02-03-25

Data handling Quiz

Tabular data:-

↳ Data which is in rows and columns.

Necessary libraries for Data Science

→ numpy, Pandas, Matplotlib, Seaborn, etc.

Numpy Used for numerical purposes.
Like creating matrix or arrays.

→ import numpy as np

→ np.array([1,2,3]) → It will create array of 1,2,3.

Matplotlib is used for creating plots.

→ import matplotlib.pyplot as plt

fig, ax = plt.subplots()
↳ starts a plot.

ax.plot(x, y, 'o') → 'o' uses for creating circle markers for data.
↳ adds the data in plot

plt.show().

→ Making graph look good we have to create labels for x and y axis and adding grid in the graph.

→ ax.grid (color='lightgray').
↳ creates grid in the plot with lightgray color.

→ ax.set_xlabel('x', fontsize=14)

→ ax.set_ylabel('y') → fontsize=16)

This will create x and y labels with their respective fontsize.

→ ax.tick_params(axis='both', which='major', labelsize=14)

↳ it basically shows the point on both axises a bit large.

Regression Line

↳ used to represent a relationship between a dependant variable (y) and an independant variable (x) in a linear equation

↳ regression is also a modal that tries to predict the data Also called fitting the data.

Equation

$y = ax + b$

Lecture 2

Data Types

int → used to store numbers 1, 2, 3 → often 4 bytes but sometimes more

float → used to store decimal number 5.2, 3.1, 9.2.

char → used to store character ('a', 'b')
size of char is 1 byte.

often 4 bytes (single precision) or 8 bytes (double precision)

a 32-bit signed integer value is
from -2147483648 to 2147483647

32

max value is → $2^{31} - 1 = 2147483647$

So, if we tries to add anything extra than its max value it will give us error.

Dataframes in Python :-

↳ dataframe is a two dimensial array or a table with rows and ~~data~~ columns

And in order to create rows and columns in python we can use pandas library.

→ import pandas as pd.

```
data = {
    'temperature' : [22,17,19],
    "Humidity"    : [1,15,19],
    'windSpeed'   : [15,19,25]
    }

df = pd.dataframe(data, index=['day_1','day_2','day_3'])

print(df)
```

→ pd.dataframe will create rows and colums of our data with the indexes which we have given.

Strings

↳ sequence of Character.

```
a = 'Hello World'
print(a)
type(a) → str
```

a[0] → 'H'
type(a[0]) → str.

→ a = '37.4'
type(a) → str

# Type conversion.
x = float(a)
type(x) → float.

Text file which we mostly used in DS.

→ Comma Separated values (CSV) → mostly used.
→ White-Space Separated
→ Fixed Width

to read csv in pandas.
→ pd.read_csv("file.csv")
white space separated
→ pd.read_csv('file.txt', delim_whitespace = True)
fixed width
→ col_width = [7, 6, 7]
→ pd.read_fwf('file.txt', widths = col_width)

Parquet file format :-

↳ column-based file format
↳ binary encoding
↳ Uses data compression.

Save dataframe to Parquet file
→ df.to_parquet('file.parquet')

reading the data from Parquet file

→ pd.read_parquet('file.parquet')

Representing Images

→ we can create images with array.
→ Once we convert the images into numpy arrays we can do ordinary math with them.

Lecture 3

Building Tables with Python:-

→ df = pd.dataframe(data).

→ df.columns
↳ shows the list of columns.

→ df.length
↳ shows the data of length column.

→ df['width']
↳ shows the data of width column.

df.length AND df['length'].

Both are same
And both of item shows the data of length column.

→ df.length[1]
↳ shows the data of second value in length column.

→ df['width'][2]

↳ shows the third data in the width column.

→ df['width'][2] = df['width'][2] + 1

↳ it is basically updating third data in width column.

→ type(df['width']).

↳ pandas series.

Series object in Pandas :-

↳ one dimensional array and can hold data of any type

data = pd.Series([1,2,3,4], index = ['a','b','c','d']).

Slicing in DataFrame :-

∵ array starts from 0

→ df[2:5]

↳ it will show dataframe data from 3rd row to 5th row.

→ df[3:]

↳ shows data from 4th row to last.

→ df.loc[0,'width'].

↳ shows first value in width column.

→ df.iloc[0, 1]

↳ shows the data of 0th means first row on 1st column.

→ df.iloc[3:4, ['width', 'height']]

↳ it will show row 3rd and 4th of width and height column.

~~→ df.iloc[3:5, 1~~

→ df.iloc[3:5, 1:3]

↳ it will select row 4th and 5th from column 2nd & and 3rd.

if we do df.width > 6 it will return us data in the form of true and false if based upon condition.

Concatination of Dataframe:-

→ pd.concat([df, df], axis=0)

↳ it adds the second dataframe into one vertically means after the first dataframe.

→ pd.dataframe([df, df], axis=1)

↳ it ~~vertical~~ horizentally add new columns of 2nd dataframe after the columns of first data frame.

Joing

Joining Data frames:-

→ data_a OR df_a

→ data_b OR df_b

→ pd.merge(df_a, df_b, on='id')

↳ it will merge both dataframes data based on id column

For example:-

df_a =

	id	name
0	1	Alice
1	2	Bob
2	3	Aakash
3	4	David

df_b =

	id	age
0	3	20
1	4	25
2	5	40
3	6	30

So in above it will give us this below result.

	id	name	age
0	3	Aakash	20
1	4	David	25

Joins types:-

Inner Join:-

↳ returns all the rows where there is a match in both columns / dataframes.

→ ~~pd.df~~ → pd.merge(df_a, df_b, on='id', how='inner')

it will return above result.

Left Join:-

↳ returns all rows from first dataframe

→ pd.merge(df_a, df_b, on='id', how='left')

↳ it will show data from 1 to 4 Id and add age column from second df and add NaN value if there is no value is given respective to Id.

Right Join:-

↳ returns all the data from 2nd dataframe

→ pd.merge(df_a, df_b, on='id', how='right')

↳ It will show the all data of 2nd dataframe and add the columns of 1st dataframe. And add NaN if there is no data for any row.

Outer Join:-

↳ Returns all rows from the both dataframes.

→ pd.merge(df_a, df_b, on='id', how='outer')

↳ return all the data of both dataframes and add NaN values for any data which is not found.

Lecture 4 Visualising Distributions:-

Mainly use slides. No need for any explanation.