Pandas

What is Pandas?

Pandas is a Python library used for working with data sets.

It has functions for analyzing, cleaning, exploring, and manipulating data.

The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Why Use Pandas?

Pandas allows us to analyze big data and make conclusions based on statistical theories.

Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science.

What Can Pandas Do?

Pandas gives you answers about the data. Like:

- Is there a correlation between two or more columns?
- What is average value?
- Max value?
- Min value?

Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values. This is called *cleaning* the data.

Where is the Pandas Codebase?

The source code for Pandas is located at this

github repository

https://github.com/pandas-dev/pandas

Pandas Series

What is a Series?

A Pandas Series is like a column in a table.

It is a one-dimensional array holding data of any type.

The Pandas Series can be defined as a one-dimensional array that is capable of storing various data types. We can easily convert the list, tuple, and dictionary into series using "**series**' method. The row labels of series are called the index. A Series cannot contain multiple columns. It has the following parameter:

- **data:** It can be any list, dictionary, or scalar value.
- **index:** The value of the index should be unique and hashable. It must be of the same length as data. If we do not pass any index, default **np.arrange(n)** will be used.
- **dtype:** It refers to the data type of series.
- **copy:** It is used for copying the data.

Create an Empty Series

A basic series, which can be created is an Empty Series.

Example

```
#import the pandas library and aliasing as pd
import pandas as pd
s = pd.Series()
print s
```

Its **output** is as follows –

```
Series([], dtype: float64)
```

Create a Series from ndarray

If data is an ndarray, then index passed must be of the same length. If no index is passed, then by default index will be **range(n)** where **n** is array length, i.e., [0,1,2,3…. **range(len(array))-1].**

Example 1

```
#import the pandas library and aliasing as pd
import pandas as pd
import numpy as np
data = np.array(['a','b','c','d'])
s = pd.Series(data)
print s
```

Its **output** is as follows –

```
0 a
1 b
2 c
3 d
dtype: object
```

Example 2

```
#import the pandas library and aliasing as pd
import pandas as pd
import numpy as np
data = np.array(['a','b','c','d'])
s = pd.Series(data,index=[100,101,102,103])
print s
```

Its **output** is as follows –

```
100 a
101 b
102 c
103 d
dtype: object
```

Retrieve Data Using Label (Index)

A Series is like a fixed-size **dict** in that you can get and set values by index label.

Example 1

Retrieve a single element using index label value.

```
import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

#retrieve a single element
print s['a']
```

Its **output** is as follows –

```
1
```

Example 2

Retrieve multiple elements using a list of index label values.

```
import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

#retrieve multiple elements
print s[['a','c','d']]
```

Its **output** is as follows –

```
a 1
c 3
d 4
dtype: int64
```

Example 3

If a label is not contained, an exception is raised.

```
import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

#retrieve multiple elements
print s['f']
```

Its **output** is as follows –

```
…
KeyError: 'f'
```

Python Pandas DataFrame

Pandas DataFrame is a widely used data structure which works with a two-dimensional array with labeled axes (rows and columns). DataFrame is defined as a standard way to store data that has two different indexes, i.e., **row index** and **column index**. It consists of the following properties:

- The columns can be heterogeneous types like int, bool, and so on.
- It can be seen as a dictionary of Series structure where both the rows and columns are indexed. It is denoted as "columns" in case of columns and "index" in case of rows.

Parameter & Description:

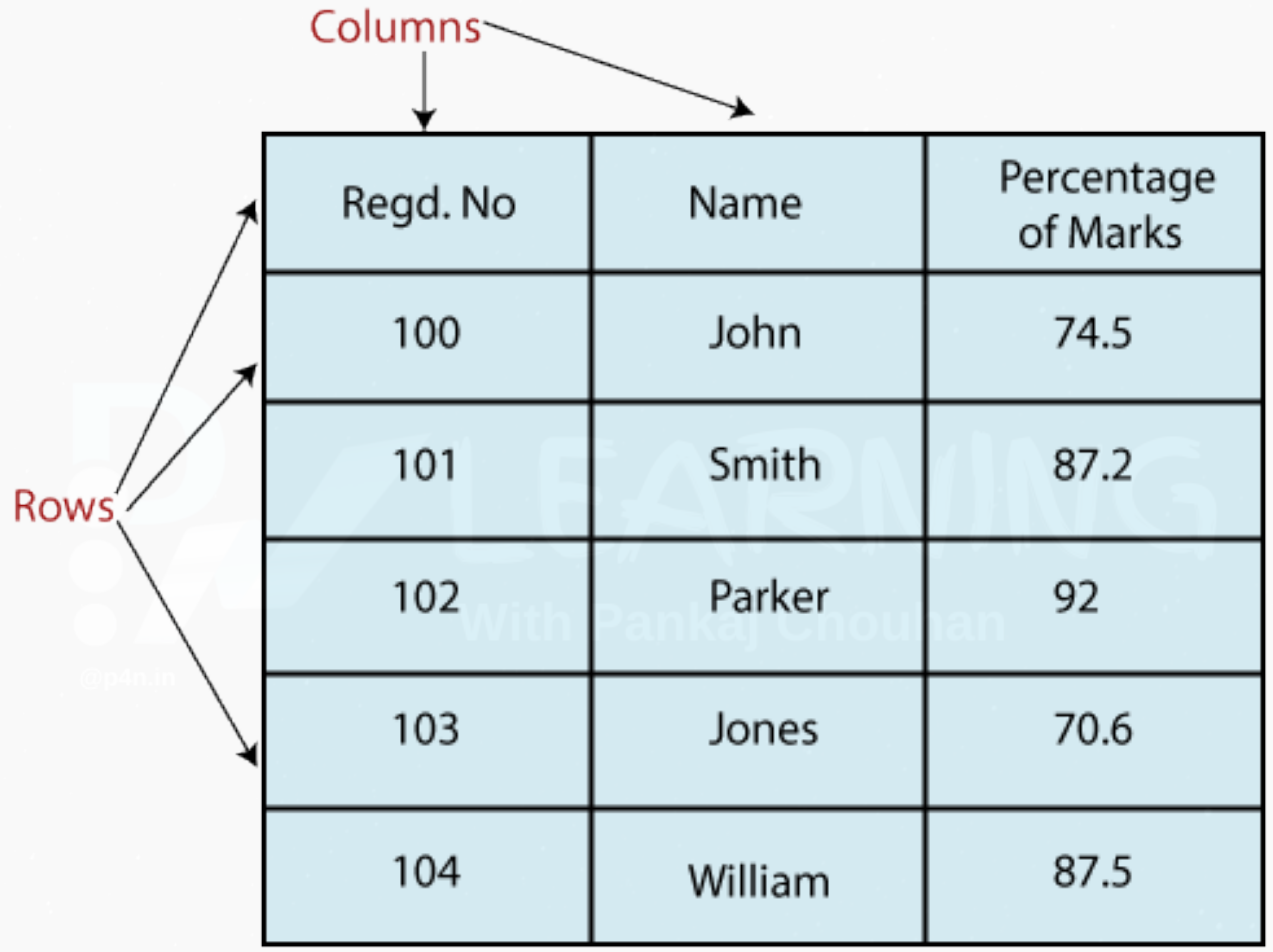
data: It consists of different forms like ndarray, series, map, constants, lists, array.

index: The Default np.arrange(n) index is used for the row labels if no index is passed.

columns: The default syntax is np.arrange(n) for the column labels. It shows only true if no index is passed.

dtype: It refers to the data type of each column.

copy(): It is used for copying the data.



Regd. No	Name	Percentage of Marks
100	John	74.5
101	Smith	87.2
102	Parker	92
103	Jones	70.6
104	William	87.5

Create a DataFrame

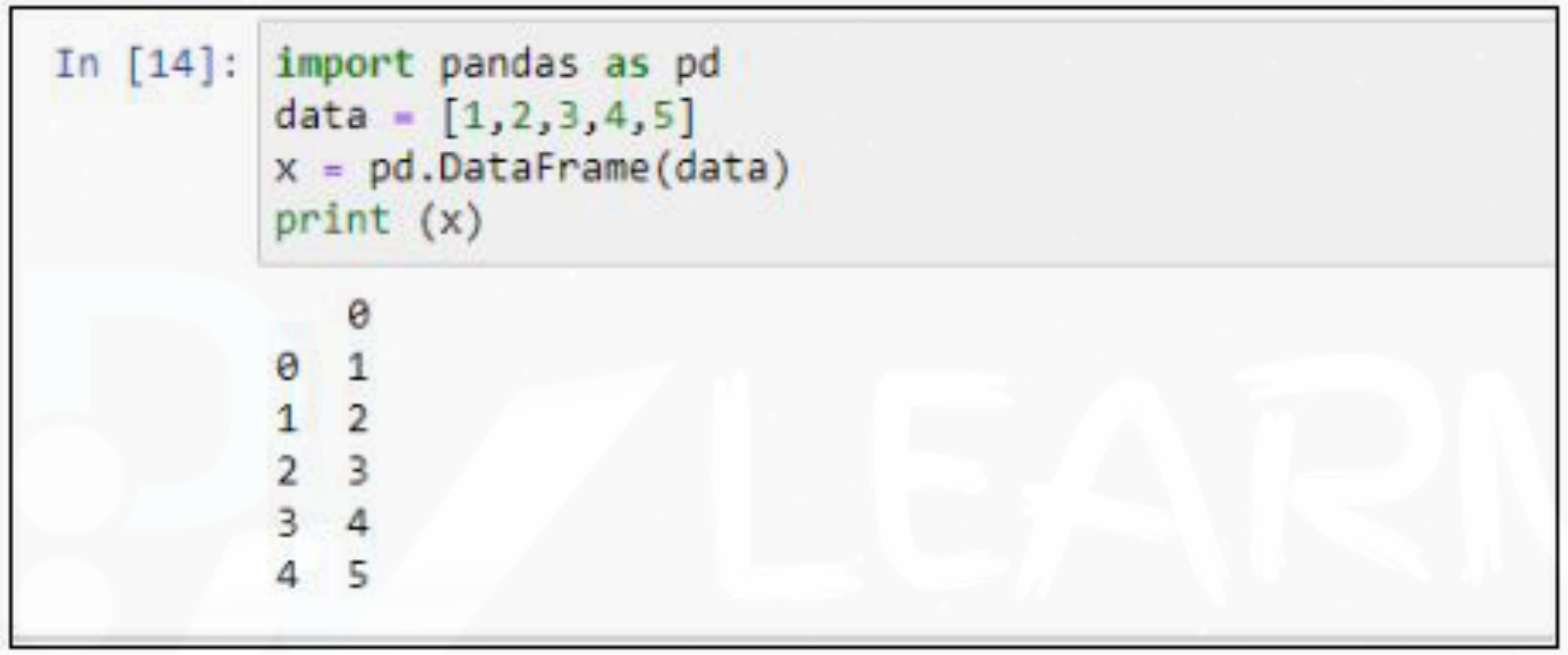
We can create a DataFrame using following ways:

- dict
- Lists
- Numpy ndarrrays
- Series

Basic Operations on DataFrames

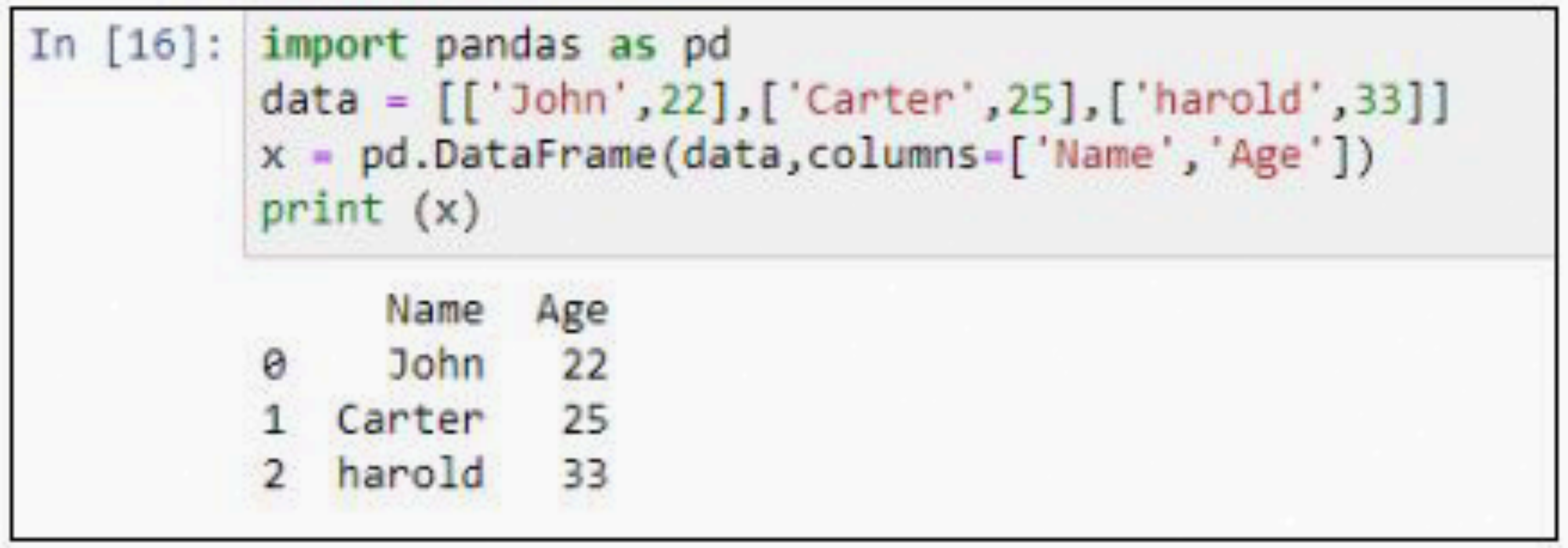
- Create a DataFrame from lists

A DataFrame can be created using a list:



```
In [14]: import pandas as pd
         data = [1,2,3,4,5]
         x = pd.DataFrame(data)
         print (x)

            0
         0  1
         1  2
         2  3
         3  4
         4  5
```

Fig: DataFrame



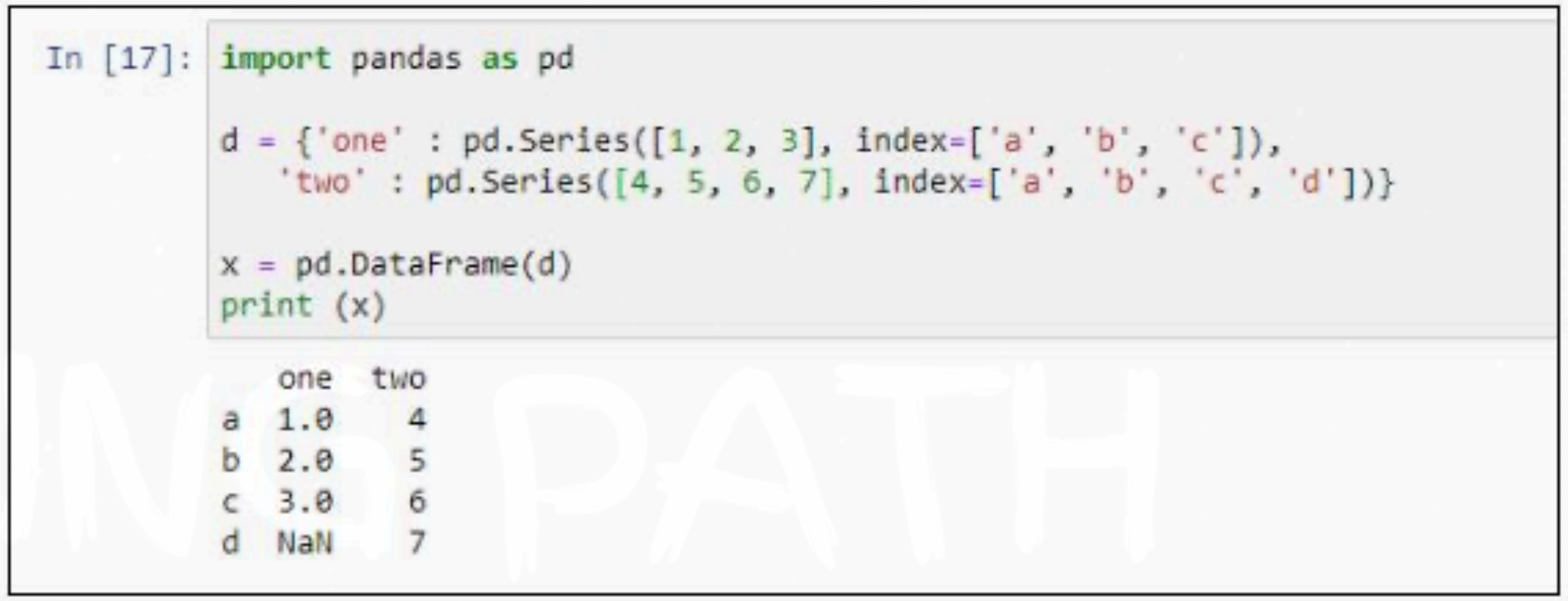
```
In [16]: import pandas as pd
         data = [['John',22],['Carter',25],['harold',33]]
         x = pd.DataFrame(data,columns=['Name','Age'])
         print (x)

              Name  Age
         0    John   22
         1  Carter   25
         2  harold   33
```

Fig: 2-D DataFrame

- Creating a DataFrame from a series dictionary

A series dictionary can be passed to form a DataFrame.

Example:



```
In [17]: import pandas as pd

         d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
            'two' : pd.Series([4, 5, 6, 7], index=['a', 'b', 'c', 'd'])}

         x = pd.DataFrame(d)
         print (x)

            one  two
         a  1.0    4
         b  2.0    5
         c  3.0    6
         d  NaN    7
```

Fig: DataFrame from a Series dictionary

Create an Empty DataFrame

A basic DataFrame, which can be created is an Empty Dataframe.

Example

```
#import the pandas library and aliasing as pd
import pandas as pd
df = pd.DataFrame()
print df
```

Its **output** is as follows –

```
Empty DataFrame
Columns: []
Index: []
```

Create a DataFrame from Lists

The DataFrame can be created using a single list or a list of lists.

Example 1

```
import pandas as pd
data = [1,2,3,4,5]
df = pd.DataFrame(data)
print df
```

Its **output** is as follows –

```
0
0 1
1 2
2 3
3 4
4 5
```

Example 2

```
import pandas as pd
data = [['Alex',10],['Bob',12],['Clarke',13]]
df = pd.DataFrame(data,columns=['Name','Age'])
print df
```

Its **output** is as follows –

```
Name Age
0 Alex 10
1 Bob 12
2 Clarke 13
```

Example 3

```
import pandas as pd
data = [['Alex',10],['Bob',12],['Clarke',13]]
df = pd.DataFrame(data,columns=['Name','Age'],dtype=float)
print df
```

Its **output** is as follows –

```
Name Age
0 Alex 10.0
1 Bob 12.0
2 Clarke 13.0
```

Create a DataFrame from Dict of ndarrays / Lists

All the **ndarrays** must be of same length. If index is passed, then the length of the index should equal to the length of the arrays.

If no index is passed, then by default, index will be range(n), where **n** is the array length.

Example 1

```
import pandas as pd
data = {'Name':['Tom', 'Jack', 'Steve', 'Ricky'],'Age':[28,34,29,42]}
df = pd.DataFrame(data)
print df
```

Its **output** is as follows –

```
Age Name
0 28 Tom
1 34 Jack
2 29 Steve
3 42 Ricky
```

Note – Observe the values 0,1,2,3. They are the default index assigned to each using the function range(n).

Example 2

Let us now create an indexed DataFrame using arrays.

Live

```
import pandas as pd
data = {'Name':['Tom', 'Jack', 'Steve', 'Ricky'],'Age':[28,34,29,42]}
df = pd.DataFrame(data, index=['rank1','rank2','rank3','rank4'])
print df
```

Its **output** is as follows –

```
Age Name
rank1 28 Tom
rank2 34 Jack
rank3 29 Steve
rank4 42 Ricky
```

Create a DataFrame from List of Dicts

List of Dictionaries can be passed as input data to create a DataFrame. The dictionary keys are by default taken as column names.

Example 1

The following example shows how to create a DataFrame by passing a list of dictionaries.

```
import pandas as pd
data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
df = pd.DataFrame(data)
print df
```

Its **output** is as follows –

```
a b c
0 1 2 NaN
1 5 10 20.0
```

Note – Observe, NaN (Not a Number) is appended in missing areas.

Example 2

The following example shows how to create a DataFrame by passing a list of dictionaries and the row indices.

```
import pandas as pd
data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
df = pd.DataFrame(data, index=['first', 'second'])
print df
```

Its **output** is as follows –

```
a b c
first 1 2 NaN
second 5 10 20.0
```

Create a DataFrame from Dict of Series

Dictionary of Series can be passed to form a DataFrame. The resultant index is the union of all the series indexes passed.

Example

```
import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
   'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)
print df
```

Its **output** is as follows –

```
one two
a 1.0 1
b 2.0 2
c 3.0 3
d NaN 4
```

Note – Observe, for the series one, there is no label **'d'** passed, but in the result, for the **d** label, NaN is appended with NaN.

Column Selection

We will understand this by selecting a column from the DataFrame.

Example

```
import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
   'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)
print df ['one']
```

Its **output** is as follows –

```
a 1.0
b 2.0
c 3.0
d NaN
Name: one, dtype: float64
```