



# **Data Handling and Visualization (CSE2026) - Record**

**Submitted From,**

**Name: Vishnu Reddy Kotam**

**Roll No: 20201ISE0032**

**Section & Semester: 8ISE-1 & 8<sup>th</sup> Sem**

**Submitted To,**

**Ms. Poornima S - Asst. Professor (CSE)**

**Presidency University, Bengaluru**

# INDEX

Sl. Number	Labsheets	Page number
1	LABSHEET 1(NUMPY)	1
2	LABSHEET 2(PANDAS)	6
3	LABSHEET 3(DATA CLEANING)	9
4	LABSHEET 4(Z-SCORE NORMALIZATION)	18
5	LABSHEET 5(OUTLIER DETECTION)	19
6	LABSHEET 6(MATPLOTLIB)	23
7	LABSHEET 7(DATA WRANGLING)	33
8	LABSHEET 8(COLORMAPS)	36
9	LABSHEET 9(HEATMAPS)	38
10	LABSHEET 10(SEABORN COLOR PALETTE)	40
11	LABSHEET 11(PLOTS IN SEABORN)	48
12	LABSHEET 12(TEXT DATA VISUALIZATION)	56
13	LABSHEET 13(TIME SERIES DATA)	59

## BASIC PLOTTING IN MATPLOTLIB

```
!pip install matplotlib==3.4
Collecting matplotlib==3.4
  Downloading matplotlib-3.4.0.tar.gz (37.1 MB)
    37.1/37.1 MB 26.7 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.4) (0.12.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.4) (1.4.5)
Requirement already satisfied: numpy>=1.16 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.4) (1.25.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.4) (9.4.0)
Requirement already satisfied: pyparsing>=2.2.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.4) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.4) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib==3.4) (1.16.0)
Building wheels for collected packages: matplotlib
  Building wheel for matplotlib (setup.py) ... done
  Created wheel for matplotlib: filename=matplotlib-3.4.0-cp310-cp310-linux_x86_64.whl size=10425804 sha256=a604bf5dc05fbb0e67f2a62efe3e
  Stored in directory: /root/.cache/pip/wheels/27/ea/35/0964d59ed4c7270bbeabc79c0984b58d72f9e4463746bf7062
Successfully built matplotlib
Installing collected packages: matplotlib
  Attempting uninstall: matplotlib
    Found existing installation: matplotlib 3.7.1
      Uninstalling matplotlib-3.7.1:
```

```
from matplotlib import pyplot as plt
plt.style.use('seaborn-whitegrid')

import numpy as np
print("step 1")
```

## TO CREATE EMPTY PLOT

```
fig=plt.figure()
ax=plt.axes()
ax.grid()
```

## PLOT LINESPACE

```
fig=plt.figure()
ax=plt.axes()
x=np.linspace(0,10,1000)
ax.plot(x,np.sin(x));
```

## ADDING TITLE AND LABELS TO PLOT

```
fig=plt.figure()
ax=plt.axes()
x=np.linspace(0,10,1000)
ax.plot(x,np.sin(x))
ax.set_title('simple plot')
ax.set_xlabel('x label')
ax.set_ylabel('y label')
```

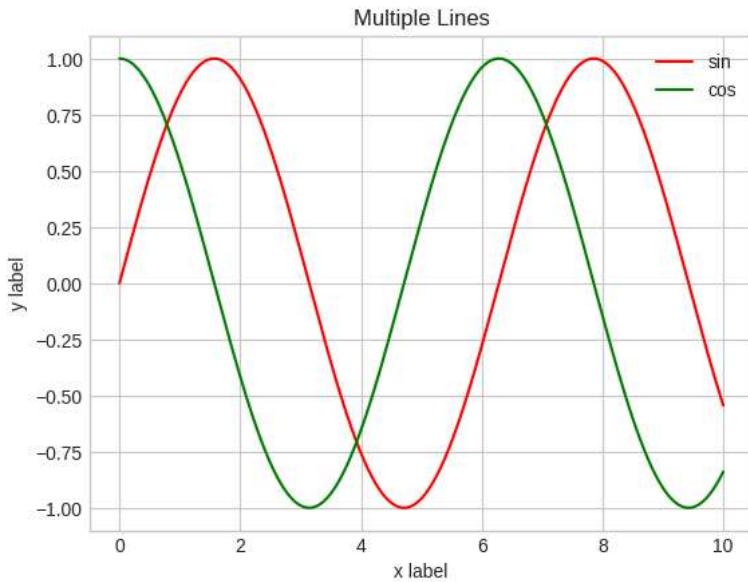
## ADD A TITLE TO THE PLOT ABOVE

```
fig=plt.figure()
ax=plt.axes()
x=np.linspace(0,10,1000)
ax.plot(x,np.sin(x),label='sin')
ax.plot(x,np.cos(x),label='cos')
ax.set_title('Multiple Lines')
ax.set_xlabel('x label')
ax.set_ylabel('y label')
ax.legend()
```

## SPECIFY COLOR BY NAME

```
fig=plt.figure()
ax=plt.axes()
x=np.linspace(0,10,1000)
ax.plot(x,np.sin(x),label='sin',color='red')
ax.plot(x,np.cos(x),label='cos',color='g')
ax.set_title('Multiple Lines')
ax.set_xlabel('x label')
ax.set_ylabel('y label')
ax.legend()
```

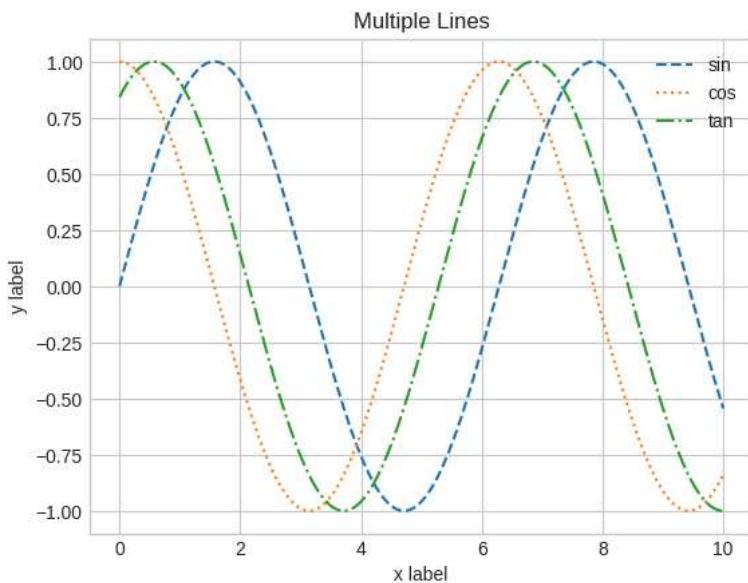
↳ <matplotlib.legend.Legend at 0x7be9aa7599c0>



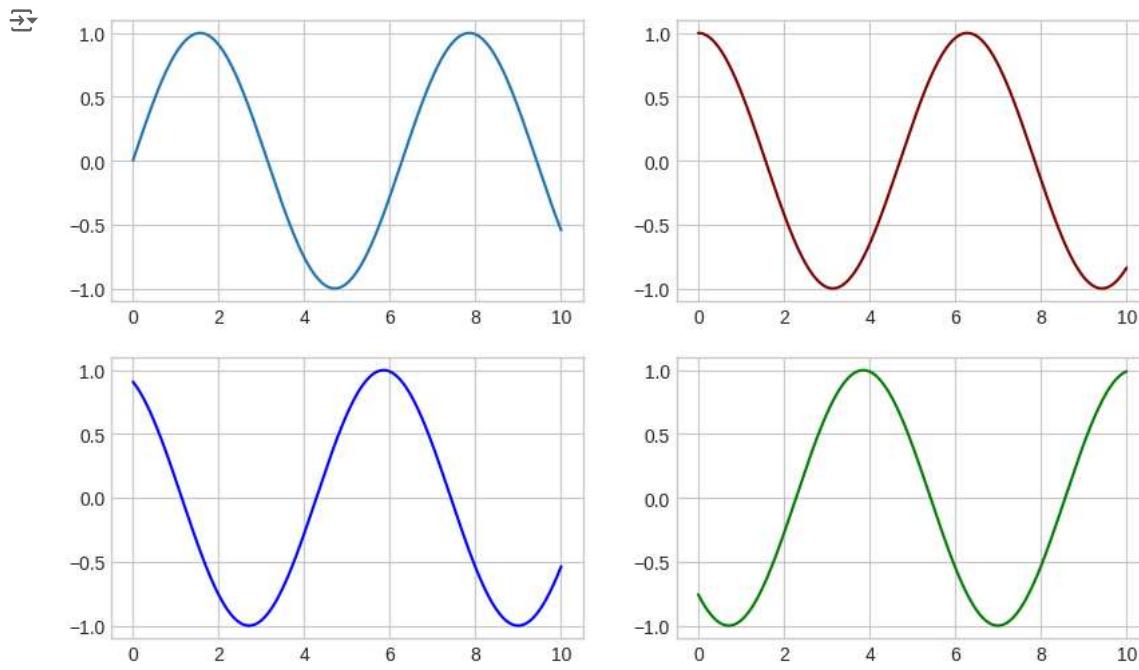
## LINESTYLE

```
fig=plt.figure()
ax=plt.axes()
x=np.linspace(0,10,1000)
ax.plot(x,np.sin(x),label='sin',linestyle='dashed')
ax.plot(x,np.cos(x),label='cos',linestyle='dotted')
ax.plot(x,np.sin(x+1),label='tan',linestyle='dashdot')
ax.set_title('Multiple Lines')
ax.set_xlabel('x label')
ax.set_ylabel('y label')
ax.legend()
```

↳ <matplotlib.legend.Legend at 0x7be9aa870d90>

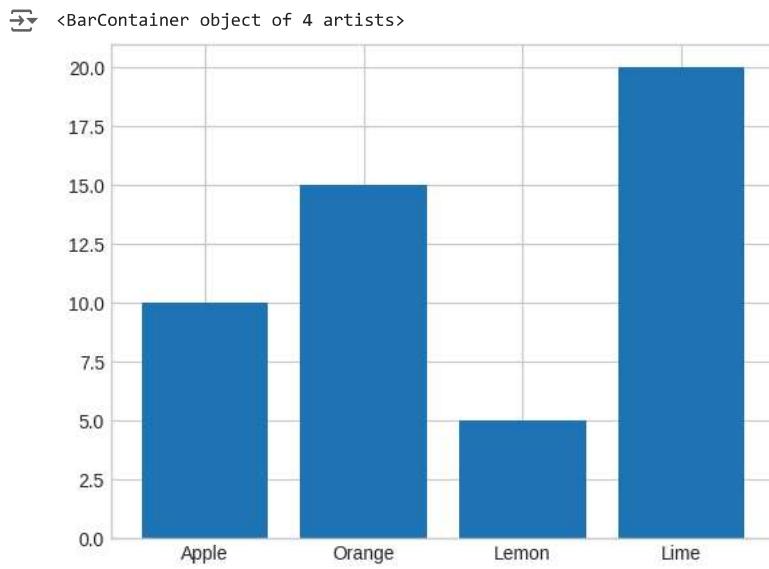


```
#SUBPLOTS
fig,axs=plt.subplots(2,2,figsize=(10,6))
x=np.linspace(0,10,1000)
axs[0,0].plot(x,np.sin(x))
axs[0,1].plot(x,np.cos(x),color='maroon')
axs[1,0].plot(x,np.sin(x+2),color='blue')
axs[1,1].plot(x,np.sin(x+4),color='green');
```



## BAR CHART

```
data={'Apple':10,'Orange':15,'Lemon':5,'Lime':20}
names=list(data.keys())
values=list(data.values())
fig=plt.figure()
ax=plt.axes()
ax.bar(names,values)
```



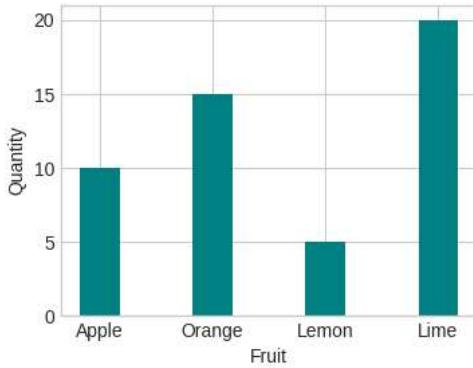
Detailing bar chart

```

data={'Apple':10,'Orange':15,'Lemon':5,'Lime':20}
names=list(data.keys())
values=list(data.values())
fig=plt.figure(figsize=(4,3))
ax=plt.axes()
ax.bar(names,values,color='teal',width=0.35)
ax.set_xlabel('Fruit')
ax.set_ylabel('Quantity')

```

→ Text(0, 0.5, 'Quantity')



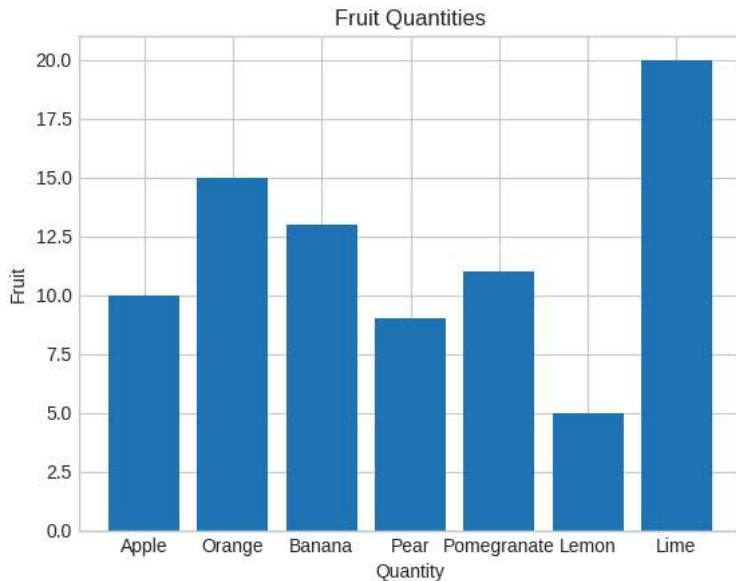
```
#creating a dictionary
```

```

data={'Apple':10,
      'Orange':15,
      'Banana':13,
      'Pear':9,
      'Pomegranate':11,
      'Lemon':5,
      'Lime':20}
names=list(data.keys())
values=list(data.values())
ax=plt.axes()
ax.bar(names,values)
ax.set_xlabel('Quantity')
ax.set_ylabel('Fruit')
ax.set_title('Fruit Quantities')

```

→ Text(0.5, 1.0, 'Fruit Quantities')





## TO CREATE SERIES

```
import pandas as pd
import numpy as np
data=np.array([ Add text cell ])
s=pd.Series(data)
print(s)
```

0 a  
1 b  
2 c  
dtype: object

## TO CREATE DICTIONARY

```
import pandas as pd
import numpy as np
data={'a':0,'b':1,'c':2}
s=pd.Series(data)
print(s)
```

a 0  
b 1  
c 2  
dtype: int64

## ARRAY WITH INDEX

```
import pandas as pd
import numpy as np
data=np.array(['a','b','c'])
s=pd.Series(data,index=[100,102,104])
print(s)
```

100 a  
102 b  
104 c  
dtype: object

## DICTIONARY WITH INDEX

```
import pandas as pd
import numpy as np
data={'a':0,'b':1,'c':2}
s=pd.Series(data,index=['a','d','b','c'])
print(s)
```

a 0.0  
d NaN  
b 1.0  
c 2.0  
dtype: float64

## CREATING A SERIES FROM SCALAR

```
import pandas as pd
import numpy as np
s=pd.Series(5,index=[0,1,2,3])
print(s)
```

0 5  
1 5  
2 5  
3 5  
dtype: int64

## RETRIEVE THE 1ST ELEMENT

```
import pandas as pd
s=pd.Series([1,2,3,4,5],index=['a','b','c','d','e'])
print(s[0])
```

→ 1

SLICING

Add text cell

```
import pandas as pd
s=pd.Series([1,2,3,4,5],index=['a','b','c','d','e'])
print(s[1:4])
```

```
→ b    2
  c    3
  d    4
  dtype: int64
```

```
import pandas as pd
s=pd.Series([1,2,3,4,5,6,7,8,9,10],index=['a','b','c','d','e','f','g','h','i','j'])
print(s[3:8])
```

```
→ d    4
  e    5
  f    6
  g    7
  h    8
  dtype: int64
```

retrieve 1st 3 elements using slicing

```
import pandas as pd
s=pd.Series([1,2,3,4,5,6,7,8,9,10],index=['a','b','c','d','e','f','g','h','i','j'])
print(s[:3])
```

```
→ a    1
  b    2
  c    3
  dtype: int64
```

retrieve a single element

```
import pandas as pd
s=pd.Series([1,2,3,4,5,6,7,8,9,10],index=['a','b','c','d','e','f','g','h','i','j'])
print(s['a'])
```

→ 1

retrieve a multiple element

```
import pandas as pd
s=pd.Series([1,2,3,4,5,6,7,8,9,10],index=['a','b','c','d','e','f','g','h','i','j'])
print(s[['a','j','e']])
```

```
→ a    1
  j    10
  e    5
  dtype: int64
```

## OPERATIONS ON DATASET

```
import pandas as pd
df=pd.read_csv("/content/nyc_weather.csv")
```

```

→----- UnicodeDecodeError ----- Traceback (most recent call last)
<ipython-input-28-c8675a4efe38> in <cell line: 2>()
      1 import pandas as pd
----> 2 df=pd.read_csv("/content/nyc_weather.csv")

Add text cell  10 frames /usr/local/lib/python3.10/dist-packages/pandas/_libs/parsers.pyx in pandas._libs.parsers.raise_parser_error()

UnicodeDecodeError: 'utf-8' codec can't decode bytes in position 15-16: invalid continuation byte

```

## CREATING DATAFRAME

```
import pandas as pd
```

```
df=pd.DataFrame()
print(df)
```

```
→ Empty DataFrame
Columns: []
Index: []
```

```
import pandas as pd
data=[1,2,3,4]
df=pd.DataFrame(data)
print(df)
```

```
→ 0
 0 1
 1 2
 2 3
 3 4
```

## MERGING TO FORM DATAFRAME

```
import pandas as pd
data=[['sa',10],['kav',20]]
df=pd.DataFrame(data,columns=['Name','Age'])
print(df)
```

```
→ Name  Age
 0  sa   10
 1  kav  20
```

## APPLYING DATATYPE

```
import pandas as pd
data=[['sa',10],['kav',20]]
df=pd.DataFrame(data,columns=['Name','Age'],dtype=float)
print(df)
```

```
→ Name  Age
 0  sa  10.0
 1  kav 20.0
<ipython-input-38-58b87c03f679>:3: FutureWarning: Could not cast to float64, falling back to object. This behavior is deprecated. In a f
 df=pd.DataFrame(data,columns=['Name','Age'],dtype=float)
```

```
import pandas as pd
data={'name':['tom','jack'],'age':[1,2]}
df=pd.DataFrame(data,index=['rank1','rank2'])
print(df)
```

```
→ name  age
 rank1  tom    1
 rank2  jack   2
```

Add text cell

```
# import the pandas library
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
'h'],columns=['one', 'two', 'three'])
print(df)
df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])

print (df)
```

```
→      one      two      three
a  0.447414 -0.156942  0.798209
c  1.266992 -1.194218 -0.402855
e -0.647991  1.435391 -0.072793
f -0.581821  1.038771 -0.776008
h  0.686654 -1.064446  1.137672
      one      two      three
a  0.447414 -0.156942  0.798209
b    NaN      NaN      NaN
c  1.266992 -1.194218 -0.402855
d    NaN      NaN      NaN
e -0.647991  1.435391 -0.072793
f -0.581821  1.038771 -0.776008
g    NaN      NaN      NaN
h  0.686654 -1.064446  1.137672
```

```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
'h'],columns=['one', 'two', 'three'])

df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])

print (df['one'].isnull())
```

```
→ a  False
b  True
c  False
d  True
e  False
f  False
g  True
h  False
Name: one, dtype: bool
```

```
#Replace the missing values by 0
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(3, 3), index=['a', 'c', 'e'],columns=['one',
'two', 'three'])
df = df.reindex(['a', 'b', 'c'])
print (df)
print ("NaN replaced with '0':")
print (df.fillna(0))
```

```
→      one      two      three
a  0.635750 -2.285400  0.248117
b    NaN      NaN      NaN
c -0.287978 -1.188414  1.378976
NaN replaced with '0':
      one      two      three
a  0.635750 -2.285400  0.248117
b  0.000000  0.000000  0.000000
c -0.287978 -1.188414  1.378976
```

```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
'h'],columns=['one', 'two', 'three'])
df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
print(df)
print (df.fillna(method='pad'))
```

```
→      one      two      three
a -0.366934  0.335056 -0.193792
b      NaN      NaN      NaN
c  1.056679  0.756225  0.912214
d      NaN      NaN      NaN
e -0.377422  0.985928  1.222426
f -0.929130 -0.094656 -0.679399
g      NaN      NaN      NaN
h  1.602288 -0.676405 -0.521540
      one      two      three
a -0.366934  0.335056 -0.193792
b -0.366934  0.335056 -0.193792
c  1.056679  0.756225  0.912214
d  1.056679  0.756225  0.912214
e -0.377422  0.985928  1.222426
f -0.929130 -0.094656 -0.679399
g -0.929130 -0.094656 -0.679399
h  1.602288 -0.676405 -0.521540
```

```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
'h'], columns=['one', 'two', 'three'])
df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])

print (df.fillna(method='bfill'))
```

```
→      one      two      three
a -0.692098  1.531090  0.364882
b -0.490656  0.770679  0.338704
c -0.490656  0.770679  0.338704
d -1.281391  0.103691 -0.117571
e -1.281391  0.103691 -0.117571
f -1.063626  0.533522  2.010359
g  1.511229 -0.596010  2.030684
h  1.511229 -0.596010  2.030684
```

```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
'h'], columns=['one', 'two', 'three'])
#print(df)
df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
print(df)
print (df.dropna())
```

```
→      one      two      three
a -1.215042  1.457607 -0.241829
b      NaN      NaN      NaN
c  1.454749 -0.087851  0.659018
d      NaN      NaN      NaN
e  1.132439 -0.333103  0.232882
f  2.031967  0.474128 -0.456087
g      NaN      NaN      NaN
h  1.438924 -0.782758  1.578343
      one      two      three
a -1.215042  1.457607 -0.241829
c  1.454749 -0.087851  0.659018
e  1.132439 -0.333103  0.232882
f  2.031967  0.474128 -0.456087
h  1.438924 -0.782758  1.578343
```

```
import pandas as pd
import numpy as np
df = pd.DataFrame({'one':[10,20,30,40,50,2000],
'two':[1000,0,30,40,50,60]})
print(df)
print (df.replace({1000:10,2000:60}))
```

```
→      one    two
0     10  1000
1     20      0
2     30     30
3     40     40
4     50     50
5   2000     60
      one    two
```

0	10	10
1	20	0
2	30	30
3	40	40
4	50	50
5	60	60

Start coding or generate with AI.

```
import numpy as np
data = [1,2,2,2,3,1,1,15,2,2,2,3,1,1,2]
mean = np.mean(data)
std = np.std(data)
print('Mean of the dataset is' ,mean)
print('Standard deviation is' ,std)
threshold = 3
outlier = []
for i in data:
    z = (i-mean)/std
    if z>threshold:
        outlier.append(i)
        print('Outlier in dataset is' ,outlier)
```

→ Mean of the dataset is 2.6666666666666665  
Standard deviation is 3.3598941782277745  
Outlier in dataset is [15]

```
import numpy as np
data = [41,26,25,21,38,19,1,157,27,24,233,37,17,13,25]
mean = np.mean(data)
std = np.std(data)
print('Mean of the dataset is' ,mean)
print('Standard deviation is' ,std)
threshold = 3
outlier = []
for i in data:
    z = (i-mean)/std
    if z>threshold:
        outlier.append(i)
        print('Outlier in dataset is' ,outlier)
```

→ Mean of the dataset is 46.93333333333333  
Standard deviation is 60.488529674825315  
Outlier in dataset is [233]

Start coding or generate with AI.

```

import numpy as np
data= [1,2,2,2,3,1,1,15,2,2,2,3,1,1,2]
mean= np.mean(data)
std= np.std(data)
print("mean of the dataset ids", mean)
print("std is", std)
threshold=3
outlier=[]
for i in data:
z=(i-mean)/std
if z>threshold:
outlier.append(i)
print("outlier in dataset is", outlier)

→ mean of the dataset ids 2.6666666666666665
std is 3.3598941782277745
outlier in dataset is [15]

```

Strating labsheets

```

import requests

pip install --upgrade 'library'

→ Collecting library
  Downloading Library-0.0.0.tar.gz (1.4 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: library
  Building wheel for library (setup.py) ... done
  Created wheel for library: filename=Library-0.0.0-py3-none-any.whl size=2054 sha256=bff50854623545f3632a3b05f8cb48a7f3ae12ce62498365f4
  Stored in directory: /root/.cache/pip/wheels/e0/71/7d/b0e29b944e43374597cd4e3b88c85197001c9bfc5dce191f4
Successfully built library
Installing collected packages: library
Successfully installed library-0.0.0

```



the romechange website:

```

r=requests.get('https://www.romexchange.com/')

r

→ <Response [406]>

r.status_code

→ 406

url='https://www.romexchange.com/'
headers={'Content-type':'application/json'}

url

→ 'https://www.romexchange.com/'

headers

→ {'Content-type': 'application/json'}

r=requests.get(url,headers=headers)

url='https://www.romexchange.com/'
headers={'User-Agent':'XY','Content-type':'application/json'}
r=requests.get(url,headers=headers)

url

```

```
→ 'https://www.romexchange.com/'
```

```
headers
```

```
→ {'User-Agent': 'XY', 'Content-type': 'application/json'}
```

```
r
```

```
→ <Response [200]>
```

```
r.status_code
```

```
→ 200
```

```
url='https://www.romexchange.com/api?item=mastela&exact=false'  
headers={'User-Agent':'XY','Content-type':'application/json'}
```

```
r=requests.get(url,headers=headers)  
r.status_code
```

```
→ 500
```

```
r.text
```

```
→ ''
```

(+ Code) (+ Text)

## BASIC PLOTTING IN MATPLOTLIB

```
!pip install matplotlib==3.4
```

```
→ Collecting matplotlib==3.4
  Downloading matplotlib-3.4.0.tar.gz (37.1 MB)
    37.1/37.1 MB 25.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.4) (0.12.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.4) (1.4.5)
Requirement already satisfied: numpy>=1.16 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.4) (1.23.5)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.4) (9.4.0)
Requirement already satisfied: pyParsing>=2.2.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.4) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib==3.4) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib==3.4) (1.16.0)
Building wheels for collected packages: matplotlib
  Building wheel for matplotlib (setup.py) ... done
  Created wheel for matplotlib: filename=matplotlib-3.4.0-cp310-cp310-linux_x86_64.whl size=10425194 sha256=ad0e297b54bf32bbf136b3ec021c
  Stored in directory: /root/.cache/pip/wheels/27/ea/35/0964d59ed4c7270bbeabc79c0984b58d72f9e4463746bf7062
Successfully built matplotlib
Installing collected packages: matplotlib
  Attempting uninstall: matplotlib
    Found existing installation: matplotlib 3.7.1
    Uninstalling matplotlib-3.7.1:
      Successfully uninstalled matplotlib-3.7.1
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source
lida 0.0.10 requires fastapi, which is not installed.
lida 0.0.10 requires kaleido, which is not installed.
lida 0.0.10 requires python-multipart, which is not installed.
lida 0.0.10 requires uvicorn, which is not installed.
mizani 0.9.3 requires matplotlib>=3.5.0, but you have matplotlib 3.4.0 which is incompatible.
plotnine 0.12.4 requires matplotlib>=3.6.0, but you have matplotlib 3.4.0 which is incompatible.
Successfully installed matplotlib-3.4.0
```

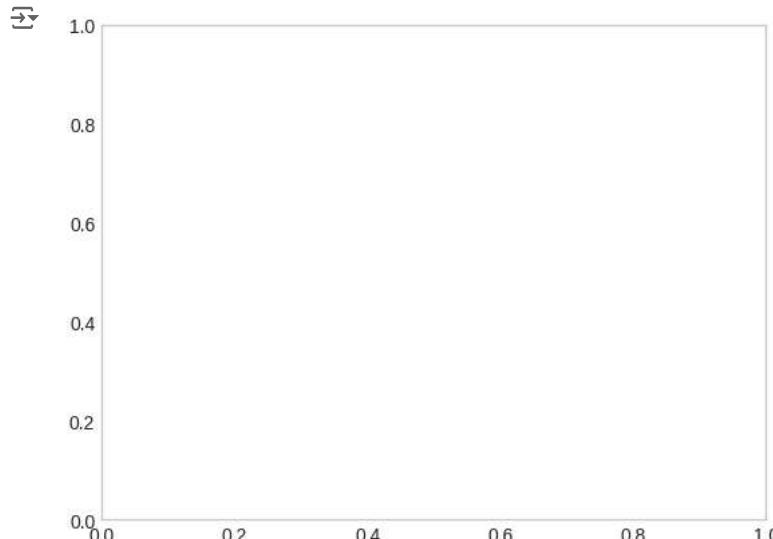
```
from matplotlib import pyplot as plt
plt.style.use('seaborn-whitegrid')
```

```
import numpy as np
print("step 1")
```

```
→ step 1
<ipython-input-3-240c5389bdd3>:2: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6, as th
  plt.style.use('seaborn-whitegrid')
```

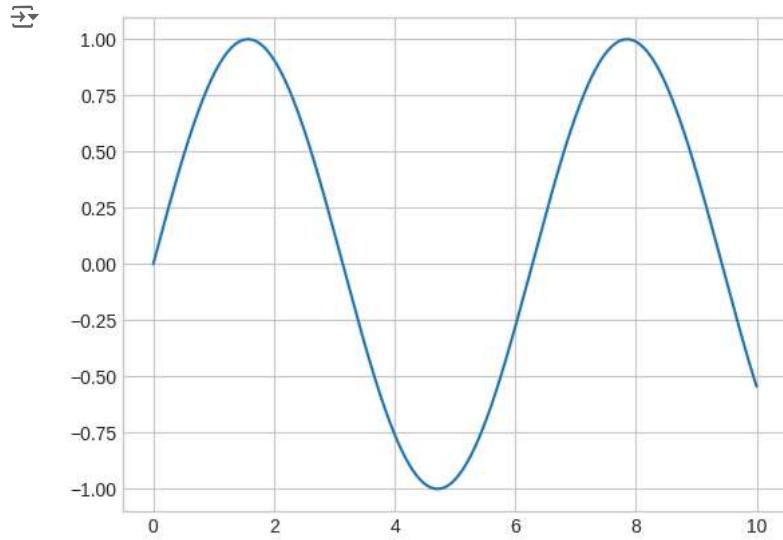
## TO CREATE EMPTY PLOT

```
fig=plt.figure()
ax=plt.axes()
ax.grid()
```



## PLOT LINESPACE

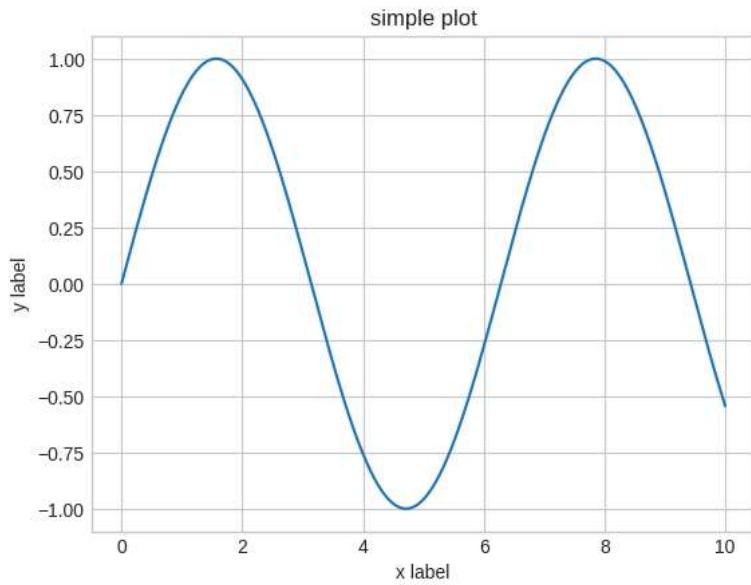
```
fig=plt.figure()
ax=plt.axes()
x=np.linspace(0,10,1000)
ax.plot(x,np.sin(x));
```



## ADDING TITLE AND LABELS TO PLOT

```
fig=plt.figure()
ax=plt.axes()
x=np.linspace(0,10,1000)
ax.plot(x,np.sin(x))
ax.set_title('simple plot')
ax.set_xlabel('x label')
ax.set_ylabel('y label')

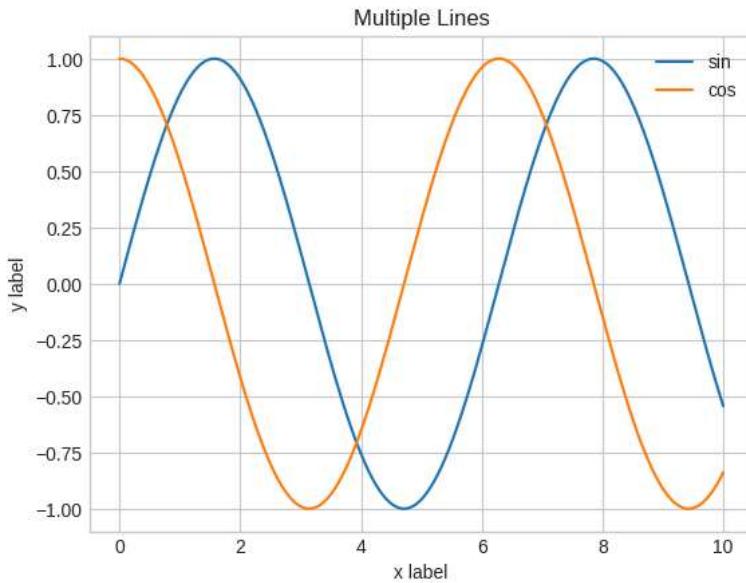
Text(0, 0.5, 'y label')
```



## ADD A TITLE TO THE PLOT ABOVE

```
fig=plt.figure()
ax=plt.axes()
x=np.linspace(0,10,1000)
ax.plot(x,np.sin(x),label='sin')
ax.plot(x,np.cos(x),label='cos')
ax.set_title('Multiple Lines')
ax.set_xlabel('x label')
ax.set_ylabel('y label')
ax.legend()
```

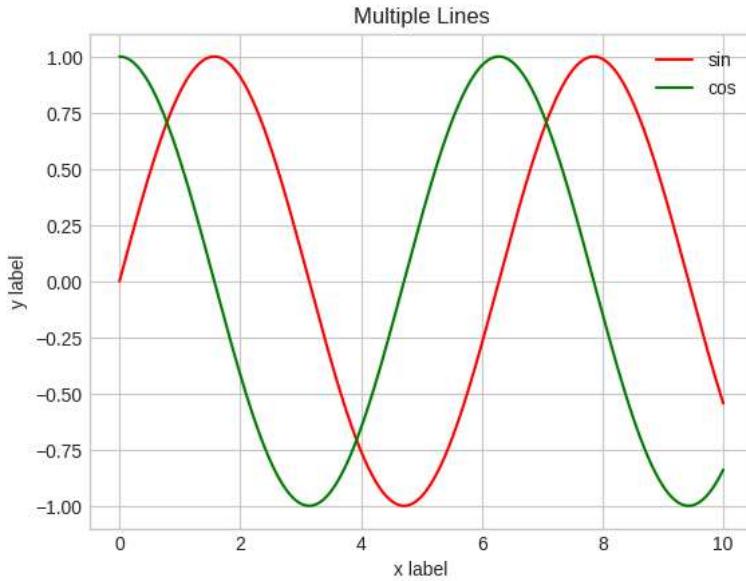
↳ <matplotlib.legend.Legend at 0x7be9c6f80df0>



#### SPECIFY COLOR BY NAME

```
fig=plt.figure()
ax=plt.axes()
x=np.linspace(0,10,1000)
ax.plot(x,np.sin(x),label='sin',color='red')
ax.plot(x,np.cos(x),label='cos',color='green')
ax.set_title('Multiple Lines')
ax.set_xlabel('x label')
ax.set_ylabel('y label')
ax.legend()
```

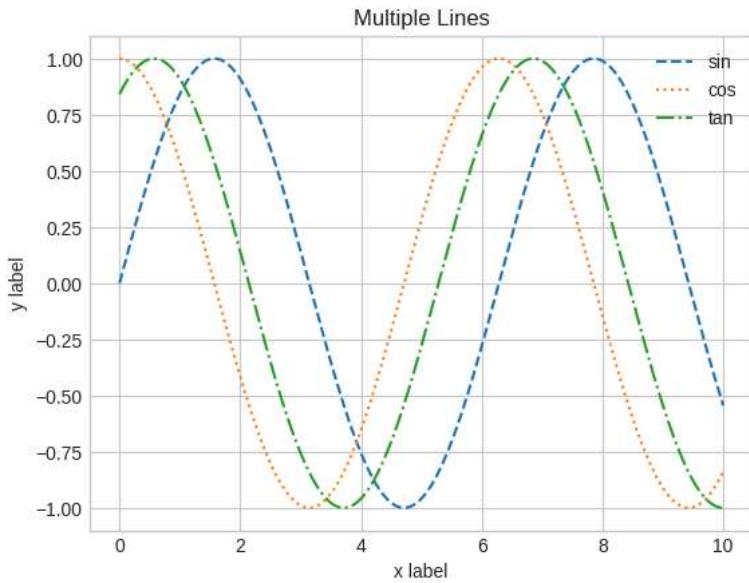
↳ <matplotlib.legend.Legend at 0x7be9aa7599c0>



#### LINESTYLE

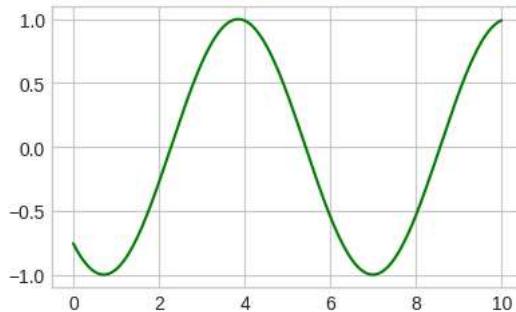
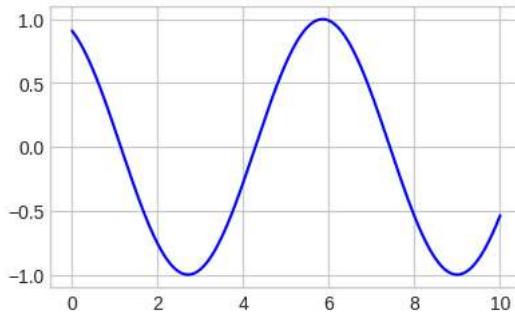
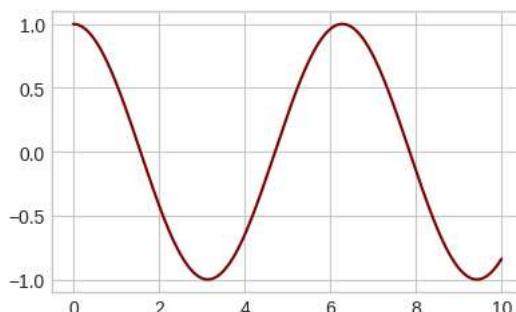
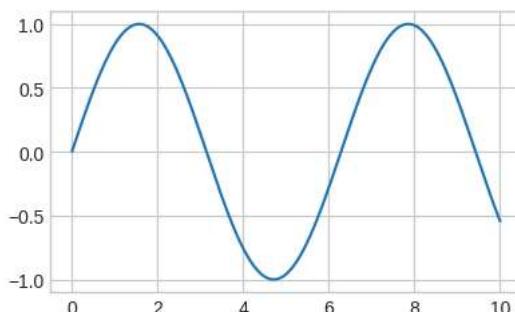
```
fig=plt.figure()
ax=plt.axes()
x=np.linspace(0,10,1000)
ax.plot(x,np.sin(x),label='sin',linestyle='dashed')
ax.plot(x,np.cos(x),label='cos',linestyle='dotted')
ax.plot(x,np.sin(x+1),label='tan',linestyle='dashdot')
ax.set_title('Multiple Lines')
ax.set_xlabel('x label')
ax.set_ylabel('y label')
ax.legend()
```

↳ <matplotlib.legend.Legend at 0x7be9aa870d90>



```
#SUBPLOTS
fig,axs=plt.subplots(2,2,figsize=(10,6))
x=np.linspace(0,10,1000)
axs[0,0].plot(x,np.sin(x))
axs[0,1].plot(x,np.cos(x),color='maroon')
axs[1,0].plot(x,np.sin(x+2),color='blue')
axs[1,1].plot(x,np.sin(x+4),color='green');
```

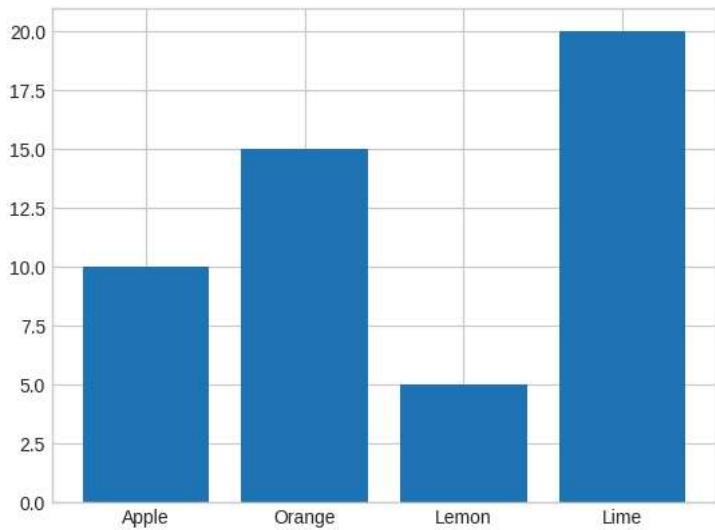
↳



## BAR CHART

```
data={'Apple':10,'Orange':15,'Lemon':5,'Lime':20}
names=list(data.keys())
values=list(data.values())
fig=plt.figure()
ax=plt.axes()
ax.bar(names,values)
```

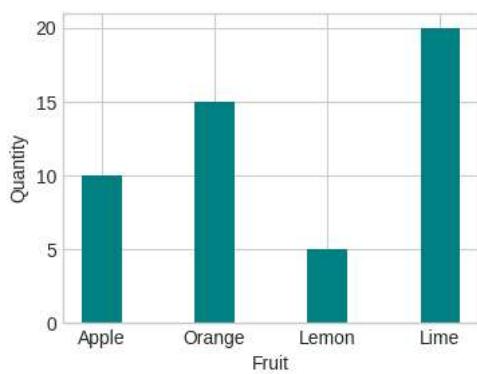
→ <BarContainer object of 4 artists>



## Detailing bar chart

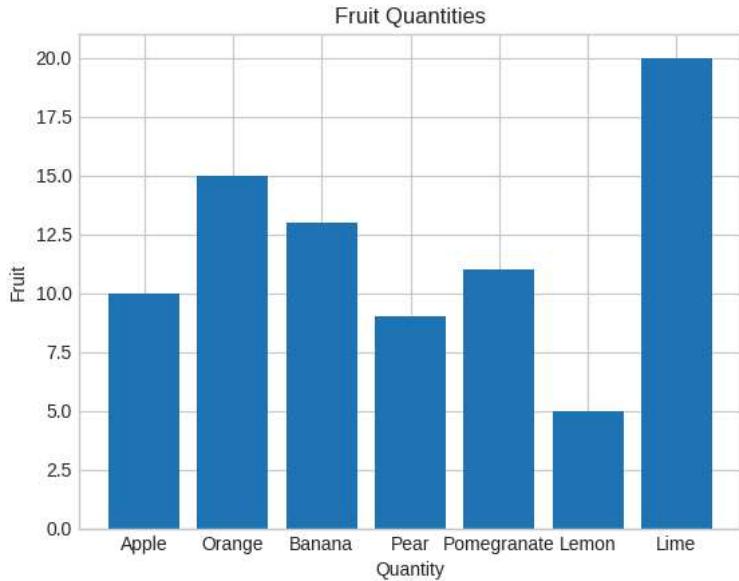
```
data={'Apple':10,'Orange':15,'Lemon':5,'Lime':20}
names=list(data.keys())
values=list(data.values())
fig=plt.figure(figsize=(4,3))
ax=plt.axes()
ax.bar(names,values,color='teal',width=0.35)
ax.set_xlabel('Fruit')
ax.set_ylabel('Quantity')
```

→ <Text(0, 0.5, 'Quantity')>



```
#creating a dictionary
data={'Apple':10,
      'Orange':15,
      'Banana':13,
      'Pear':9,
      'Pomegranate':11,
      'Lemon':5,
      'Lime':20}
names=list(data.keys())
values=list(data.values())
ax=plt.axes()
ax.bar(names,values)
ax.set_xlabel('Quantity')
ax.set_ylabel('Fruit')
ax.set_title('Fruit Quantities')
```

→ Text(0.5, 1.0, 'Fruit Quantities')



```
import pandas as pd
```

```
df = pd.read_csv('train.csv')
df
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Heikkinen, Miss. Laina	female	38.0	1	0	PC 17599 STON/O2. 3101282	71.2833 7.9250	C85 NaN	C S
2	3	1	3		female	26.0	0	0				
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...		...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W.C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C

```
df.dtypes
```

PassengerId	int64
Survived	int64
Pclass	int64
Name	object
Sex	object
Age	float64
SibSp	int64
Parch	int64
Ticket	object
Fare	float64
Cabin	object
Embarked	object
dtype:	object

```
df.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

```
df.isna().sum()
```

PassengerId	0
Survived	0
Pclass	0
Name	0
Sex	0
Age	177
SibSp	0
Parch	0
Ticket	0
Fare	0
Cabin	687

```
Embarked      2
dtype: int64
```

```
age_mean_value=df['Age'].mean()
df['Age']=df['Age'].fillna(age_mean_value)
```

```
df.drop("Cabin",axis=1,inplace=True)
```

```
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Th...	female	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S

```
filtered_age = df[df.Age>40]
filtered_age
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	S
15	16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55.0	0	0	248706	16.0000	S
33	34	0	2	Wheadon, Mr. Edward H	male	66.0	0	0	C.A. 24579	10.5000	S
35	36	0	1	Holverson, Mr. Alexander Oskar	male	42.0	1	0	113789	52.0000	S
...	...	...	...	...	...	...	...	...	...	...	...
862	863	1	1	Swift, Mrs. Frederick Joel (Margaret Welles Ba... Ba...	female	48.0	0	0	17466	25.9292	S
865	866	1	2	Bystrom, Mrs. (Karolina)	female	42.0	0	0	236852	13.0000	S
871	872	1	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female	47.0	1	1	11751	52.5542	S
873	874	0	3	Vander Cruyssen, Mr. Victor	male	47.0	0	0	345765	9.0000	S
879	880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0	1	11767	83.1583	C

150 rows × 11 columns

```
# let's sort the column Name in ascending order
sorted_passengers = df.sort_values('Name',ascending=True,kind ='heapsort')
```

```
sorted_passengers.head(10)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
845	846	0	3	Abbing, Mr. Anthony	male	42.0	0	0	C.A. 5547	7.5500	S
746	747	0	3	Abbott, Mr. Rossmore Edward	male	16.0	1	1	C.A. 2673	20.2500	S
279	280	1	3	Abbott, Mrs. Stanton (Rosa Hunt)	female	35.0	1	1	C.A. 2673	20.2500	S
308	309	0	2	Abelson, Mr. Samuel	male	30.0	1	0	P/PP 3381	24.0000	C
874	875	1	2	Abelson, Mrs. Samuel (Hannah Wizosky)	female	28.0	1	0	P/PP 3381	24.0000	C
365	366	0	3	Adahl, Mr. Mauritz Nils Martin	male	30.0	0	0	C 7076	7.2500	S
401	402	0	3	Adams, Mr. John	male	26.0	0	0	341826	8.0500	S
40	41	0	3	Ahlin, Mrs. Johan (Johanna Persdotter Larsson)	female	40.0	1	0	7546	9.4750	S
855	856	1	3	Aks, Mrs. Sam (Leah Rosen)	female	18.0	0	1	392091	9.3500	S
207	208	1	3	Albimona, Mr. Nassef Cassem	male	26.0	0	0	2699	18.7875	C

```
merged_df = pd.merge(df.head(2),df.tail(2),how='outer',indicator=True)  
merged_df
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	_merge
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S	left_only
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th... ...	female	38.0	1	0	PC 17599	71.2833	C	left_only
2	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C	right_only

```
group_df = df.groupby('Name')
```

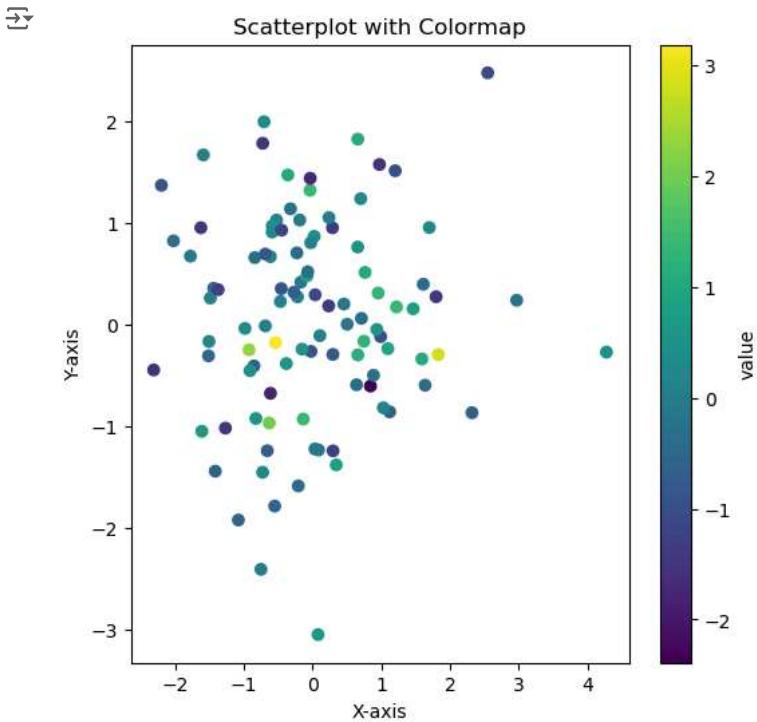
group\_df

→ <pandas.core.groupby.generic.DataFrameGroupBy object at 0x111f7ad50>

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# Sample dataframe with multiple columns
data = pd.DataFrame({
    "x": np.random.randn(100),
    "y": np.random.randn(100),
    "value": np.random.randn(100)
})
#Define the colormap and alpha values
cmap = "viridis"
alpha = 1
#Create the Scatterplot
plt.figure(figsize=(6,6))
plt.scatter(data["x"],data["y"], c=data["value"], cmap=cmap, alpha=alpha)
#Customize the plot (optional)
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Scatterplot with Colormap")
plt.colorbar(label="value")
#show the plot
plt.show()

```



```

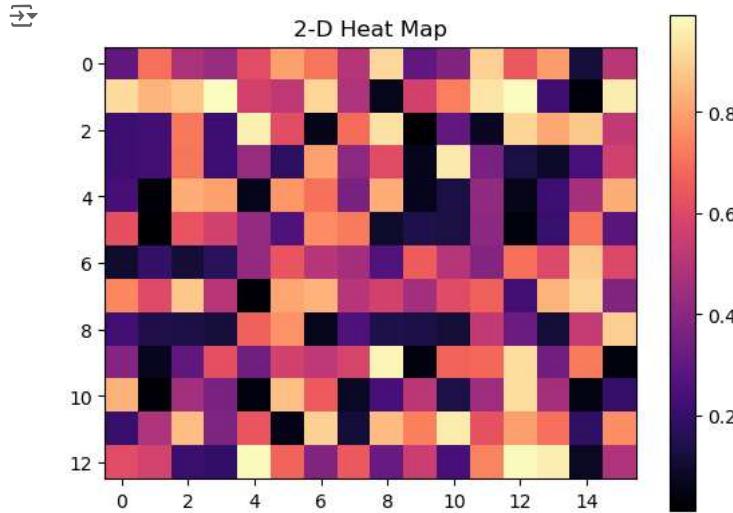
import pandas as pd
import numpy as np
print(np.random.randn(100))

[ 0.93603133 -1.63722979 -0.10017444 -0.60149469  0.39932534 -0.45703361
  0.25057015  1.16184784 -0.79708678  2.88347889 -0.14618649  0.337601
 -2.0345703   0.40240422 -0.77514298 -1.04586704 -0.28505791 -0.23513006
 -0.14975113  0.23885146  1.72175188  0.83628108  0.44163425  0.18695965
  0.80570507 -0.24603939  0.28883352 -0.68801307 -1.83107853 -0.5669822
 -0.1057094  -0.0189177  0.26834436  0.18755706 -0.84127166  0.52464179
 -1.14570934 -0.21483583  2.34927447  0.35593397  1.35393218 -1.24366076
  0.91521568 -1.14074783 -1.42731147  1.24097388  0.91417475  0.15797778
 -1.38603285  0.97211733 -0.35319504 -0.13251576  0.16290939  1.00081601
  0.26846481  0.05458016 -0.25696446 -2.3473615   1.22469537  0.41355171
 -0.10080791  1.09875899  0.95038585 -2.41815866 -0.79853071 -0.16964941
 -1.67448663 -0.40002767 -0.42075586  0.35686112  0.44291465  1.19193174
  0.53157364 -0.47701345  0.263171   -0.0443464  -0.07068025  0.2340464
 -0.28149185 -0.19582088 -0.69668824 -1.26743537 -0.16443496  0.47244768
  1.15742809  0.02413712  0.34102243  0.56784084  1.08765417  0.46394361
 -0.20817849 -0.47558734 -0.62140393  0.1354527   1.16532663 -0.88561804
 -0.71428299  1.21782237 -0.34087043  0.44290545]

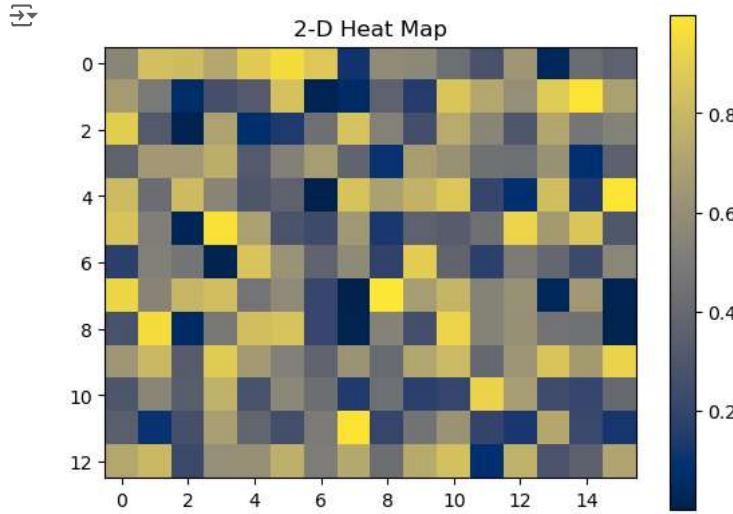
```

Start coding or generate with AI.

```
import numpy as np
import matplotlib.pyplot as plt
data = np.random.random((13,16))
plt.imshow(data,cmap="magma")
plt.title("2-D Heat Map")
plt.colorbar()
plt.show()
```



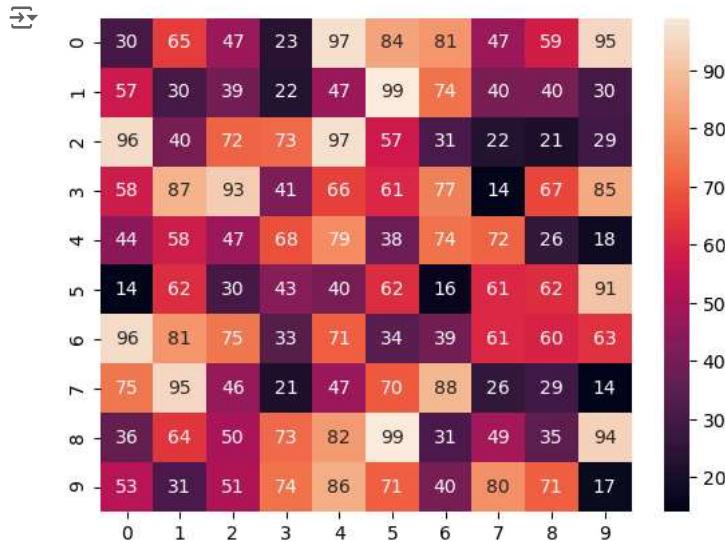
```
import numpy as np
import matplotlib.pyplot as plt
data = np.random.random((13,16))
plt.imshow(data,cmap="cividis")
plt.title("2-D Heat Map")
plt.colorbar()
plt.show()
```



```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

#generating 2-D 10x10 matrix of random numbers
data = np.random.randint(low=14,
                        high=100,
                        size=(10, 10))

#plotting heatmap
hm = sns.heatmap(data=data, annot=True)
plt.show()
```



Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

[Start coding](#) or [generate](#) with AI.

```
pip install seaborn
```

```
Requirement already satisfied: seaborn in c:\users\reddy\anaconda3\lib\site-packages (0.11.0)
Requirement already satisfied: pandas>=0.23 in c:\users\reddy\anaconda3\lib\site-packages (from seaborn) (1.4.4)
Requirement already satisfied: numpy>=1.15 in c:\users\reddy\anaconda3\lib\site-packages (from seaborn) (1.21.5)
Requirement already satisfied: matplotlib>=2.2 in c:\users\reddy\anaconda3\lib\site-packages (from seaborn) (3.5.2)
Requirement already satisfied: scipy>=1.0 in c:\users\reddy\anaconda3\lib\site-packages (from seaborn) (1.9.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\reddy\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn) (2.8.2)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\reddy\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn) (3.0.9)
Requirement already satisfied: packaging>=20.0 in c:\users\reddy\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn) (21.3)
Requirement already satisfied: pillow>=6.2.0 in c:\users\reddy\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn) (9.2.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\reddy\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn) (4.25.0)
Requirement already satisfied: cycler>=0.10 in c:\users\reddy\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\reddy\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn) (1.4.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\reddy\anaconda3\lib\site-packages (from pandas>=0.23->seaborn) (2022.1)
Requirement already satisfied: six>=1.5 in c:\users\reddy\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib>=2.2->seaborn) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

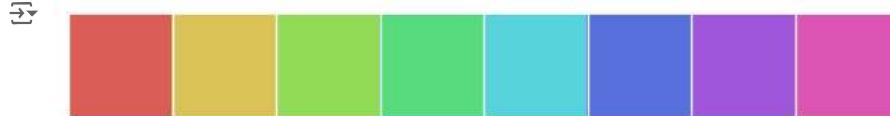
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
sns.set(rc={"figure.figsize": (6,6)})
```

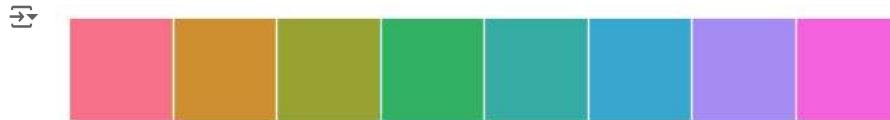
```
current_palette = sns.color_palette()
sns.palplot(current_palette)
```



```
sns.palplot(sns.color_palette("hls",8))
```



```
sns.palplot(sns.color_palette("husl",8))
```



```
sample_colors = ["windows blue", "amber", "greyish", "faded green", "dusty purple", "pale red", "medium green", "denim blue"]
sns.palplot(sns.xkcd_palette(sample_colors))
```



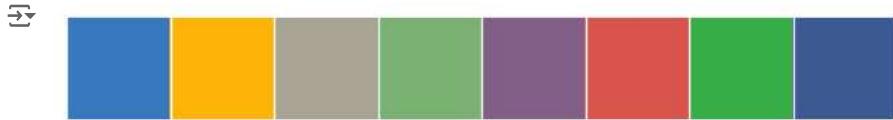
```
%pip install seaborn==0.11.0
```

```
Collecting seaborn==0.11.0
  Note: you may need to restart the kernel to use updated packages.
```

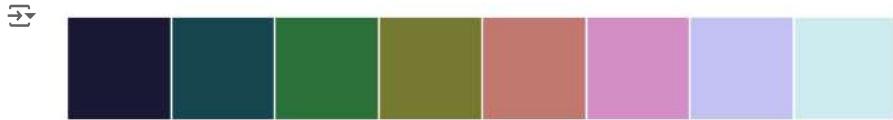
```
  Downloading seaborn-0.11.0-py3-none-any.whl (283 kB)
  283.1/283.1 kB 2.9 MB/s eta 0:00:00
Requirement already satisfied: pandas>=0.23 in c:\users\reddy\anaconda3\lib\site-packages (from seaborn==0.11.0) (1.4.4)
Requirement already satisfied: scipy>=1.0 in c:\users\reddy\anaconda3\lib\site-packages (from seaborn==0.11.0) (1.9.1)
Requirement already satisfied: numpy>=1.15 in c:\users\reddy\anaconda3\lib\site-packages (from seaborn==0.11.0) (1.21.5)
```

```
Requirement already satisfied: matplotlib>=2.2 in c:\users\reddy\anaconda3\lib\site-packages (from seaborn==0.11.0) (3.5.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\reddy\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn==0.11.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\reddy\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn==0.11.0) (9.2)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\reddy\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn==0.11.0) (4.22.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\reddy\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn==0.11.0) (1.0.1)
Requirement already satisfied: cycler>=0.10 in c:\users\reddy\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn==0.11.0) (0.11.0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\reddy\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn==0.11.0) (3.1.2)
Requirement already satisfied: packaging>=20.0 in c:\users\reddy\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn==0.11.0) (21.0)
Requirement already satisfied: pytz>=2020.1 in c:\users\reddy\anaconda3\lib\site-packages (from pandas>=0.23->seaborn==0.11.0) (2022.1)
Requirement already satisfied: six>=1.5 in c:\users\reddy\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib>=2.2->seaborn==0.11.0)
Installing collected packages: seaborn
  Attempting uninstall: seaborn
    Found existing installation: seaborn 0.11.2
    Uninstalling seaborn-0.11.2:
      Successfully uninstalled seaborn-0.11.2
Successfully installed seaborn-0.11.0
```

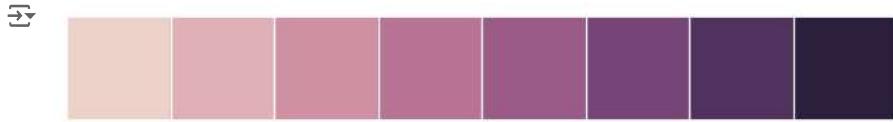
```
sample_colors = ["windows blue", "amber", "greyish", "faded green", "dusty purple", "pale red", "medium green", "denim blue"]
sns.palplot(sns.xkcd_palette(sample_colors))
```



```
# Default matplotlib Cubehelix version:
sns.palplot(sns.color_palette("cubehelix", 8))
```



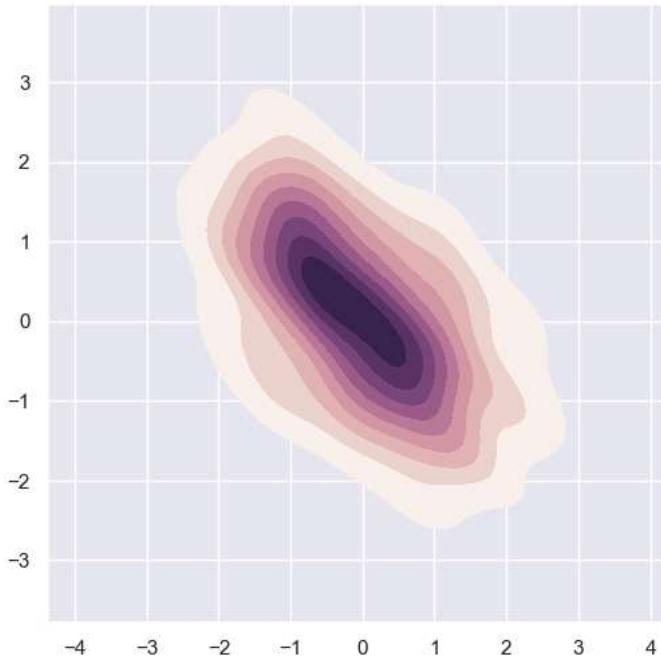
```
# Default Seaborn Cubehelix version:
sns.palplot(sns.cubehelix_palette(8))
```



```
x, y = np.random.multivariate_normal([0, 0], [[1, -.5], [-.5, 1]], size = 300).T

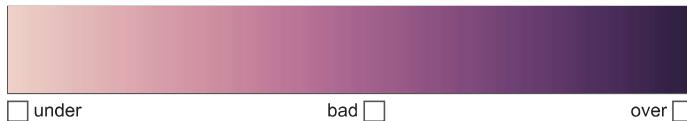
sample_cmap = sns.cubehelix_palette(light=1, as_cmap=True)
sns.kdeplot(x=x, y=y, cmap=sample_cmap, shade=True)
```

↳ <AxesSubplot:>



```
sns.choose_cubehelix_palette(as_cmap=True)
```

↳ interactive(children=(IntSlider(value=9, description='n\_colors', max=16, min=2), FloatSlider(value=0.0, description='light', max=1.0, min=0.0), ColorPicker(value=(255, 0, 0), description='color'), RadioButtons(options=['under', 'bad', 'over'], value='under', description='order')), disabled=False)



```
sns.palplot(sns.cubehelix_palette(n_colors=8, start=1.7, rot=0.2, dark=0, light=.95, reverse=True))
```



```
# Loading up built-in dataset:
tips = sns.load_dataset("C:/Users/reddy/OneDrive/Desktop/tips.csv")
```

```
# Creating Strip plot for day-wise revenue:
sns.stripplot(x="day", y="total_bill", data=tips, color="g")
```

↳

```

ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_2320\3471772796.py in <module>
      1 # Loading up built-in dataset:
----> 2 tips = sns.load_dataset("C:/Users/reddy/OneDrive/Desktop/tips.csv")
      3
      4 # Creating Strip plot for day-wise revenue:
      5 sns.stripplot(x="day", y="total_bill", data=tips, color="g")

~\anaconda3\lib\site-packages\seaborn\utils.py in load_dataset(name, cache, data_home, **kws)
    484     if not os.path.exists(cache_path):
    485         if name not in get_dataset_names():
--> 486             raise ValueError(f"'{name}' is not one of the example datasets.")
    487         urlretrieve(full_path, cache_path)
    488         full_path = cache_path
```

ValueError: 'C:/Users/reddy/OneDrive/Desktop/tips.csv' is not one of the example datasets.

```
# Set Theme:  
sns.set_style('whitegrid')  
  
# Creating Strip plot for day-wise revenue:  
sns.swarmplot(x="day", y="total_bill", data=tips, palette="viridis")  
  
NameError  
Traceback (most recent call last)  
~\AppData\Local\Temp\ipykernel_2320\3511107052.py in <module>  
      1  
      2  
      3  
      4  
      5  
      6  
      7  
      8  
      9  
     10  
     11  
     12  
     13  
     14  
     15  
     16  
     17  
     18  
     19  
     20  
     21  
     22  
     23  
     24  
     25  
     26  
     27  
     28  
     29  
     30  
     31  
     32  
     33  
     34  
     35  
     36  
     37  
     38  
     39  
     40  
     41  
     42  
     43  
     44  
     45  
     46  
     47  
     48  
     49  
     50  
     51  
     52  
     53  
     54  
     55  
     56  
     57  
     58  
     59  
     60  
     61  
     62  
     63  
     64  
     65  
     66  
     67  
     68  
     69  
     70  
     71  
     72  
     73  
     74  
     75  
     76  
     77  
     78  
     79  
     80  
     81  
     82  
     83  
     84  
     85  
     86  
     87  
     88  
     89  
     90  
     91  
     92  
     93  
     94  
     95  
     96  
     97  
     98  
     99  
     100  
     101  
     102  
     103  
     104  
     105  
     106  
     107  
     108  
     109  
     110  
     111  
     112  
     113  
     114  
     115  
     116  
     117  
     118  
     119  
     120  
     121  
     122  
     123  
     124  
     125  
     126  
     127  
     128  
     129  
     130  
     131  
     132  
     133  
     134  
     135  
     136  
     137  
     138  
     139  
     140  
     141  
     142  
     143  
     144  
     145  
     146  
     147  
     148  
     149  
     150  
     151  
     152  
     153  
     154  
     155  
     156  
     157  
     158  
     159  
     160  
     161  
     162  
     163  
     164  
     165  
     166  
     167  
     168  
     169  
     170  
     171  
     172  
     173  
     174  
     175  
     176  
     177  
     178  
     179  
     180  
     181  
     182  
     183  
     184  
     185  
     186  
     187  
     188  
     189  
     190  
     191  
     192  
     193  
     194  
     195  
     196  
     197  
     198  
     199  
     200  
     201  
     202  
     203  
     204  
     205  
     206  
     207  
     208  
     209  
     210  
     211  
     212  
     213  
     214  
     215  
     216  
     217  
     218  
     219  
     220  
     221  
     222  
     223  
     224  
     225  
     226  
     227  
     228  
     229  
     230  
     231  
     232  
     233  
     234  
     235  
     236  
     237  
     238  
     239  
     240  
     241  
     242  
     243  
     244  
     245  
     246  
     247  
     248  
     249  
     250  
     251  
     252  
     253  
     254  
     255  
     256  
     257  
     258  
     259  
     260  
     261  
     262  
     263  
     264  
     265  
     266  
     267  
     268  
     269  
     270  
     271  
     272  
     273  
     274  
     275  
     276  
     277  
     278  
     279  
     280  
     281  
     282  
     283  
     284  
     285  
     286  
     287  
     288  
     289  
     290  
     291  
     292  
     293  
     294  
     295  
     296  
     297  
     298  
     299  
     300  
     301  
     302  
     303  
     304  
     305  
     306  
     307  
     308  
     309  
     310  
     311  
     312  
     313  
     314  
     315  
     316  
     317  
     318  
     319  
     320  
     321  
     322  
     323  
     324  
     325  
     326  
     327  
     328  
     329  
     330  
     331  
     332  
     333  
     334  
     335  
     336  
     337  
     338  
     339  
     340  
     341  
     342  
     343  
     344  
     345  
     346  
     347  
     348  
     349  
     350  
     351  
     352  
     353  
     354  
     355  
     356  
     357  
     358  
     359  
     360  
     361  
     362  
     363  
     364  
     365  
     366  
     367  
     368  
     369  
     370  
     371  
     372  
     373  
     374  
     375  
     376  
     377  
     378  
     379  
     380  
     381  
     382  
     383  
     384  
     385  
     386  
     387  
     388  
     389  
     390  
     391  
     392  
     393  
     394  
     395  
     396  
     397  
     398  
     399  
     400  
     401  
     402  
     403  
     404  
     405  
     406  
     407  
     408  
     409  
     410  
     411  
     412  
     413  
     414  
     415  
     416  
     417  
     418  
     419  
     420  
     421  
     422  
     423  
     424  
     425  
     426  
     427  
     428  
     429  
     430  
     431  
     432  
     433  
     434  
     435  
     436  
     437  
     438  
     439  
     440  
     441  
     442  
     443  
     444  
     445  
     446  
     447  
     448  
     449  
     450  
     451  
     452  
     453  
     454  
     455  
     456  
     457  
     458  
     459  
     460  
     461  
     462  
     463  
     464  
     465  
     466  
     467  
     468  
     469  
     470  
     471  
     472  
     473  
     474  
     475  
     476  
     477  
     478  
     479  
     480  
     481  
     482  
     483  
     484  
     485  
     486  
     487  
     488  
     489  
     490  
     491  
     492  
     493  
     494  
     495  
     496  
     497  
     498  
     499  
     500  
     501  
     502  
     503  
     504  
     505  
     506  
     507  
     508  
     509  
     510  
     511  
     512  
     513  
     514  
     515  
     516  
     517  
     518  
     519  
     520  
     521  
     522  
     523  
     524  
     525  
     526  
     527  
     528  
     529  
     530  
     531  
     532  
     533  
     534  
     535  
     536  
     537  
     538  
     539  
     540  
     541  
     542  
     543  
     544  
     545  
     546  
     547  
     548  
     549  
     550  
     551  
     552  
     553  
     554  
     555  
     556  
     557  
     558  
     559  
     560  
     561  
     562  
     563  
     564  
     565  
     566  
     567  
     568  
     569  
     570  
     571  
     572  
     573  
     574  
     575  
     576  
     577  
     578  
     579  
     580  
     581  
     582  
     583  
     584  
     585  
     586  
     587  
     588  
     589  
     590  
     591  
     592  
     593  
     594  
     595  
     596  
     597  
     598  
     599  
     600  
     601  
     602  
     603  
     604  
     605  
     606  
     607  
     608  
     609  
     610  
     611  
     612  
     613  
     614  
     615  
     616  
     617  
     618  
     619  
     620  
     621  
     622  
     623  
     624  
     625  
     626  
     627  
     628  
     629  
     630  
     631  
     632  
     633  
     634  
     635  
     636  
     637  
     638  
     639  
     640  
     641  
     642  
     643  
     644  
     645  
     646  
     647  
     648  
     649  
     650  
     651  
     652  
     653  
     654  
     655  
     656  
     657  
     658  
     659  
     660  
     661  
     662  
     663  
     664  
     665  
     666  
     667  
     668  
     669  
     670  
     671  
     672  
     673  
     674  
     675  
     676  
     677  
     678  
     679  
     680  
     681  
     682  
     683  
     684  
     685  
     686  
     687  
     688  
     689  
     690  
     691  
     692  
     693  
     694  
     695  
     696  
     697  
     698  
     699  
     700  
     701  
     702  
     703  
     704  
     705  
     706  
     707  
     708  
     709  
     710  
     711  
     712  
     713  
     714  
     715  
     716  
     717  
     718  
     719  
     720  
     721  
     722  
     723  
     724  
     725  
     726  
     727  
     728  
     729  
     730  
     731  
     732  
     733  
     734  
     735  
     736  
     737  
     738  
     739  
     740  
     741  
     742  
     743  
     744  
     745  
     746  
     747  
     748  
     749  
     750  
     751  
     752  
     753  
     754  
     755  
     756  
     757  
     758  
     759  
     760  
     761  
     762  
     763  
     764  
     765  
     766  
     767  
     768  
     769  
     770  
     771  
     772  
     773  
     774  
     775  
     776  
     777  
     778  
     779  
     780  
     781  
     782  
     783  
     784  
     785  
     786  
     787  
     788  
     789  
     790  
     791  
     792  
     793  
     794  
     795  
     796  
     797  
     798  
     799  
     800  
     801  
     802  
     803  
     804  
     805  
     806  
     807  
     808  
     809  
     810  
     811  
     812  
     813  
     814  
     815  
     816  
     817  
     818  
     819  
     820  
     821  
     822  
     823  
     824  
     825  
     826  
     827  
     828  
     829  
     830  
     831  
     832  
     833  
     834  
     835  
     836  
     837  
     838  
     839  
     840  
     841  
     842  
     843  
     844  
     845  
     846  
     847  
     848  
     849  
     850  
     851  
     852  
     853  
     854  
     855  
     856  
     857  
     858  
     859  
     860  
     861  
     862  
     863  
     864  
     865  
     866  
     867  
     868  
     869  
     870  
     871  
     872  
     873  
     874  
     875  
     876  
     877  
     878  
     879  
     880  
     881  
     882  
     883  
     884  
     885  
     886  
     887  
     888  
     889  
     890  
     891  
     892  
     893  
     894  
     895  
     896  
     897  
     898  
     899  
     900  
     901  
     902  
     903  
     904  
     905  
     906  
     907  
     908  
     909  
     910  
     911  
     912  
     913  
     914  
     915  
     916  
     917  
     918  
     919  
     920  
     921  
     922  
     923  
     924  
     925  
     926  
     927  
     928  
     929  
     930  
     931  
     932  
     933  
     934  
     935  
     936  
     937  
     938  
     939  
     940  
     941  
     942  
     943  
     944  
     945  
     946  
     947  
     948  
     949  
     950  
     951  
     952  
     953  
     954  
     955  
     956  
     957  
     958  
     959  
     960  
     961  
     962  
     963  
     964  
     965  
     966  
     967  
     968  
     969  
     970  
     971  
     972  
     973  
     974  
     975  
     976  
     977  
     978  
     979  
     980  
     981  
     982  
     983  
     984  
     985  
     986  
     987  
     988  
     989  
     990  
     991  
     992  
     993  
     994  
     995  
     996  
     997  
     998  
     999  
     1000  
     1001  
     1002  
     1003  
     1004  
     1005  
     1006  
     1007  
     1008  
     1009  
     10010  
     10011  
     10012  
     10013  
     10014  
     10015  
     10016  
     10017  
     10018  
     10019  
     10020  
     10021  
     10022  
     10023  
     10024  
     10025  
     10026  
     10027  
     10028  
     10029  
     10030  
     10031  
     10032  
     10033  
     10034  
     10035  
     10036  
     10037  
     10038  
     10039  
     10040  
     10041  
     10042  
     10043  
     10044  
     10045  
     10046  
     10047  
     10048  
     10049  
     10050  
     10051  
     10052  
     10053  
     10054  
     10055  
     10056  
     10057  
     10058  
     10059  
     10060  
     10061  
     10062  
     10063  
     10064  
     10065  
     10066  
     10067  
     10068  
     10069  
     10070  
     10071  
     10072  
     10073  
     10074  
     10075  
     10076  
     10077  
     10078  
     10079  
     10080  
     10081  
     10082  
     10083  
     10084  
     10085  
     10086  
     10087  
     10088  
     10089  
     10090  
     10091  
     10092  
     10093  
     10094  
     10095  
     10096  
     10097  
     10098  
     10099  
     100100  
     100101  
     100102  
     100103  
     100104  
     100105  
     100106  
     100107  
     100108  
     100109  
     100110  
     100111  
     100112  
     100113  
     100114  
     100115  
     100116  
     100117  
     100118  
     100119  
     100120  
     100121  
     100122  
     100123  
     100124  
     100125  
     100126  
     100127  
     100128  
     100129  
     100130  
     100131  
     100132  
     100133  
     100134  
     100135  
     100136  
     100137  
     100138  
     100139  
     100140  
     100141  
     100142  
     100143  
     100144  
     100145  
     100146  
     100147  
     100148  
     100149  
     100150  
     100151  
     100152  
     100153  
     100154  
     100155  
     100156  
     100157  
     100158  
     100159  
     100160  
     100161  
     100162  
     100163  
     100164  
     100165  
     100166  
     100167  
     100168  
     100169  
     100170  
     100171  
     100172  
     100173  
     100174  
     100175  
     100176  
     100177  
     100178  
     100179  
     100180  
     100181  
     100182  
     100183  
     100184  
     100185  
     100186  
     100187  
     100188  
     100189  
     100190  
     100191  
     100192  
     100193  
     100194  
     100195  
     100196  
     100197  
     100198  
     100199  
     100200  
     100201  
     100202  
     100203  
     100204  
     100205  
     100206  
     100207  
     100208  
     100209  
     100210  
     100211  
     100212  
     100213  
     100214  
     100215  
     100216  
     100217  
     100218  
     100219  
     100220  
     100221  
     100222  
     100223  
     100224  
     100225  
     100226  
     100227  
     100228  
     100229  
     100230  
     100231  
     100232  
     100233  
     100234  
     100235  
     100236  
     100237  
     100238  
     100239  
     100240  
     100241  
     100242  
     100243  
     100244  
     100245  
     100246  
     100247  
     100248  
     100249  
     100250  
     100251  
     100252  
     100253  
     100254  
     100255  
     100256  
     100257  
     100258  
     100259  
     100260  
     100261  
     100262  
     100263  
     100264  
     100265  
     100266  
     100267  
     100268  
     100269  
     100270  
     100271  
     100272  
     100273  
     100274  
     100275  
     100276  
     100277  
     100278  
     100279  
     100280  
     100281  
     100282  
     100283  
     100284  
     100285  
     100286  
     100287  
     100288  
     100289  
     100290  
     100291  
     100292  
     100293  
     100294  
     100295  
     100296  
     100297  
     100298  
     100299  
     100300  
     100301  
     100302  
     100303  
     100304  
     100305  
     100306  
     100307  
     100308  
     100309  
     100310  
     100311  
     100312  
     100313  
     100314  
     100315  
     100316  
     100317  
     100318  
     100319  
     100320  
     100321  
     100322  
     100323  
     100324  
     100325  
     100326  
     100327  
     100328  
     100329  
     100330  
     100331  
     100332  
     100333  
     100334  
     100335  
     100336  
     100337  
     100338  
     100339  
     100340  
     100341  
     100342  
     100343  
     100344  
     100345  
     100346  
     100347  
     100348  
     100349  
     100350  
     100351  
     100352  
     100353  
     100354  
     100355  
     100356  
     100357  
     100358  
     100359  
     100360  
     100361  
     100362  
     100363  
     100364  
     100365  
     100366  
     100367  
     100368  
     100369  
     100370  
     100371  
     100372  
     100373  
     100374  
     100375  
     100376  
     100377  
     100378  
     100379  
     100380  
     100381  
     100382  
     100383  
     100384  
     100385  
     100386  
     100387  
     100388  
     100389  
     100390  
     100391  
     100392  
     100393  
     100394  
     100395  
     100396  
     100397  
     100398  
     100399  
     100400  
     100401  
     100402  
     100403  
     100404  
     100405  
     100406  
     100407  
     100408  
     100409  
     100410  
     100411  
     100412  
     100413  
     100414  
     100415  
     100416  
     100417  
     100418  
     100419  
     100420  
     100421  
     100422  
     100423  
     100424  
     100425  
     100426  
     100427  
     100428  
     100429  
     100430  
     100431  
     100432  
     100433  
     100434  
     100435  
     100436  
     100437  
     100438  
     100439<br
```

## PLOT CATEGORIES IN SEABORN

- 1.Relational plot : This plot is used to understand the relation between two variables
- 2.Categorical plot : This plot deals with categorical variables and how they can be visualized
- 3.Distribution plot : This plot is used for examining
- 4.Matrix plot
- 5.Regression plot

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure

import seaborn as sns
%matplotlib inline

dates = ['1981-01-01', '1981-01-02', '1981-01-03', '1981-01-04', '1981-01-05', '1981-01-06', '1981-01-07', '1981-01-08', '1981-01-09', '1981-01-10']
min_temperature=[20,7,17.9,10.0,14.6,15.8,15.8,17.4,21.8,20.0]
max_temperature=[34.7,28.9,31.8,25.6,28.8,21.8,22.8,28.4,30.8,32.0]

fig,axes = plt.subplots(nrows = 1, ncols = 1, figsize = (15,10))
axes.plot(dates, min_temperature, label = 'Min Temperature')
axes.plot(dates, max_temperature, label = 'Max Temperature')
axes.legend()
```

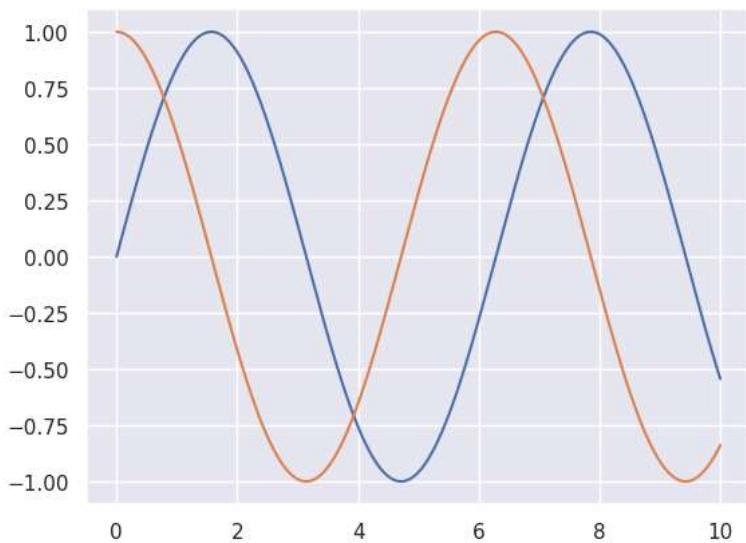
→ <matplotlib.legend.Legend at 0x7fda4ffde560>



sns.set()

```
x = np.linspace(0,10,1000)
plt.plot(x, np.sin(x), x, np.cos(x))

[<matplotlib.lines.Line2D at 0x7fd44fd8a2f0>,
 <matplotlib.lines.Line2D at 0x7fd44fd8a350>]
```



```
sns.set(style='dark')
fig,ax = plt.subplots(nrows = 1, ncols = 2, figsize = (15,10))
df = sns.load_dataset("tips")

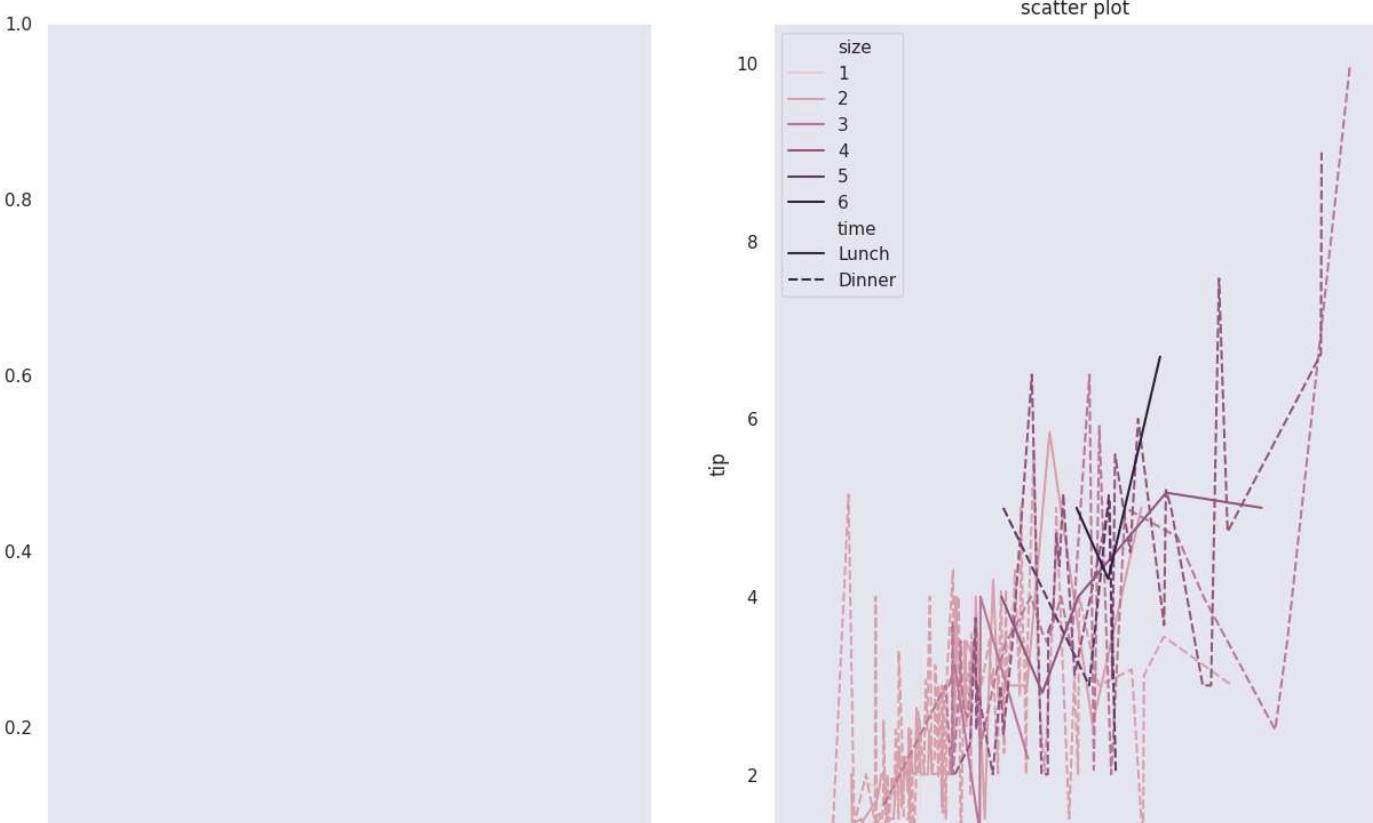
print(df.head())

sns.lineplot(x='total_bill', y='tip', hue='size', style='time', data=df, ax=ax[1]).set_title('scatter plot')

[<ipython.core.debugger.Debugger at 0x7fd44fd8a2f0>]
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

Text(0.5, 1.0, 'scatter plot')



```

sns.set(style='darkgrid')
fig,ax = plt.subplots(nrows = 5, ncols = 2)
fig.set_size_inches(18.5, 10.5)
df = sns.load_dataset("tips")

sns.barplot(x='tip', y='total_bill', data = df, palette = 'plasma', estimator = np.std, ax=ax[0,0]).set_title('Bar Plot')
sns.countplot(x ='sex', data=df, ax=ax[0,1]).set_title('Bar Plot')
sns.boxplot(x = 'day', y = 'total_bill', data = df, hue = 'smoker', ax=ax[1,0]).set_title('Box plot')
sns.violinplot(x = 'day', y = 'total_bill', data = df, hue = 'sex', split = True, ax=ax[1,1]).set_title('violin plot')
sns.stripplot(x = 'day', y = 'total_bill', data = df, hue = 'smoker', jitter = True, dodge = True, ax=ax[2,0]).set_title('strip plot')
sns.swarmplot(x = 'day', y = 'total_bill', data = df, ax=ax[2,1]).set_title('swarm plot')

sns.violinplot(x = 'day', y = 'total_bill', data = df, ax=ax[3,0])
sns.swarmplot(x = 'day', y = 'total_bill', color = 'black', data = df, ax=ax[3,0]).set_title('Combined plot')

sns.scatterplot(x = 'day', y = 'total_bill', data = df, ax=ax[3,1]).set_title("Scatter Plot: Tip vs Total Bill")

sns.boxenplot(x = 'day', y = 'total_bill', data = df, color = 'b', scale='linear', ax=ax[4,0])

sns.pointplot(x = 'day', y = 'total_bill', data = df, hue = 'sex', color = 'b', ax=ax[4,1])

```

→ <ipython-input-15-de4fcc9757c1>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend`

```

sns.barplot(x='tip', y='total_bill', data = df, palette = 'plasma', estimator = np.std, ax=ax[0,0]).set_title('Bar Plot')
<ipython-input-15-de4fcc9757c1>:18: FutureWarning:

```

The `scale` parameter has been renamed to `width\_method` and will be removed in v0.15. Pass `width\_method='linear'` for the same effect.

```

sns.boxenplot(x = 'day', y = 'total_bill', data = df, color = 'b', scale='linear', ax=ax[4,0])
<ipython-input-15-de4fcc9757c1>:20: FutureWarning:

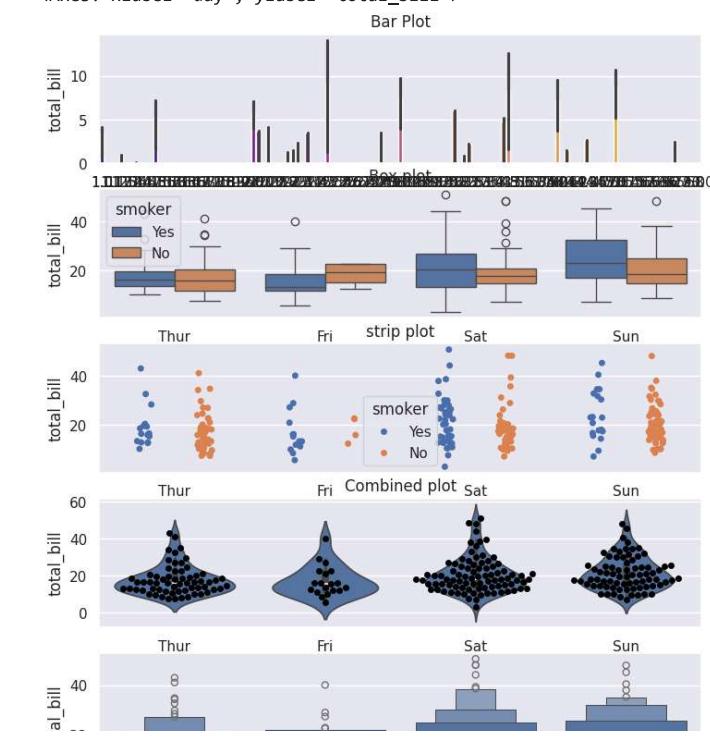
```

Setting a gradient palette using color= is deprecated and will be removed in v0.14.0. Set `palette='dark:b'` for the same effect.

```

sns.pointplot(x = 'day', y = 'total_bill', data = df, hue = 'sex', color = 'b', ax=ax[4,1])
<Axes: xlabel='day', ylabel='total_bill'>

```

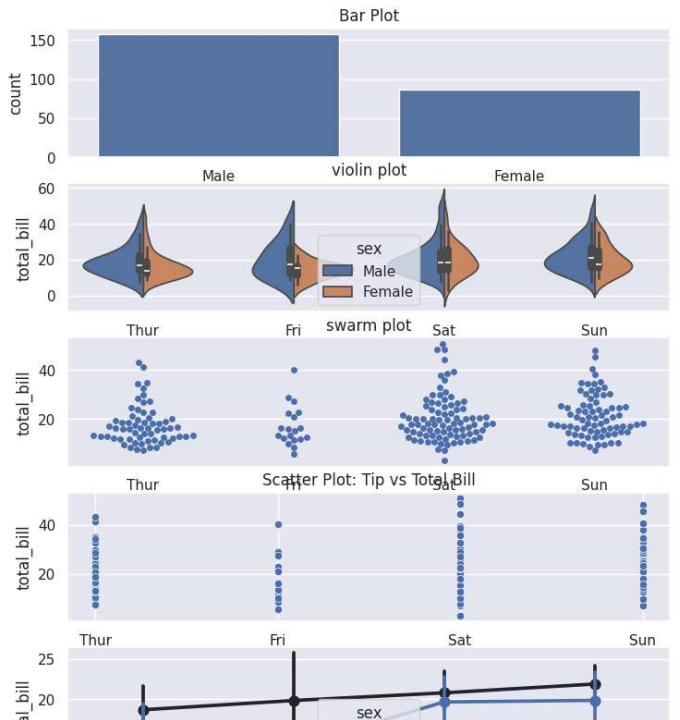


#Distribution Plots

```

joinplot
distplot
pairplot
rugplot

```



```

sns.set_style('whitegrid')

df = sns.load_dataset('iris')
print(df.head())

```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
sns.distplot(df['petal_length'], kde = True, color = 'red', bins = 30).set_title('Dist Plot')
```

`<ipython-input-17-6691fe3df3d8>:1: UserWarning:`

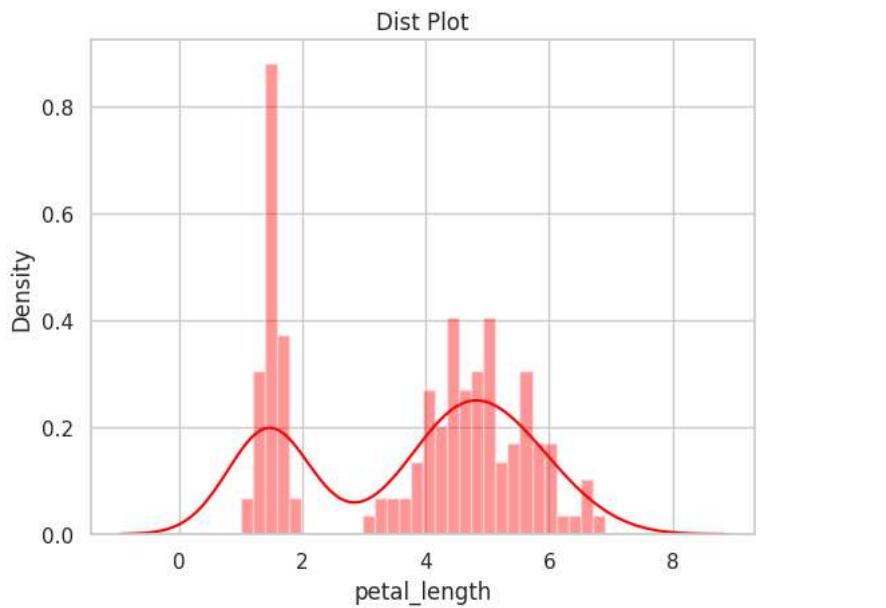
``distplot` is a deprecated function and will be removed in seaborn v0.14.0.`

`Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).`

For a guide to updating your code to use the new functions, please see

<https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(df['petal_length'], kde = True, color = 'red', bins = 30).set_title('Dist Plot')
Text(0.5, 1.0, 'Dist Plot')
```



```

jointgrid = sns.JointGrid(x = 'petal_length', y = 'petal_width', data = df)
jointgrid.plot_joint(sns.scatterplot)
jointgrid.plot_marginals(sns.distplot)

```

```
→ /usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:1886: UserWarning:
```

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see  
<https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

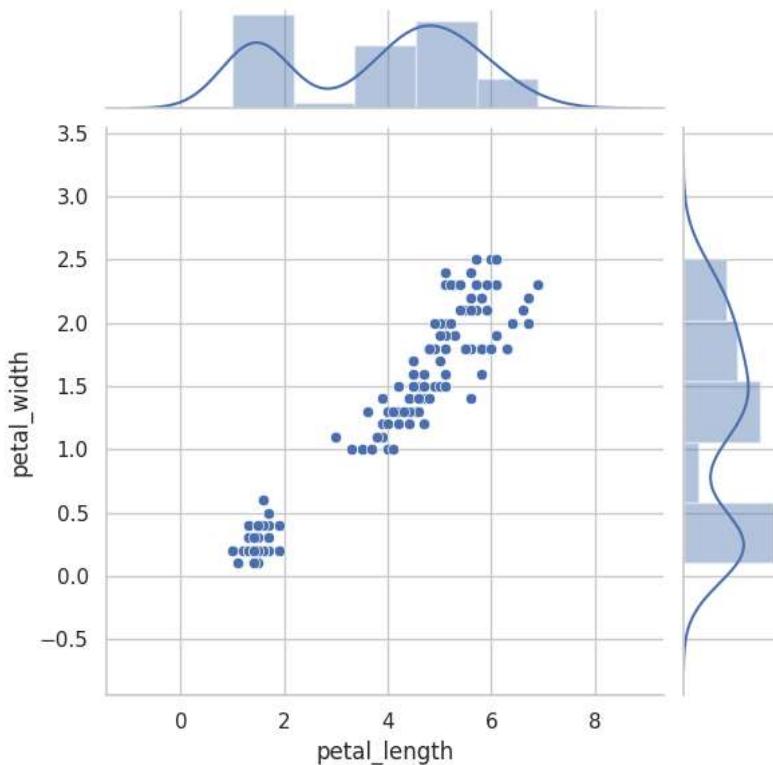
```
func(self.x, **orient_kw_x, **kwargs)
/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:1892: UserWarning:
```

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see  
<https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

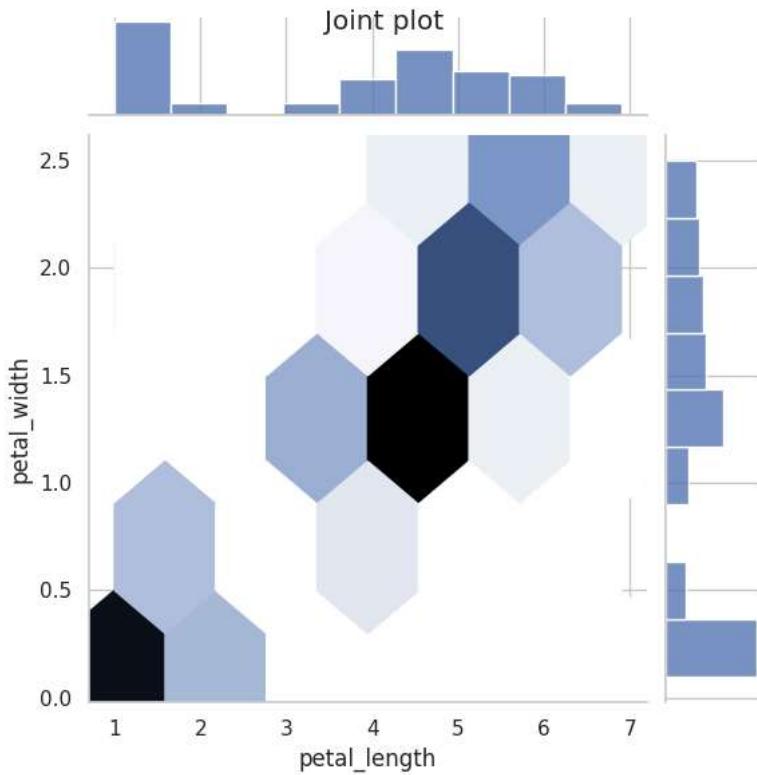
```
func(self.y, **orient_kw_y, **kwargs)
<seaborn.axisgrid.JointGrid at 0x7fd3d58bc10>
```



Start coding or [generate](#) with AI.

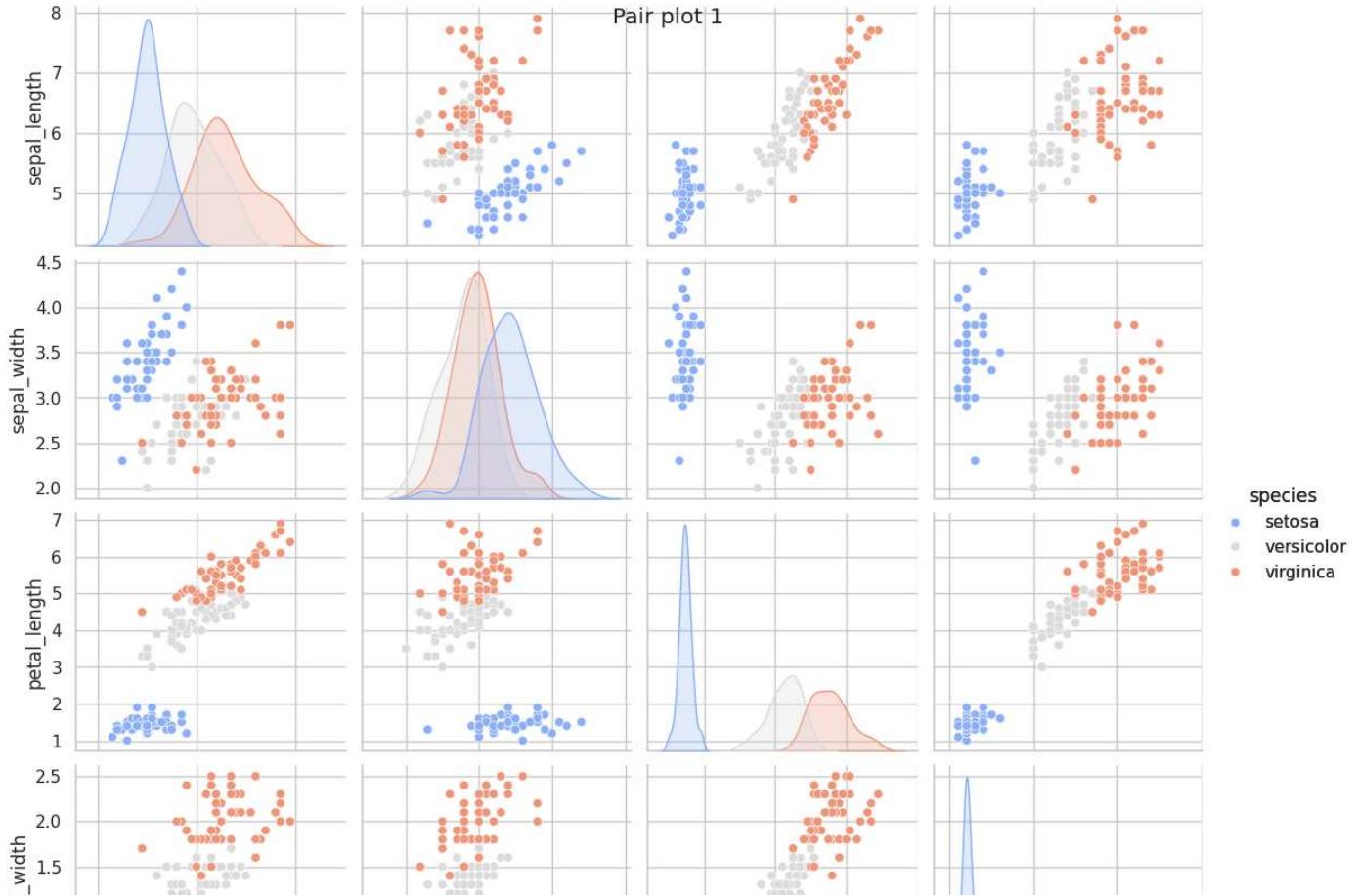
```
g = sns.jointplot(x = 'petal_length', y = 'petal_width', data = df, kind = 'hex')
g.fig.suptitle('Joint plot')
```

Joint plot



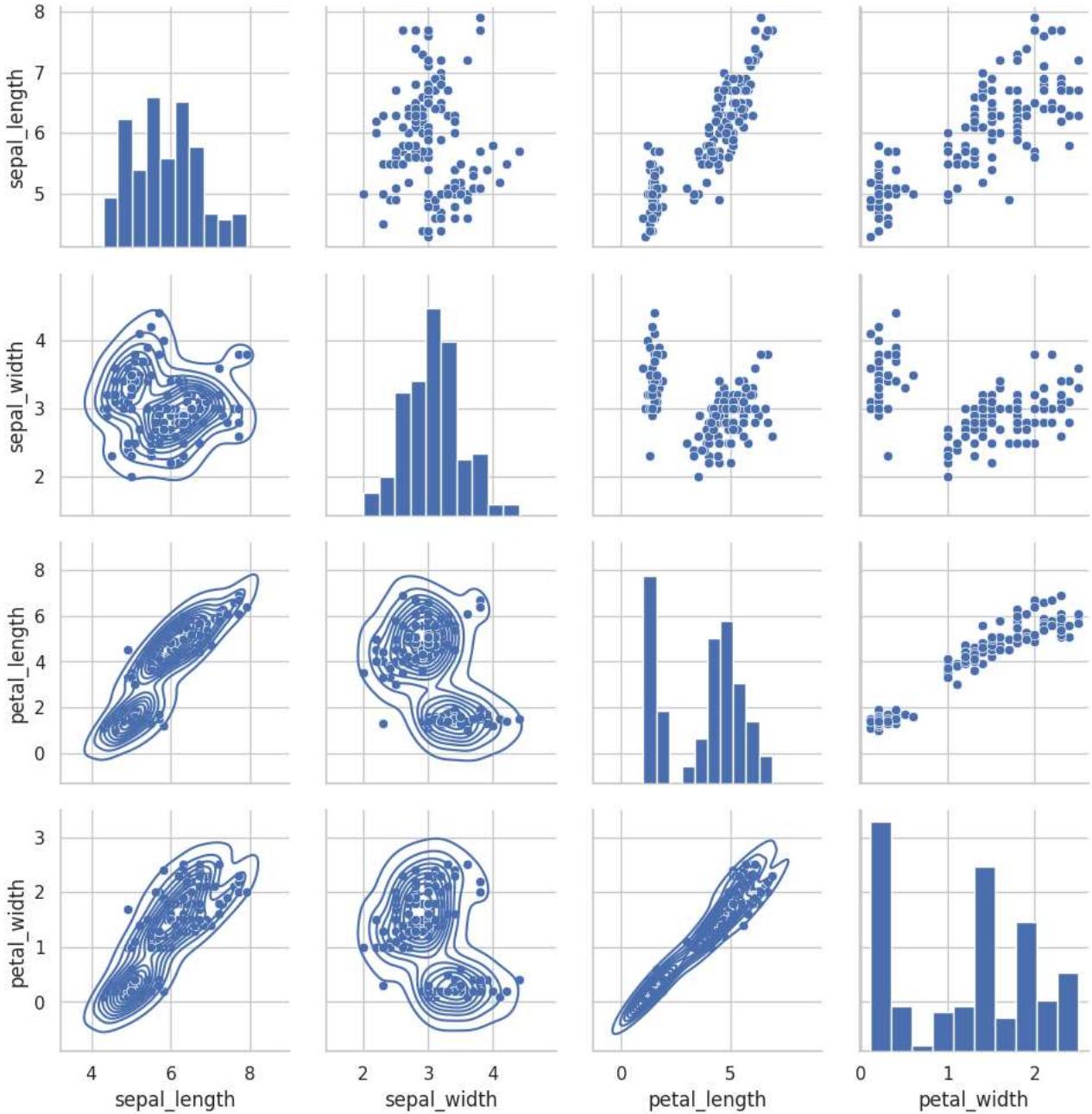
```
g = sns.pairplot(df,hue = "species", palette = "coolwarm")
g.fig.suptitle("Pair plot 1")
g.add_legend()
```

Pair plot 1

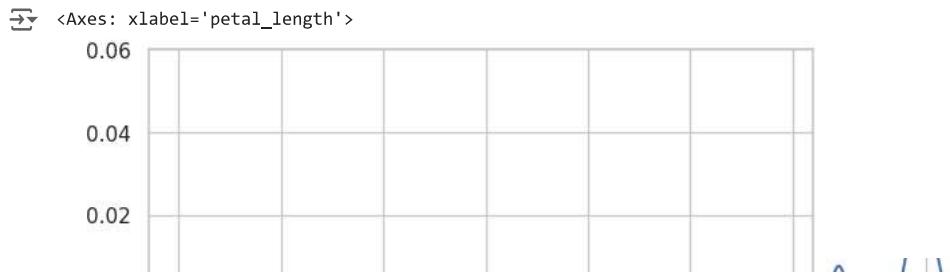


```
pairgrid = sns.PairGrid(data = df)
pairgrid = pairgrid.map_offdiag(sns.scatterplot)
pairgrid = pairgrid.map_diag(plt.hist)
pairgrid = pairgrid.map_lower(sns.kdeplot)
g = sns.PairGrid(df, diag_sharey = False, corner = True)
g.map_lower(sns.scatterplot)
g.map_diag(sns.kdeplot)
```

↳ <seaborn.axisgrid.PairGrid at 0x7fd3bc50be0>



```
sns.rugplot(df['petal_length'])
```



#Matrix Plots

```
fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(15,10))

df1 = sns.load_dataset('flights')
df2 = sns.load_dataset('iris')

df11 = pd.pivot_table(values = 'passengers', index = 'month', columns = 'year', data = df1)

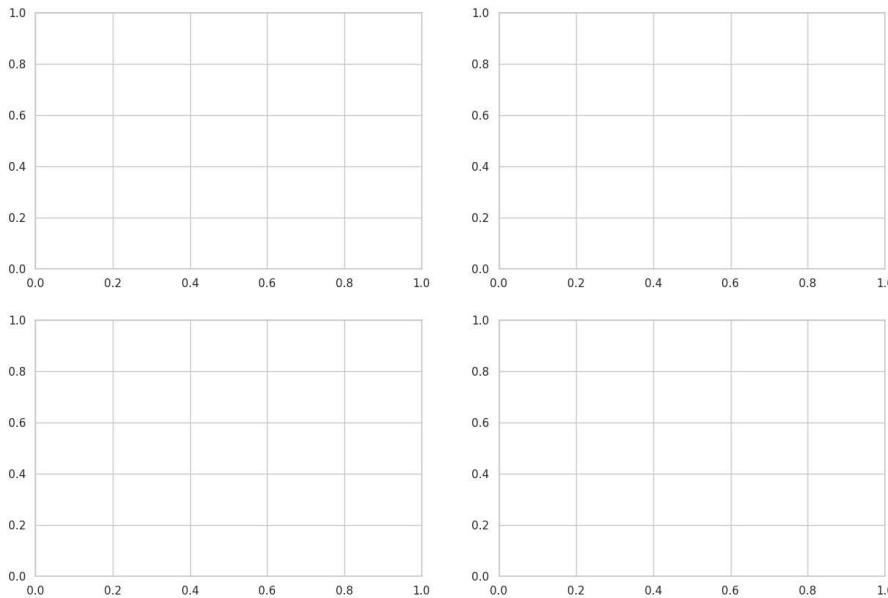
dfc1 = df1.corr()
dfc2 = df2.corr()
```

ValueError Traceback (most recent call last)  
<ipython-input-41-c4253d8ab63d> in <cell line: 8>()  
6 df11 = pd.pivot\_table(values = 'passengers', index = 'month', columns = 'year',  
data = df1)  
7  
----> 8 dfc1 = df1.corr()  
9 dfc2 = df2.corr()

6 frames

```
/usr/local/lib/python3.10/dist-packages/pandas/core/arrays/categorical.py in
__array__(self, dtype)
1342     ret = take_nd(self.categories._values, self._codes)
1343     if dtype and not is_dtype_equal(dtype, self.categories.dtype):
-> 1344         return np.asarray(ret, dtype)
1345     # When we're a Categorical[ExtensionArray], like Interval,
1346     # we need to ensure __array__ gets all the way to an
```

ValueError: could not convert string to float: 'Jan'



```
sns.heatmap(df11, cmap = "YlGnBu", linecolor = "r", linewidths = 0.5, annot = True, fmt = "d", square = True, ax = ax[0,0]).set_title("heat")
sns.heatmap(dfc2, cmap = "coolwarm", linecolor = "black", linewidths = 1, annot = True, ax = ax[0,0]).set_title("heat map iris")
```

```
import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from wordcloud import STOPWORDS

df = pd.read_csv('/content/netflix_titles.csv',usecols = ['cast'])
df.head()
```

	cast
0	NaN
1	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...
2	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...
3	NaN
4	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...

```
ndf = df.dropna()
ndf.head()
```

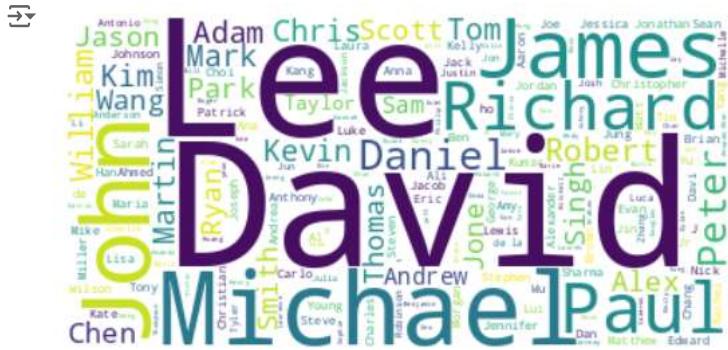
	cast
1	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...
2	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...
4	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...
5	Kate Siegel, Zach Gilford, Hamish Linklater, H...
6	Vanessa Hudgens, Kimiko Glenn, James Marsden, ...

```
text = " ".join(item for item in ndf['cast'])
print(text)
```

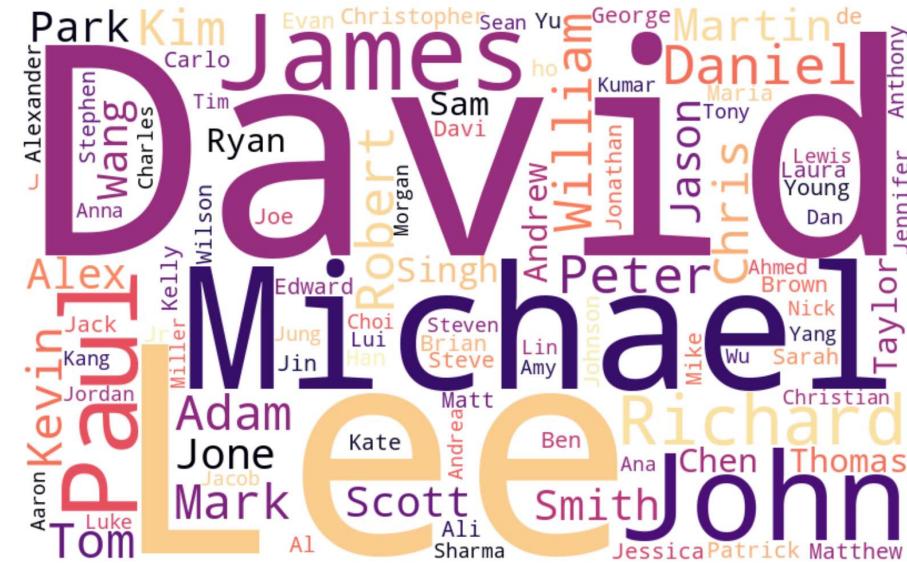
→ Ama Qamata, Khosi Ngema, Gail Mabalane, Thabang Molaba, Dillon Windvogel, Natasha Thahane, Arno Greeff, Xolile Tshabalala, Getmore Sitho  
◀ ▶

```
stopwords = set(STOPWORDS)
```

```
wordcloud = WordCloud(background_color = "white").generate(text)
plt.imshow(wordcloud, interpolation = "bilinear")
plt.axis("off")
plt.margins(x = 0, y = 0)
plt.show()
```



```
wordcloud=WordCloud(background_color='white',max_words=100,max_font_size=300,width=800,height=500,colormap='magma').generate(text)
plt.figure(figsize=(20,20))
plt.imshow(wordcloud,interpolation='bilinear')
plt.axis('off')
plt.margins(x=0,y=0)
```



```
wordcloud=WordCloud(background_color='white',max_words=100,max_font_size=300,width=800,height=500,colormap='magma').generate(text)
plt.figure(figsize=(20,20))
plt.imshow(wordcloud,interpolation='nearest',alpha=0.5)
plt.axis('off')
plt.margins(x=0,y=0)
```



```

methods = [None, 'none', 'nearest', 'bilinear', 'bicubic', 'spline16','spline36', 'hanning', 'hamming', 'hermite', 'kaiser', 'quadric','catr
fig, axs = plt.subplots(nrows=3, ncols=6, figsize=(15, 10), subplot_kw={'xticks': [], 'yticks': []})
wordcloud=WordCloud(stopwords=stopwords,background_color='white').generate(text)
for ax, interp_method in zip(axs.flat, methods):
    ax.imshow(wordcloud, interpolation=interp_method, cmap='viridis')
    ax.set_title(str(interp_method))

plt.tight_layout()
plt.show()

```



```
methods = [None, 'none', 'nearest', 'bilinear', 'bicubic', 'spline16','spline36', 'hanning', 'hamming', 'hermite', 'kaiser', 'quadric','catr
fig, axs = plt.subplots(nrows=3, ncols=6, figsize=(15, 10), subplot_kw={'xticks': [], 'yticks': []})
cmap=['viridis','plasma','inferno','magma','cividis']
i=0
wordcloud=WordCloud(stopwords=stopwords,background_color='white').generate(text)
for ax, interp_method in zip(axs.flat, methods):
    ax.imshow(wordcloud, interpolation=interp_method, cmap=cmap[i])
    ax.set_title(str(interp_method))
    if(i==4):
        i=0
    else:
        i=i+1

plt.tight_layout()
plt.show()
```



spline36

hanning

hamming

hermite

kaiser

quadric



catrom

gaussian

bessel

mitchell

sinc

lanczos



A time series is the series of data points listed in time order.

A time series is a sequence of successive equal interval points in time.

A time-series analysis consists of methods for analyzing time series data in order to extract meaningful insights and other useful characteristics of data.

For performing time series analysis download stock\_data.csv

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# reading the dataset using read_csv
df = pd.read_csv("stock_data.csv",
                  parse_dates=True,
                  index_col="Date")

# displaying the first five rows of dataset
df.head()
```

	Open	High	Low	Close	Volume	Name
Date						
2006-01-03	39.69	41.22	38.79	40.91	24232729	AABA
2006-01-04	41.22	41.90	40.77	40.97	20553479	AABA
2006-01-05	40.93	41.73	40.85	41.53	12829610	AABA
2006-01-06	42.88	43.57	42.80	43.21	29422828	AABA
2006-01-09	43.10	43.66	42.82	43.42	16268338	AABA

We have used the 'parse\_dates' parameter in the read\_csv function to convert the 'Date' column to the DatetimeIndex format.

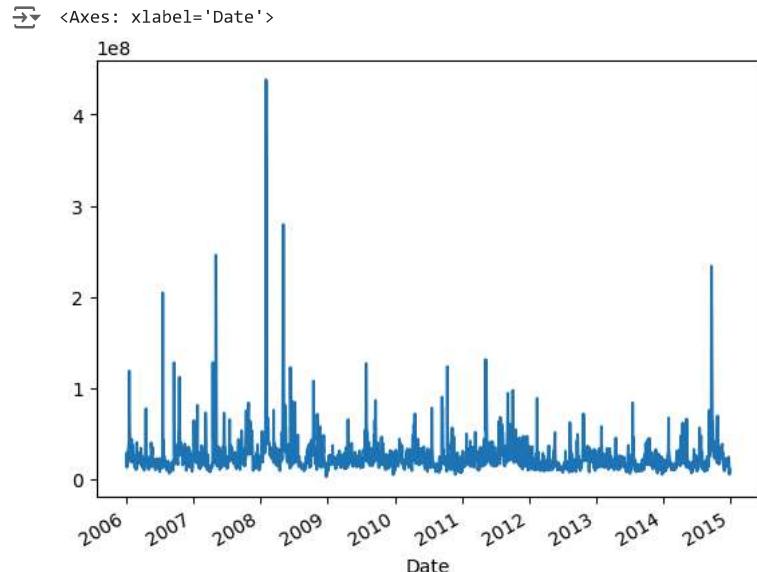
By default, Dates are stored in string format which is not the right format for time series data analysis.

Now, removing the unwanted columns from dataframe i.e. 'Unnamed: 0'.

```
# deleting column
df = df.drop(columns='Name')
```

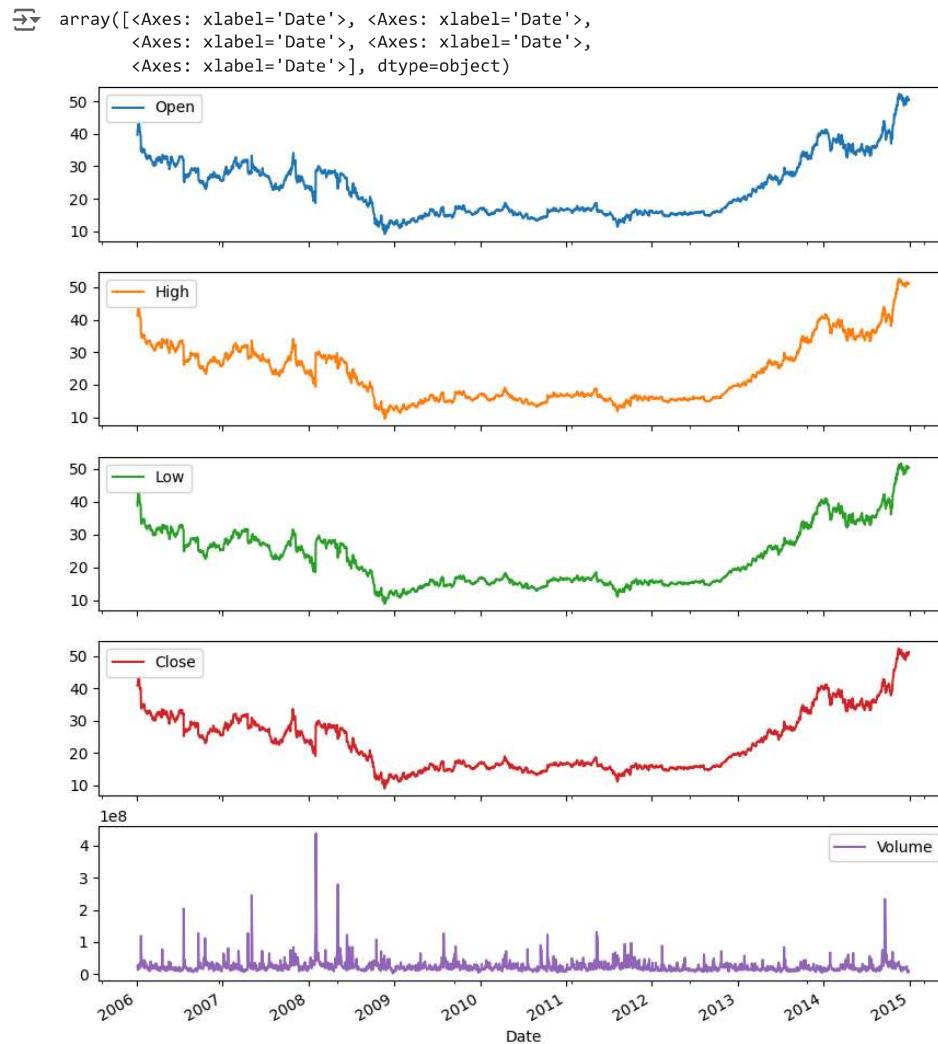
Example 1: Plotting a simple line plot for time series data.

```
df['Volume'].plot()
```



Example 2: Now let's plot all other columns using subplot.

```
df.plot(subplots=True, figsize=(10, 12))
```



```
print(df['Volume'].dtype)
```

```
int64
```

**Resampling:** Resampling is a methodology of economically using a data sample to improve the accuracy and quantify the uncertainty of a population parameter. Resampling for months or weeks and making bar plots is another very simple and widely used method of finding seasonality. Here we are going to make a bar plot of month data for 2016 and 2017.

Example 3:

```
# Resampling the time series data based on monthly 'M' frequency
```

```
df_month = df.resample("M").mean()
```

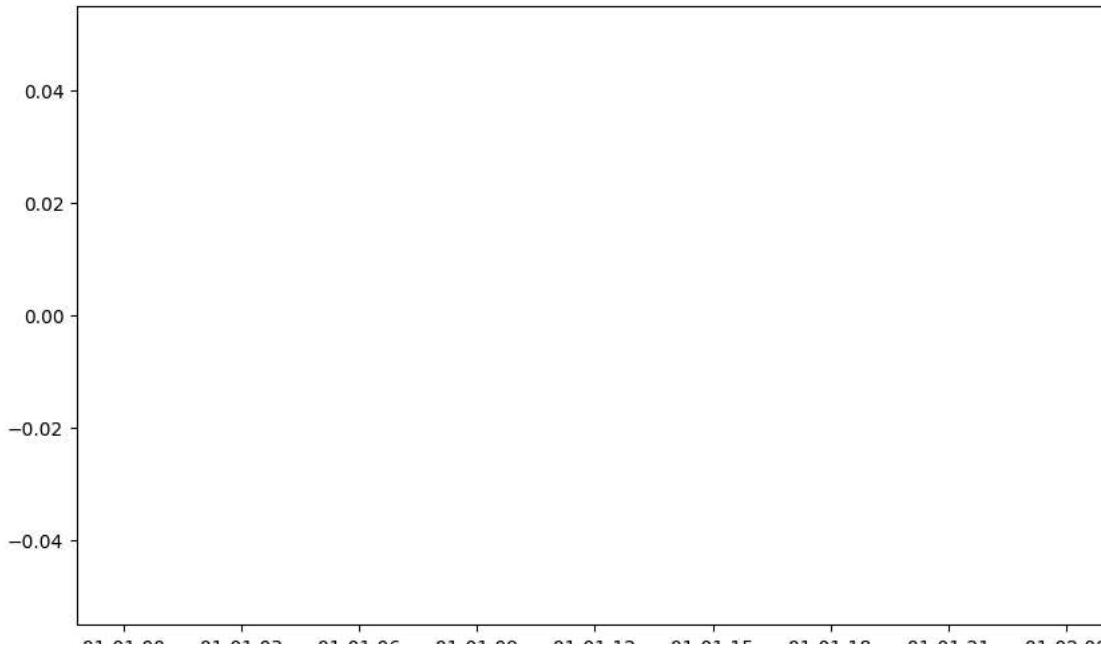
```
# using subplot
```

```
fig, ax = plt.subplots(figsize=(10, 6))
```

```
# plotting bar graph
```

```
ax.bar(df_month['2016':'2008'].index,
       df_month.loc['2016':'2008', "Volume"],
       width=25, align='center')
```

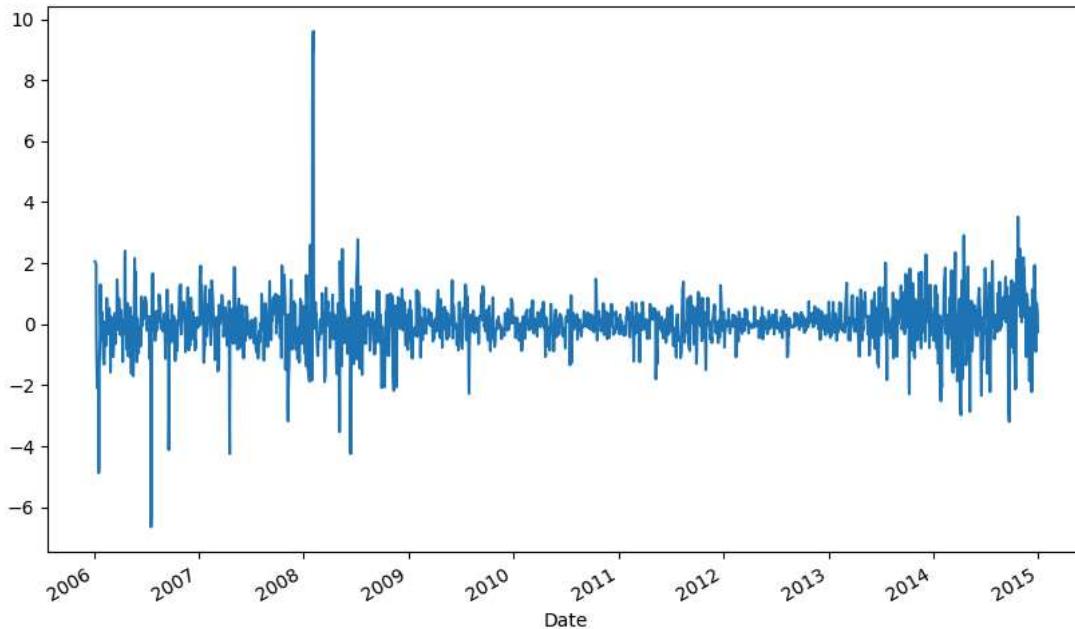
→ <BarContainer object of 0 artists>



Differencing: Differencing is used to make the difference in values of a specified interval. By default, it's one, we can specify different values for plots. It is the most popular method to remove trends in the data.

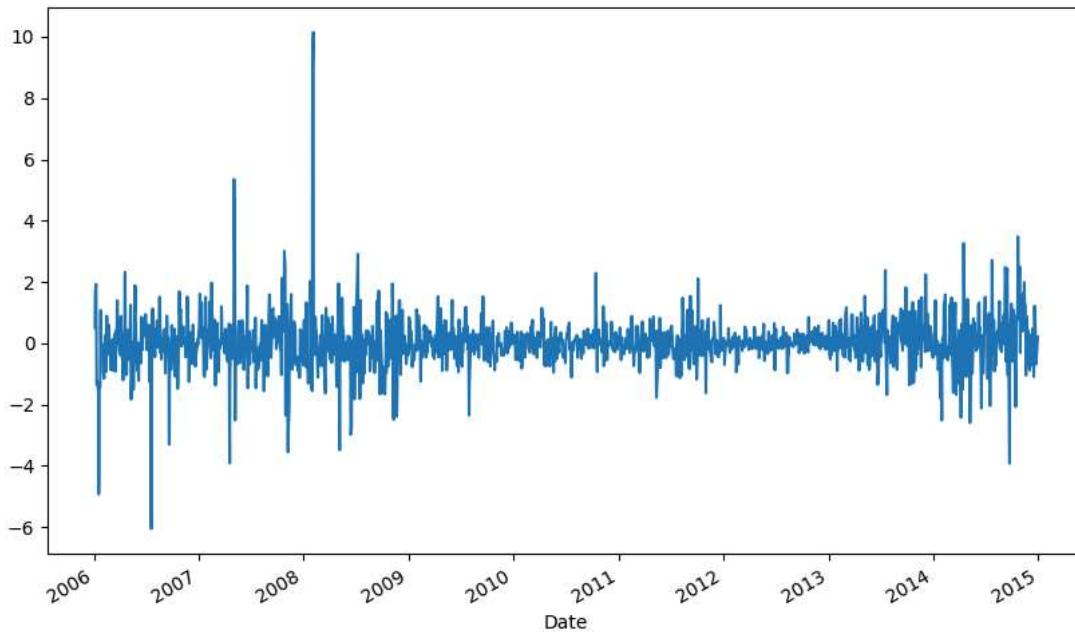
```
df.Low.diff(2).plot(figsize=(10, 6))
```

→ <Axes: xlabel='Date'>



```
df.High.diff(2).plot(figsize=(10, 6))
```

&lt;Axes: xlabel='Date'&gt;

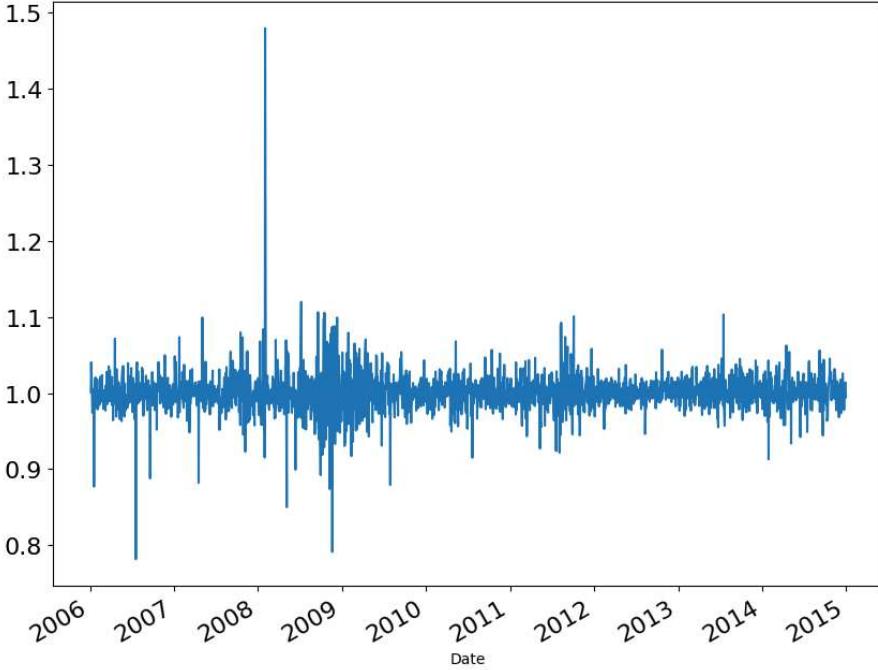
**Plotting the Changes in Data**

We can also plot the changes that occurred in data over time. There are a few ways to plot changes in data.

**Shift:** The shift function can be used to shift the data before or after the specified time interval. We can specify the time, and it will shift the data by one day by default. That means we will get the previous day's data. It is helpful to see previous day data and today's data simultaneously side by side.

```
df[ 'Change' ] = df.Close.div(df.Close.shift())
df[ 'Change' ].plot(figsize=(10, 8), fontsize=16)
```

&lt;Axes: xlabel='Date'&gt;



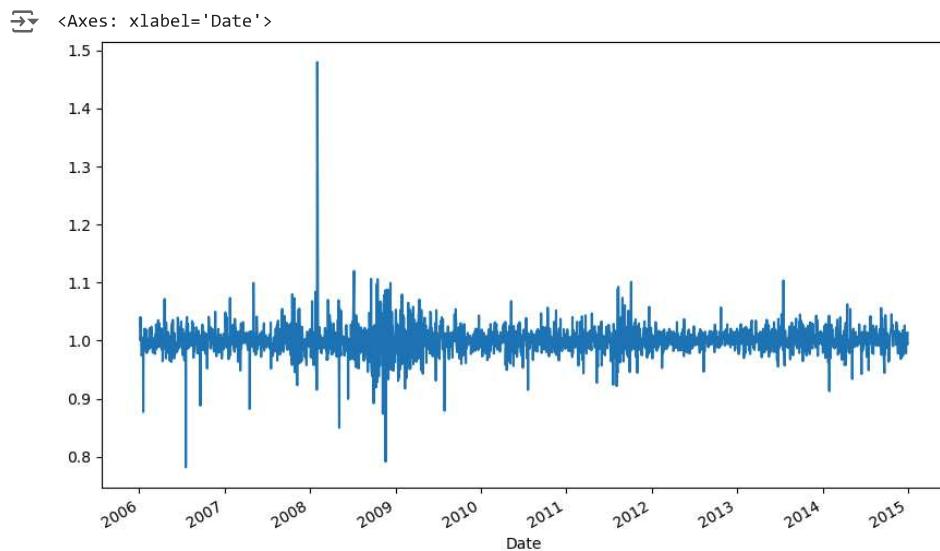
.div() function helps to fill up the missing data values.

Actually, div() means division.

If we take df. div(6) it will divide each element in df by 6.

We do this to avoid the null or missing values that are created by the 'shift()' operation.

```
df['Change'].plot(figsize=(10, 6))
```



```
df.info()
```

```
→ <class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2263 entries, 2006-01-03 to 2014-12-30
Data columns (total 7 columns):
 #   Column   Non-Null Count  Dtype  
--- 
 0   Open      2263 non-null   float64
 1   High      2263 non-null   float64
 2   Low       2263 non-null   float64
 3   Close     2263 non-null   float64
 4   Volume    2263 non-null   int64  
 5   Name      2263 non-null   object 
 6   Change    2262 non-null   float64
dtypes: float64(5), int64(1), object(1)
memory usage: 141.4+ KB
```

Double-click (or enter) to edit