**PANDAS**

**What is Pandas?**

**Pandas is a fast, powerful, flexible, and easy to use open-source data analysis and manipulation tool, built on top of the Python programming language.**

**about pandas: https://pandas.pydata.org/about/index.html**

**Basic data structure in pandas**

**Pandas provide two types of classes for handling data:**

**~ Series  
~ DataFrame**

**Importing Pandas**

**There are different way to import pandas library:**

**1.Most common way is import statement with alias:**

**~ import pandas as pd**

**2.Importing all the function and class:**

**~ from pandas import \***

**3.Importing the specific function and class library:**

**~ from pandas import DataFrame, read\_csv**

**Pandas Series**

**A Pandas Series is like a column in a table.**

**It is a one-dimenational labelled array capable of holding of any datatype.**

**Integers, string, floating point numbers, python object etc.**

**Each value in a pandas series is associated with the index.**

**The default index value of it is from 0 to number – 1, or you can specify your own index values.**

**Note:**

**‘Series’ is a class provided by the pandas library.**

**When you create a ‘Series’ in your code, means your creating an ‘Series’ object of that ‘Series’ class.**

**Syntax: pd.Series( data, index, name )**

**data: it can be array, list, dictionary, csv file or excel**

**file**

**index: it is default, also we can create custom index**

**name: series name**

**Creating a pandas series**

**Pandas series can be created from the lists, dictionary and from other scalar values etc.**

**Series can be created in different ways, here are some ways to create a series.**

**Create an empty series:**

**Code:**

**import pandas as pd**

**empty\_series = pd.Series()**

**print(empty\_series)**

**Output:**

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

**Create a series from lists:**

**We first creating a list after that we can create series of list.**

**# String**

**Code:**

**import pandas as pd**

**import numpy as np**

**countries = ['India','Nepal','Srilanka','Bhutan']**

**print(pd.Series(countries))**

**Output:**

**0 India**

**1 Nepal**

**2 Srilanka**

**3 Bhutan**

**dtype: object**

**# Custom index and name**

**Code:**

**marks = [ 58, 93, 89, 60 ]**

**subjects = [ 'C++' , 'Python', 'R', 'Java' ]**

**print(pd.Series(marks, index=subjects, name='student'))**

**Output:**

**C++ 58**

**Python 93**

**R 89**

**Java 60**

**Name: student, dtype: int64**

**Create a series from dictionary:**

**We must first create a dictionary then we can pefrom series on dictionary.**

**Code:**

**import pandas as pd**

**marks = { 'maths' : 78, 'english' : 70, 'science' : 89 }**

**pd.Series(marks, name='student score')**

**Output:**

**maths 78**

**english 70**

**science 89**

**Name: student score, dtype: int64**

**Pandas Series Attribute:**

**In Pandas, a Series object has several important attributes that is commonly used attributes of a Pandas Series include:**

**size attribute:**

**Returns the number of elements in a Series, including any elements that might contain missing or NaN (Not-a-Number) values.**

**Code:**

**import pandas as pd**

**data = [ 10, 20, 30, None, 50 ]**

**print(pd.Series(data).size)**

**data1 = [ 10, 20, 30, 50 ]**

**print(pd.Series(data1).size)**

**Output:**

**5**

**4**

**dtype attribute:**

**Returns the data type of the elements in the Series.**

**Used to check the data type of the data within the Series.**

**Code:**

**import pandas as pd**

**data = [ 10, 20, 30, 50 ]**

**print(pd.Series(data).dtype)**

**Output:**

**int64**

**name attribute:**

**Assign a name to a eries when creating it or later using the name attribute.**

**The name is typically used in the context of DataFrames, where a Series can represent a column.**

**Code:**

**import pandas as pd**

**data = [ 10, 20, 30, 50 ]**

**print(pd.Series(data**, **name='my\_data').name)**

**Output:**

**my\_data**

**is\_unique attribute:**

**Returns a boolean value indicating whether all the values in the Series are unique (no duplicates) or not.**

**Code:**

**import pandas as pd**

**data = [ 10, 20, 30, 40, 50 ]**

**print(pd.Series(data).is\_unique)**

**data1 = [ 10, 20, 30, 40, 50 ]**

**print(pd.Series(data1).is\_unique)**

**Output:**

**True**

**False**

**values attribute:**

**Returns the data in the Series as a NumPy array.**

**Code:**

**import pandas as pd**

**marks = { 'maths' : 78, 'english' : 70, 'science' : 89 }**

**print(pd.Series(marks).values)**

**Output:**

**[ 78 70 89 ]**

**Series using read\_csv() function:**

**~ squeeze=True:**

**Attribute specifies that the result should be squeezed into a series if it has only one column.**

**Here we intentionally use this attribute to avoid the dataframe to understand the about series.**

**Dataset link:**

**Code:**

**import pandas as pd**

**subs = pd.read\_csv('subs.csv', squeeze=True )**

**print(subs)**

**print(type(subs))**

**Output:**

**0 48**

**1 57**

**...**

**363 144**

**364 172**

**Name: Subscribers gained, Length: 365, dtype: int64**

**<class 'pandas.core.series.Series'>**

**~ index\_col:**

**used to specify which column in the CSV file should be used as the index for the resulting DataFrame/Series.**

**The index is a way to uniquely identify each row in the DataFrame/Series.**

**By default, if you don't specify the index\_col parameter, Pandas will create a default integer index starting from 0.**

**Dataset link:**

**Code:**

**import pandas as pd**

**vk = pd.read\_csv('kohli\_ipl.csv', index\_col= 'match\_no',**

**squeeze=True )**

**print(vk)**

**print(type(vk))**

**Output:**

**match\_no**

**1 1**

**2 23**

**..**

**214 25**

**215 7**

**Name: runs, Length: 215, dtype: int64**

**<class 'pandas.core.series.Series'>**

**Dataset link:**

**Code:**

**import pandas as pd**

**m = pd.read\_csv('bollywood.csv', index\_col= 'movie',**

**squeeze=True )**

**print(m)**

**print(type(m))**

**Output:**

**movie**

**Uri: The Surgical Strike Vicky Kaushal**

**Battalion 609 Vicky Ahuja**

**...**

**Company (film) Ajay Devgn**

**Awara Paagal Deewana Akshay Kumar**

**Name: lead, Length: 1500, dtype: object**

**<class 'pandas.core.series.Series'>**

**Note: above dataset used to perform series methods**

**Series Methods:**

**These methods are most used in pandas**

**head():**

**Used to display the first few rows (default is 5) of a DataFrame or Series.**

**If we provide negative number as parameter then return all**

**rows.**

**Code:**

**import pandas as pd**

**print(vk.head())  
 print(vk.head(2))**

**Output:**

**match\_no**

**1 1**

**2 23**

**3 13**

**4 12**

**5 1**

**Name: runs, dtype: int64**

**match\_no**

**1 1**

**2 23**

**Name: runs, dtype: int64**

**tail():**

**Used to display the last few rows (default is 5) of a DataFrame or Series.**

**Code:**

**import pandas as pd**

**print(vk.tail())**

**print(vk.tail(2))**

**Output:**

**match\_no**

**211 0**

**212 20**

**213 73**

**214 25**

**215 7**

**Name: runs, dtype: int64**

**match\_no**

**214 25**

**215 7**

**Name: runs, dtype: int64**

**sample():**

**To randomly select a specified number of rows (default is 1) or elements from a dataframe or series.**

**Useful to obtain a random sample from your data for data exploration, analysis, or testing.**

**Ex. if you have biased in your datasets then we can use this method.**

**Code:**

**import pandas as pd**

**print(vk.sample()) # default 1**

**print(vk.sample(3)) # three random row from dataset**

**Output:**

**match\_no**

**118 33**

**Name: runs, dtype: int64**

**match\_no**

**202 5**

**150 8**

**110 82**

**Name: runs, dtype: int64**

**value\_counts():**

**To count the frequency of values that occur multiple times in a Series.**

**Code:**

**import pandas as pd**

**print(m.value\_counts())**

**Output:**

**Akshay Kumar 48**

**Amitabh Bachchan 45**

**..**

**Akanksha Puri 1**

**Edwin Fernandes 1**

**Name: lead, Length: 566, dtype: int64**

**sort\_values():**

**Used to sort the values within a pandas series.**

**By default, it sorts the values in ascending order, but you can specify the sorting order using the ascending parameter.**

**syntax: series.sort\_values( ascending=True, inplace=False )**

**inplace: when true then sort and replace sorted data**

**with original data**

**If false (default), it returns a new series with**

**the sorted values while leaving the original**

**series unchanged**

**ascending: true(ascending) or false(descending)**

**Code:**

**import pandas as pd**

**print(vk.sort\_values()) # default is ascending=True**

**Output:**

**match\_no**

**135 0**

**8 0**

**..**

**126 109**

**128 113**

**Name: runs, Length: 215, dtype: int64**

**# with inplace=True for permanent changes in series**

**Code:**

**print(vk.sort\_values(inplace=True)**

**print(vk)**

**Output:**

**match\_no**

**128 113**

**126 109**

**...**

**8 0**

**135 0**

**Name: runs, Length: 215, dtype: int64**

**Method chaining:**

**It is practice of applying multiple operations or methods to a Series in a single line of code.**

**This approach is both efficient and readable, making it easier to perform complex data manipulations and transformations.**

**Code:**

**import pandas as pd**

**print(vk.sort\_values(ascending=False).head(1).values[0])**

**Output:**

**113**

**sort\_index():**

**It similar in concept to the sort\_values() method, but instead of sorting the values within the Series, it sorts the index (row labels) of the Series.**

**Both methods allow you to control the sorting order, either ascending or descending, and both can be used with the inplace parameter to modify the original Series.**

**syntax: series.sort\_index( ascending=True, inplace=False )**

**Code:**

**import pandas as pd**

**print(vk.sort\_index()) # default is ascending=True**

**Output:**

**match\_no**

**1 1**

**2 23**

**...**

**214 25**

**215 7**

**Name: runs, Length: 215, dtype: int64**

**# descending with inplace=True for permanent changes**

**Code:**

**print(vk.sort\_index(ascending=False, inplace=True)**

**print(vk)**

**Output:**

**match\_no**

**215 7**

**214 25**

**..**

**2 23**

**1 1**

**Name: runs, Length: 215, dtype: int64**

**Series Mathematical Methods:**

**Common statistical methods in Pandas Series for analyzing data:**

**count():**

**Count the non-null elements in the Series.**

**Code:**

**import pandas as pd**

**Output:**

**sum():**

**Used to calculate the sum of all the elements in a Series**

**Code:**

**import pandas as pd**

**Output:**

**product() :**

**The product() method in Pandas Series is used to calculate the product of all elements in the Series.**

**It multiplies all the values together and returns the result.**

**Code:**

**import pandas as pd**

**Output:**

**Common statistical methods in Pandas Series for analyzing data:**

**mean():**

**This method calculates the mean (average) of the elements in a Series.**

**Code:**

**import pandas as pd**

**Output:**

**median():**

**The median() method calculates the median of the elements in a Series, which is the middle value when the data is sorted.**

**It's a measure of central tendency.**

**Code:**

**import pandas as pd**

**Output:**

**mode():**

**The mode() method returns the mode(s) of the elements in a Series, which is the most frequently occurring value(s).**

**Code:**

**import pandas as pd**

**Output:**

**std():**

**The std() method computes the standard deviation of the elements in a Series, which measures the spread or dispersion of the data.**

**Code:**

**import pandas as pd**

**Output:**

**var():**

**The var() method calculates the variance of the elements in a Series, which is the average of the squared differences from the mean.**

**Code:**

**import pandas as pd**

**Output:**

**min():**

**The min() method returns the minimum value in a Series or DataFrame.**

**Code:**

**import pandas as pd**

**Output:**

**max():**

**The max() method returns the maximum value in a Series or DataFrame.**

**Code:**

**import pandas as pd**

**Output:**

**describe():**

**It is a convenient function to generate descriptive statistics of a numeric Series.**

**It provides a summary of various statistical measures, giving you insights into the data's distribution and central tendency.**

**It’s provides the following statistics:**

**count: number of non-null elements in the Series.**

**mean: mean (average) of the Series.**

**std: standard deviation, which measures the spread of**

**the data.**

**min: The minimum value in the Series.**

**25%: The 25th percentile (lower quartile).**

**50%: The median (50th percentile).**

**75%: The 75th percentile (upper quartile).**

**max: The maximum value in the Series.**

**Note: the describe() works on numeric data**

**Code:**

**import pandas as pd**

**Output:**

**Some Important Series Methods:**

**These are some common methods and functions available for working with Pandas Series in Python:**

**astype():**

**This method is used to cast the data type of the elements in a Series to the specified data type (e.g., int, float, str).**

**Useful to reduce the memory space**

**Syntax: series.astype(dtype)**

**Code:**

**import sys**

**print('Original size of dataset:',sys.getsizeof(vk))**

**vk\_size =vk.astype('int32')**

**print('Reduce size of dataset:',sys.getsizeof(vk\_size))**

**Output:**

**Original size of dataset: 3456**

**Reduce size of dataset: 2596**

**between():**

**Checks if each element in the Series falls within the specified range.**

**Returns a boolean Series.**

**Syntax: series.between(left, right, inclusive=True)**

**Code:**

**import pandas as pd**

**print(vk.between(50,100)) # return boolean value print(vk[vk.between(50, 100)]) # values**

**Output:**

**clip():**

**Clips values in the Series to be within the specified lower and upper bounds.**

**Syntax: series.clip(lower, upper)**

**Code:**

**import pandas as pd**

**Output:**

**drop\_duplicates():**

**Removes duplicate values from the Series.**

**Syntax: Series.drop\_duplicates(keep='first', inplace=False)**

**Code:**

**import pandas as pd**

**Output:**

**Output:**

**duplicated():**

**the duplicated() method is used to identify and mark duplicate values in a Series (column) of a DataFrame.**

**It returns a Boolean Series**

**Syntax: Series.duplicated()**

**Code:**

**import pandas as pd**

**temp = pd.Series([1,1,3,3,3,5,5])**

**print(temp.duplicated()) # True means duplicate**

**print('Duplicate value count:',temp.duplicated().sum())**

**Output:**

**isnull():**

**Returns a boolean Series indicating whether each element is NaN (missing data).**

**Syntax: series.isnull()**

**Code:**

**import pandas as pd**

**import numpy as np**

**temp = pd.Series([1,3,np.nan,np.nan,5,np.nan,7,np.nan])**

**print(temp)**

**# isnull**

**print(temp.isnull()) # return boolean**

**print('Missing values:',temp.isnull().sum())**

**dropna():**

**Removes missing (NaN) values from the Series.**

**Syntax: series.dropna(axis=0, inplace=False)**

**Code:**

**import pandas as pd**

**import numpy as np**

**fillna():**

**Fills missing (NaN) values in the Series with the specified value.**

**Syntax: series.fillna(value)**

**Code:**

**import pandas as pd**

**import numpy as np**

**temp = pd.Series([1,3,np.nan,np.nan,5,np.nan,7,np.nan])**

**print(temp)**

**print(temp.fillna(0))**

**isin():**

**Checks if each element in the Series is in the provided list of values.**

**Returns a boolean Series.**

**Syntax: series.isin(values)**

**Code:**

**import pandas as pd**

**print(vk.isin([49,99])) # return Boolean**

**print(vk[vk.isin([49,99])])**

**Output:**

**apply():**

**Applies a given function to each element in the Series and returns a new Series with the results.**

**Syntax: series.apply(func)**

**Code:**

**import pandas as pd**

**print(m.apply(lambda x:x.split()[0].upper()).head(5))**

**Output:**

**movie**

**Uri: The Surgical Strike VICKY**

**Battalion 609 VICKY**

**The Accidental Prime Minister (film) ANUPAM**

**Why Cheat India EMRAAN**

**Evening Shadows MONA**

**Name: lead, dtype: object**

**EXTRA Series method that is used in Exercise:**

**to\_numeric():**

**Used to convert the values in a Series (or DataFrame column) to numeric data types.**

**Useful when you have a Series containing strings or other non-numeric data, and you want to convert them to numeric types like integers or floating-point numbers.**

**Syntax: pd.to\_numeric(series, errors='coerce',**

**downcast ='integer')**

**Here are some common options for the errors parameter:**

**raise(default): Raises an error if any value cannot be**

**converted to a number.**

**coerce: Replaces non-convertible values with NaN.**

**ignore: Ignores non-convertible values and leaves**

**them as they are.**

**quantile():**