Pandas Series

- **Type Conversion - .astype()**

for e.g.

Code:

```python
import numpy as np
import pandas as pd

num_array = pd.Series(range(5))

num_array
```

Output:

```
0    0
1    1
2    2
3    3
4    4
dtype: int64
```

Code:

```python
num_array.astype("float")
```

Output:

```
0    0.0
1    1.0
2    2.0
3    3.0
4    4.0
dtype: float64
```

Code:

```
num_array.astype("object")
```

Output:

```
0    0
1    1
2    2
3    3
4    4
dtype: object
```

Code:

```
num_array.astype("string")
```

Output:

```
0    0
1    1
2    2
3    3
4    4
dtype: string
```

Code:

```
num_array = pd.Series(["a","b","c"])
num_array
```

Output:

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

Note: We cannot convert string to integer, if the value provided is not numeric

Code:

```
num_array.astype("int")
```

Output:

```
ValueError: invalid literal for int() with base 10: 'a'
```

- **Pandas index series and custom indices**

Code:

```
import numpy as np
import pandas as pd
my_series = pd.Series(range(5))

my_series
```

```
>>
0    0
1    1
2    2
3    3
4    4
dtype: int64
```

```
my_series[3]
```

```
>> 3
```

```
my_series[0]
```

```
>> 0
```

```
my_series[1:3]   # 1 to 2
```

```
>>

1    1
2    2
dtype: int64
```

```
my_series[1::2]   # step size of 2
```

```
>>

1    1
3    3
dtype: int64
```

```
my_series = pd.Series(range(5), index =["Day 0","Day 1","Day 2","Day 3","Day 4"])

my_series
```

```
>>

Day 0    0
Day 1    1
Day 2    2
Day 3    3
Day 4    4
dtype: int64
```

```
my_series["Day 1"]
```

```
>> 1
```

```
my_series["Day 3"]
```

```
>> 3
```

```
my_series["Day 1":"Day 3"]
```

```
>>

Day 1    1
Day 2    2
Day 3    3
dtype: int64
```

Note: Here stop point is inclusive, because we have index in a non-numerical format.

```
my_series[::2]   # Step size of two
```

```
>>

Day 0    0
Day 2    2
Day 4    4
dtype: int64
```

- **iloc method**

```
my_series.iloc[1]
```

```
>> 1
```

```
my_series.iloc[3]
```

```
>> 3
```

```
my_series.iloc[-1]
```

```
>> 4
```

```
my_series.iloc[[1,3,4]]  # multiple row selection
```

```
>>

Day 1    1
Day 3    3
Day 4    4
dtype: int64
```

```
my_series.iloc[1:4]
```

```
>>

Day 1    1
Day 2    2
Day 3    3
dtype: int64
```

```
my_series.iloc[1:]
```

```
>>

Day 1    1
Day 2    2
Day 3    3
Day 4    4
dtype: int64
```

- **loc Method**

```
import pandas as pd
import numpy as np
my_series = pd.Series(range(5), index =["Day 0","Day 1","Day 2","Day 3","Day 4"])

my_series
```

```
>>

Day 0    0
Day 1    1
Day 2    2
Day 3    3
Day 4    4
dtype: int64
```

```
my_series.loc["Day 3"]
```

```
>> 3
```

```
my_series.loc["Day 1":"Day 3"]
```

```
>>

Day 1    1
Day 2    2
Day 3    3
dtype: int64
```

```
my_series.index = [0,2,44,100,5]

my_series
```

```
>>

0      0
2      1
44     2
100    3
5      4
```

```
dtype: int64
```

```python
my_series.loc[5]
```

```
>> 4
```

```python
my_series.loc[[2,44,5]]     # multiple row
```

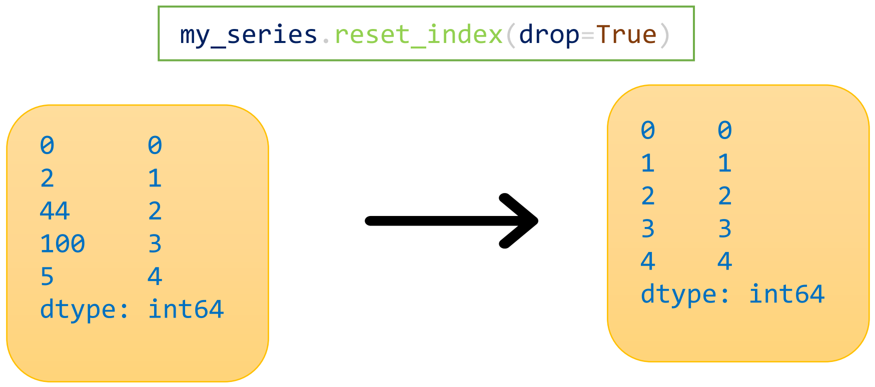
```
>>

2      1
44     2
5      4
dtype: int64
```

```python
my_series.loc[0:5]
```

```
>>

0       0
2       1
44      2
100     3
5       4
dtype: int64
```

```python
my_series.reset_index(drop=True)    # resets the index to default type
```

```
>>

0    0
1    1
2    2
3    3
4    4
dtype: int64
```



```python
my_series.reset_index(drop=True)
```

```
0       0
2       1
44      2
100     3
5       4
dtype: int64
```

```
0    0
1    1
2    2
3    3
4    4
dtype: int64
```

- **Duplicating Index values and resseting index**

```
import numpy as np
import pandas as pd
my_series = pd.Series(range(5), index = ["Day 0","Day 0","Day 0","Day 2","Day 2"])

my_series
```

```
>>

Day 0    0
Day 0    1
Day 0    2
Day 2    3
Day 2    4
dtype: int64
```

```
my_series["Day 0"]
```

```
>>

Day 0    0
Day 0    1
Day 0    2
dtype: int64
```

```
my_series["Day 2"]
```

```
>>

Day 2    3
Day 2    4
dtype: int64
```

```
my_series["Day 0":"Day 2"]
```

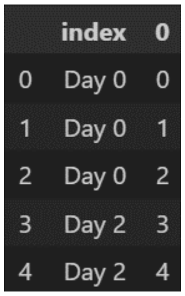
```
>>

Day 0    0
Day 0    1
Day 0    2
Day 2    3
Day 2    4
dtype: int64
```

```
my_series["Day 0"][1] # Further slicing the row
```

```
>> 1
```

```
my_series.reset_index()
```

```
>>
```



	index	0
0	Day 0	0
1	Day 0	1
2	Day 0	2
3	Day 2	3
4	Day 2	4

By default it will be converted into dataframe.

```
my_series.reset_index(drop=True)
>>

0    0
1    1
2    2
3    3
4    4
dtype: int64
```

```
my_series.reset_index(drop=True).iloc[1:4]
>>

1    1
2    2
3    3
dtype: int64
```

IMP

```
my_series.reset_index(drop=True).loc[1:4]
>>

1    1
2    2
3    3
4    4
dtype: int64
```

IMP

➢ **Filtering Series and logical test**

```
import numpy as np
import pandas as pd

my_series = pd.Series(
    [0,1,2,3,4], index= ["day 0","day 1","day 2","day 3","day 4"])

my_series
```

```
>>

day 0    0
day 1    1
day 2    2
day 3    3
day 4    4
dtype: int64
```

```
my_series == 2
```

```
>>

day 0    False
day 1    False
day 2     True
day 3    False
day 4    False
dtype: bool
```

```
my_series != 2
```

```
>>

day 0     True
day 1     True
day 2    False
day 3     True
day 4     True
dtype: bool
```

```
my_series.loc[my_series != 2]
```

```
>>

day 0    0
day 1    1
day 3    3
day 4    4
dtype: int64
```

```
my_series.loc[~(my_series != 2)]  # inverting
```
```
>>

day 2    2
dtype: int64
```
```
my_series.loc[my_series.isin([1,2])]
```
```
>>

day 1    1
day 2    2
dtype: int64
```
```
my_series[~(my_series.isin([1,2]))]  # inverting
```
```
>>

day 0    0
day 3    3
day 4    4
dtype: int64
```
```
my_series.loc[my_series.gt(2)] # greater than
```
```
>>

day 3    3
day 4    4
dtype: int64
```
```
my_series.loc[~my_series.gt(2)] # inverting
```
```
>>

day 0    0
day 1    1
day 2    2
dtype: int64
```
```
mask = (my_series.isin([1,2])) | (my_series.gt(2))

my_series.loc[mask]
```
```
>>

day 1    1
day 2    2
day 3    3
day 4    4
dtype: int64
```
```
mask = (my_series.isin([1,2])) & (my_series.gt(2))

my_series.loc[mask]
```
```
>>

Series([], dtype: int64)
```