Date 30 | 01 | 22

Pandas in Python. (self study).

① → Software library written for python for data manipulation & analysis.

i) → pd1 = pd.Dataframe (dict1)
  ↳ An already defined struc.
  ↳ sort of like an excel sheet is made.
  ↳ import Pandas as pd.

ii) → pd1.to_csv('friends.csv') } fn to convert data ~~to excel sheet.~~ (transfer)
  ↳ Name of the excel sheet that will be made

iii) → pd1. head (2) → shows ~~first 2 rows of table~~ ~~[useful when data has thousands of rows]~~.
- .tail() also there
- .describe() → shows stats analysis of numerical columns.

(iv) ~~to read & edit~~ excel sheet in pandas → read1 = pd.read_csv('read2.csv')
  ↳ to read.
  ~~read1 ['ColumnName'] [0] = 50~~
  ↳ ~~to edit~~

---

② Pandas is Excel with power & functions provided by python. Especially for data analysis. Also open source. Generally used with numpy.

③ 2. data structures / → Series (1D) (just one column (or) row).
  → Dataframes (2D) (like in excel).

→ Created a dataframe for showing operations:—

```
newdf1 = pd.Dataframe (np.random.rand(334,5),
                        index = np.arange(334))
```
↳ optional.

Page No.
Date

shape(), describe(), info()   (optional) extra [convert to numpy array → newdf1.to_numpy()] → for other operations in numpy.

(4) Super imp function foundation
newdf1.columns/rows = list("ABCD").
newdf1.index   newdf1.describe()   newdf1.drop(0, axis) } commonly shown
newdf1.dtypes   type(newdf1)
newdf1.sort_index(axis = x, ascending = false/true).
(or) sort_values
↳ ~~doesn't modify original data.~~

(5) ~~warning → A value is trying - - - - - slice from Dframe.~~

→ shown bcz :-

of copy & view functions.

newdf2 = newdf1.
~~newdf2[0][0] = 93.~~
~~then newdf1 also changed.~~ } newdf2 as view executed not copy.
~~bcz unless you specify newdf2 = newdf1.copy()~~
~~newdf2 will act as a pointer & changes in it will~~
~~affect newdf1 too.~~

∴ Bcz of this confusion, we use newdf1.loc[0,0] = 50
↳ loc func. to change values.
lists for all ':' can be used.

Eg:- 1.loc[(newdf1['0'] <0.3)]
↳ Returns copy of all rows satisfying given condition. (AND can also be used) in column O.

(6)
→ ~~iloc also there. Like loc selects rows & columns~~
wrt to their name.
iloc → uses only index (0,1, - - -) not names given by you.

(7) Note:-
~~newdf1 = newdf1.drop[0] → changes the database~~
~~newdf1.drop[0] → shows how change will be, no change~~ made unless, '=' used
(OR) inplace can be used.

03/02/22

Pandas Masterclass..

(1) Dataframe → A table having row & column.. (2D) heterogeneous.

(2) → merge, concatenate (or) reshape data. (slice and date)
Good at missing data handling.
Has powerful time series tool to work.
} Adv of Pandas.

(3) → Dataframe → data, → index, → column. } 3 components of dataframe.

Most convenction way to create dataframe is by ndarray.

(4) Diff formats from where data can be imported from:-
→ .csv , .tsv , .pkl.
.csv ↓ comma seperated
.tsv ↓ tab seperated.

(5) → data = np.array(· - - -) → can be dict, list also.
s = pd. Series(data)
↳ other parameters also present.
Simi syntax for data frame..

when extracting info. at col
[ ] → info abt single column.
[[ ]] → info abt multiple column.

(6) How errors happen

→ when multiple paragraph error given error is only one place but so many functions & classes are called that all error are shown.
You can use traceback to see where first point is.

df.value_counts() normalise = True ↳ for percentage.

bfill → below fill.

df.fillna('4')
↳ fills all null value by 4.

★ df[['Name', 'Age']] ,
df['Name'] .

(7) df[df['Age'> 20]] → returns row.
df['Age'> 20] → returns true/false.

(8) df.loc[0] → returns first row.

For editing.

df.loc[0, ['Name']] → returns name from first row.
df.loc[0:2 ['Name', 'Age']]
↳ start to end (slicing)

iloc → only uses index
↳ if slicing used, goes to start to end-1.
↳ Generally iloc & loc only used for slicing..

(9) → iloc → only for index specified in parameter.
Not columns, for manipulating (or) taking (or) editing columns use loc .

(10) Inplace = True & drop → imp concept.
↳ for making changes permanent.

(11) → While reading file into pandas, path is to be specified if .csv not in same folder. [★★ r to be mentioned before "].

(12) → df.iloc[0:5, -2:] → try.

★★ df.drop('columnname', axis=1, inplace = True) .
↳ if dropping column give axis=1 (or) else won't execute.

Other func → ⭐ nunique() → no parameters

→ isin()

df.isnull().sum()
↳ imp.

Many ways ← notnull() or isnull() == false

rename()
& inplace()
both similar, see once

(13) Groupby concept (imp) } little adv

zip ⟶ store in something (like dict (or) list to access).
merge() } adv
eg:- dict1{zip(...)}

df.columnname

⭐ unique() (or) nunique()

Group by

→ df.groupby('columnname').size()

df.groupby('Sex')['Survived'].mean().

	S.No.	City	Dept	Manager
0	1	Mumbai	Sales	Mr. M
1	2	Mumbai	Sales	"
2	3	Goa	Sales	"
3	4	Goa	Tech	"

groupby('City') → 2 block Goa & Mum

groupby('Manager') → 1 block whole t

↳ used instead of for loop. → Never used

(14). Joining 2 different datasets

i) → Inner Join, Outer join, etc.

ii) → Combining df1 & df2 ⟶ $\boxed{\text{merge()}}$
↳ can only be used for common columns.

→ pd.merge(df1, df2, on='city').

iii) → on='', how='' & other parameters can also be passed on.

(15) → pd.Concat() → Merge dataframe vertically & horizontally entirely.
(without particular condition like inner (or) outer join).