Pandas Functions:

STEP 1: Import Pandas in the python file

import pandas as pd

STEP 2: See the methods according to the need

1. **pd.read_csv(***file_path***)**:
   i. Desc: Reads the data from the CSV File
   ii. Parameters:
      - *file_path*: Relative Path to the CSV File.
   iii. Ex:
      - dataset = pd.read_csv('../dataset/ds.csv')
   iv. Returns: *DataFrame*
2. **dataset.columns:**
   i. Desc: Get the columns names
   ii. Ex:
      - cols = dataset.columns
   iii. Returns: *Object of names*
3. **isnull()** :
   i. Desc: Checks whether the data in the attribute is null or not
   ii. Ex:
      - dataset.isnull()
   iii. Returns: *Dataframe with boolean values*
4. **isnull().sum()**:
   i. Desc: Sum of all the null values attribute wise
   ii. Ex:
      - dataset.isnull().sum()
   iii. Returns: *dtype object*
5. **isnull().sum().sum()**:
   i. Desc: Total null values in the whole DataFrame
   ii. Ex:
      - dataset.isnull().sum().sum()
   iii. Returns: *Integer Value*

6. **Filling the null values**:
    i. **Fill the whole table**:
        - Method: dataset.fillna(*value, inplace = False*)
        - Fills all the null values
        - Parameters:
            - *value* : value use to fill the null values.
            - *inplace*: boolean [replaces in the same dataframe and doesnot create another object]
    ii. **Fill particular column**:
        - Method: dataset[*column_name*].fillna(*value, inplace = False*)
        - Fills the null values in the respective column.
        - Parameter:
            - *column_name*: Name of the column
            - *value*: replace the null data with.
            - *inplace:* boolean [replaces in the same dataframe and doesnot create another object]
        - Ex:
            - Fill the null values with **max** value of the column with name ‘col1’: dataset[‘col1’].fillna(dataset[‘col1’].max())
            - Fill the null values with **mode** value of the column with name ‘col1’: dataset[‘col1’].fillna(dataset[‘col1’].mode())

7. **to_datetime()**:
    i. Desc: Converting the date fields in dataset to DateTime object:
    ii. Ex:
        - pd.to_datetime(data[*column_name*])
    iii. Parameters:
        - *column_name*: name of the column containing dates
8. **count()**:
    i. Desc: Calculating the length of data in columns:
    ii. Ex:
        - data[*column_name*].count()
    iii. Parameters:
        - *column_name*: name of the columns

9. **DataFrame.shape:**
    i. Returns: *A tuple representing the dimensions of dataframe.*
    ii. Ex:
        - if the dataframe contains 2 rows and 2 columns
        - df = pd.read_csv(CSVFILEPATH)
        - rc = df.shape
10. **DataFrame.size**:
    i. Returns: *Integer value representing the total elements of the object.*
    ii. Ex:
        - df = pd.read_csv(CSVFILEPATH)
        - df.size
11. **DataFrame.drop(***labels=None, axis=0, inplace=False***)**:
    i. Returns: *Pandas DataFrame with columns dropped.*
    ii. Parameters:
        - *labels: Name of particular columns to be dropped or list of columns names.*
        - *axis: 0 or 1*
            - *axis: 0 -> removes the rows*
            - *axis: 1 -> remove the columns*
        - *inplace: Boolean value*
    iii. Ex:
        - *df.drop(['col1','col2'], axis = 1)*
12. **DataFrame.head(***n***):**
    i. Returns: *Obj_head*
    ii. Parameters:
        - *n: Number of rows to be displayed. Default 5*
    iii. Ex:
        - df.head(n=10)

Numpy Functions:

STEP 1: Import NumPy in the python file

import numpy as np

STEP 2: See the methods according to the need

1. **np.array(***object***):**
   i. Desc: Creates an Array.
   ii. Parameter:
      - *object*: *array_like*
        - *Any object exposing the array interface(LIST)*
   iii. Ex:
      - np.array([1,2,3])
   iv. Returns: *Array object.*
2. **np.mean(***object, axis***):**
   i. Desc: Computes the mean of the object array.
   ii. Parameter:
      - *object: array_like*
        - *Any object exposing the array interface (LIST).*
      - *axis: 0 or 1 [OPTIONAL]*
        - *0: Column-Wise*
        - *1: Row-Wise*
   iii. Returns:
      - Mean: Array type object
   iv. Ex:
      - np.mean([[1,2],[3,4]], *axis*=0)
        - array([2. , 3.])
      - np.mean([[1,2],[3,4]], *axis*=1)
        - array([1.5 , 3.5])
3. **np.median(***object, axis***):**
   i. Desc: Computes the median of the object array.
   ii. Parameter:
      - *object: array_like*
        - *Any object exposing the array interface (LIST).*

- *axis: 0 or 1 [OPTIONAL]*
  - *0: Column-Wise*
  - *1: Row-Wise*

iii. Returns:

- Median: Array type object

iv. Ex:

- np.median([[10,7,4],[3,2,1]], *axis*=0)
  - array([6.5 , 4.5, 2.5])
- np.median([[10,7,4],[3,2,1]], *axis*=1)
  - array([7. , 2.])

4. **np.std(***object, axis***):**

i. Desc: Computes the standard deviation.

ii. Parameter:

- *object: array_like*
  - *Any object exposing the array interface (LIST).*
- *axis: 0 or 1 [OPTIONAL]*
  - *0: Column-Wise*
  - *1: Row-Wise*

iii. Returns:

- standard_deviation: Array type object

iv. Ex:

- np.std([[1,2],[3,4]], *axis*=0)
  - array([1., 1.])
- np.std([[1,2],[3,4]], *axis*=1)
  - array([0.5, 0.5])

5. **np.sum(***object, axis***):**

i. Desc: Computes the sum of array elements over a given axis.

ii. Parameter:

- *object: array_like*
  - *Any object exposing the array interface (LIST).*
- *axis: None, int or tuple of ints. [OPTIONAL]*
  - *If tuple of ints is used the sum is calculated over all the columns.*

iii. Returns:

- sum_along_axis: Array type object if axis given else int

iv. Ex:

- np.sum([]): 0

- np.sum([0.5,1.5]): 2.0
- np.std([[1,2],[3,4]], *axis*=1)
  - array([3, 7])

6. **np.min(***object***):**
   i. Desc: Computes the min from the list.
   ii. Parameter:
      - *object: array_like*
        - *Any object exposing the array interface (LIST).*
   iii. Returns:
      - int
   iv. Ex:
      - np.min([1,2,3]): 1
7. **np.min(***object***):**
   i. Desc: Computes the min from the list.
   ii. Parameter:
      - *object: array_like*
        - *Any object exposing the array interface (LIST).*
   iii. Returns:
      - int
   iv. Ex:
      - np.min([1,2,3]): 1