Analytics using Python

Learning outcomes

1. You will learn Python , a useful language

2. Use programming for problem solving

Great Lakes Institute of Management

A guide to learn python for analytics

P. V. Subramanian

**A workbook on Analytics using Python**

**P. V. SUBRAMANIAN**

**Visiting faculty – Great Lakes Institute of Management**

*Copyright @ 2019 Great Lakes Institute of Management : All contents in this book, including (but not limited to) its contents are proprietary to Great Lakes Institute of Management, Chennai. Any effort at re-using, modifying and circulating both through Digital or Physical means, without the express written permission of the author, will amount to infringement of copyrights, and will be liable for suitable action.*

**Chapter 3. NumPy and Pandas**

**Table of contents**

[**1.** **NumPy arrays** 5](#_Toc8020918)

[**1.1** **NumPy arrays, array manipulation and random numbers** 5](#_Toc8020919)

[**1.2** **NumPy capabilities** 7](#_Toc8020920)

[**1.3** **NumPy efficiency** 7](#_Toc8020921)

[**1.4** **NumPy random numbers** 8](#_Toc8020922)

[**1.5** **How do you create ndarray?** 10](#_Toc8020923)

[**1.6** **Vectorization property of NumPy** 14](#_Toc8020925)

[**1.7** **Basic indexing and slicing** 15](#_Toc8020926)

[**1.8** **Boolean Indexing** 15](#_Toc8020927)

[1.8.1 Indexing using integer arrays 18](#_Toc8020928)

[1.8.2 Selecting a subset of rows 18](#_Toc8020929)

[1.8.3 reshape 19](#_Toc8020930)

[1.8.4 Transposing Arrays and Swapping Axes 20](#_Toc8020931)

[1.8.5 Mathematical functions 22](#_Toc8020932)

[**2.** Pandas 26](#_Toc8020933)

[**2.1** **Series** 26](#_Toc8020934)

[**2.2** **Data Frames** 32](#_Toc8020935)

[2.2.1 Features of a data frame 32](#_Toc8020936)

[2.2.2 History 33](#_Toc8020937)

[2.2.3 Which version of pandas you are using? 33](#_Toc8020938)

[2.2.4 How do you create a pandas data frame? 33](#_Toc8020939)

[2.2.5 basic funtionalities 36](#_Toc8020940)

[2.2.6 Add and remove columns from a pandas data frame 39](#_Toc8020941)

[2.2.7 Indexing with columns of a data frame 40](#_Toc8020942)

[2.2.8 Re-indexing 43](#_Toc8020943)

[2.2.9 rename 43](#_Toc8020944)

[2.2.10 row selection, addition and deletion 44](#_Toc8020945)

[2.2.11 descriptive statistics 47](#_Toc8020946)

[2.2.12 Missing values treatment in pandas 53](#_Toc8020947)

[3. Exercises 64](#_Toc8020948)

# **NumPy arrays**

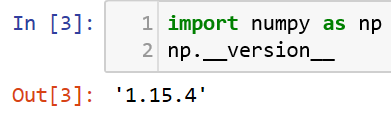
## **NumPy arrays, array manipulation and random numbers**

**NumPy**, short for Numerical Python is a cornerstone of numerical computing in Python. It pro- vides the data structures, algorithms, and library glue needed for most scientific applications involving numerical data in Python.

|  |  |
| --- | --- |
|  |  |
| *Ref: https://en.wikipedia.org/wiki/NumPy* | **Travis Oliphant**  *https://www.linkedin.com/in/teoliphant/* |

At the time of writing this chapter, I observed that the latest version of NumPy is 1.16.3 released in April 2019.

**How do you know the version of NumPy you are using? Note the use of double underscore(\_).**



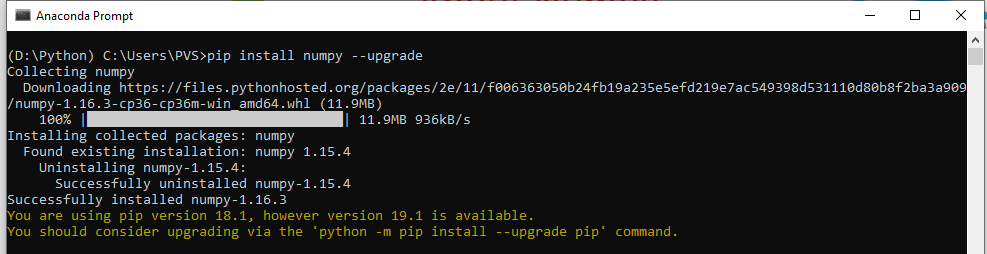
This is good since we use a version of NumPy which is not very old.

**How do you upgrade the version of NumPy when you find that you are using an old version?**

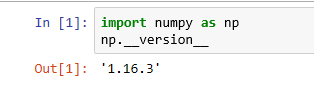
**At Anaconda prompt, you need to issue the command**

**pip install numpy –upgrade**

**Note: We have used hyphen(-) in the above command.**



Let us check the NumPy version now.



## **NumPy capabilities**

NumPy has the capabilities including the following:

1. A fast and efficient multi-dimensional array object, ndarray
2. Functions for performing element-wise computations with arrays or mathematical computations between arrays.
3. Tools for reading and writing array-based datasets to disk
4. Linear algebra operations, random number generation

**NumPy is a large topic. Let us focus on the following:**

1. Fast vectorized array operations for data munging and cleaning, sub-setting and filtering, transformation, and other computations
2. Common array algorithms like sorting, unique and set operations
3. Efficient descriptive statistics and aggregating / summarizing data
4. Expressing conditional logic as array operations instead of iterative statements( loops) with

if ….:

…

elif ….:

…

else:

….

branches.

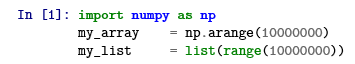
1. Group-wise data manipulation (aggregation, transformation, function application).

## **NumPy efficiency**

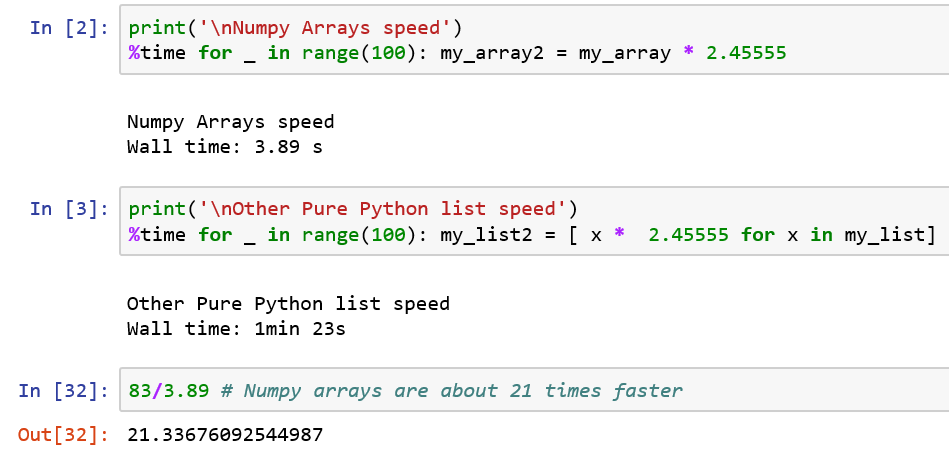
**NumPy is designed for efficiency on large arrays of data.**

**How do you check if numpy arrays are faster than other pure python counterparts?**

Consider a NumPy array of ten million integers and the equivalent Python list.



Now, let us multiply each sequence by 2.45555 ad repeat this hundred times in a loop. Using %time measure the Wall times.

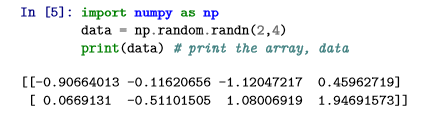


We observe that calculation using numpy arrays are around 21 times faster than performing calculations using pure python list methods.

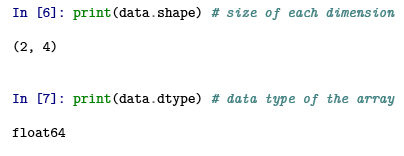
## **NumPy random numbers**

numpy.random.randn return a sample or samples from standard normal distribution with mean 1 and standard deviation 0.

Let us generate a small 2 by 4 array of random data.



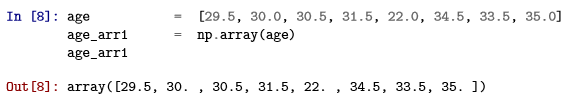
* An ndarray is a generic multidimensional container for homogeneous data (all data elements are of the same type).
* Every array has \* a shape, a tuple indicating the size of each dimension. \* a dtype, an object indicating the data type of the array
* The data type or dtype is a special object containing information the ndarray needs to interpret a chunk of memory as a particular type of data.



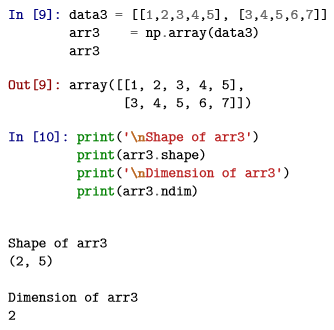
Generally, array, ndarray, numpy array all refer to the same ndarray object.

## **How do you create ndarray?**

Function, array creates a new NumPy array containing the “passed” data (sequence like object, including other arrays).

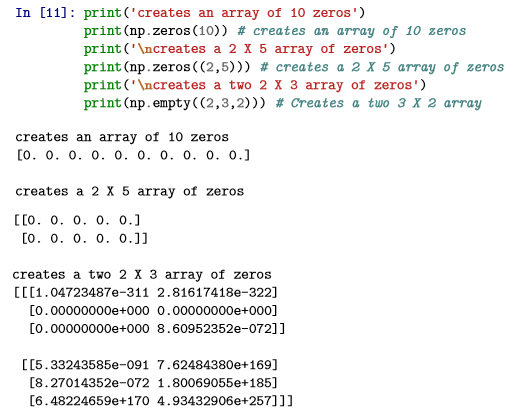


Nested sequences, like a list of equal-length lists will be converted into a multi-dimensional array:



There are several functions for creating new arrays such as zeros, ones to hold o’s and 1’s respectively with a given length of shape.

You can create an empty array without initializing its values to any value.



The function, arange is an array-valued version of the built-in Python range function. np.arange(10)

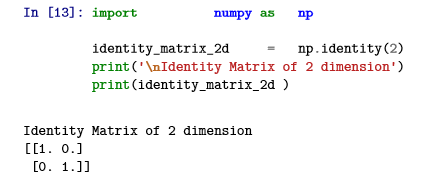
In [12]: np.arange(10)

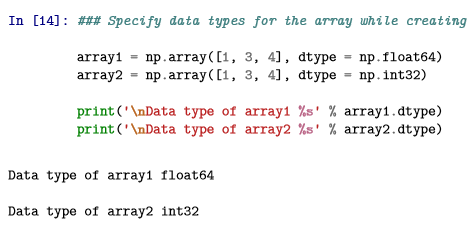
Out[12]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

### Array creation functions

|  |  |  |
| --- | --- | --- |
| **#** | **Function** | **Description** |
| 1 | array | Convert input data containing any one of list, tuple, array, or other sequence type to an ndarray either by explicitly specifying a dtype or inferring a dtype; by default copies the input data |
| 2 | asarray | Convert input to ndarray, but do not copy if if the input is already an ndarray |
| 3 | arange | Same as built-in range, but returns an ndarray instead |
| 4 | ones, ones\_like | Produce an array of all 1s with the given shape and dtype; ones\_like takes another array and produces a ones array of the same shape and dtype |
| 5 | zeros, zeros\_like | Produce an array of all 0s with the given shape and dtype; zeros\_like takes another array and produces a zeros array of the same shape and dtype |
| 6 | empty, empty\_like | Create new arrays by allocating new memory, but do not populate with any values like ones or zeros |
| 7 | full, full\_like | Produce an array of given shape and dtype; with all values set to the indicated ’fill value’ full\_like takes another array and produces a filled array of the same shape and dtype |
| 8 | eye, identity | Produce an array of given shape and dtype; with all values set to the indicated ’fill value’ full\_like takes another array and produces a filled array of the same shape and dtype Create a square n X n Identity matrix, 1s on diagonal and 0 elsewhere |
| 9 | matrix | This returns a matrix from an array-like object, or from a string of data. |

**How to create a 2d Identity matrix with 1s on the diagonal and 0s elsewhere?**



****

**You can explicitly convert an array from one data type to another using “astype” method.**

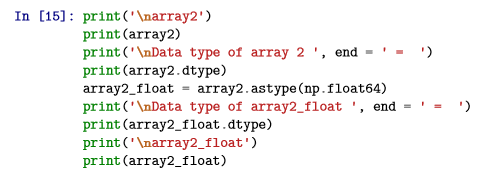
In [15]: print('**\n**array2')

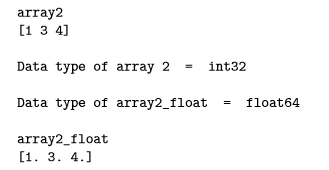
print(array2)

print('**\n**Data type of array 2 ', end = ' = ') print(array2.dtype)

array2\_float = array2.astype(np.float64) print('**\n**Data type of array2\_float ', end = ' = ') print(array2\_float.dtype)

print('**\n**array2\_float') print(array2\_float)

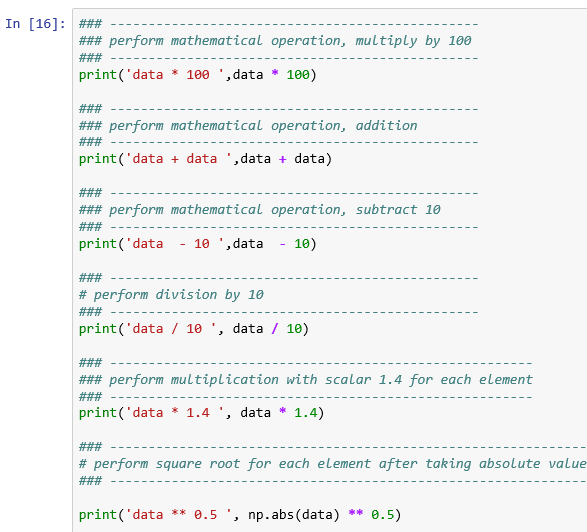


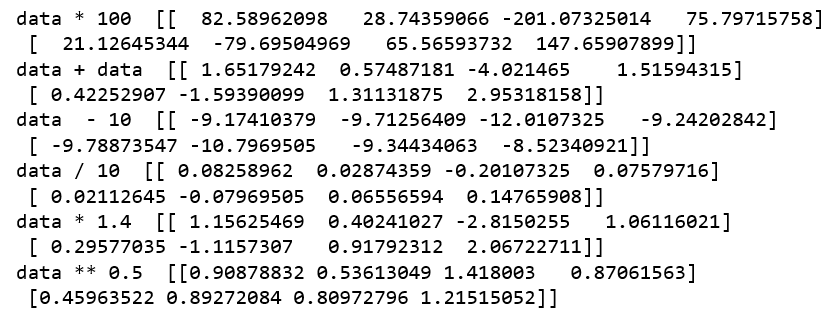


## **Vectorization property of NumPy**

* Arrays are important enables you to do any arithmetic operations between equal-size arrays without writing any for loops. It applies the operation element-wise.
* Arithmetic operations include, multiplication (\*), subtraction (-), addition (+) and division (/) and power ( ). Arithmetic operations with scalars propagate the scalar argument to each element in the array.

∗∗





## **Basic indexing and slicing**

**One dimensional arrays are simple and they act similarly to Python lists.**

In [17]: array = np.arange(20) print(array)

print(array[5]) *# Sixth element* print(array[5:8]) *# Sixth element to 8th element* array[5:8] = 100

print(array) *# Value given to a slice is propagated to the entire selection*

[ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19]

5

[5 6 7]

[ 0 1 2 3 4 100 100 100 8 9 10 11 12 13 14 15 16 17

18 19]

## **Boolean Indexing**

**Consider an example where we have some data in an array and an array of names with duplicates.**

**Use the randn function in numpy.random to generate some random normally distributed data:**

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

names = np.array(['Arun', 'Bill', 'Jay', 'Arun', 'Jay', 'Butler']) data = np.random.randn(6, 4)

In [2]: print('names',names)

names ['Arun' 'Bill' 'Jay' 'Arun' 'Jay' 'Butler']

In [3]: print('data', data)

data [[-0.37872613 1.17223464 -0.99871173 -0.38798951]

[-0.46201744 0.41672788 0.08557869 1.32186669]

|  |  |
| --- | --- |
| [-0.51540156 | -0.64393741 -0.07013691 0.35116435] |
| [ 0.31843458 | -0.19230838 -0.82800615 -0.78421484] |
| [ 0.72898486 | 0.43962914 0.21475248 0.01778945] |
| [-0.53407184 | -1.82479575 0.99885712 -1.52308785]] |

In [4]: names == 'Jay' *# Get the corresponding names with Jay*

Out[4]: array([False, False, True, False, True, False])

In [5]: data[names == 'Jay'] *# Select number from the rows where names == 'Jay'*

|  |  |  |  |
| --- | --- | --- | --- |
| Out[5]: | array([[-0.51540156, | -0.64393741, -0.07013691, | 0.35116435], |
|  | [ 0.72898486, | 0.43962914, 0.21475248, | 0.01778945]]) |

In [6]: data[names == 'Jay', 2:]*# Select from the rows where names == 'Jay' index the columns,*

Out[6]: array([[-0.07013691, 0.35116435],

[ 0.21475248, 0.01778945]])

**To select everything but ’Jay’, you can either use != or negate the condition using ~:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| In [7]: | data[names != 'Jay'] |  | | |
| Out[7]: | array([[-0.37872613, | 1.17223464, | -0.99871173, | -0.38798951], |
|  | [-0.46201744, | 0.41672788, | 0.08557869, | 1.32186669], |
|  | [ 0.31843458, | -0.19230838, | -0.82800615, | -0.78421484], |
|  | [-0.53407184, | -1.82479575, | 0.99885712, | -1.52308785]]) |
| In [8]: | condition = names == data[~condition] | 'Jay' |  |  |
| Out[8]: | array([[-0.37872613, | 1.17223464, | -0.99871173, | -0.38798951], |
|  | [-0.46201744, | 0.41672788, | 0.08557869, | 1.32186669], |
|  | [ 0.31843458, | -0.19230838, | -0.82800615, | -0.78421484], |
|  | [-0.53407184, | -1.82479575, | 0.99885712, | -1.52308785]]) |

**To set all the negative values in data to 0 we need only do:**

In [9]: data[data < 0] = 0

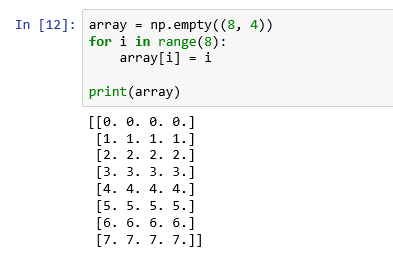
In [10]: print(data)

|  |  |  |
| --- | --- | --- |
| [[0. 1.17223464 | 0. | 0. ] |
| [0. 0.41672788 | 0.08557869 | 1.32186669] |
| [0. 0. | 0. | 0.35116435] |
| [0.31843458 0. | 0. | 0. ] |
| [0.72898486 0.43962914 | 0.21475248 | 0.01778945] |
| [0. 0. | 0.99885712 | 0. ]] |

In [11]: data[names == 'Jay'] = 10 print(data)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [[ 0. | 1.17223464 | 0. | 0. ] | |
| [ 0. | 0.41672788 | 0.08557869 | 1.32186669] | |
| [10. | 10. | 10. | 10. | ] |
| [ 0.31843458 | 0. | 0. | 0. | ] |
| [10. | 10. | 10. | 10. | ] |
| [ 0. | 0. | 0.99885712 | 0. | ]] |

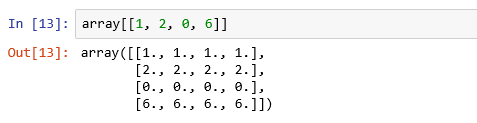
### Indexing using integer arrays



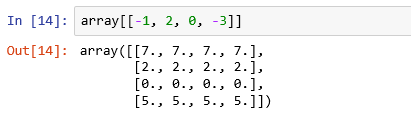
### Selecting a subset of rows

For selecting out a subset of the rows in a particular order, you pass a list or ndarray of integers specifying the desired order.

For example, to select the second row, third row, first row and seventh row of the object, array, we can code as follows:

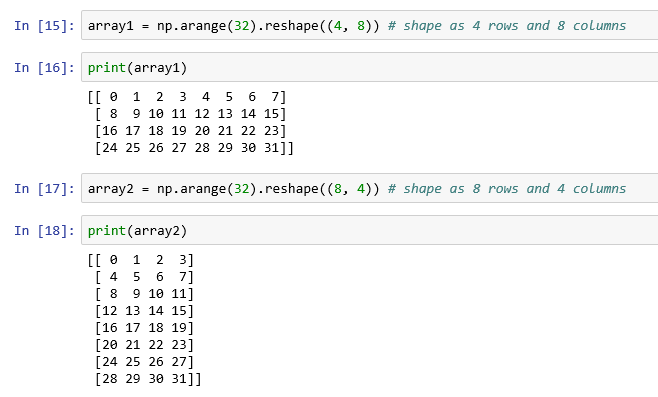


For example, to select the last row, third row, first row and third row from the last row of the object, array, we can code as follows:

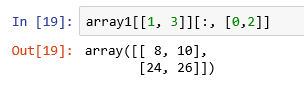


### reshape

**Reshape() gives a new shape to the array without changing the data.**



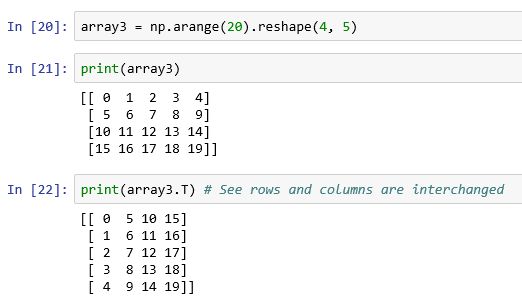
**Select first and second columns of the second and fourth row of the array, array1.**



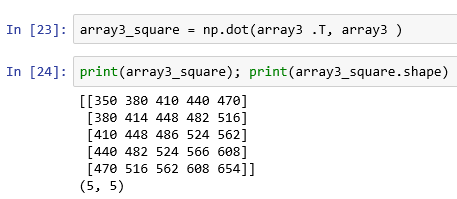
### Transposing Arrays and Swapping Axes

**Arrays have the transpose method and the special T attribute:**

*This will result in interchanging the rows and columns of the array.*

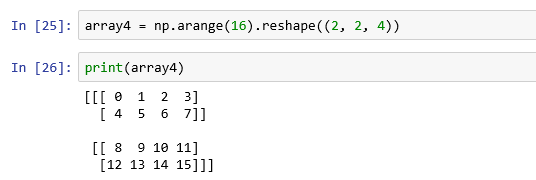


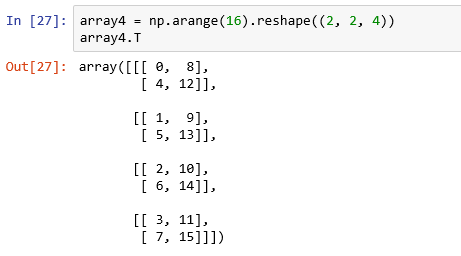
**Compute the inner matrix product using np.dot**



**If you multiply any array with its transpose, you get a square matrix.**

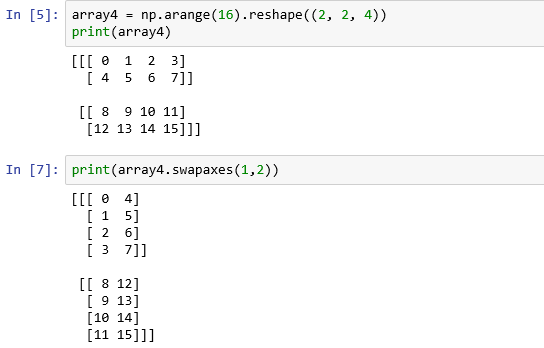
**For higher dimensional array (3-d),transpose will accept a tuple of axis numbers to permute the axes.**





**Swap axes**

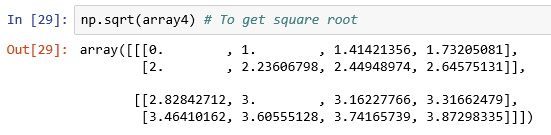
**The ndarray has the method swap-axes, which takes a pair of axis numbers and switches the indicated axes to rearrange the data**



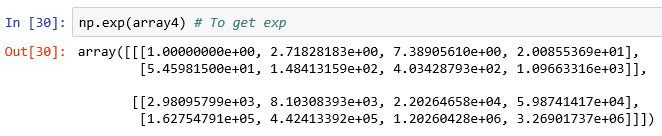
### Mathematical functions

|  |  |  |
| --- | --- | --- |
| **#** | **Function** | **Descritpion** |
| 1 | np.sqrt() | To get square root of the array elements |
| 2 | np.exp() | To get the exponential value of the array elements |
| 3 | np.min() | To get the minimum value in the array elements |
| 4 | np.max() | To get the maximum value in the array elements |
| 5 | np.abs() | To get the absolute value in the array elements |
| 6 | np.floor() | To get the floor value in the array elements |
| 7 | np.mean() | To get the arithmetic mean of the array elements |
| 8 | np.sum() | To get the sum of the array elements |
| 9 | cumsum() | cumsum return an array of the same size, but with the partial aggregates computed along the indicated axis according to each lower dimensional slice |
| 10 | cumprod() | To compute the cumulative product of array elements over a given axis |
| 11 | std() | To get the standard deviation of the array elements |
| 12 | var() | To get the variance of the array elements |

1. **Get square root of the array elements**



1. **Get the exponential value of the array elements**



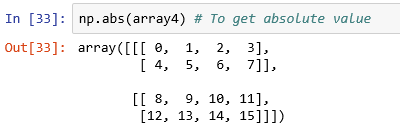
1. **Get the minimum value in the array elements**



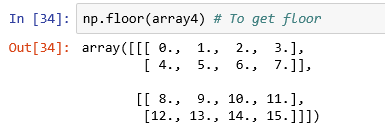
1. **Get the maximum value in the array elements**



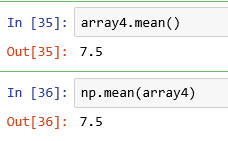
1. **Get the absolute value in the array elements**



1. **Get the floor value in the array elements**



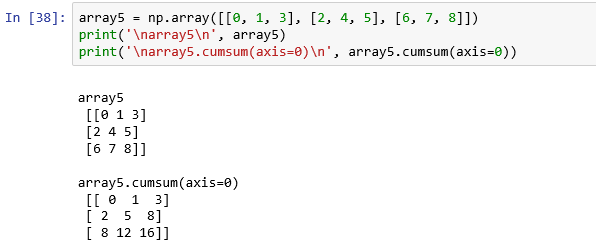
1. **Get the arithmetic mean of the array elements**



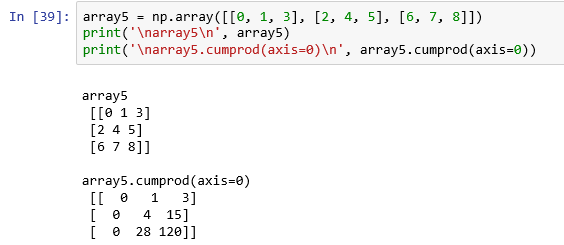
1. **Get the sum of the array elements**



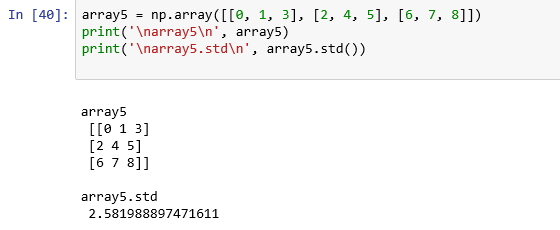
1. **Get the partial aggregates of the array elements**



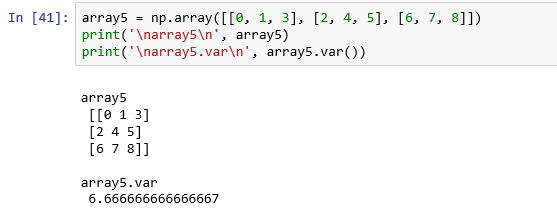
1. **Get the cumprod of the array elements**



1. **Get the standard deviation of the array elements**



1. **Get the variance of the array elements**



# Pandas

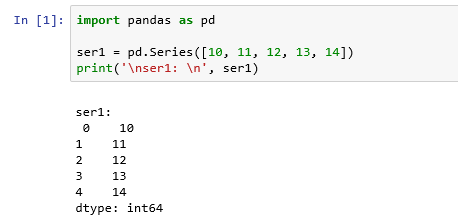
**Pandas core components**

* Series and DataFrames are two core components of Pandas.
* A series is essentially a column while a DataFrame is a multi-dimensional table made up of a collection of Series.
* Many operations such as calculating mean, finding shape(dimension) can be performed on both DataFrames and Series as they are quite similar.



## **Series**

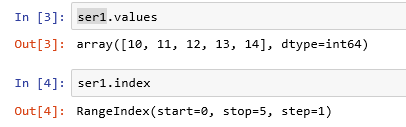
* A series is a one-dimensional object (like array) containing a sequence of values and an index ( associated array of data labels).



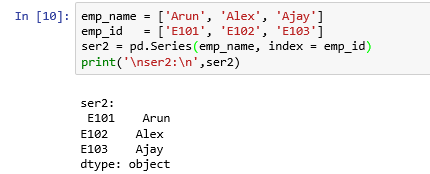
The output shows the index on the left and the values on the right.

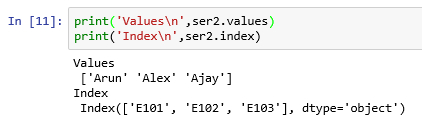
We have not specified any index and default values ranging from 0 to n-1 (where n is the length of the data) is used as index.

**How do you get the array representation and index of the series?**



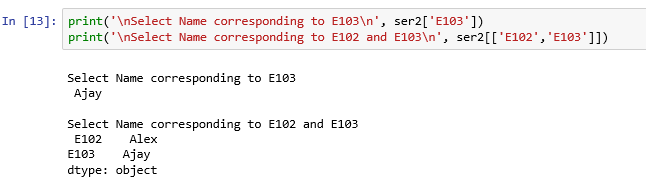
**Don't you think, it is good to create a series with an index identifying each data point with a label? If yes, how?**



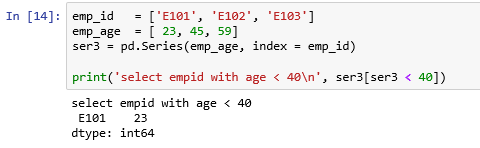


**Access values in a Pandas Series**

Use labels in the index for selecting a single value or a set of values.

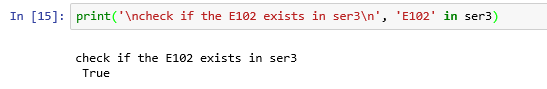


We can apply NumPy functions such as math function, fileting a Boolean array to Pandas series.

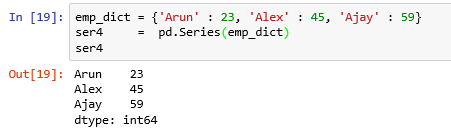


You can treat pandas Series as an ordered dictionary and use in many contexts.

**Check for the existence of a value in the series.**

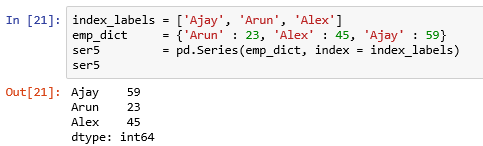
****

**You can create a Pandas series from a dictionary.**

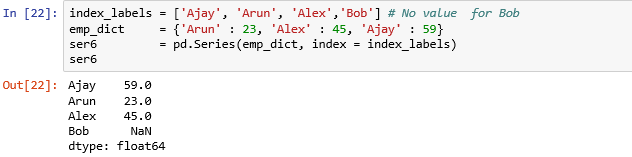
****

**When you pass only a dictionary, the index in the resulting Series is having the keys of the dictionary in sorted order.**

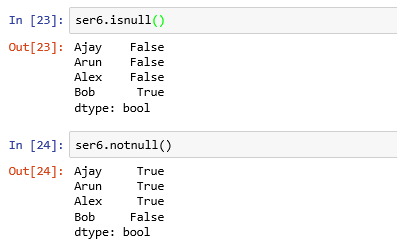
**You can change this order.**

****

**When you pass an index for a non-existing value, missing value (NaN) appears.**

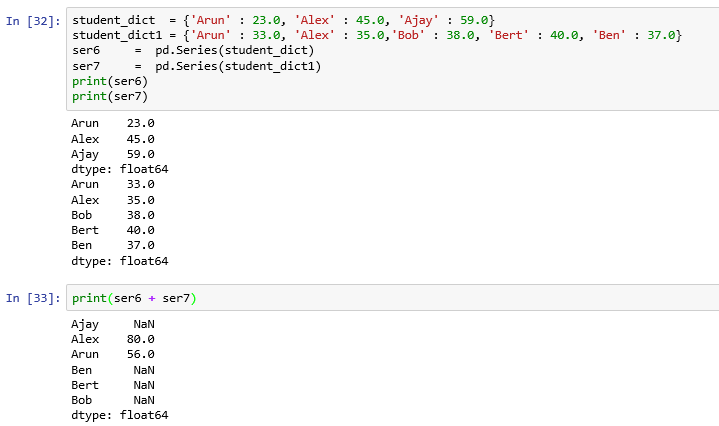


**The isnull() and notnull() functions in pandas are used to detect missing values.**

****

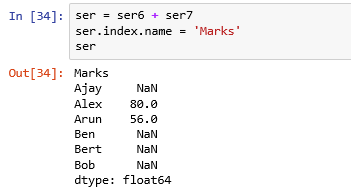
**Arithmetic operations on Series**

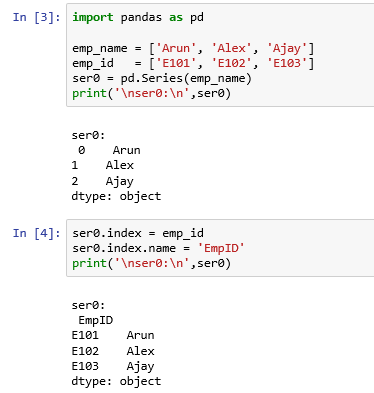
*In arithmetic operations, series automatically aligns by index label as shown below:*

****

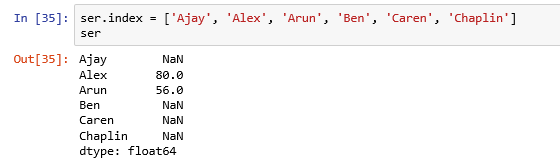
**NaN appears when there is no value for the key in both series.**

**Both series object and its index have a name attribute and a name can be assigned.**

****

****

**A series index can be re-assigned.**

****

## **Data Frames**

A data frame is a two-dimensional data structure, aligned like a table with rows and columns.

**Word Pandas is derived from ‘panel’ and ‘data’**

### Features of a data frame

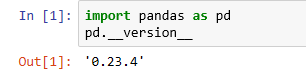
1. Size of the data frame is mutable meaning that you add rows and / or columns
2. Labeled axes (rows and columns)
3. Columns can have different data types
4. Allows arithmetic operations on rows or columns

### History

|  |  |
| --- | --- |
|  | ***http://wesmckinney.com/pages/about.html***  ***https://www.linkedin.com/in/wesmckinn/*** |

Ref: https://en.wikipedia.org/wiki/Pandas\_(software)

### Which version of pandas you are using?



I am using the version 0.23.4. Though it is not latest, it is not very old.

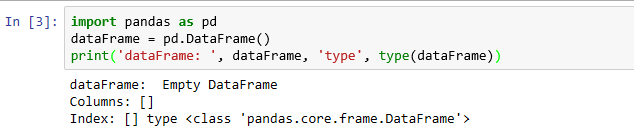
### How do you create a pandas data frame?

A data frame can be created using the following constructor:

pandas.DataFrame(data, index, columns, dtype, copy)

|  |  |  |
| --- | --- | --- |
| **#** | **Parameter** | **Description** |
| 1 | data | Data can be of various forms such as series or ndarray or map or lists or dictionaries or constants or another data frame. |
| 2 | index | Index is used for the row labels, the index to be used. Optional. Default is np.arange(n). |
| 3 | columns | Columns is used for the column labels. Default np.arange(n). |
| 4 | dtype | Describes the data type of each column. |
| 5 | copy | copy is used for copying of data. |

**How do you create an empty DataFrame?**



A pandas DataFrame can be creates using various inputs like:

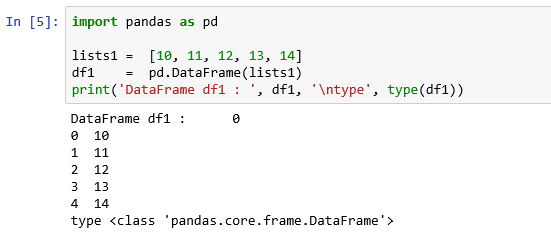
1. Lists

2. Dictionary

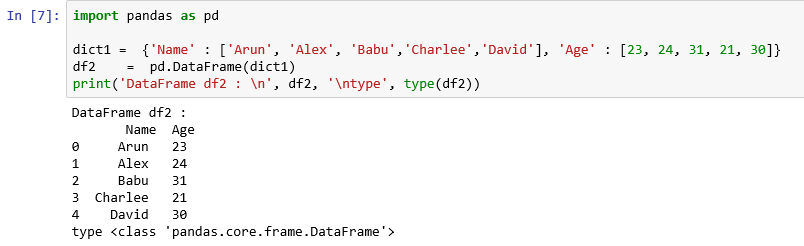
3. NumPy ndarrays

4. Another dataframe

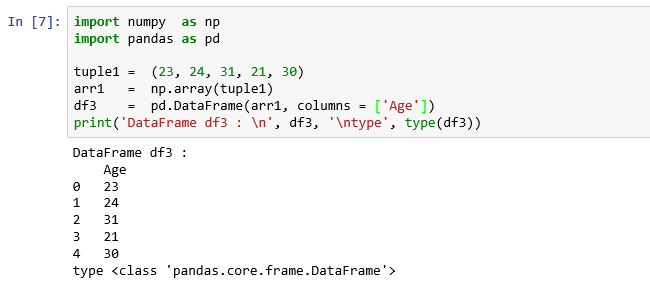
1. **Using Lists as input to create a data frame**



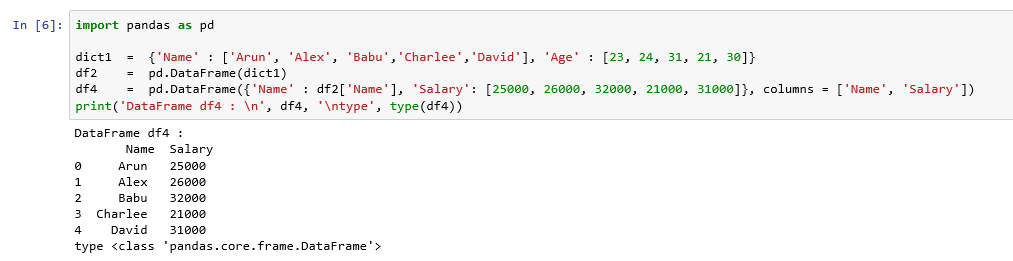
1. **Using dictionary as input to create a data frame**

****

1. **Using NumPy arrays as input to create a data frame**

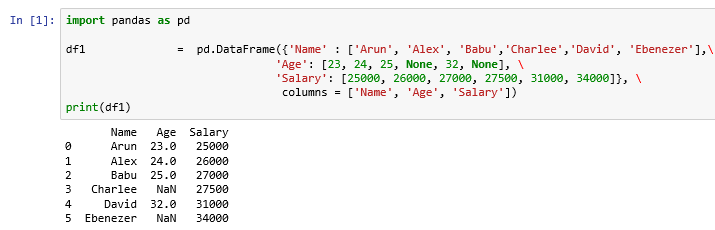


1. **Using another data frame as input to create a data frame**

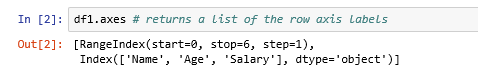
****

### basic funtionalities

* axes returns a list of the row axis labels
* dtype returns the data type of the object
* empty returns True if the given object is empty
* ndim() returns the number of dimensions of the underlying object
* size() returns the number of elements in the underlying object
* head() returns the first n rows, 5 is the default
* tail() returns the last n rows, 5 is the default



1. **axes returns a list of the row axis labels**

****

We have obtained the column labels of the data frame, df1 and also the data type of the column as object.

1. **dtype returns the data type of the object**

****

We have obtained the data type of the column, Age and it is float 64.

1. **empty returns True if the given object is empty**

****

The output of the command empty is False as the data frame, df1 is not empty.

1. **ndim() returns the number of dimensions of the underlying object**

****

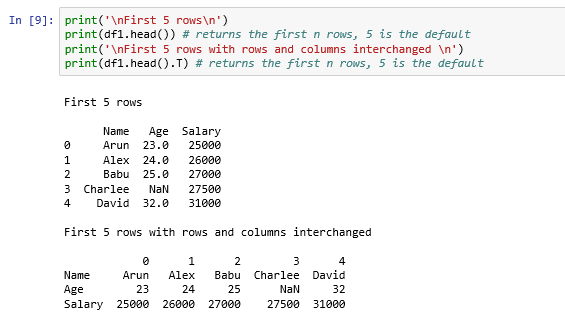
The output of the command ndim is 2 as the data frame is of dimension 2.

1. **size() returns the number of elements in the underlying object**

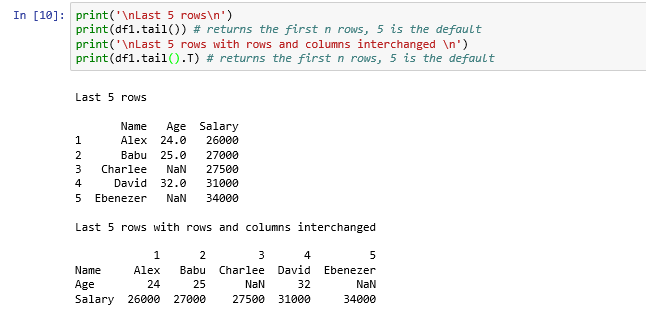
****

The output of the command size is 18 as the data frame, df1 contains six rows and three columns.

1. **head() returns the first n rows, 5 is the default**

****

1. **tail() returns the last n rows, 5 is the default.**



### Add and remove columns from a pandas data frame

To add a new column, declare a list of values to be converted to a column and assign a label to the column name and assign it to the list.



To remove the column from a data frame, use del() command.

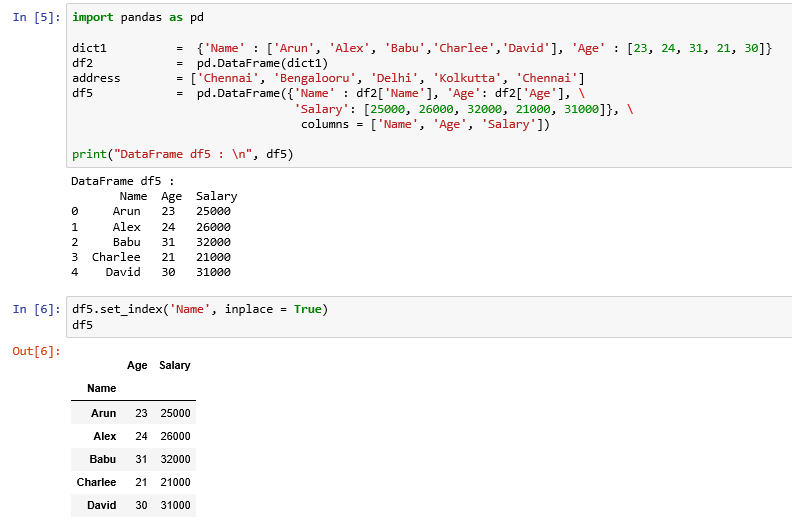


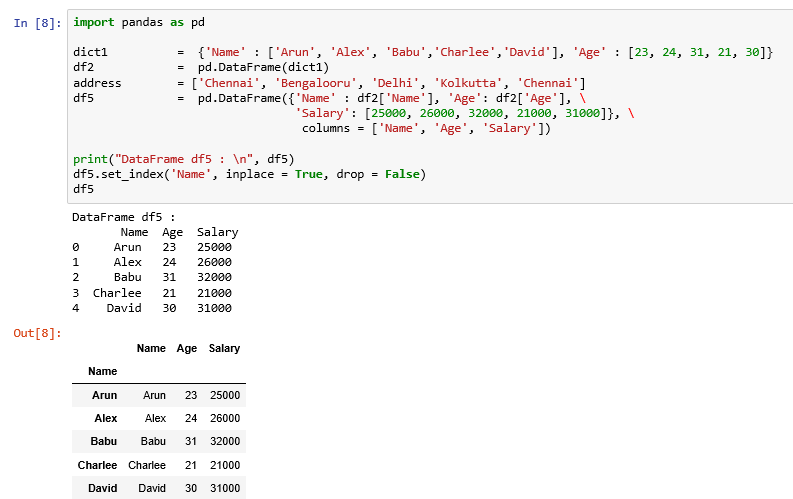
### Indexing with columns of a data frame

There are occasions, when you want to use one or more columns of the data frame as the row index. Sometimes, you want to move the row index to the columns of the data frame.

#### Set index with one or more columns of the data frame

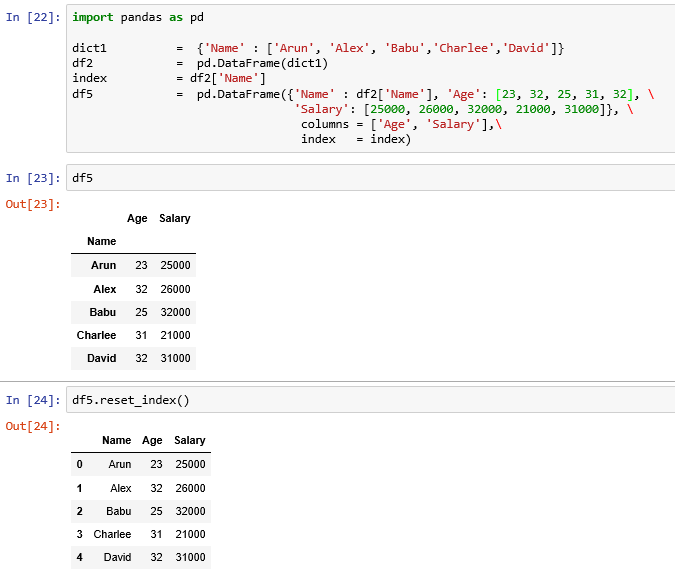
set\_index(), function will create a new DataFrame using one or more of the data frame columns.



By default, the columns that are used in the index are removed from the Data frame.

#### Move Index levels to the columns

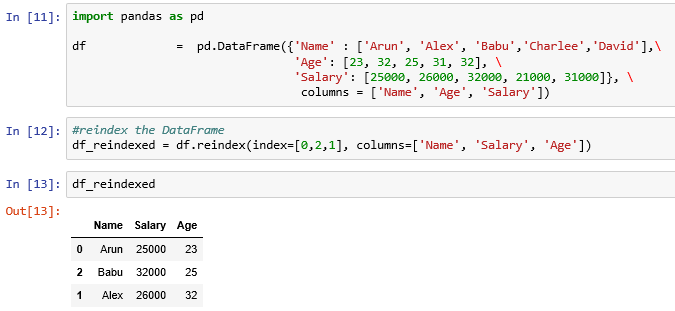
Using reset\_index() function, we can move the index back to the columns.



### Re-indexing

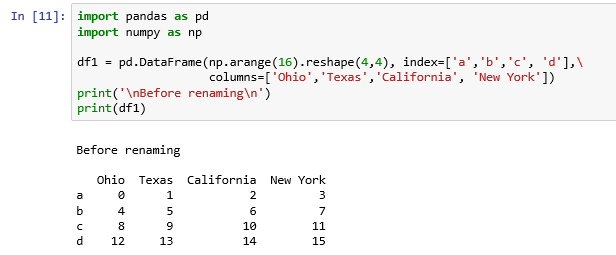
Re-indexing changes the row labels and column labels of a DataFrame.

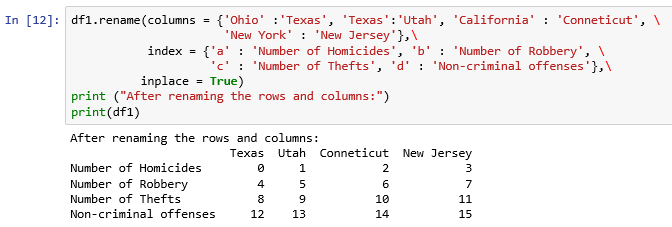
#### Reorder the existing data



### rename

The rename() method allows you to relabel an axis based on mapping a dictionary or series or an arbitrary function.





<https://www.tutorialspoint.com/python_pandas/python_pandas_reindexing.htm>

<https://www.geeksforgeeks.org/python-pandas-dataframe-reindex/>

https://www.tutorialspoint.com/python\_pandas/python\_pandas\_indexing\_and\_selecting\_data.htm

<https://www.tutorialspoint.com/python_pandas/python_pandas_reindexing.htm>

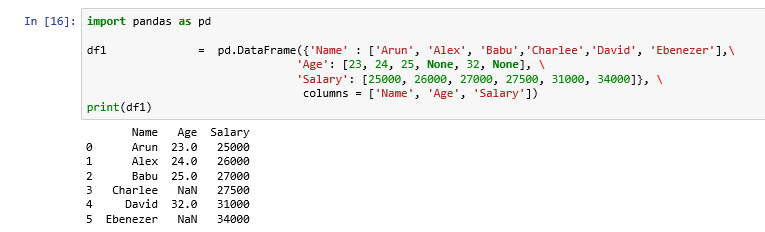
<https://www.edureka.co/blog/python-pandas-tutorial/>

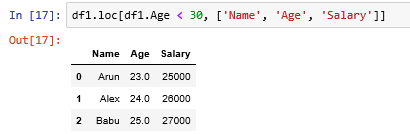
### row selection, addition and deletion

#### Row selection

By passing row label to a loc() function, rows can be selected. From the df5, data frame, we are selecting rows with age < 30 and only the column, Age.

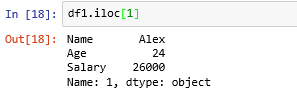
1. **Select by label**

****

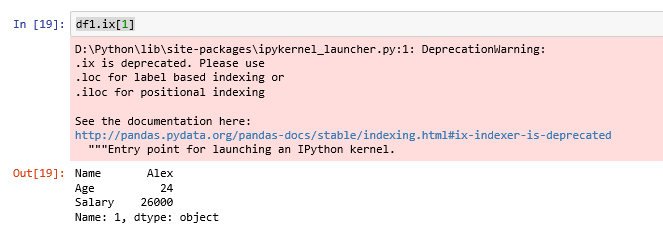


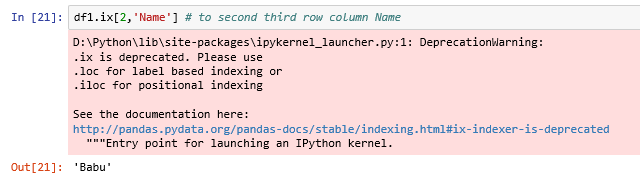
1. **Select by integer location**

Select second row of the data frame, df5.



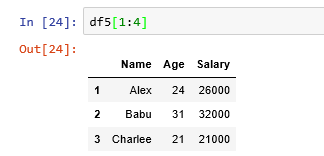
1. **Select by hybrid – integer & label based.**

**This method is deprecated as per the warning message. Use loc and iloc methods.**

****

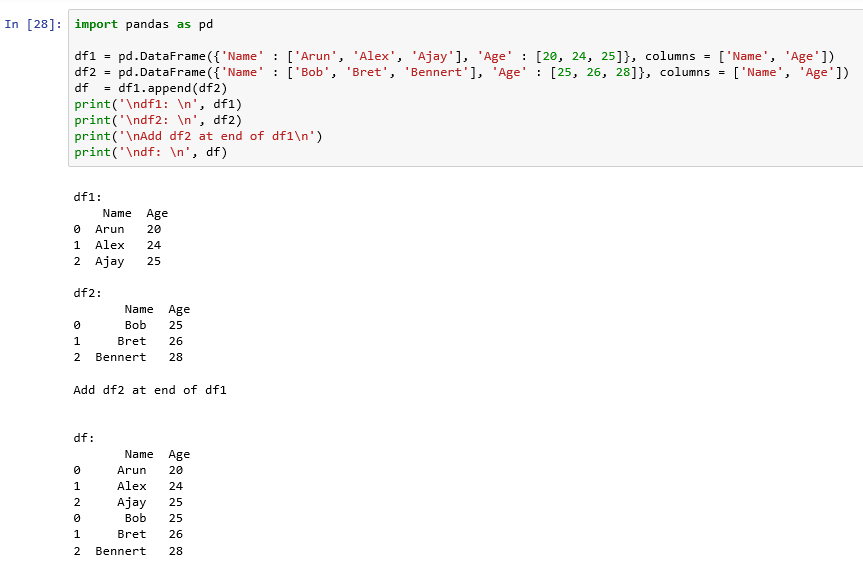
1. **Multiple rows selection**

Using : operator, multiple rows can be selected. We select second to fourth row from df5, data frame.



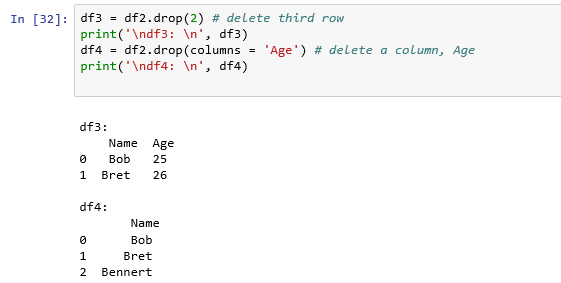
#### Row addition

By using the append() function, we can add new rows to a Data Frame.



#### rows deletion

Use drop() to delete or drop rows from a DataFrame.



### descriptive statistics

Descriptive statistics provide simple summaries about the sample and about the observations that have been made. Such summaries may be either quantitative, i.e. summary statistics, or visual, i.e. simple-to-understand graphs. - <https://en.wikipedia.org/wiki/Descriptive_statistics>

**Important functions in pandas**

1. count() returns the number of non-null observations

2. sum() returns the sum of values

3. min() returns the minimum value

4. max() returns the maximum value

5. abs() returns the absolute value

6. prod() returns the product of values

7. cumsum() returns the cumulative sum of values

8. cumprod() returns the cumulative product of values

9. mean() returns the arithmetic mean of values

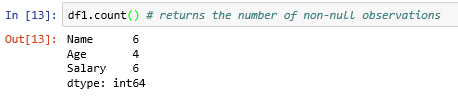
10. median() returns the median of values

11. mode() returns the mode of values

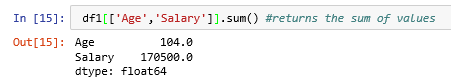
12. std() returns the standard deviation of values

13. info() returns the summary information about the data frame

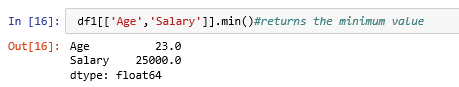
1. **count() returns the number of non-null observations**



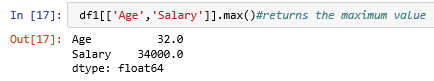
1. **sum() returns the sum of values**



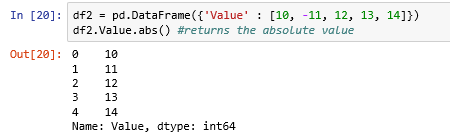
1. **min() returns the minimum value**



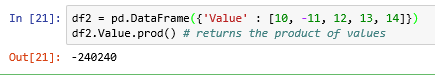
1. **max() returns the maximum value**



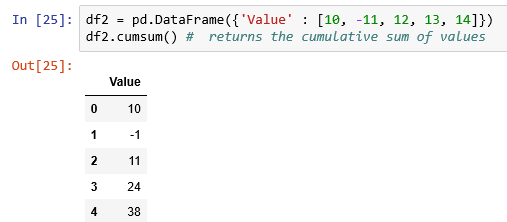
1. **abs() returns the absolute value**



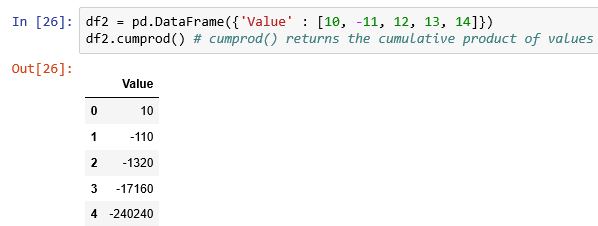
1. **prod() returns the product of values**



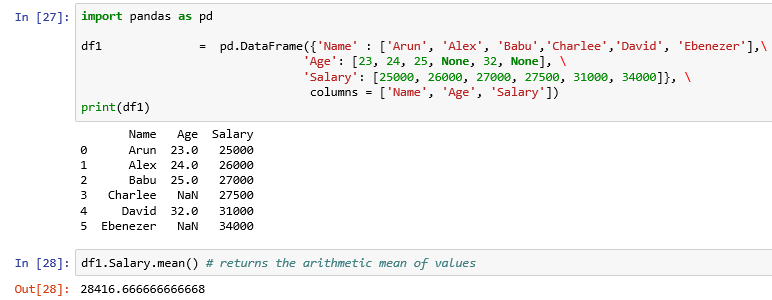
1. **cumsum() returns the cumulative sum of values**



1. **cumprod() returns the cumulative product of values**



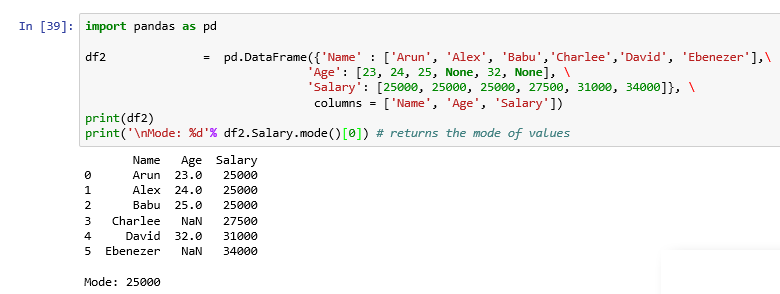
1. **mean() returns the arithmetic mean of values**

****

1. **median() returns the median of values**



1. **mode() returns the mode of values**

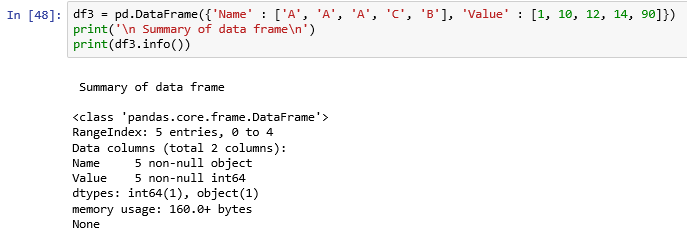


1. **std() returns the standard deviation of values**

****

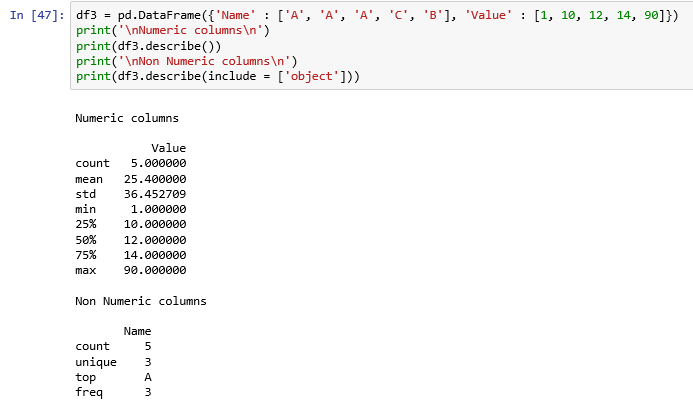
1. **info() returns the summary of the data frame**

The info() method prints information such as column names. data type, index, non null values and memory usage about a data frame.

****

**Data summary in pandas**

* The describe() function produces a summary of statistics pertaining to the columns of the data frame.
* The describe() function produces a summary of statistics pertaining to the columns of the data frame.
* By default, it gives summary for the numeric columns. Summary contains measures such as count, mean, standard deviation, min, Quartile1 (25%), Quartile 2 (Median), Quartile 3 (75%) and max.
* For columns other than numeric such as object, you need to specify in the include parameter of the describe() function as include = ['object']. It gives measures such as count, unique elements, top (element that occurs most) and its frequency.



### Missing values treatment in pandas

#### Missing data is always an issue in real world problems

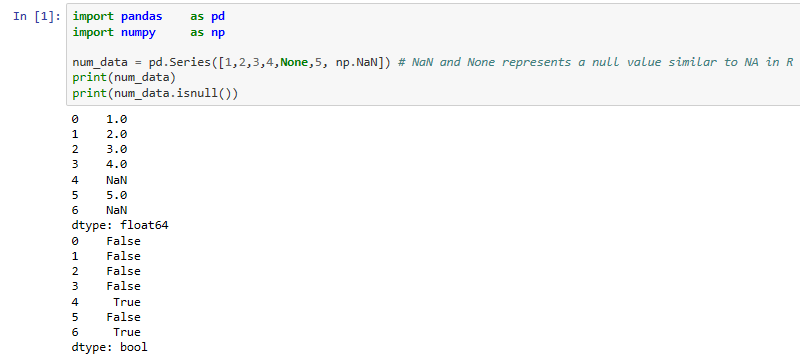
* Data can have missing values for reasons such as data corruption, failure to record etc.
* We need to understand:
* How to identify the missing values in a large dataset?[¶](file:///D:\RRD\Missing+Values+Treatment.html#1.-How-to-identify-the-missing-values-in-a-large-dataset?)
* How to make invalid data or corrupt data as missing values?[¶](file:///D:\RRD\Missing+Values+Treatment.html#2.-How-to-make-invalid-data-or-corrupt-data-as-missing-values?)
* How to treat the missing values?[¶](file:///D:\RRD\Missing+Values+Treatment.html#3.-How-to-treat-the-missing-values?)

Please refer to:

* [*https://www.tutorialspoint.com/python\_pandas/python\_pandas\_missing\_data.htm*](https://www.tutorialspoint.com/python_pandas/python_pandas_missing_data.htm)
* [*https://machinelearningmastery.com/handle-missing-data-python/*](https://machinelearningmastery.com/handle-missing-data-python/)
* [*https://towardsdatascience.com/the-tale-of-missing-values-in-python-c96beb0e8a9d*](https://towardsdatascience.com/the-tale-of-missing-values-in-python-c96beb0e8a9d)
* *https://www.tutorialspoint.com/python\_pandas/python\_pandas\_basic\_functionality.htm*

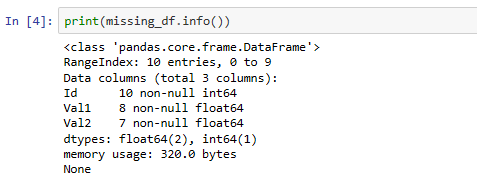
#### Check for Missing values

* Pandas has two functions, isnull() and notnull() to detect missing values.
* Pandas handles missing data efficiently. All of the descriptive statistics on pandas objects exclude missing data by default.
* Pandas uses the floating point value, NaN (Not a number), a sentinel value to represent missing data.
* Numpy NaN and None represents a null value similar to NA in R





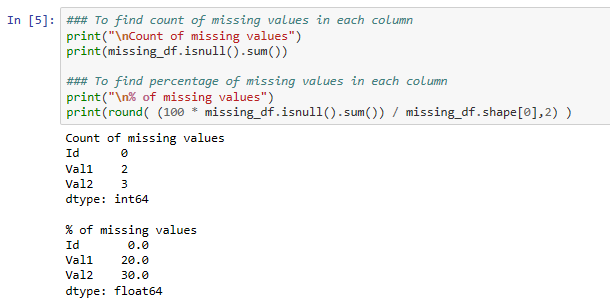
**How will you the count of missing values in each of the pandas data frame?**



* From the info() function, we observe that there are 10 observations and three columns in the data frame, missing\_df.
* There are only 8 non null entries for the column, Val1 indicating that there are 2 (= 10 - 8) missing values for Var1 and 7 non null entries for the column, Val2 indicating that there are 3 (= 10 - 7) missing values for Var2.

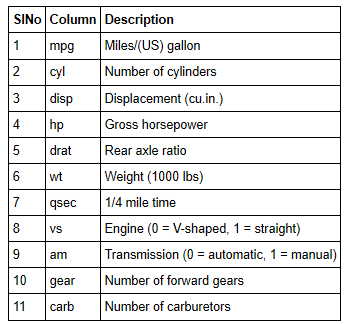
**My question is how do you find this programmatically?**

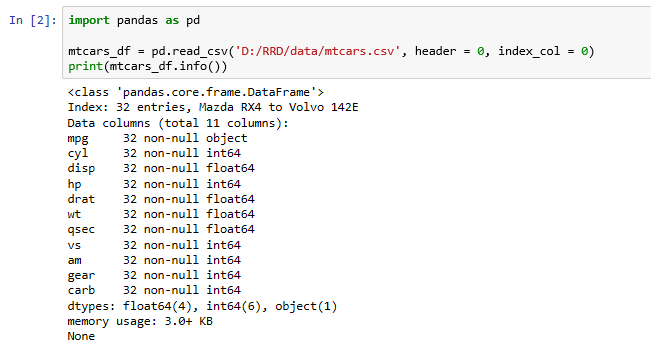
*By applying the function, sum() to the output of missing\_df.isnull() command, you get column-wise missing counts.*

****

**How to make invalid data or corrupt data as missing values?**

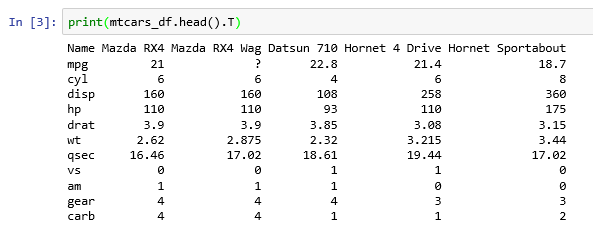
* Consider the dataset mtcars
* The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models).
* This data set has 32 observations on 11 (numeric) variables.





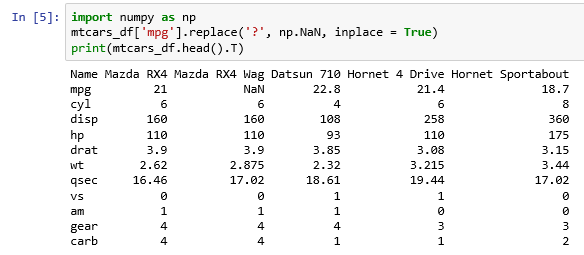
**Inference**

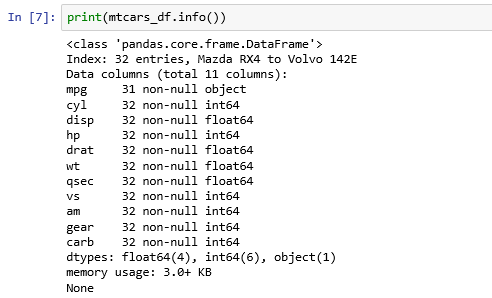
* There are no missing values as per the above result of info() function.
* But the Data type of the column, mpg which is supposed to be integer is string (object).
* Check first five rows

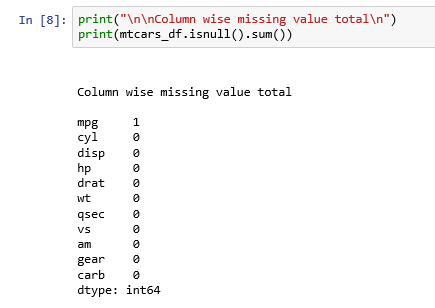


**Inference**

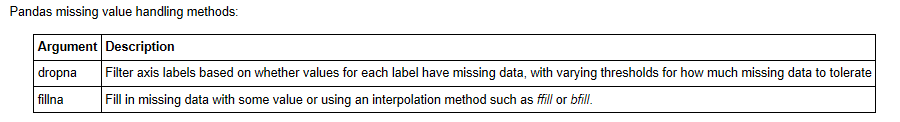
* There is a non-numeric value of ? in the column, mpg which is invalid. We shall replace it with a null value as follows:





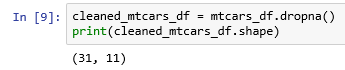


#### How to treat the missing values?



**a) Dropping null or missing value**

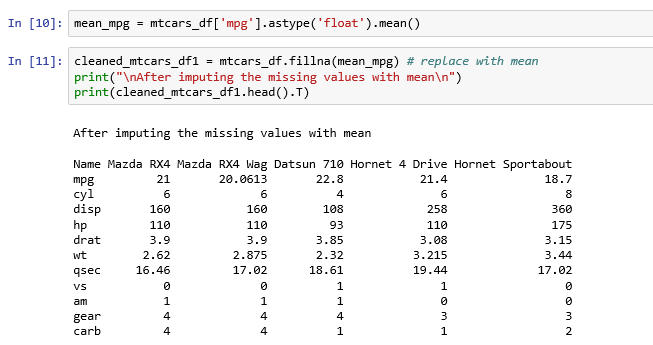
* This is the fastest and easiest way to handle missing values. We drop row-wise.
* If the percentage of missing values > 40%, we drop the column.
* This method reduces the number of observations of the model.

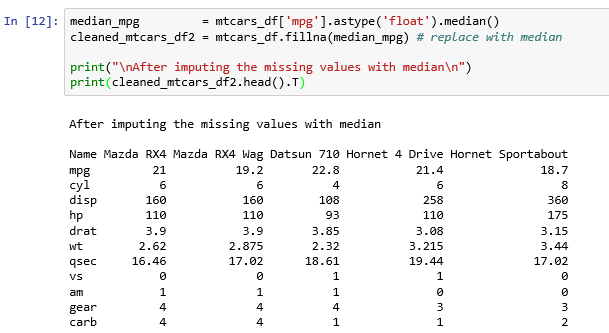


* We observe that the number of observations is reduced to 31 from 32.
* It is advisable to report to the customer about missing values and outliers.

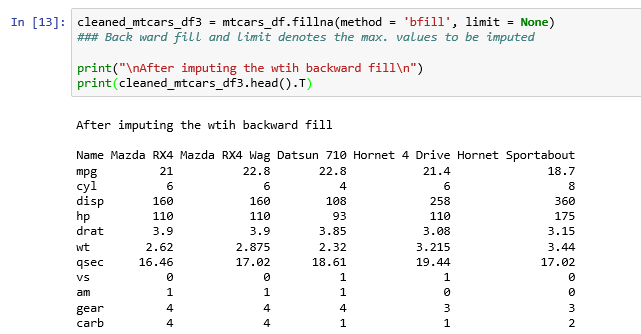
**b) Filling missing values**

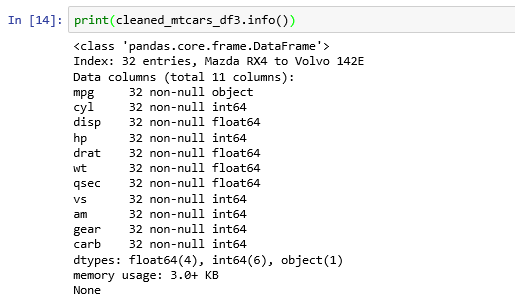
1. Replace null or missing value with a test statistic such as mean or median

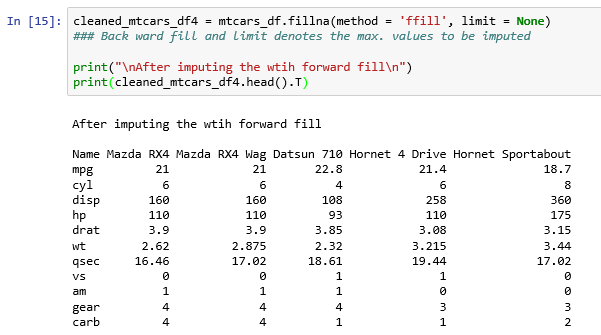


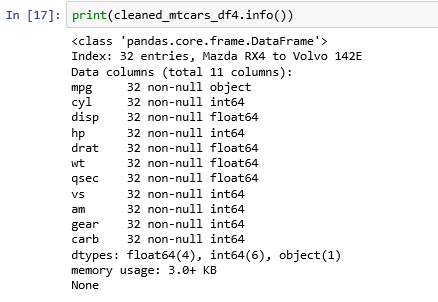


1. Use back fill or forward fill to propagate next or previous values respectively.









# Exercises

You have seen the how the data set, mtcars is used to understand the data using descriptive statistics, missing values and its imputation etc.

Understand the following data sets:

1. Iris data set
2. Boston Housing Prices data set
3. Pima Indians Diabetes data set