Python :- Basics To Advance

Created	@September 29, 2024 2:58 PM

Python Topics covered :- List , Dictionary , tuples

Basics of Libraries:

1. Numpy
2. pandas
3. Matplotlib
4. Seaborn

Python:

Data Structures

List

- Append → To add at last
  ex: list.append(["John", "David"])

- Insert → To add in particular index
  ex: list.insert(1, "John")
  index value

Sets {no duplicate element}
ex set_var = {1,2,3,4,3}
set_var
{1,2,3,4} output

Here we can't access through indexing

Dictionary → key value pairs

Adding items in dictionary
my_dict['car4'] = 'Audi2.0'
→ if it already present- it will replaced by new value

# How to access nested dictionary
print(car_type)
{'car1': {'Mercedes': 1960}, 'car2': {'Audi': 1979} ...

print(car_type['car1'])
{'Mercedes': 1960}

print(car_type['car1']['Mercedes'])
1960

Tuples: → not mutable

ex: my_tuple = (1, 2, 3)
my_tuple[0] = ("John")
This will throw error

Library: Numpy
It provides a high-performance multidimensional array object, and tool for working with these arrays.
It is the fundamental package for scientific computing with Python.

{import numpy as np} # importing.

ex. my_list = [1, 2, 3, 4, 5]
arr = np.array(my_list)

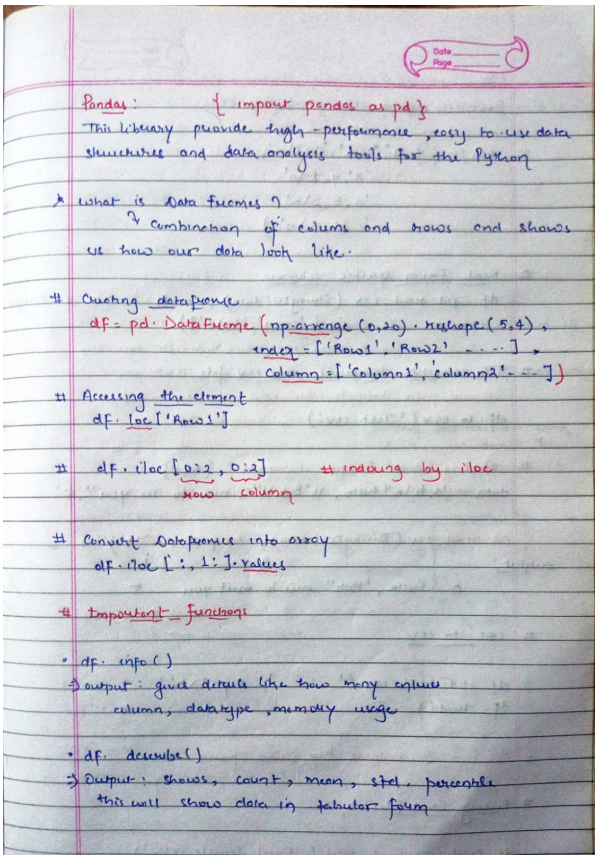
type(arr)
⇒ numpy.ndarray output.

Creating Multidimensional array
ex arr = np.array(my_list1, my_list2, my_list3)

Indexing
ex: arr[3]
⇒ 4 output.

In Multidimensional
ex arr [row, column]
arr [0:2, 3:4]

Copy function and broadcasting
arr[3:] = 100
⇒ array([1, 2, 3, 100, 100 ....]) output.



Pandas: { import pandas as pd }
This library provide high-performance, easy to use data structures and data analysis tools for the Python

* What is Data Frames ?
  ↳ combination of columns and rows and shows us how our data look like.

# Creating dataframes
df = pd.DataFrame(np.arrange(0,20).reshape(5,4), index = ['Row1','Row2' ...], column = ['Column1','column2'...])

Accessing the element
df.loc['Row1']

df.iloc[0:2, 0:2] # indexing by iloc
row column

Convert Dataframes into array
df.iloc[:, 1:].values

Important functions

• df.info()
⇒ output: gives details like how many entries column, datatype, memory usage

• df.describe()
⇒ Output: shows, count, mean, std, percentile this will show data in tabular form

CSV

```
from io import StringIO, BytesIO

# data = ('col1,col2,col3\n',
          'x,y,z\n'
          'a,b,2\n'
          'c,d,3\n')

# Read from specific columns
df = pd.read_csv(StringIO(data), usecols = lambda x: x.upper()
                 in ['COL1','COL3'])

# To convert dataframe into csv file

df.to_csv('Test.csv')

# Quoting and Escape characters : {NLP}
data = 'a,b\n"hello, \"Bob\", nice to see you",5'

pd.read_csv(StringIO(data), escapechar='\')
```

output:

	a	b
0	hello, "Bob" nice to met you	5

```
# URL to CSV

df = pd.read_csv('------', sep='\t')
df.head()

# Json :- It contains key value pairs

# Reading Excel files

df_excel = pd.read_excel('Excel_Sample.xlsx')
df_excel.head()
```

↗ Python object into byte stream

Pickling: All pandas objects are equipped with to_pickle methods which use Python's cPickle module to save data structures to disk using pickle format

```
df_excel.to_pickle('df_excel')
df = pd.read_pickle('df_excel')
df.head()
```

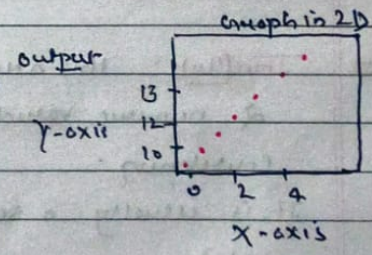
Matplotlib : is a plotting library for the python and its numerical mathematics extension Numpy.
using GUI toolkits like Tkinter, wxpython, qt or GTK+

```
import matplotlib.pyplot as plt
% matplotlib inline
```

↳ This reduces our work so that we don't have to write plt.show() everytime

- `plt.scatter(x,y,c='r')` ← colour
  `plt.xlabel('x-axis')`
  `plt.ylabel('y-axis')`
  `plt.title('Graph in 2D')`

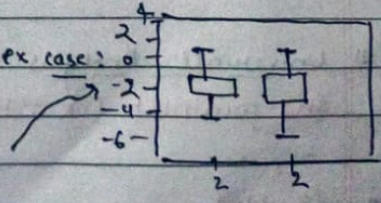
⇒ output



- `plt.plot(x,y,'r')`
  ↳ to get straight line

`plt.box(..)`
`plt.hist(..)`

ex case:



Box Plot using Matplotlib

Q: `data = [np.random.normal(0, std, 100) for std in range(1,4)]`

`plt.boxplot(data, vert=True, patch_artist=True);`

Pie chart:

plt.pie (sizes, explode = explode, labels = labels, colors = colors, autopct = '%.1f%%', shadow = False)

→ How far we want

This will show plot form our data

* Seaborn :- import seaborn as sns

↳ Helps us to visualise data with multiple features

eg: displot . Jointplot . pairplot

Correlation with Heatmap

A correlation heatmap uses coloured cells - typically in a monochromatic scale to show a 2D correlation matrix (table) between two discrete dimensions or event types.

It is very informative in feature selection.

{values range -1 to +1}

df.corr()

sns.heatmap(df.corr())

Jointplot: this allows to study the relationship between 2 numeric variables. The central chart display their correlation.

It is usually a scatterplot, a hexbin plot or a 2D density plot.

Univariate Analysis

sns.jointplot (x = 'Hip', y = 'total bill', data = df, kind = 'reg')

this will also show regression line

Pairplot: * features more than 2.

sns.pairplot(df)