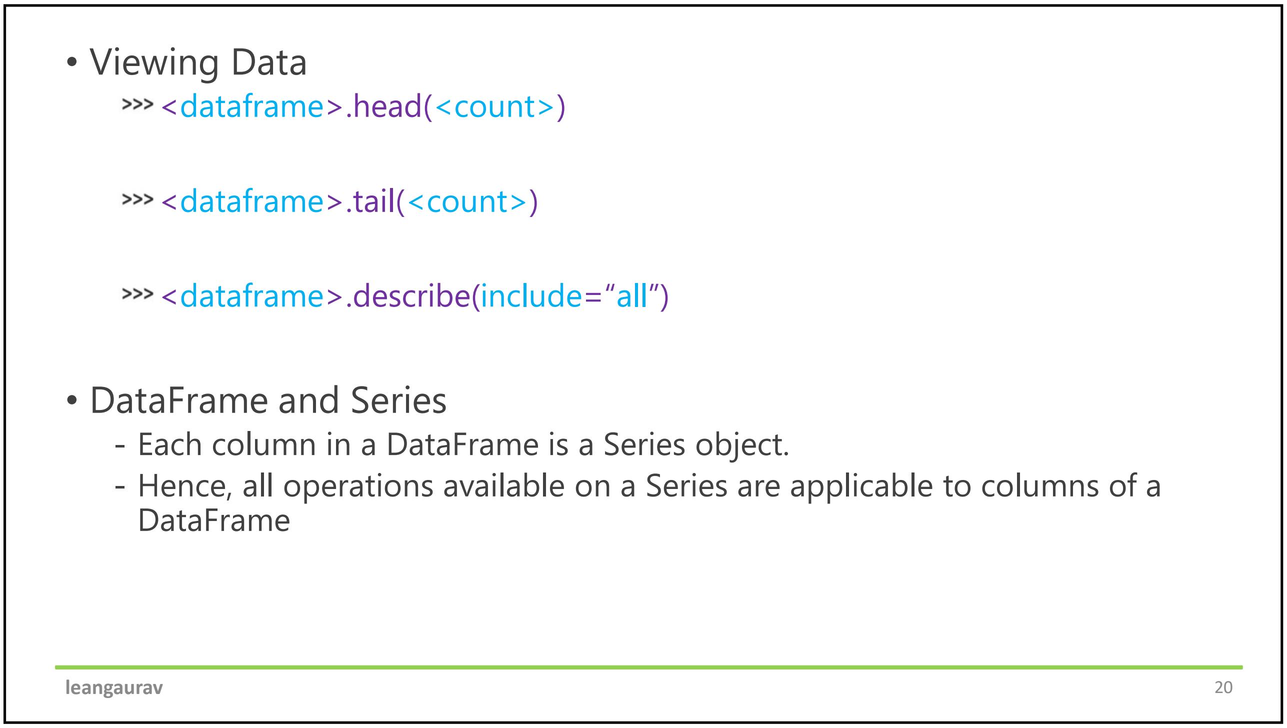
- Aggregate functions (axis =0,1 controls column or row major)
  - min, max
  - sum, mean etc.
  - Unique



- Indexing

```
>>> <dataframe> [ <column name> ]

>>> <dataframe> . <column name>
```

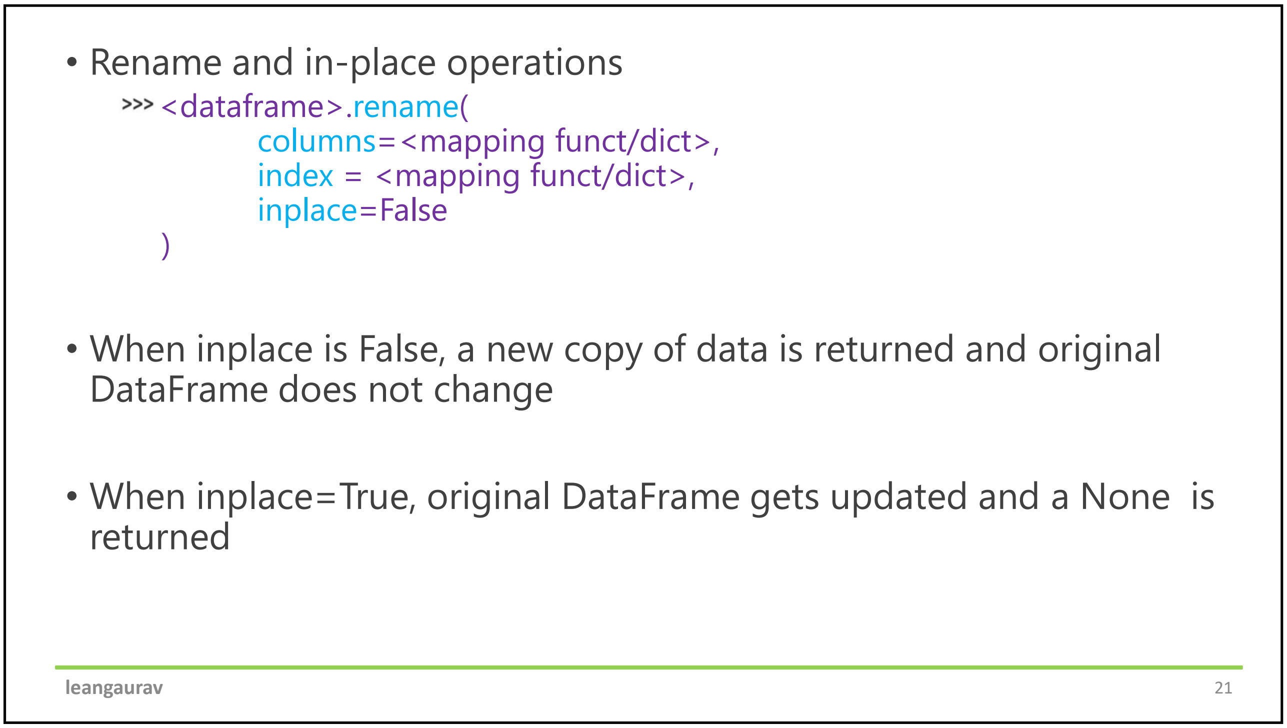
- For using second option, the column name must be a valid python identifier
- Since column names can be types other then string, hence first syntax can be used for all kinds of column names. Whether string or numeric type.



- Viewing Data

```
>>> <dataframe>.head(<count>)

>>> <dataframe>.tail(<count>)

>>> <dataframe>.describe(include="all")
```

- DataFrame and Series
  - Each column in a DataFrame is a Series object.
  - Hence, all operations available on a Series are applicable to columns of a DataFrame



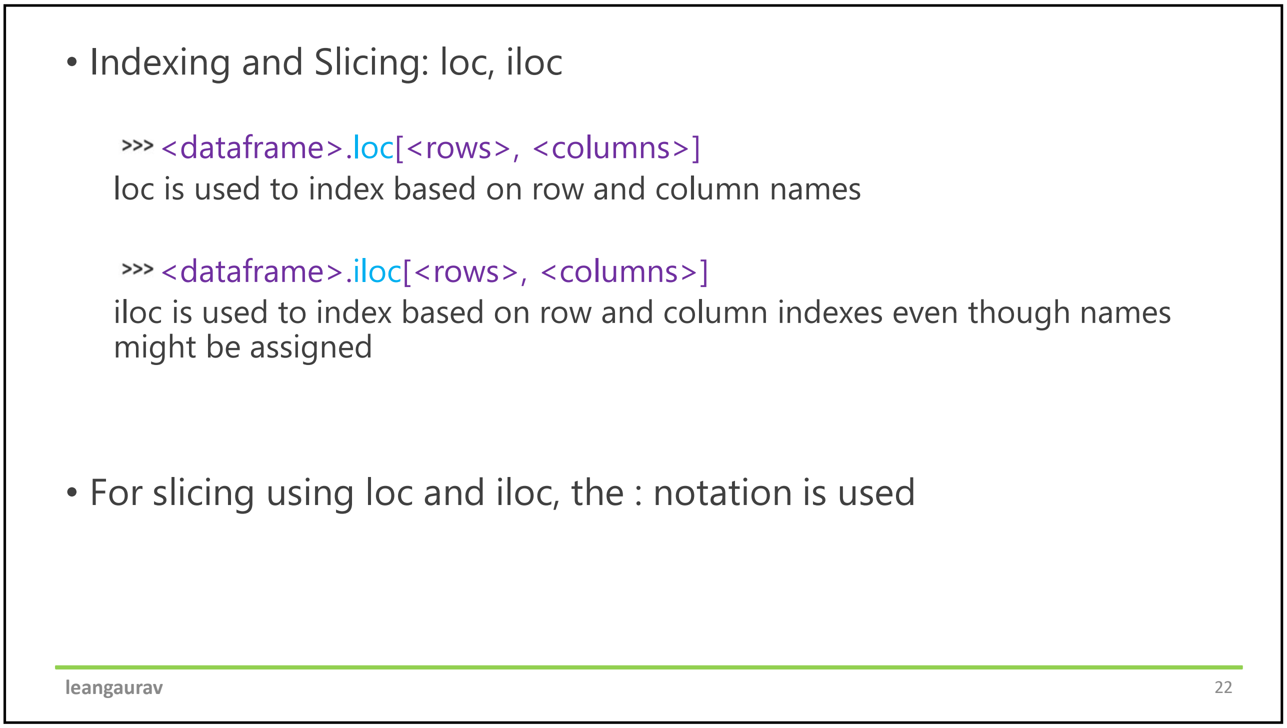
- Rename and in-place operations

```
>>> <dataframe>.rename(
              columns=<mapping funct/dict>,
              index = <mapping funct/dict>,
              inplace=False
    )
```

- When inplace is False, a new copy of data is returned and original DataFrame does not change

- When inplace=True, original DataFrame gets updated and a None is returned

21



- Indexing and Slicing: loc, iloc

```
>>> <dataframe>.loc[<rows>, <columns>]
```

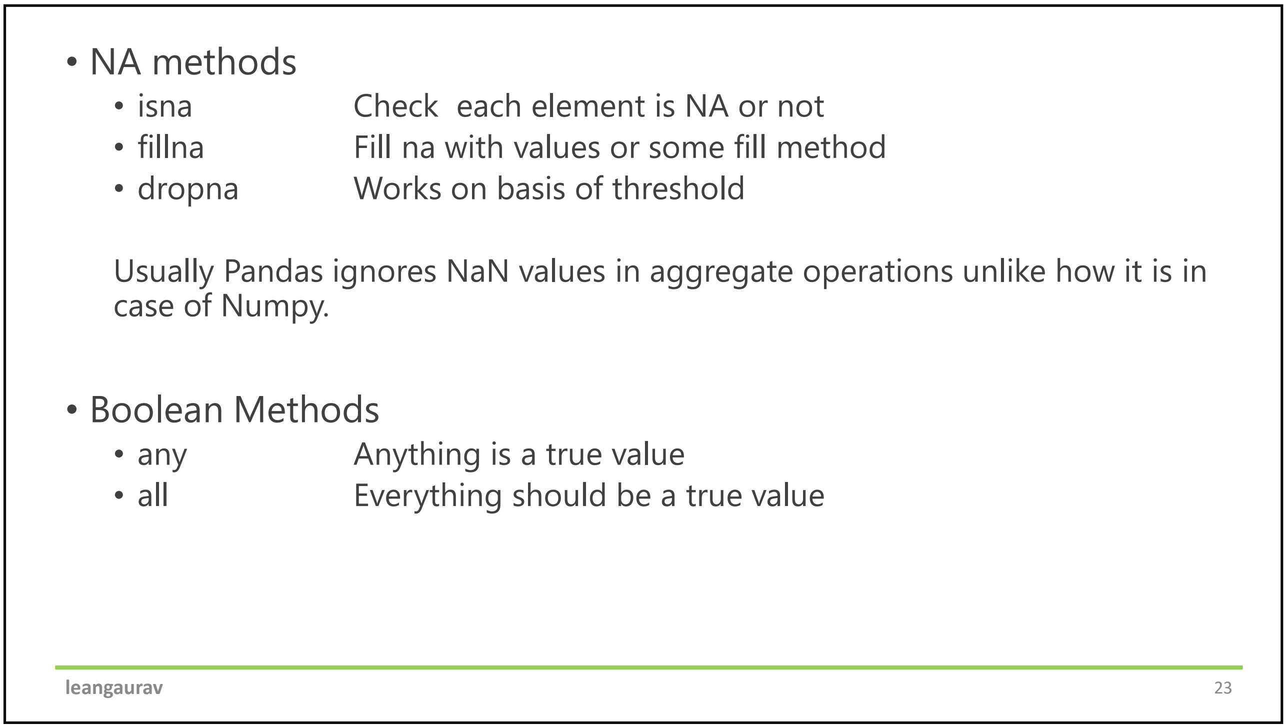
loc is used to index based on row and column names

```
>>> <dataframe>.iloc[<rows>, <columns>]
```

iloc is used to index based on row and column indexes even though names might be assigned

- For slicing using loc and iloc, the : notation is used

22



- NA methods

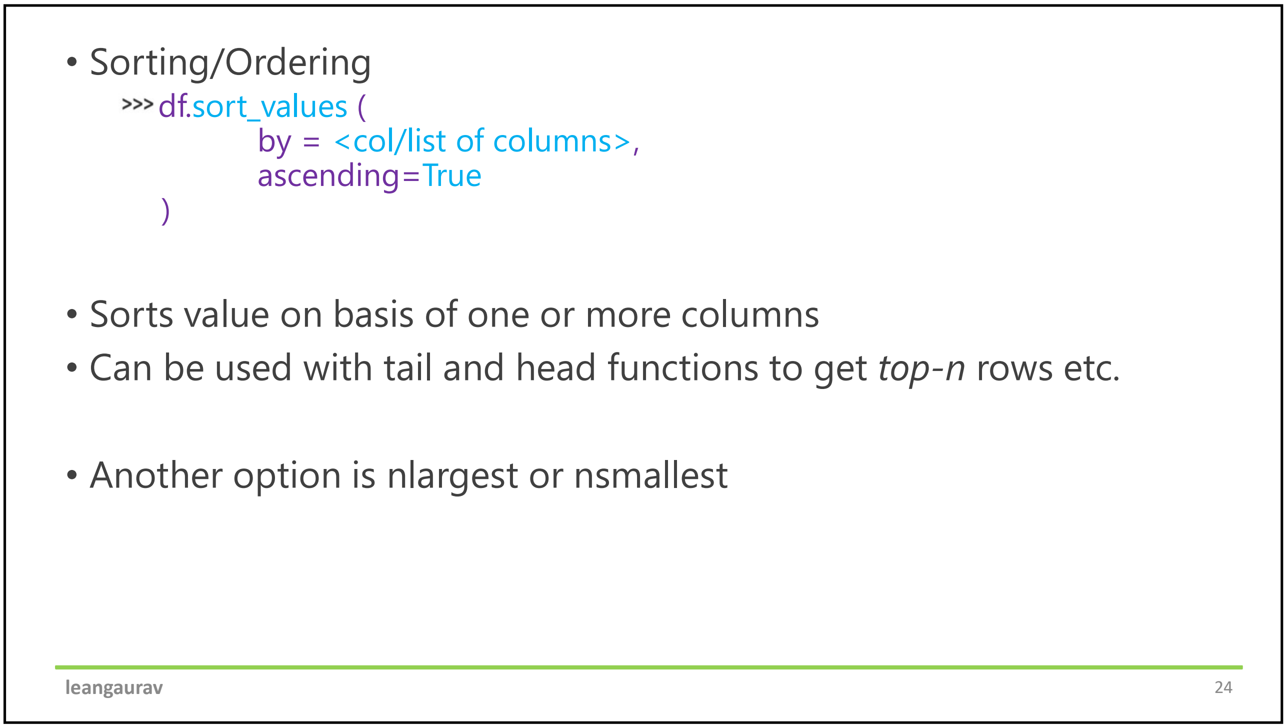
• isna	Check each element is NA or not
• fillna	Fill na with values or some fill method
• dropna	Works on basis of threshold

Usually Pandas ignores NaN values in aggregate operations unlike how it is in case of Numpy.

- Boolean Methods

• any	Anything is a true value
• all	Everything should be a true value

23



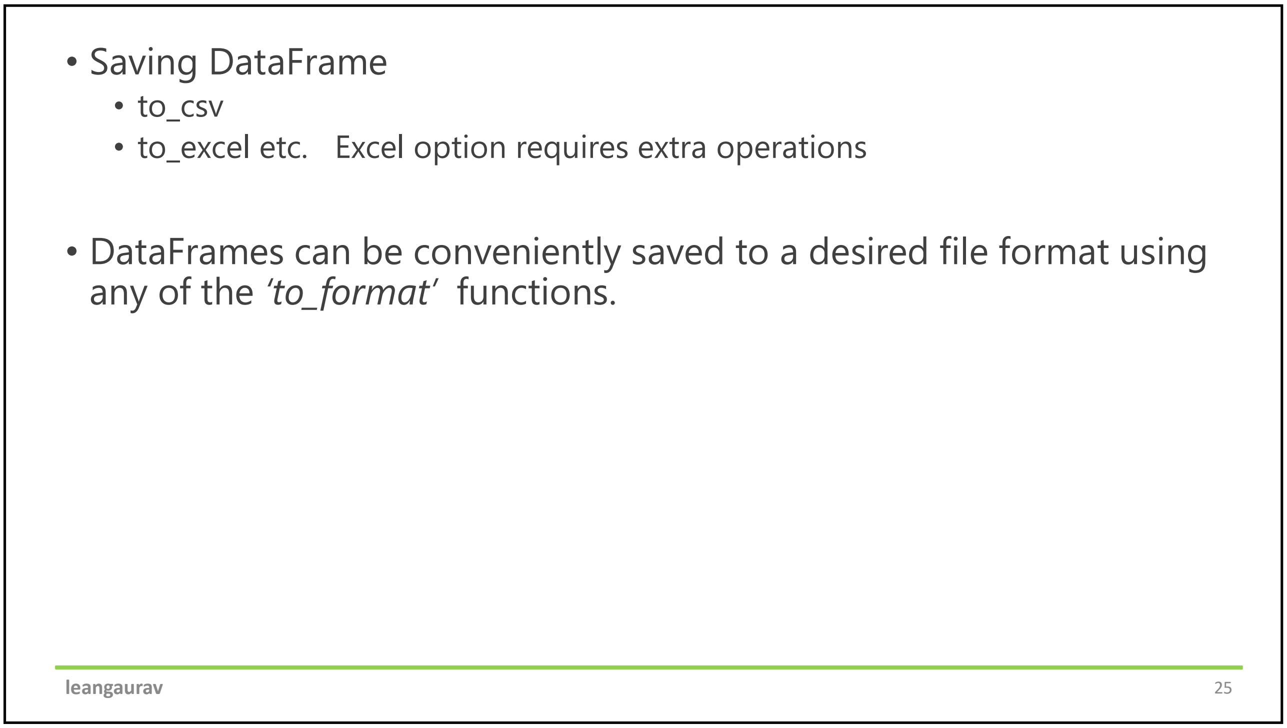
- Sorting/Ordering

```
>>> df.sort_values (
              by = <col/list of columns>,
              ascending=True
      )
```

- Sorts value on basis of one or more columns
- Can be used with tail and head functions to get *top-n* rows etc.

- Another option is nlargest or nsmallest

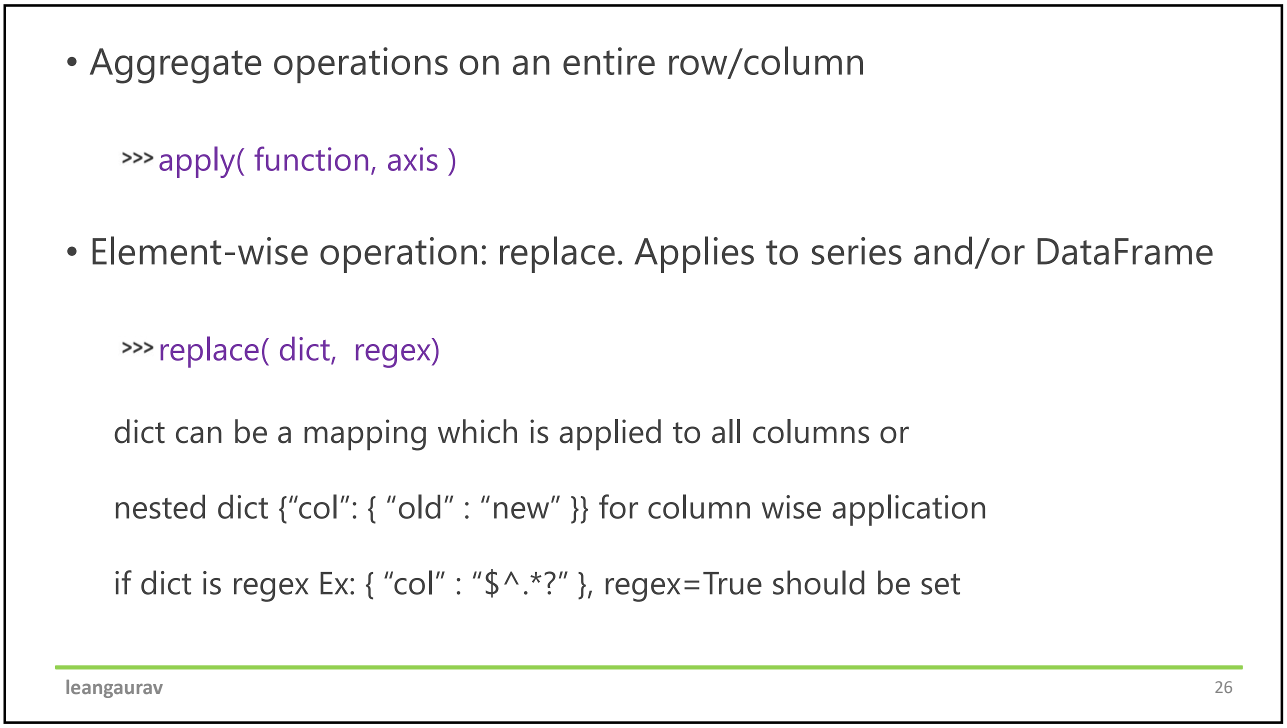
24



- Saving DataFrame
  - to_csv
  - to_excel etc.   Excel option requires extra operations

- DataFrames can be conveniently saved to a desired file format using any of the *'to_format'* functions.

25



- Aggregate operations on an entire row/column

```
>>>apply( function, axis )
```

- Element-wise operation: replace. Applies to series and/or DataFrame

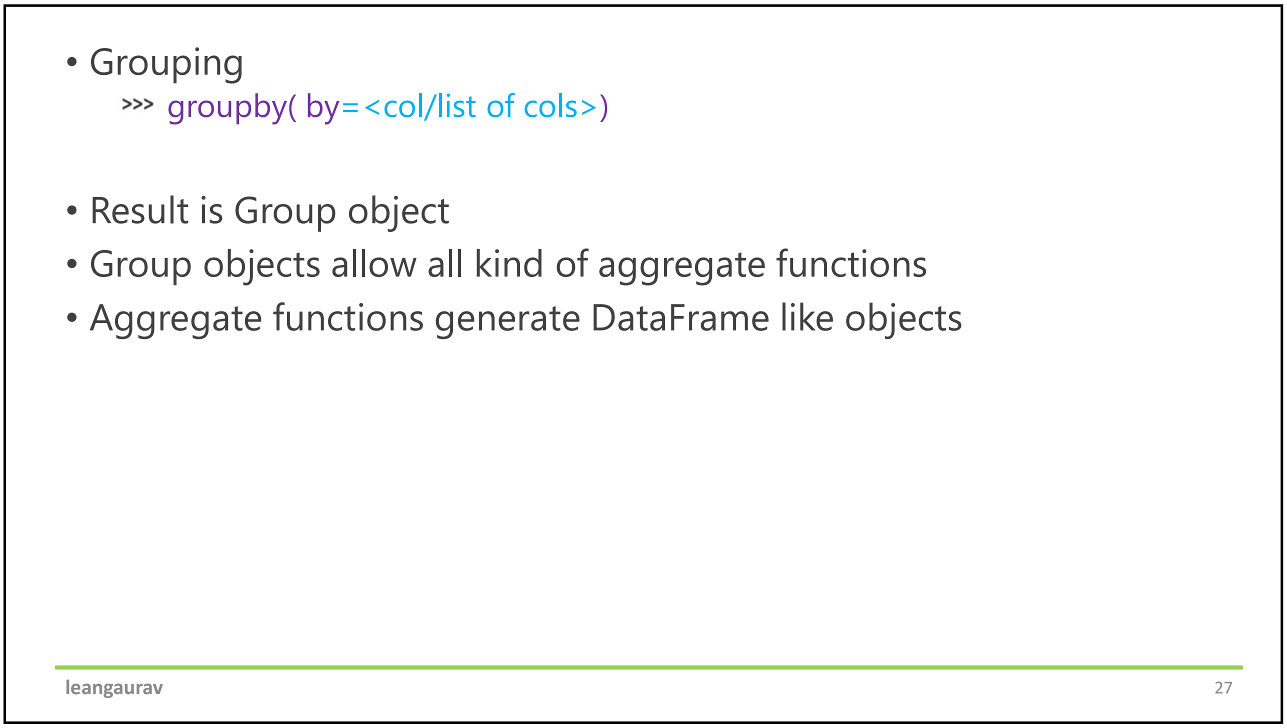
```
>>>replace( dict,  regex)
```

dict can be a mapping which is applied to all columns or

nested dict {“col”: { “old” : “new” }} for column wise application

if dict is regex Ex: { “col” : “$^.*?” }, regex=True should be set

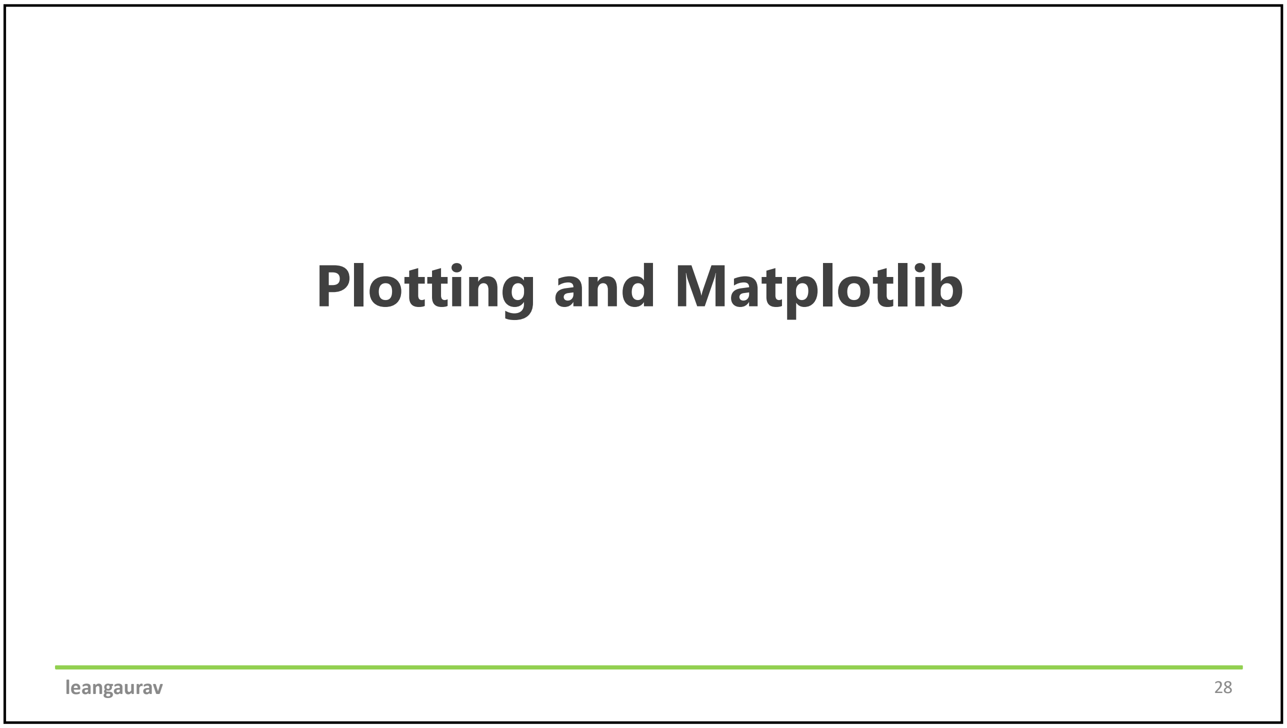
26



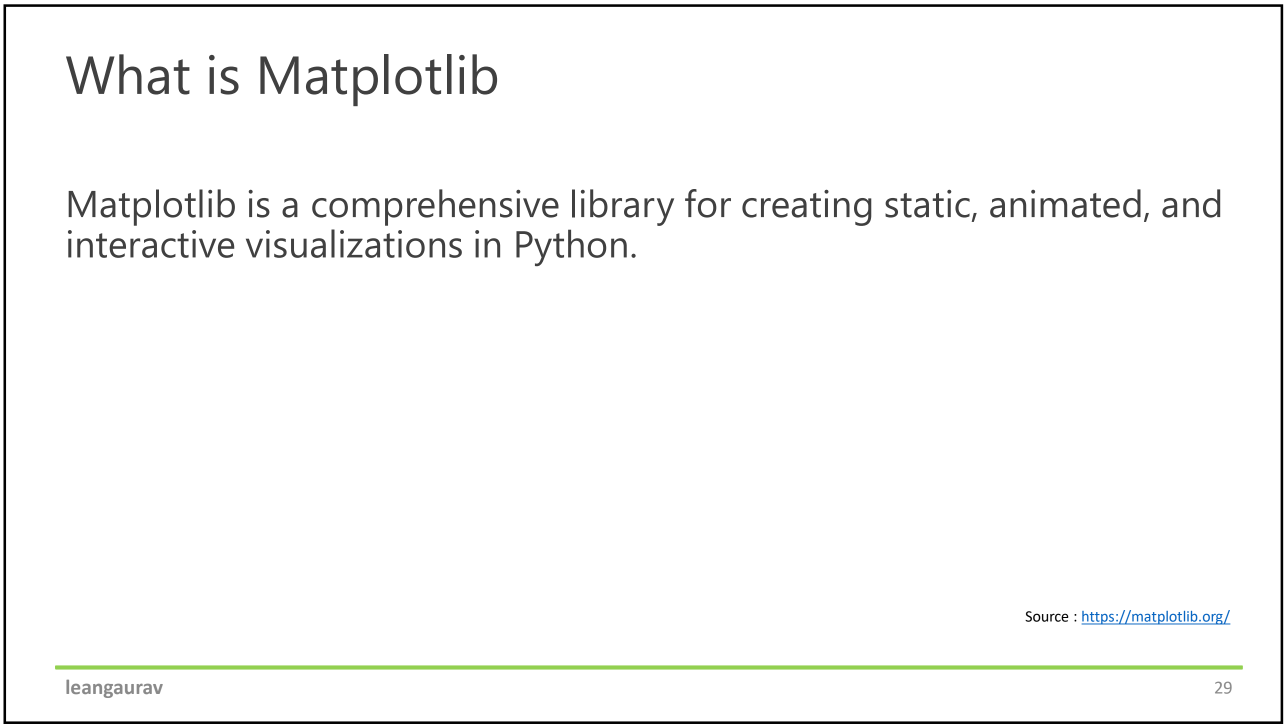
- Grouping

```
>>> groupby( by=<col/list of cols>)
```

- Result is Group object
- Group objects allow all kind of aggregate functions
- Aggregate functions generate DataFrame like objects



Plotting and Matplotlib

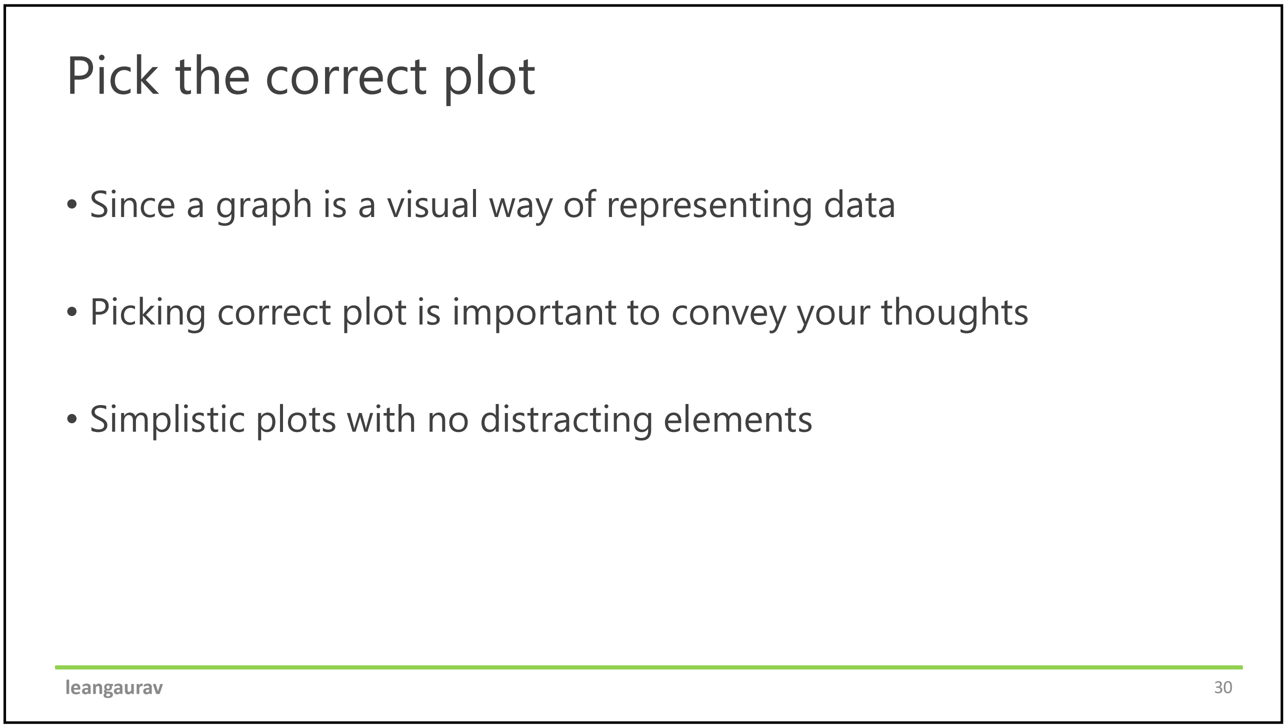


What is Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

Source : https://matplotlib.org/

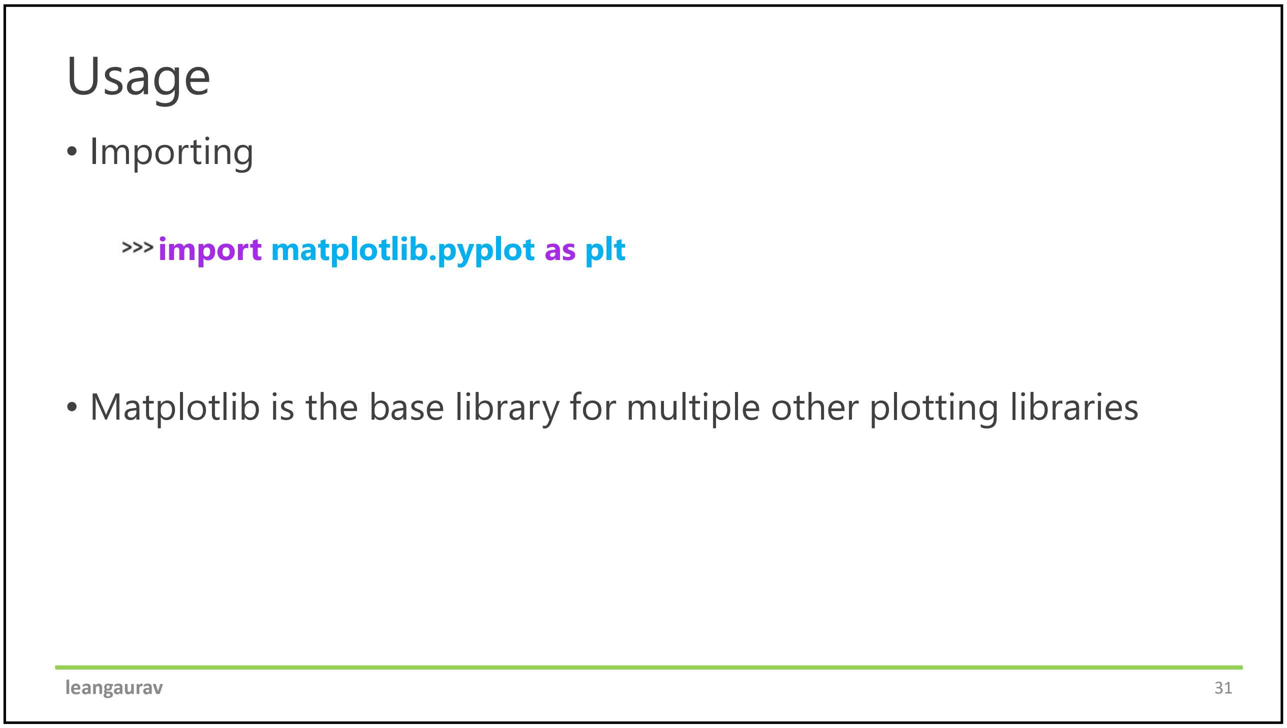
29



Pick the correct plot

- Since a graph is a visual way of representing data
- Picking correct plot is important to convey your thoughts
- Simplistic plots with no distracting elements

30



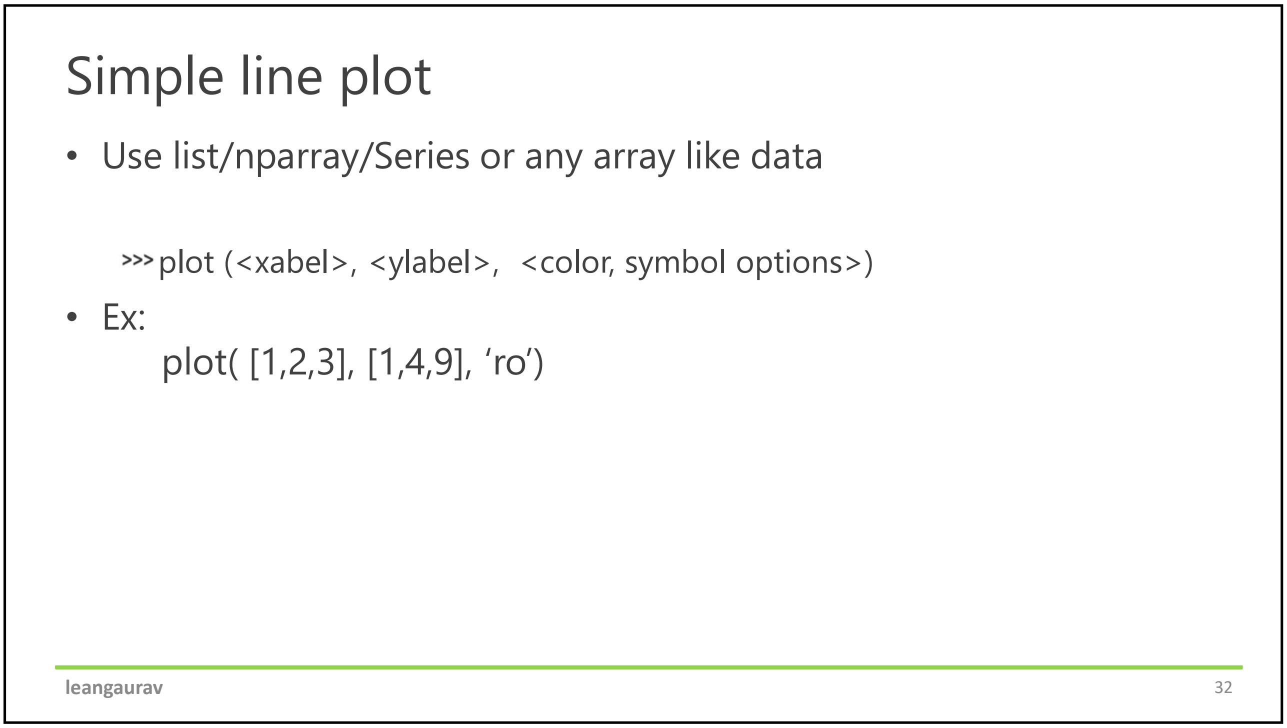
Usage

- Importing

```
>>>import matplotlib.pyplot as plt
```

- Matplotlib is the base library for multiple other plotting libraries

31



Simple line plot

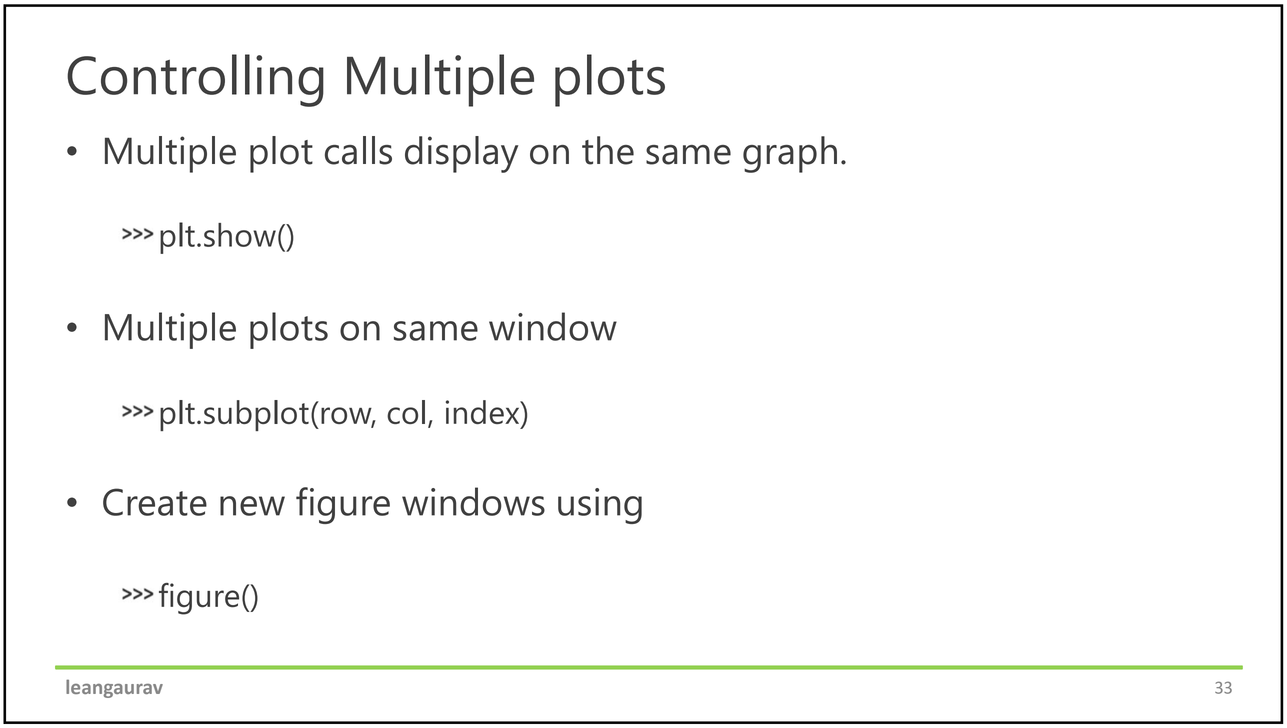
- Use list/nparray/Series or any array like data

```
>>>plot (<xabel>, <ylabel>,  <color, symbol options>)
```

- Ex:

  plot([1,2,3], [1,4,9], ‘ro’)

32



Controlling Multiple plots

- Multiple plot calls display on the same graph.

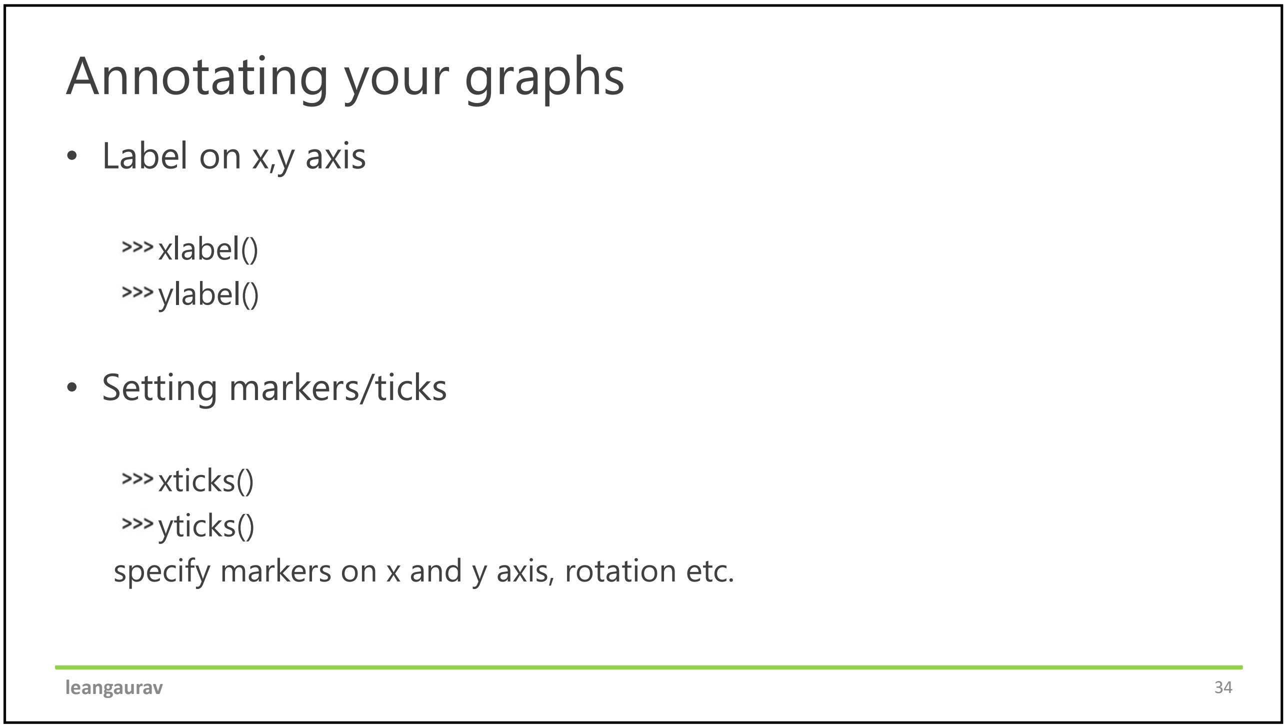
```
>>> plt.show()
```

- Multiple plots on same window

```
>>> plt.subplot(row, col, index)
```

- Create new figure windows using

```
>>> figure()
```



Annotating your graphs

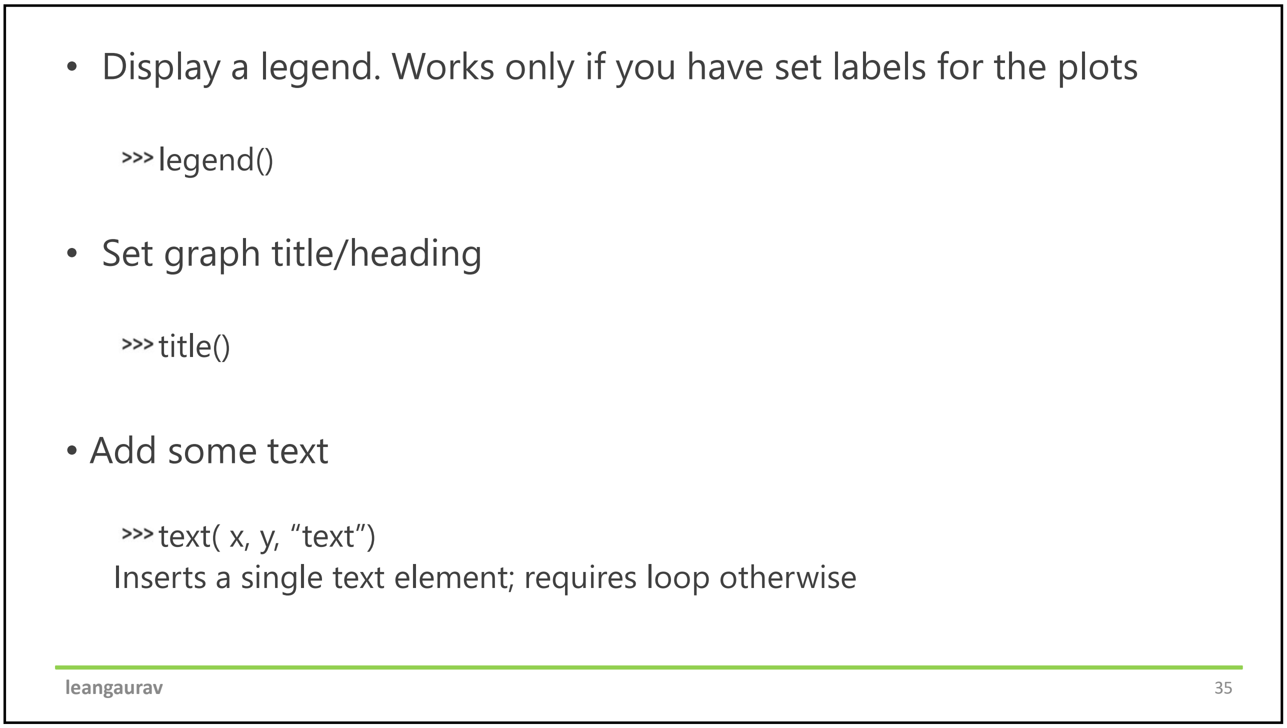
- Label on x,y axis

```
>>> xlabel()
>>> ylabel()
```

- Setting markers/ticks

```
>>> xticks()
>>> yticks()
```

specify markers on x and y axis, rotation etc.



- Display a legend. Works only if you have set labels for the plots

```
>>>legend()
```

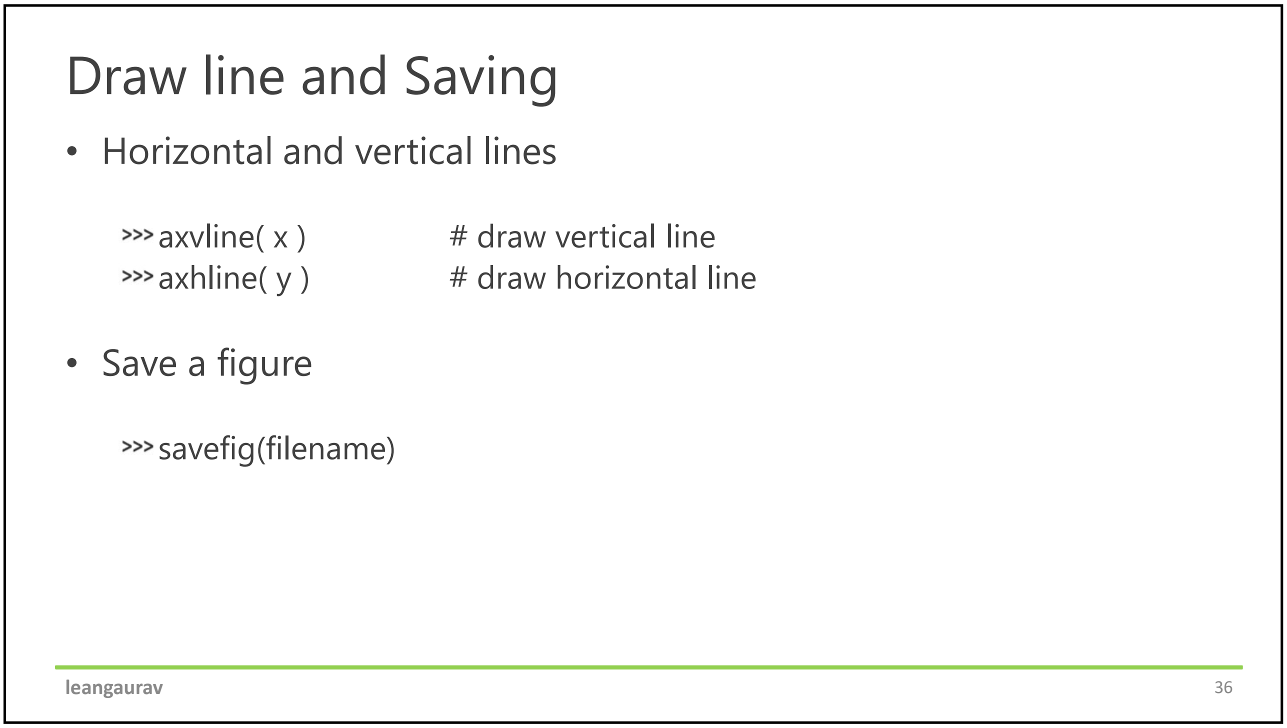
- Set graph title/heading

```
>>>title()
```

- Add some text

```
>>>text( x, y, “text”)
```

Inserts a single text element; requires loop otherwise



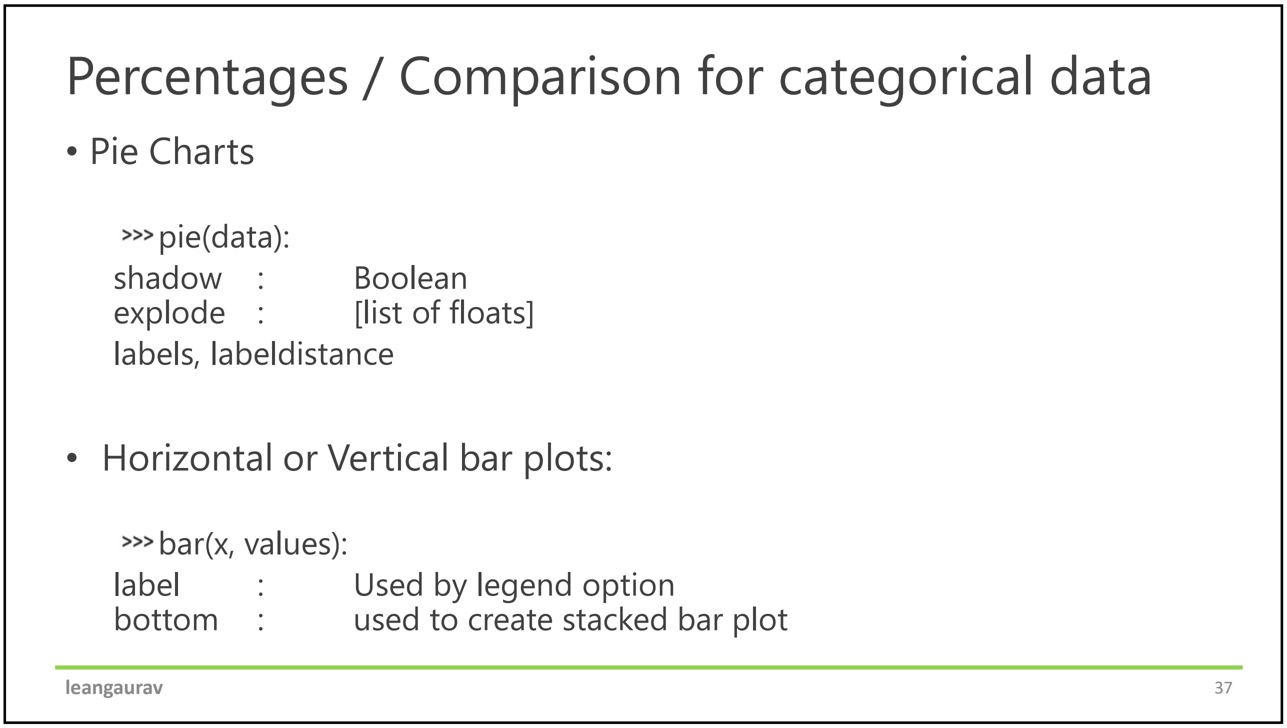
Draw line and Saving

- Horizontal and vertical lines

```
>>>axvline( x )          # draw vertical line
>>>axhline( y )          # draw horizontal line
```

- Save a figure

```
>>>savefig(filename)
```



Percentages / Comparison for categorical data

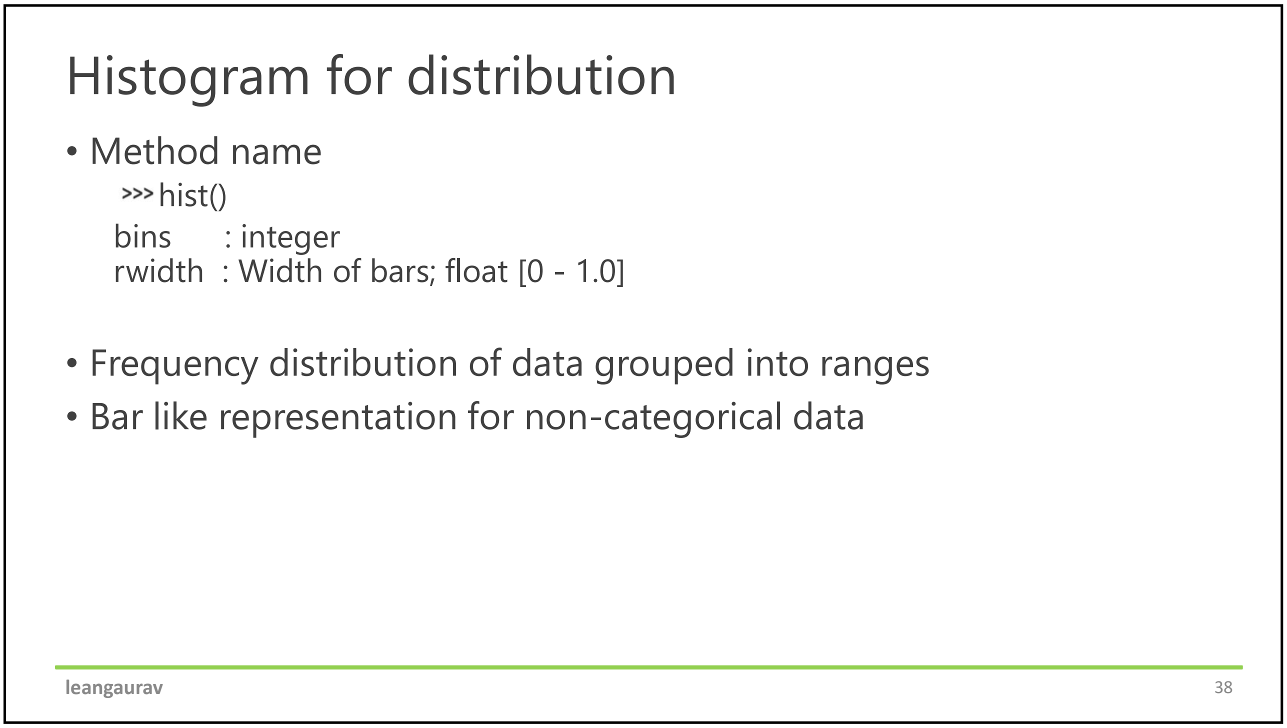
- Pie Charts

```
>>> pie(data):
shadow    :       Boolean
explode   :       [list of floats]
labels, labeldistance
```

- Horizontal or Vertical bar plots:

```
>>> bar(x, values):
label     :       Used by legend option
bottom    :       used to create stacked bar plot
```

37



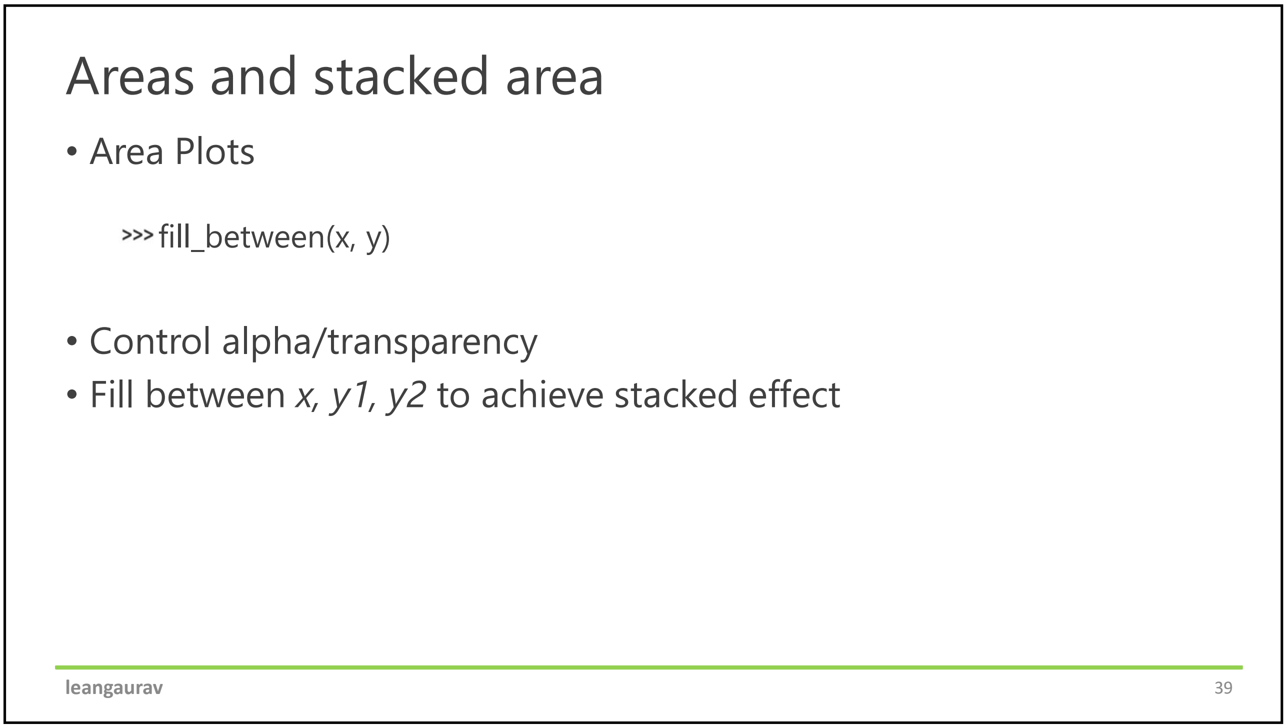
Histogram for distribution

- Method name

```
>>> hist()
bins    : integer
rwidth  : Width of bars; float [0 - 1.0]
```

- Frequency distribution of data grouped into ranges
- Bar like representation for non-categorical data

38

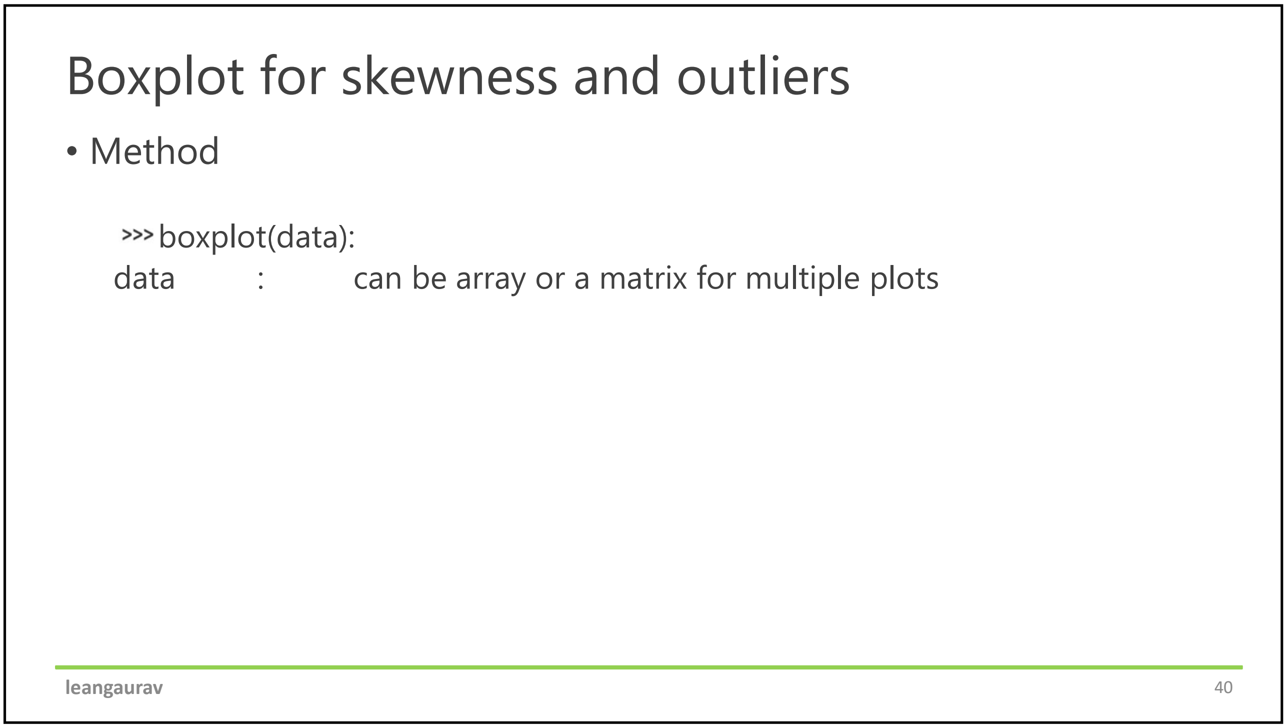


Areas and stacked area

- Area Plots

```
>>> fill_between(x, y)
```

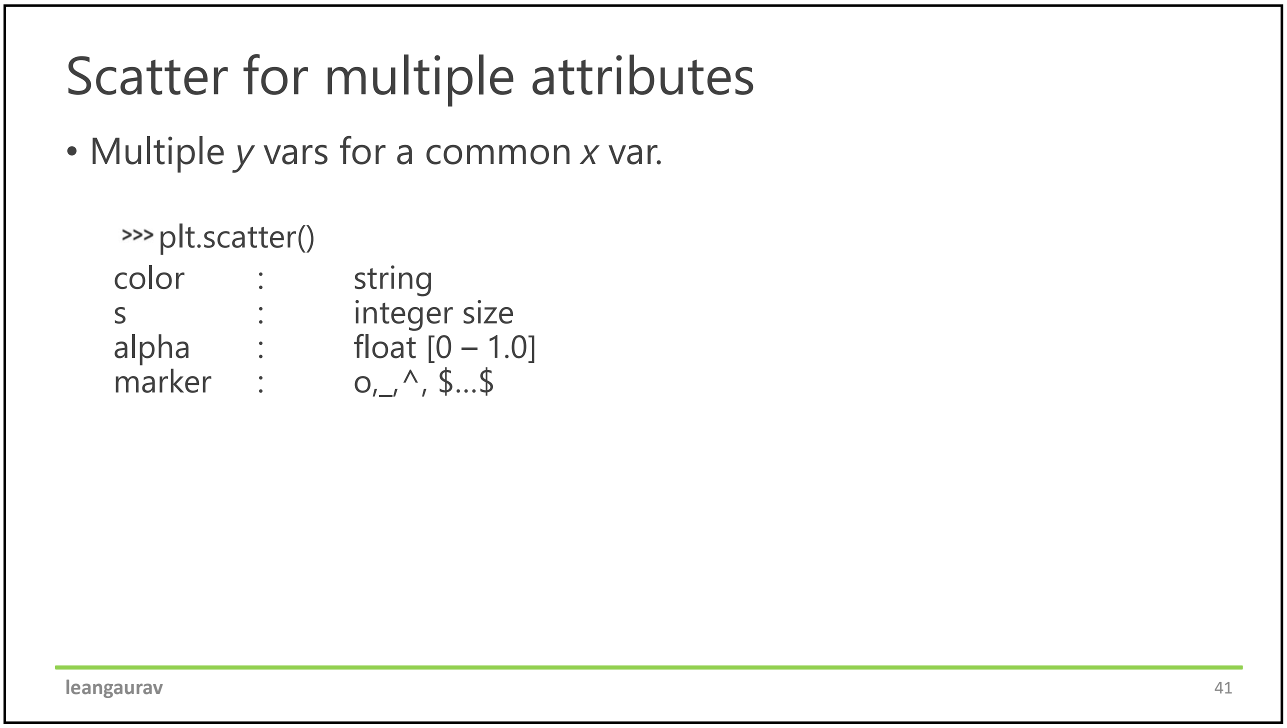
- Control alpha/transparency
- Fill between *x, y1, y2* to achieve stacked effect



Boxplot for skewness and outliers

- Method

```
>>> boxplot(data):
data        :        can be array or a matrix for multiple plots
```



Scatter for multiple attributes

- Multiple *y* vars for a common *x* var.

```
>>> plt.scatter()
```

color	:	string
s	:	integer size
alpha	:	float [0 – 1.0]
marker	:	o,_,^, $...$

leangaurav 41

41

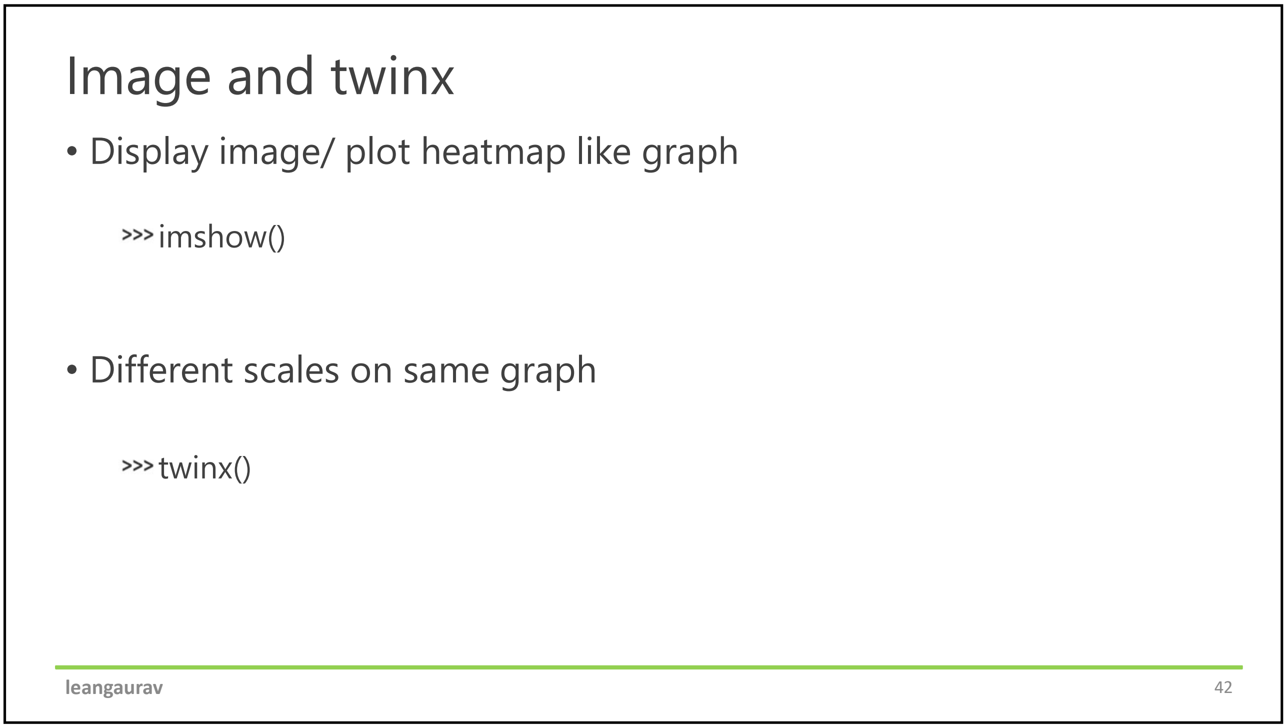


Image and twinx

- Display image/ plot heatmap like graph

```
>>> imshow()
```

- Different scales on same graph

```
>>> twinx()
```

leangaurav 42

42