

# Pandas

## Table of Contents

**\*\*1. Working with Pandas Series -----** a) Creating Series Series through list Series through Numpy Array Setting up our own index Series through dictionary Using repeat function along with creating a Series Accessing data from Series ----- b) Aggregate function on Pandas Series ----- c) Series Absolute Function ----- d) Appending Series ----- e) Astype Function ----- f) Between Functions ----- g) All strings functions can be used to extract or modify texts in a series Upper and Lower Function Len function Strip Function Split Function Contains Function Replace Function Count Function Startswith and Endswith Function Find Function ----- h) Converting a Series to List

1. Detailed Coding Implementations on Pandas DataFrame -----a) Creating Data Frames -----b) Slicing in DataFrames Using Iloc and Loc Basic Loc Operations Basic Iloc Operations Slicing Using Conditions -----c) Column Addition in DataFrames Using List Using Pandas Series Using an Existing Column -----d) Deleting Column in DataFrame Using del Using pop function ----- e) Addition of rows ----- f) Drop function ----- g) Transposing a DataFrame ----- h) A set of more DataFrame Functionalities axes function ndim function dtypes function shape function head function tail function empty function ----- i) Statistical or Mathematical Functions Sum Mean Median Mode Variance Min Max Standard Deviation ----- j) Describe Function ----- k) Pipe Functions: Pipe function Apply Function Applymap Function ----- l) Reindex Function ----- m) Renaming Columns in Pandas DataFrame ----- n) Sorting in Pandas DataFrame ----- o) Groupby Functions Adding Statistical Computation on groupby Using Filter Function with Groupby
2. Working with csv files and basic data Analysis Using Pandas -----a) Reading CSV -----b) Info Function -----c) isnull() Function -----d) Quantile Function -----e) Copy Function -----f) Value Counts Function -----g) Unique and Nunique function -----h) dropna() function -----i) fillna() function -----j) sample Functions -----k) to\_csv() functions
3. A detailed Pandas Profile Report\*\*

```
import pandas as pd
import numpy as np
pd.__version__ #This will give us the current version of pandas
'2.1.1'
```

Series through a list

```
lst = [1,2,4,5,6]
arr = np.array(lst)
#series from numpy array
print(arr, type(arr))
srr = pd.Series(arr) #Converts any PYTHON LIST or NUMPY ARRAY to
PANDAS SERIES and the series will have the labels as 0,1,2 etc. by
default
print(srr, type(srr))
```

```

#Giving our own labels
srr = pd.Series(arr, index=['one','two','four','five','six',]) #We can
label the rows by ourselves
print(srr, type(srr))
#Series through dictionary values
steps = {'day1': 4000, 'day2': 3000, 'day3': 4500, 'day4': 5000,}
srr = pd.Series(steps)
print(srr, type(srr))

[1 2 4 5 6] <class 'numpy.ndarray'>
0      1
1      2
2      4
3      5
4      6
dtype: int32 <class 'pandas.core.series.Series'>
one      1
two      2
four     4
five     5
six      6
dtype: int32 <class 'pandas.core.series.Series'>
day1     4000
day2     3000
day3     4500
day4     5000
dtype: int64 <class 'pandas.core.series.Series'>

```

Using repeat along with creating a series

```

print(pd.Series(5)) #creates a series with an element 5 at label 0
print(pd.Series(5).repeat(5)) #Repeats the same element n number of
times with the same index
print(pd.Series(5).repeat(5).reset_index()) #Resets the index to
integers starting with 0,1,2 etc, but in a data frame
pd.Series(5).repeat(5).reset_index(drop=True) #This wil not give a
dataframe but a series

0      5
dtype: int64
0      5
0      5
0      5
0      5
0      5
0      5
dtype: int64
   index  0
0      0  5
1      0  5
2      0  5

```

```
3      0  5
4      0  5

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

### Accessing the elements

```
series = pd.Series([10,20]).repeat([5,2]).reset_index(drop=True) #
Repeats 10 five times and 20 two times
print(series)
print(series[2]) #We can access the elements in the same way as we
access data in arrays in any other programming language

0      10
1      10
2      10
3      10
4      10
5      20
6      20
dtype: int64
10

print(series) #This will print all the values
print('-'*20)
print(series[2:5]) #Returns element from given index to the other
given index
print('-'*20)
print(series[:]) #Again returns all the elements
print('-'*20)
print(series[:4]) #Returns elements from index 0 to index 4
print('-'*20)
print(series[5:]) #Returns elements from index 5 to end
print('-'*20)
print(series[2:-1]) #Returns elements starting from index 2 till the
elements in the second index from the last

0      10
1      10
2      10
3      10
4      10
5      20
6      20
dtype: int64
```

```

-----
2      10
3      10
4      10
dtype: int64
-----
0      10
1      10
2      10
3      10
4      10
5      20
6      20
dtype: int64
-----
0      10
1      10
2      10
3      10
dtype: int64
-----
5      20
6      20
dtype: int64
-----
2      10
3      10
4      10
5      20
dtype: int64

```

### Aggregate functions

```

sr = pd.Series([1,2,3,4,5,6,7,8,9])
print(sr)
sr.agg(['min', 'max', 'sum'])

```

```

0      1
1      2
2      3
3      4
4      5
5      6
6      7
7      8
8      9
dtype: int64

min      1
max      9

```

```
sum    45
dtype: int64
```

### Series absolute function

```
sr = pd.Series([1,-2,5,-5,9,-6])
print(sr)
sr.abs() #Makes negative numbers as positive. Returns absolute of a number.
```

```
0    1
1   -2
2    5
3   -5
4    9
5   -6
dtype: int64
```

```
0    1
1    2
2    5
3    5
4    9
5    6
dtype: int64
```

### Appending Series

```
sr1 = pd.Series([1,-2,5,-5,9,-6])
sr2 = pd.Series([55,44,88,99,525])
sr3 = pd.concat([sr1,sr2]) #To concat/merge two series together, but the indexes are also taken in the same order
sr3[0] #This will give two output
```

```
0    1
0   55
dtype: int64
```

```
pd.concat([sr1,sr2]).reset_index(drop=True) #By doing this, pandas is assigning default integer values in the index
```

```
0    1
1   -2
2    5
3   -5
4    9
5   -6
6   55
7   44
8   88
9   99
```

```
10    525
dtype: int64
```

### Astype Function

```
print(type(sr1))
print(sr1.astype('float')) #Changes the type of data in the series to float
print(type(sr1.astype('float'))))
str = sr1.astype('str') #Converts the type of data in the series to string
print(type(str[2]))
```

```
<class 'pandas.core.series.Series'>
0    1.0
1   -2.0
2    5.0
3   -5.0
4    9.0
5   -6.0
dtype: float64
<class 'pandas.core.series.Series'>
<class 'str'>
```

### Between Function

```
str1 = pd.Series([1,22,3,4,55,65,7,8,9,10])
str1.between(2,6) #Tells us if the particular element in the series is in between the given range

0    False
1    False
2     True
3     True
4    False
5    False
6    False
7    False
8    False
9    False
dtype: bool
```

All strings functions can be used to extract or modify texts in a series

Upper and Lower Function   Len function   Strip Function   Split Function   Contains Function  
Replace Function   Count Function   Startswith and Endswith Function   Find Function

## Upper and lower case

```
ser = pd.Series(["Luv Ratan" , "Data Science" , "Geeks for Geeks" ,  
'Hello World' , 'Machine Learning', 56])  
print(ser)
```

*#Converting all the string value series to upper case*

```
print(ser.str.upper())
```

*#Converting all the string value series to lower case*

```
print(ser.str.lower())
```

```
0      Luv Ratan  
1      Data Science  
2      Geeks for Geeks  
3      Hello World  
4      Machine Learning  
5              56  
dtype: object
```

```
0      LUV RATAN  
1      DATA SCIENCE  
2      GEEKS FOR GEEKS  
3      HELLO WORLD  
4      MACHINE LEARNING  
5              NaN  
dtype: object
```

```
0      luv ratan  
1      data science  
2      geeks for geeks  
3      hello world  
4      machine learning  
5              NaN  
dtype: object
```

## Length Function

```
ser = pd.Series(["Luv Ratan" , "Data Science" , "Geeks for  
Geeks" , 'Hello World' , 'Machine Learning'])
```

```
print(len(ser) ) #Returns the length of the series, that means the  
number of items on the series
```

```
print('-'*15)
```

*#Printing the length of each item which is a string in the series*

```
for i in ser:  
    print(len(i))
```

```
5  
-----  
15  
17  
15  
13  
16
```

## Strip Function

```
print(ser)
ser = ser.str.strip() #Removes all the extra spaces from left and right of the string but does not remove from between two strings
print(ser)
```

```
0      Luv Ratan
1      Data Science
2      Geeks for Geeks
3      Hello World
4      Machine Learning
dtype: object
0      Luv Ratan
1      Data Science
2      Geeks for Geeks
3      Hello World
4      Machine Learning
dtype: object
```

## Split functions

```
print(ser)

print(ser.str.split()) #Splits all the items from the space using a comma and returns a nested Series
ser.str.split()[0]
```

```
0      Luv Ratan
1      Data Science
2      Geeks for Geeks
3      Hello World
4      Machine Learning
dtype: object
0      [Luv, Ratan]
1      [Data, Science]
2      [Geeks, for, Geeks]
3      [Hello, World]
4      [Machine, Learning]
dtype: object
```

```
['Luv', 'Ratan']
```

```
dateser = pd.Series(['10/12/24', '23/02/24', '18/6,24'])
print(dateser.str.split('/')[0]) #By default, it splits from space but we can explicitly tell pandas from where we want to split the string
print(dateser.str.split('/')[1])
print(dateser.str.split('/')[2])
```



```
['10', '12', '24']  
['23', '02', '24']  
['18', '6,24']
```

### Contain Function

```
print(ser)  
ser.str.contains('for') #Will give True for the index where the given  
string is found
```

```
0      Luv Ratan  
1      Data Science  
2      Geeks for Geeks  
3      Hello World  
4      Machine Learning  
dtype: object
```

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

### Replace Function

```
ser.str.replace('r', '@') #Replaces the item with another given item  
wherever found
```

```
0      Luv Ratan  
1      Data Science  
2      Geeks fo@ Geeks  
3      Hello Wo@ld  
4      Machine Lea@ning  
dtype: object
```

### Count Function

```
ser.str.count('Luv') #Counts the number of times the item has appeared  
in all the strings of each element
```

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

Starts with and ends with

```
print(ser.str.startswith('Luv'))    #Returns True for the indexes of  
element where this particular word starts with  
print(ser.str.endswith('ld'))    #Returns True for the indexes of  
element where this particular word ends with
```

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

Find Function

```
ser = pd.Series(["Luv Ratan" , "Data Science" , "Geeks for Geeks" ,  
'Hello World' , 'Machine Learning', 56])  
ser.str.find('Luv')    #Returns 0 for the index of the element where it  
found that item
```

```
0    0.0  
1   -1.0  
2   -1.0  
3   -1.0  
4   -1.0  
5    NaN  
dtype: float64
```

Converting a Series into a list

```
print(type(ser))  
print(type(ser.to_list())) #Converts a Pandas Series to Python list  
  
<class 'pandas.core.series.Series'>  
<class 'list'>
```