# Series in Pandas

In this lesson, the series object of pandas is explained

**We'll cover the following**

## Series #

`Series` can be described as a single column of a 2-D array or a matrix. It has specific index values attached to each row for identification. It can also be imagined as one column of an excel file.

The index values are automatically defined for each `Series` when it is created. These values can also be explicitly defined. Let's look at an example to understand this:

```
# importing pandas in our program
import pandas as pd

# Defining a series object
srs = pd.Series([1,2,3,4,5])

# printing series values
print("The Series values are:")
print(srs.values)

# printing series indexes
print("\nThe Index values are:")
print(srs.index.values)
```

| Index | Values |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 3 |
| 3 | 4 |
| 4 | 5 |

The `srs.values` function on **line 9** returns the values stored in the `Series` object and the function `srs.index.values` on **line 13** returns the index values.

## Defining indexes

In the above example, only series values were defined, but its indexes were automatically generated. Each index corresponds to its respective value in the `Series` object. However, explicitly defining the indexes can be of great help if every value needs to be identified with something meaningful.

Let’s look at an example where the population growth rate of different countries is assigned to the country name.

```
#importing pandas in our program
import pandas as pd

# Defining a series object
srs = pd.Series([11.9, 36.0, 16.6, 21.8, 34.2], index = ['China', 'India', 'USA', 'Brazil', 'Pakis

# Set Series name
srs.name = "Growth Rate"

# Set index name
srs.index.name = "Country"

# printing series values
print("The Indexed Series values are:")
print(srs)
```

| Country | Rate % |
|---|---|
| China | 11.9 |
| India | 36.0 |
| USA | 16.6 |
| Brazil | 21.8 |
| Pakistan | 34.2 |

A `list` can also be passed in the index parameter, and the list values are assigned as indexes.

Two attributes of the `Series` object are used on **line 8** and **line 11**. The attribute `srs.name` sets the name of our series object as can be seen at the end of the output. The attribute `srs.index.name` sets the name for the indexes as can be seen on top of the index names.

## Index consistency #

If there are two `Series` objects with some common index names and some operations need to be performed on them, the `Series` object automatically performs operations on the values of the same index names while maintaining its consistency.

Let's understand this with an example:

```
#importing pandas in our program
import pandas as pd

# Defining two series object
srs1 = pd.Series([11.9, 36.0, 16.6, 21.8, 34.2, 62.4], index = ['China', 'India', 'USA', 'Brazil',

srs2 = pd.Series([20.3, 11.9, 36.0, 16.6, 21.8, 34.2], index = ['Africa', 'China', 'India', 'USA',

srs = srs1 / srs2
# Set Series name
srs.name = "Resultant Object"

# Set index name
srs.index.name = "Country"

# printing series values
```

```
print("The new Indexed Series values are:")
print(srs)
```

It can be seen from the output that in the new `Series` object the index names that are not common are assigned `NaN`, meaning `null` values in `pandas`. The divide operation is performed on the index values with common names, and as all values are the same, `1.0` is obtained as the result.

Just like the division operator in this example, any mathematical operation can be performed on `Series` objects.

## More functions from indexes #

Some more useful functions that make use of indexes to produce the relevant results are explained below in the example.

```
#importing pandas in our program
import pandas as pd
# Defining two series object
srs1 = pd.Series([11.9, 36.0, 16.6, 21.8, 34.2, 62.4], index = ['China', 'India', 'USA', 'Brazil',
srs2 = pd.Series([20.3, 11.9, 36.0, 16.6, 21.8, 34.2], index = ['Africa', 'China', 'India', 'USA',
# Dividing series 1 values with seies 2 values
srs = srs1 / srs2
# Set Series name
srs.name = "Resultant Object"
# Set index name
srs.index.name = "Country"


print("The indexes that are assigned ‘True’ are Null:")
print(pd.isnull(srs))

print("\nThe indexes that are assigned ‘True’ are not Null:")
print(pd.notnull(srs))

print("\nIndexes that satisfy the condition are:")
print(srs[srs == 1.0])

print("\nIndexes that satisfy the condition are:")
print(srs[srs != 1.0])
```

- The `isnull()` function returns a series object with indexes assigned to `True` whose values are `NaN`, and the rest of the indexes are assigned to `False`.
- The `notnull()` function returns a series object with indexes assigned to `False`

whose values are `NaN`, and the rest of the indexes are assigned to `True`.

- The condition `srs[srs == 1.0]` returns a series object containing indexes with values equal to `1.0`.
- The condition `srs[srs != 1.0]` returns a series object containing indexes with values not equal to `1.0`.

Similarly, more logical operators can be used in the same way to access certain conditions. For more information on `Series` object's functions refer here.

---

In the next lesson, the pandas `DataFrame` object is explained.