PANDAS

↳ library used for Data manipulation and analysis

↓
cleaning
Handling missing data
Merging or joining data

- **Data Manipulation**

→ Cleaning
→ Handling missing data
→ Merging, joining data

- Data Analysis

→ Read Data
→ Mean, Median, variance

- Handling Different Data Format:

→ CSV → JSON
→ Excel → SQL databases

PANDAS DATASTRUCTURE

(I) Series (1D labelled Array)

- Can hold any datatype → integers, float, strings, Python objects

(II) DataFrame (Table)

- Can store different type datas e.g ⇒ integer, float, strings.
- Tabular Spread Sheet like str. containg rows, each of which have one or more column

Series e.g

↳ data = [10, 20, 30, 40]

```
series = pd.Series(data,
                   index=['a', 'b', 'c',
                          'd'])
```

→

a	10
b	20
c	30
d	40

} stored vertically

~~pd.series(data, index=False)~~

DataFra

data =

'N

'A

'S

pd.D

o/p ⇒

0
1
2

Q How

Dato

→ By u

⇒ 1D Dat

DataFrame e.g

```
data = {
    'Name' : ['Alice', 'Bob'
              'Charlie'],
    'Age' : [ , , ]
    'Salary' : [ ]
    }
pd.DataFrame(data)
```

o/p ⇒

	Name	Age	Salary
0	Alice	25	50,000
1	Bob	30	60,000
2	Charlie	35	70,000

Q How you create 2D data for DataFrame?

→ By using list in Dictionary.

⇒ 1D Data ⇒ list

Data Series	DataFrame
(i) 1D	(i) 2D
(ii) Single column or row of data	(ii) Multiple rows or columns
(iii) Access element directly by their index	(iii) Access element using .iloc[] or .loc[]

Q Difference in .loc & .iloc[] used in Dataframe of Pandas.

↳ can also be used in .loc[] in Series

→

Index	Name	Age
100	Alice	25
101	Bob	30
102	charlie	35

→ df.iloc[1, 0] ⇒ Bob.

row, column

→ df.loc['101', 'Name'] => Bob

Q Okay that's all right Vinayak what's the actual difference?

→ is How they select data (access data)

iloc. => select data based on native integer location

loc => select data according to index of rows & labels of column

PANDAS (Code with Harry)

PANDAS FXN

① → To Excel (save)

pd df.to_csv('ABC.csv')

② No index (save to excel)

df.in
df.to_csv('Abc.csv', index = False)

(iii) df.head(2) ⇒ Two values starting

(iv) df.tail(5) ⇒ Last 5 rows.

(v) df.describe() ↳ i) Work on Numerical column

ii) Calculate

↳ (i) Count
(ii) Mean
(iii) Std
(iv) Min
(v) Max
(vi) 25%
(vii) 50%
(viii) Max
(ix) 75%

Q = What happen if we use describe() fxn on not number data or semi number data?

** ↳ when applied on Table which have numeric column + Non-Numeric + Semi-Numeric → only do

→ Can be applied on Non-Numeric data

↳ But we usually don't do

↳ So, say it is only used on Numeric Data

• Non-Numeri

E.g. ⇒

Name

Hello
Yellow
Mellow

• Semi-Num

↳ count
↳ freq
↳ top
↳ uniq

for Numeric

Mixed
100
200
abc
200

- Non-Numeric Data → i) count
  ii) Unique
  iii) top (most occuring)
  iv) freq ⇒ freq. of most occuring

E.g ⇒

Name	City
Hello	Path
Yellow	Amsr
Mellow	Cith

} object ↓ DataType

- Semi-Numeric
  - ↳ count
  - ↳ freq
  - ↳ top
  - ↳ unique

for Numeric

Mixed
100
200
abc
200

} count unique top freq

- Mixed DataFrame

Name	Age	City
Alice	10	Pune
Bob	20	Chdl
Cily	40	Amsr

⇒ dfp ⇒

	Age
count	3
mean	30
Std	
min	

(VI) pd.read_csv('Abc.csv') ⇒ To Read CSV.

(VII) ~~Hardy K~~

(VII) pd.Series(np.random.randn(34)) → 34 ele in series.

(VIII) pd.DataFrame(np.random.randn(334,5)) → Create 334 rows of with 5 column

(IX) df

(X) df

(XI) df

(XII) A

(XIII) ★ c

(XIV) df

(XV) ★

df.rows ⇒ X

(IX) df.index ⇒ rows name index.
↳ 0, 1, 2, 3

(X) df.columns ⇒ Tells column name
↳ col1, col2, ...

(XI) df.to_numpy() ⇒ DataFrame converted to Numpy.

(XII) Axis = 0 ⇒ Rows
= 1 ⇒ Columns

★ (XIII) df.drop(0, axis=1)
↳ Remove column
↳ will print by dropping ↳ But don't change in original

(XIV) df.loc[[1,2], ['C','D']]
That
rows columns

★ (XV) df.loc[(condition)]

E.g ⇒ df.loc[(new

df.loc[(df['A']<0.3)]
↳ Return all rows with column satisfying this

(XVI) df.loc[(cond1) & (cond2)]

e.g. => df.loc[(df['A']<0.3) & (df['C']>0.1)]

↳ Return all rows with col. satisfying that condit.

(XVII) df = df.drop(['A', 'D'], axis=1)

OR

df.drop(['A','D'], axis=1, inplace=True)

↳ will change in original Table

(XVIII)

(XVIII) pd.read_json('ABC.json')

(XIX) df["salary"].tolist()

↳ convert to list

PA

Q = Create Array

data

series

---> Numpy

Q

0
1
2
3
4
5
6
7

→ df.

→ df.ilo

PANDAS (GFG)

Q Create Pandas Series from Numpy Array Data?

```
data = np.array(['g', 'e', 'k',
                 's'])
series = pd.Series(data)
```

---> Numpy => converts list to array.

Q

	Col 1	Col 2
0	52	53
1	21	54
2	99	97
3	88	99
4	79	100
5	62	101
6	44	107
7	49	108

→ df.loc[3:6] => will print row 3, 4, 5, 6.

→ df.iloc[3:6] => will print 3, 4, 5.

Q You have two series d1 & d2. How to add subtract.

→ d1.add(d2)
→ d2.sub(d1)

d1 + d2 →

a	5	a	1	a	6
b	2	b	6	b	8
c	3	d	4	c	3
d	7	e	9	d	11
				e	9

Q Suppose you want to change the data type of one column. How to do that?

→ astype()

e.g ⇒ data["salary"] = data["salary"].astype(int)

- unique()
- value_coun the

★ series array

→ dict

→ pd.Ser index

O/P ⇒ Gee Boo Croo

★ DataF from

→ But

- unique() → See unique value in a particular column

- value_counts() → Method to count the number of times each

★ Series can be created from Numpy array, list or Dictionary

→ dict = { 'Geeks' : 10
'BOOKS' : 20
'CROOKS' : 30 }

→ pd.series(dict)

O/P ⇒

index	
Geeks	10
BOOKS	20
CROOKS	30.

★ DataFrame can also be created from List, Dictionary, List of List ✓ Dictionary

→ But prefer list in Dict:
{ 'Name' : [, ,] ,

① List of List : DataFrame

```
lst = [ [1, 'Geeks'], [2, 'for']
      , [3, 'Geeks'] ]

pd.DataFrame(lst, col=['Id', 'Data'
```

Id	Data
1	Geeks
2	for
3	Geeks

② List of Dictionary : Dataframe

```
lst = [ {1: 'Geeks', 2: 'For'
        3: 'Geeks'}
      , {1: 20, 2: 30, 3: 40}
      ]
```

1	2	3
Geeks	for	Geeks
20	30	40

Q How to
?

df = df.renar

Q Assign

df. df=

Q Descr

Q set

date

df = p

Q How to rename column in a Dataframe ?

df = df.rename(columns = {'col1': 'Sad', 'col2': 'Piggy'})

Q Assign custom index to Dataframe

df. df = pd.Dataframe(d1, index = [, ,])

Q Describe() → only numerical column are selected.

Q set Date as index

dates = pd.date_range('20250118', period = 6)

df = pd.Dataframe(data1, index = dates, columns = [, , ,])

classmate
Date
Page

★★★

Q = Access Method using Different Strategies in Dataframe

→ what allowed / what not

df.
df [:] ['A'] ✓
df [: , 'A'] ✗
df [:] or df [1:4] ✓ ✓
df [:] [['A', 'B'] ✓
df [:] ['A', 'B'] ✗

Syntax for Directly accessing

df [ ] [ ]
↓ col ↓ rows

★

→ using loc & iloc

df.l
df.lo
★ df.loc
★ df.loc
df.l

Q = Conc

df1
0
1
2

→ df3

df.loc[:, ['A', 'B']]

df.loc ⇒ Print memory location

df.loc[:] == df.loc[:,] == df.loc[:, :]

df.loc[:, []] ⇒ Show all index

df.loc[,] ⇒ ✗

Concatenate Two dataframe

df2

	0	1
x	a	b
y	b	c
z	c	

	0	1	2
0	a	b	c
1	d	e	f
2	g	h	i
3	j	k	l

df3 = pd.concat([df1, df2], axis=0)

↓ rowwise

Table

	0	1	2
0	x	a	NaN
1	y	b	NaN
2	z	c	NaN
0	a	b	c
1	d	e	f
2	g	h	i
3	j	k	l

→ df4 = pd.concat([df1, df2], axis=1)

column wise

→ To correct index use

↳ df4.reset_index(drop=True, inplace=True)

JOIN

Q Type

↳

↳

↳

↳

Q Left J

Df1

i) pd.mer

ii) pd.

iii) pd

JOIN IN PANDAS

Q Types of Joins?

- Inner
- Outer
- Left
- Right

Q Left Join, Outer Join, Inner two dataframe

Df1

A	B
1	11
2	12
3	13
4	14

Df2

A	C
1	21
2	22
3	23
5	24

i) pd.merge(var1, var2, on='A')
↳ inner

ii) pd.merge(var1, var2, how="left")

iii) pd.merge(var1, var2, how="outer")

classmate
Date
Page

Q How to access first row?

→ df[column][row] X
↳ can't be used in this

df.head(1) ✓
df.iloc[0] ✓
df.[:][0] X

** Need to practice Indexing ⇒ Accessing element from Kaggle

MEAN, MEDIAN, UNIQUE, VALUE_COUNTS

↳ Review ⇒ Table

(I) reviews.description.mean()

(II) reviews.points.median()
↳ middle value when data is set in order

(III)

reviews.points.mode()
↳ most frequently occuring

(V) reviews.points.unique()
↳ only shows the ~~unique~~ values every value once (col name)

---> Like Like if there are two name (duplicate) 'Roger', 'Roger', 'Hello'.
---> unique will only show it once a name
↳ 'Roger'
'Hello'

★ (VI) reviews.points.value_counts()
↳ How many times each value occurs.
↳ all unique() ones

o/p => Roger 10
Hello 5
Kane 1
;
;

MAPS

↳ are the way to modify the data in Series or DataFrame.

↳ Perform opperation element-wise.

MAP

Two Method to Apply

(I) map & lambda

(II) Apply()

(I) reviews.points.map(lambda p: p - reviews.points.mean())

(II) reviews.apply( )

↳ can't understand

O/P =>

0
1
2
3
:
:

★ → map -s a values

(II) Apply(

Map

(I) works o only

(II) Element

(III) No axi

o/p =>

0	-1.447
1	-1.499
2	1.55
3	⋮
⋮	

* → map & lambda combination returns a new series where all the values are transformed

(II) Apply()

Map()	Apply()
(I) works on series only	(I) works on series & DataFrame
(II) Element wise	(II) Element wise
(III) No axis	(III) axis = 0 (columns) axis = 1 (rows)

sum is a inbuilt fxn.

	A	B
Apply	1	4
↳	2	5
	3	6

① result = df.apply(sum, axis=0)

↳ o/p

A	5
B	15

② result = df.apply(lambda row: row*2, axis=1)

A	B
2	8
4	10
6	12

---> Built In Pandas

↳ a - b -> a - b.mean()

↳ a + b ⇒ a + b.mean()

↳

Q Return a series counting how many times word "Tropical" or "fruity" appears in column app description

(i) -> n_trop = reviews.description.map
('lambda desc: "tropical"
in desc).sum()

0	True
1	False
2	True
3	False
:	

---> True is treated as 1 in sum() & False as 0.

→ sum() ⇒ 2.

(ii) n_fruity = reviews.description.map
(lambda desc: 'fruity in desc).sum()

0	True	2	False
1	True	3	True

→ sum() ⇒ 3.

```
des.count = pd.Series([n_trop, n_fruit],
                      index=['tropical', 'fruity'])
```

Q Why not use here count()?

↳ Because statement in bracket returns Boolean value True or False, count will consider all value in count.

GROUPBY

↳ Simular to GROUP BY in DBMS (SQL)

A	B	C	D
10	60	70	1
10	70	90	2
15	90	80	3
20	100	77	5
30	190	67	4

→ df.groupby('A

A	B
10	130
15	90
20	100
30	190

→ df.groupby

A
10
15
20
30

→ df.groupby

A
10
15
20
30

→ df.groupby('A').sum()

A	B	C	D
10	130	160	1
15	90	1	1
20	100	1	1
30	190	1	1

→ df.groupby('A').B

A	B
10	[60, 70]
15	[90]
20	[100]
30	[190]

→ df.groupby('A').B.min()

A	B
10	60
15	90
20	100
30	190

★ we can also use aggregate function with groupby

↳ reviews.groupby('country').price.agg([len, min, max])

opp ---> will calculate count of prices
min price
max price

SORT FXN ()

↳ Table_name.sort_values(by='col_name')

↳ Table_name.sort_values(by='col_name', ascending=False)

↳ Table_name.sort_values(by=['country', 'len'])

first sorted by this after that in that sorted by len.

★ Column consist
don't get the
instead given

Q Change the

reviews.poin

HANDLI

VALU

Q Check al
a colum

→ reviews[

913 NAN
450 NA
1011 NA

★ Column consisting entirely of strings don't get there own type, they are instead given the object type.

Q Change the datatype of column?

reviews.points.astype('float64')

HANDLING MISSING VALUES

Q Check all the missing value in a column?

→ reviews[pd.isnull(reviews.country)]

913	NAN
450	NAN
1011	NAN

Q Fill Non-Null (NAN) value with something else?

→ reviews.region.fillna("Unknown")

Q Calculate total no. of entries is null in a column?

→ reviews.price.isnull().sum()

Q Print Correlation matrix

→ df.corr()

Q Drop the rows with missing value in Table.

df = df.dropna();

Q Drop the rows having more than two columns with missing value in Table.

df = df.dropna(thresh=2)

Q Drop the have missi

→ df.drop

Q Removes the Tabl

→ df.drop

Q Removes value e

→ user['

Q How to Dataf

df['T

df

Q Drop the rows where age column have missing value

→ df.dropna(subset=['Age'])

Q Removes Duplicate rows from the Table?

→ df.drop_duplicates()

Q Removes Raos with Duplicate value in age column?

→ user['age'].drop_duplicates()

Q How to add new Column in Dataframe?

df['Total_fare'] = 5* df[price]

df

Total_fare

NUMPY

↳ Numpy is a library which provid-es multidimensional array objed & tools for working with these array.

→ LIST to Numpy Array

① lst = [1,2,3,4]
arr = np.array(lst)

o/p ⇒ array([1,2,3,4])

type(arr) → numpy.ndarray.

arr.shape ⇒ (4,)

②

lst1 = [1,2,3,4,5]
lst2 = [2,3,4,5,6]
lst3 = [3,4,5,6,7]

arr1 = np.array([lst1, lst2, lst3, lst4])

o/p

arr1.shape ⇒ (3,5)

↳ DSA
only
↳ In NIGHT
↳ Python.

classmate
Date
Page

O/p => array([[1,2,3,4,5],
[2,3,4,5,6],
[3,4,5,6,7]])

★ If Numpy Array is 1D => Retrieve it like List.

1D Numpy Array => [1, 2, 3, 4, 5]

★ If Numpy Array is 2D or Multi
→ Retrieve it like iloc in Data frame

2D => [1, 2, 3, 4, 5]
[2, 3, 4, 5, 6]
[3, 4, 5, 6, 7]

arr[:, 1]
arr[2:4, :]
arr[2:5, 0:-1]

→ [1, 2,
→ arr < 2
↳ Tru
Q In NDA
POSE T
↳ like ar
arr1.
↳
Q Create
np.a
=> arra
→ np.ar

→ [1, 2, 3, 4, 5]

→ arr < 2
↳ True, False, False, False, False

Q In NDARRAY, WE CAN ALSO TRANSPOSE THAT ARRAY.

↳ like array is of shape =>(3,5)

arr1.reshape(5,3)

↳
1	2	3
2	3	4
3	4	5
4	5	6
5	6	7

Q Create Numpy array using range.

np.arange(1, 10, 1)

=> array([1, 2, 3, 4, 5, 6, 7, 8, 9])

-> np.arange(1, 10, 1).reshape(2,5)

(i) arr * arr => element wise multiplication
(ii) arr * 2 => 2 multiplied element wise

(iii) np.random.randint(10, 50, 4)
↳ 15, 35, 40, 39
array([15, 35, 40, 34])

(iv) np.random.randint(10, 50, 4).reshape(2,2)
↳ array([[20, 35]
[40, 49]])

(v) np.random.random_sample([4,7])
↳ B/w 0 & 1
↳ array([[ ],
[ ],
[ ],
[ ],])