

Objective: This worksheet focuses on simple operations using *Pandas* series objects.

Let us define a series data as shown in the table below (units are not shown):

KIA Seltos Car				
Mileage	Engine Displacement	Torque	ВНР	Boot Space
18	1493	250	113	433

This is created in a series object using Pandas:

```
import pandas as pd
kiaSeltos = pd.Series(data = [18, 1493, 250, 113, 433], index = ['mileage', 'engDisp', 'torque', 'bhp', 'boot'])
```

• The different indices' values can be viewed using *loc* and *iloc* as discussed in the previous worksheet:

```
kiaSeltos.loc['mileage']

18

kiaSeltos.iloc[0]

18
```

• Notice the subtle difference when the *loc* and *iloc* are used to access the values using range. When range is provided in case of *loc*, both the ends are <u>inclusive</u>. When range is provided in *iloc*, the last end is exclusive.

```
kiaSeltos.loc['mileage':'boot']
mileage
            18
           1493
engDisp
torque
            250
bhp
            113
boot
            433
dtype: int64
kiaSeltos.iloc[0:4]
mileage
            18
engDisp
           1493
            250
torque
bhp
            113
dtype: int64
```

All values for 'mileage' to 'boot' are printed.

All values for indices 0 to 3 are printed. 4^{th} value is excluded.

• To demonstrate other operations let us define another series object as per the table below:

Planet Diameters (km)							
Mercury	Earth	Saturn	Mars	Neptune	Jupiter	Venus	Uranus
4879	12742	116460	6779	49244	139820	12104	50724



Let us say if we are interested to see only those planets whose diameters are above 15000 km. How
can we do that? If we simply execute planetDia > 15000, what will happen? A new series is created
with Boolean values: False where the planet diameter is <= 15000 and True where the planet diameter
is > 15000.

```
planetDia > 15000
           False
Mercury
Earth
           False
Saturn
            True
Mars
           False
Neptue
            True
Jupiter
            True
Venus
           False
Uranus
            True
dtype: bool
```

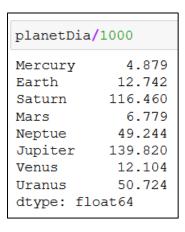
• This operation sounds useful, but how the list of planets can be obtained whose diameters are > 15000? The above conditional expression can be used as index for the series object to filter the data as shown below. This operation is used extensively in data science for conditional filtering.

```
planetDia [planetDia > 15000]

Saturn 116460
Neptue 49244
Jupiter 139820
Uranus 50724
dtype: int64
```

• More complex filtering can also be done. For example getting the planets whose diameters are more 15000 but less than 100000:

- Remember that the above two examples show conditional filtering. There will be no change in the content of original series planetDia.
- Scalar broadcasting operation will also work as it happens in NumPy. Let us say, the diameter are to be represented in the units of 1000 km. Now, all the data values are to be divided by 1000. This can be done simply by planetDia/1000 as shown below. Note that data type is also changed to float.



 Since Pandas is built over NumPy, NumPy mathematical operations can also be directly used over Pandas objects. The min, max and mean of the diameters can be calculated as shown below using NumPy methods:

```
import numpy as np
print (np.min(planetDia))
print (np.max(planetDia))
print (np.mean(planetDia))

4879
139820
49094.0
```

• Let us take another example where two bags are filled with different vegetables.

Weights in kg					
	Potato	Tomato	Cabbage	Radish	Turnip
Bag-1	2	1	3		3
Bag-2		2		4	

• Let us create two Pandas series objects from this data.

```
bag1 = pd.Series(data = [2, 1, 3, 3], index = ['potato', 'tomato', 'cabbage', 'turnip'])
bag2 = pd.Series(data = [2, 4], index = ['tomato', 'radish'])
```

- These series do not include which those respective bags do not carry. So can these two objects be added? If yes, what would be the result?
- Since only Tomato is the common vegetable, its weights are added and other values are reported as NaN (Not a Number). The reason is because addition cannot be performed with the non-existent data.

bag1+ba	ıg2
cabbage	e NaN
potato	NaN
radish	NaN
tomato	3.0
turnip	NaN
dtype:	float64

• isnull() is a function in Pandas that returns boolean values showing what all is null and what all is not.

<pre>newBag = bag1 + bag2 pd.isnull(newBag)</pre>			
cabbage	True		
potato	True		
radish	True		
tomato	False		
turnip	True		
dtype: bo	ool		

 Conditional filtering can be done using isnull() function and NOT operator which is the tilde (~) as shown below:

```
newBag[~pd.isnull(newBag)]
tomato 3.0
dtype: float64
```

- The above statement attempts to print only those values in the newBag series object which are not
- New values with indices can also be added and dropped as shown below. To drop the function *drop()* is used.

planetDia planetDia	['Moon'] = 3474
Mercury	4879
Earth	12742
Saturn	
Mars	6779
Neptue	49244
Jupiter	
Venus	12104
Uranus	50724
Moon	3474
dtype: in	
dcypc. In	
planetDia	.drop ('Moon')
Mercury	4879
Earth	12742
Saturn	116460
Mars	6779
Neptue	49244
Jupiter	139820
Venus	12104
Uranus	50724
dtype: in	t64

Exercise:

1. Find the mass (M) in kg for the planets shown in the above examples through online search and assuming these planets as sphere, calculate the densities of each planet. Volume (V) of a sphere of radius (r) can be calculated as: $V = \frac{4}{3}\pi r^3$. $Density = \frac{mass}{volume}$.

- 2. Prepare a dictionary of 10⁶ elements, repeatedly filled with the values 0 to 99. Now take an array of 100 random elements from 0 to 99. Prepare a Pandas series object also from this dictionary.
 - i. Which is faster to search all elements of the array? Dictionary or Series?
 - ii. Which is faster to sum all elements? Dictionary or Series? (Note: NumPy sum() function can be used with the series object but not on dictionary)