Data Handling and Visualization (CSE2026) - Record

Submitted From,

Name:Rohan raj

Roll No: 20201ISB0007

Section & Semester: 8ISE-1 & 8th Sem

Submitted To,

Ms. Poornima S - Asst. Professor (CSE)

Presidency University,

Bengaluru

INDEX

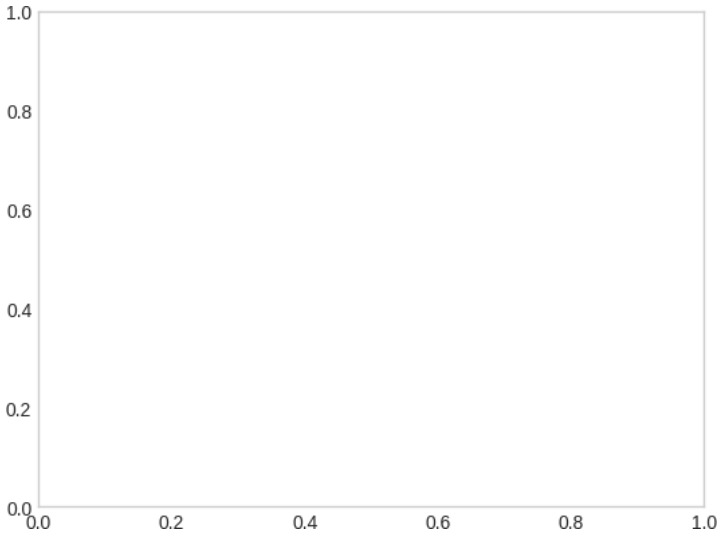
Sl. Number	Labsheets	Page number
1	LABSHEET 1(NUMPY)	1
2	LABSHEET 2(PANDAS)	6
3	LABSHEET 3(DATA CLEANING)	9
4	LABSHEET 4(Z-SCORE NORMALIZATION)	18
5	LABSHEET 5(OUTLIER DETECTION)	19
6	LABSHEET 6(MATPLOTLIB)	23
7	LABSHEET 7(DATA WRANGLING)	33
8	LABSHEET 8(COLORMAPS)	36
9	LABSHEET 9(HEATMAPS)	38
10	LABSHEET 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

LABSHEET 1

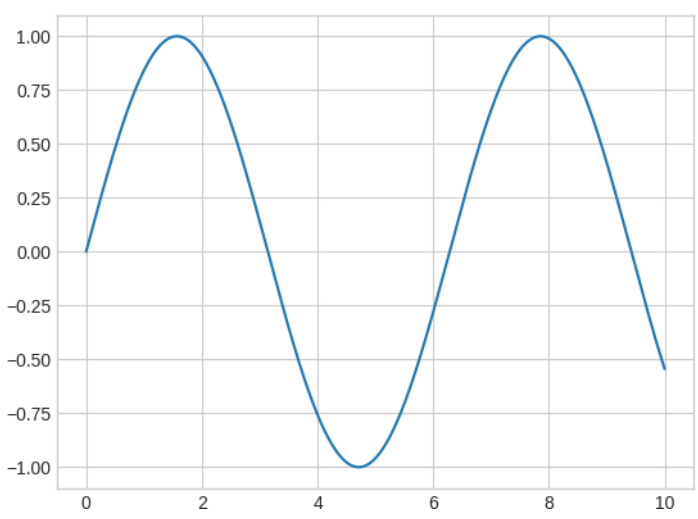
```
from matplotlib import pyplot as plt
plt.style.use('seaborn-whitegrid')

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

```
step 1
<ipython-input-4-240c5389bdd3>:2: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6,
  plt.style.use('seaborn-whitegrid')
```

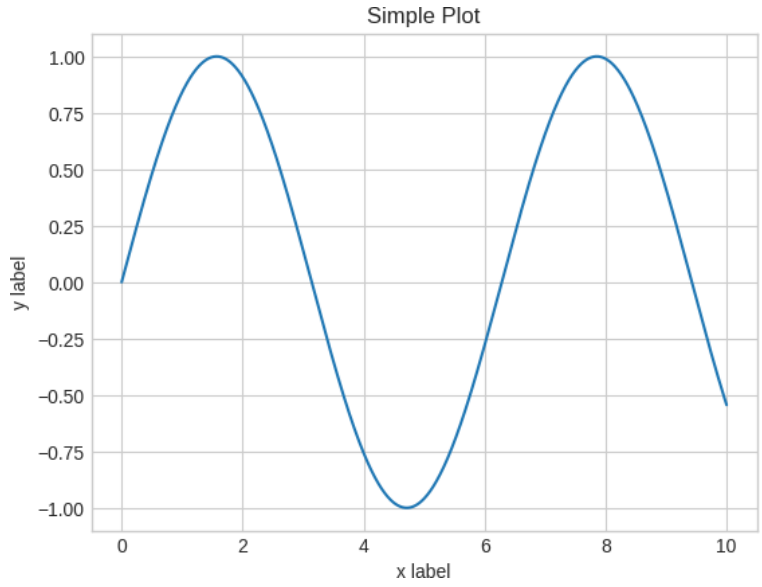
```
fig = plt.figure()
ax = plt.axes()
ax.grid()
```



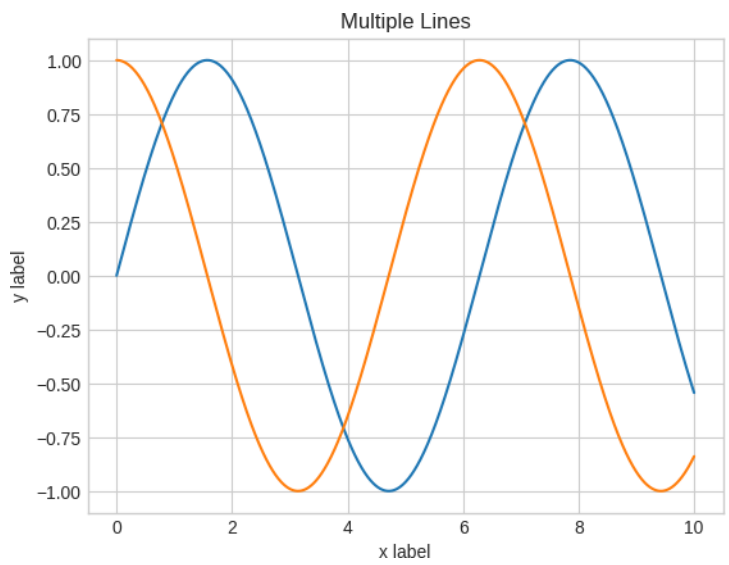

```
fig = plt.figure()
ax = plt.axes()

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



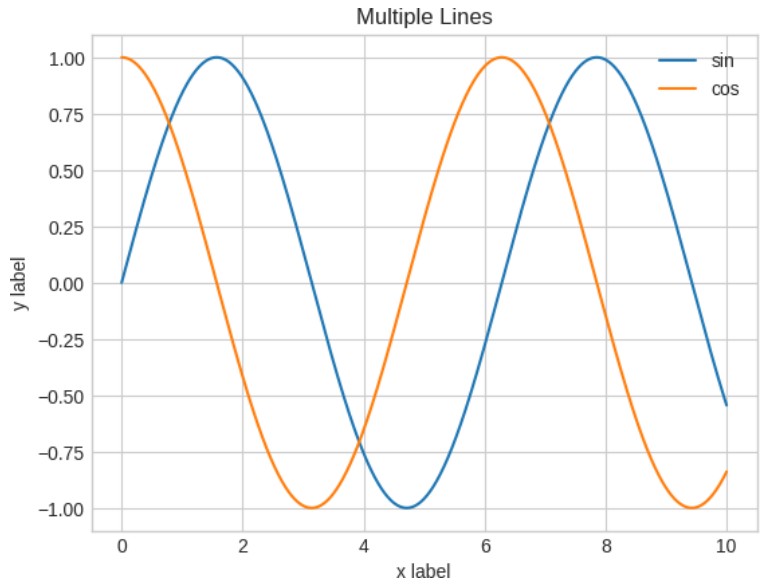
```
# Lets add a title and labels to the plot

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

x = np.linspace(0, 10, 1000)
ax.plot(x, np.sin(x))
ax.set_title('Simple Plot')   # Add a title
ax.set_xlabel('x label')          # Add x label
ax.set_ylabel('y label');         # Add y label
```



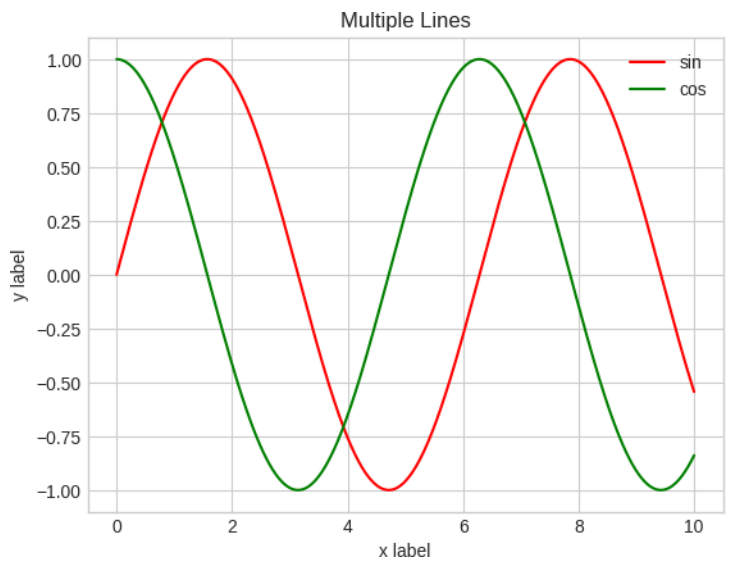

```
# Lets add a title to the plot above
fig = plt.figure()
ax = plt.axes()

x = np.linspace(0, 10, 1000)
ax.plot(x, np.sin(x))
ax.plot(x, np.cos(x))
#ax.plot(x, np.tan(x))
ax.set_title('Multiple Lines');
ax.set_xlabel('x label')
ax.set_ylabel('y label')
plt.show()
```



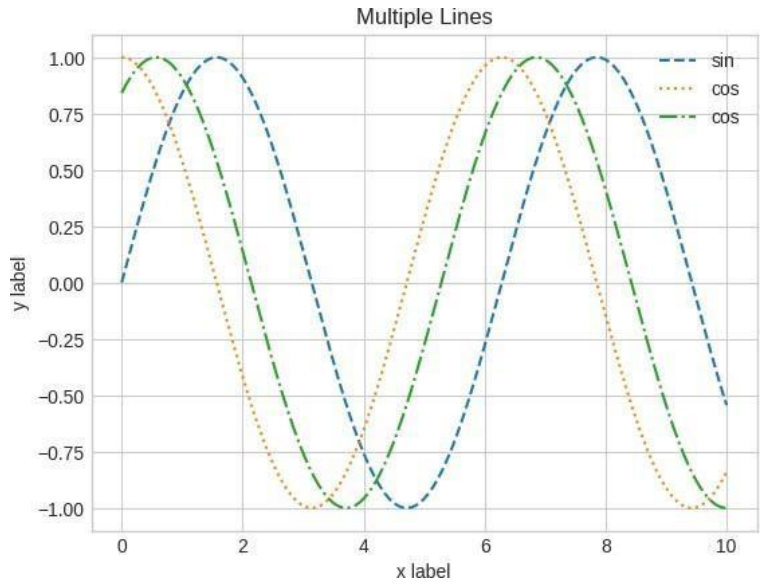
```
fig = plt.figure()
ax = plt.axes()

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()
# ax.legend(loc=1)
plt.show()
```



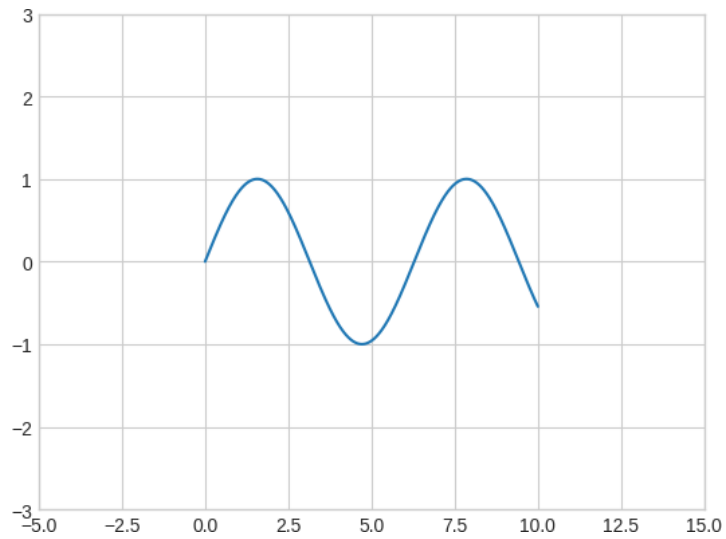

```
fig = plt.figure()
ax = plt.axes()

x = np.linspace(0, 10, 1000)
ax.plot(x, np.sin(x), label = 'sin', color = 'red')   # specify color by name
ax.plot(x, np.cos(x), label = 'cos', color = 'g')    # short color code (rgbcmyk)
ax.set_title('Multiple Lines');
ax.set_xlabel('x label')
ax.set_ylabel('y label')
ax.legend();
```



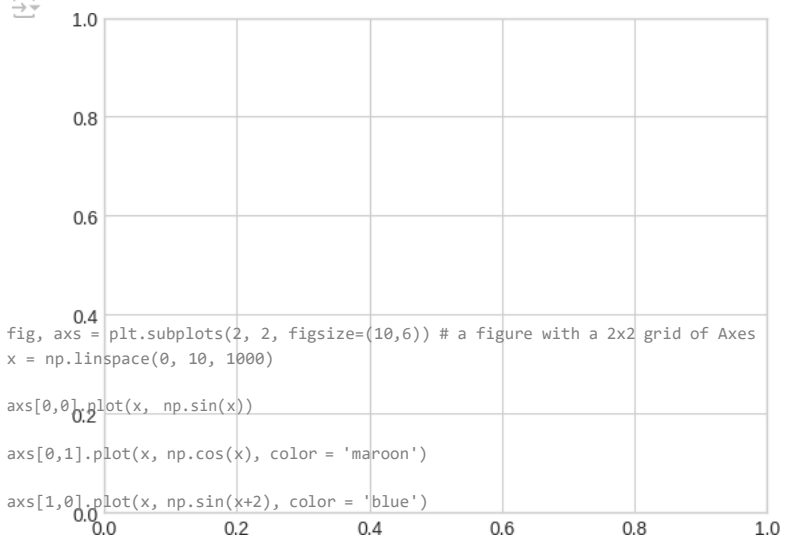
```
fig = plt.figure()
ax = plt.axes()
# ax.grid(linestyle = '--')

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 = 'cos', linestyle = 'dashdot')
ax.set_title('Multiple Lines');
ax.set_xlabel('x label')
ax.set_ylabel('y label')
ax.legend();
```



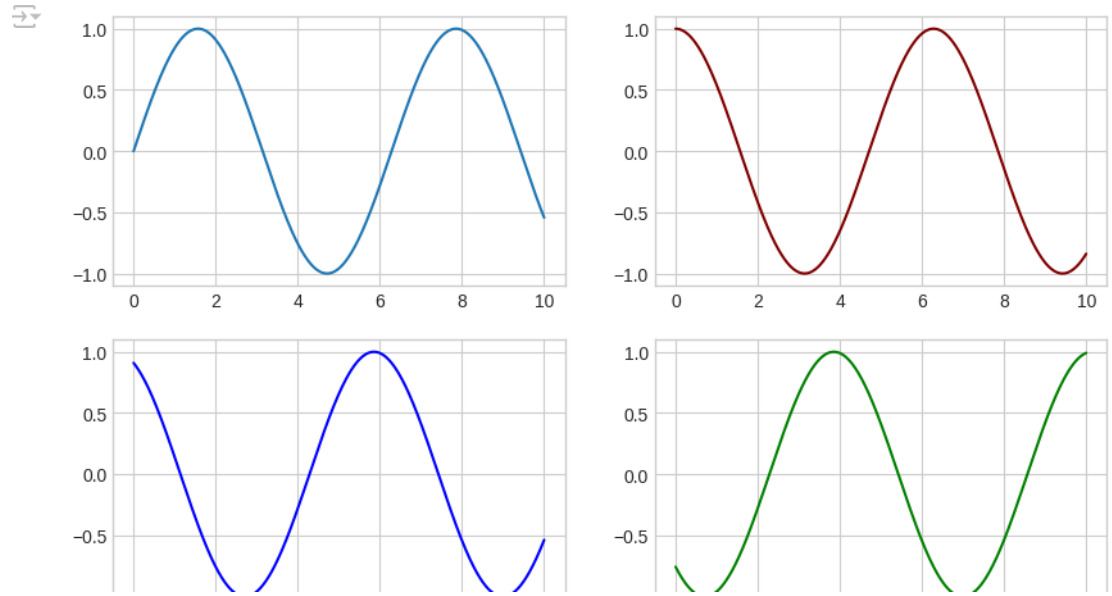

```
fig = plt.figure()
ax = plt.axes()

x = np.linspace(0, 10, 1000)
ax.plot(x, np.sin(x))

ax.set_xlim(-5, 15)
ax.set_ylim(-3, 3);
```




```
fig, ax = plt.subplots()        # a figure with a single Axes
```



```
fig, axs = plt.subplots(2, 2, figsize=(10,6)) # a figure with a 2x2 grid of Axes
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');
```



ļABSHEEĪ 2

pandas

```
import pandas as pd
data=pd.read_csv(r'C:\Users\Thejas Venugopal\Downloads\nyc_weather.csv')
data.head()
```

	EST	Temperature	DewPoint	Humidity	Sea Level PressureIn	VisibilityMiles	WindSpeedMP
0	1/1/2016	38	23	52	30.03	10	8.
1	1/2/2016	36	18	46	30.02	10	7.
2	1/3/2016	40	21	47	29.86	10	8.
3	1/4/2016	25	9	44	30.05	10	9.

pandas seíies

```
import numpy as np
d=np.array(['a','b','c','d'])
s=pd.Series(d)
print(s)
```

```
0    a
1    b
2    c
3    d
dtype: object
```

with d being a dictionaíy

```
d={'a':1.,'b':2,'c':3}
s=pd.Series(d,index=['b','c','d'])
s
```

```
b    2.0
c    3.0
d    NaN
dtype: float64
```

changing the index

```
d=np.array(['a','b','c','d'])
s=pd.Series(d,index=[100,101,102,103])
print(s)
```

```
100    a
101    b
102    c
103    d
dtype: object
```

dtype = float

```
n=np.array([1,2,3])
s1=pd.Series(n,dtype=float)
s1
```

```
0    1.0
1    2.0
2    3.0
dtype: float64
```

syntax

pd.Seíies(data,index=[ ],dtype=, name=, copy=,)

combining 2 aííays to make an object

```
a1=np.array([1,2,3])
a2=np.array(['a','b','z'])
s2=pd.Series(a1,a2)
s2
```

```
a    1
b    2
z    3
dtype: int32
```

handling missing values

```
d={'a':1.,'b':2,'c':3}
s=pd.Series(d,index=['b','c','d'])
print(s)
```

```
b    2.0
c    3.0
d    NaN
dtype: float64
```

```
s.isna().sum()
```

```
1
```

```
s.dropna()
```

```
b    2.0
c    3.0
dtype: float64
```

```
d={'a':1.,'b':2,'c':3}
s=pd.Series(d,index=['b','c','d'])
print(s)
```

```
b    2.0
c    3.0
d    NaN
dtype: float64
```

```
s.fillna(2)
```

```
b    2.0
c    3.0
d    2.0
dtype: float64
```

accessing elements fíom the index

```
series=pd.Series([1,2,3,4,5],index=['a','b','c','d','e'])
series[1]
```

```
2
```

```
series[:3]
```

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

```
series['a']
```

```
    1
series[['a','c','e']]
```

```
a    1
c    3
e    5
dtype: int64
```

```
series1=pd.Series([103,1079,978],index=[' a hundred and three','one thousand seventy nine','nine hundred seventy eight'])
series1['nine hundred seventy eight']
```

```
978
```

KAI'A IRAME

```
import pandas as pd
data = {'Name':['Alice', 'Bob', 'Claire', 'David'],
        'Age':[20, 21, 20, 22]}
df = pd.DataFrame(data)
print(df)
```

```
      Name  Age
0   Alice   20
1     Bob   21
2  Claire   20
3   David   22
```

```
# creating a dataframe from a list of dictionary
data = [{'Name': 'Alice', 'Age': 20},
        {'Name': 'Bob', 'Age': 21},
        {'Name': 'Claire', 'Age': 20},
        {'Name': 'David', 'Age': 22}]
df = pd.DataFrame(data)
print(df)
```

```
      Name  Age
0   Alice   20
1     Bob   21
2  Claire   20
3   David   22
```

```
pd.DataFrame(df)
```

	Name	Age
0	Alice	20
1	Bob	21
2	Claire	20
3	David	22

```
Start coding or ge nerate with AI.
```

ĻABSHEEĪ' 3

Kata Cleaning and Kata Píepíocessing:

1. Kata cleaning is the píocess of changing oí eliminating gaíbage, incoíiect, duplicate, coíiupted, oí incomplete data in a dataset.
2. Ī'kcíc's no such absolute way to descíibe the píecise steps in the data cleaning píocessbecause the píocesses may vaíy fíom dataset to dataset.

Kata Cleaning Cycle

Missi□g Val"cs:

```
# 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  0.375319 -0.763927 -0.762393
c -1.093644  1.335944 -0.668966
e -0.013401  0.155461 -0.843651
f  0.423813  0.900266 -0.828664
h -0.644593  2.654895  1.211697
```

Check foí Missing Values:

l'o makc dctccti g□missi g□al"cs casicí (a d a□oss diíícíc t aíía□dtQpcs),Pa das pío□dcs tkc **isnull()** a□d **notnull()** í" □tio s□ wkick aíc also mctkods o Scíi□cs a d Kat□Iíamc objccts −

```
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())
# print(df)
print(df["one"].isnull())
```

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

Rcplaci g tkc Missi g Valucs

```
#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 -0.961858 -1.671248  0.556286
b       NaN       NaN       NaN
c -0.386504 -0.709324  0.622838
NaN replaced with '0':
        one       two     three
a -0.961858 -1.671248  0.556286
b  0.000000  0.000000  0.000000
c -0.386504 -0.709324  0.622838
```

Ïill NA Ïoíwaíd a d Backwaíd

# Method	Action
pad/fill	Fill methods Forward
bfill/backfill	Fill methods Backward

```
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  0.109813 -1.940379 -0.444834
b       NaN       NaN       NaN
c -0.208020  0.309864  0.819870
d       NaN       NaN       NaN
e -0.465764  0.215614  1.031519
f  1.189843  3.814140  0.954030
g       NaN       NaN       NaN
h  0.480653  0.552598 -0.888482
        one       two     three
a  0.109813 -1.940379 -0.444834
b  0.109813 -1.940379 -0.444834
c -0.208020  0.309864  0.819870
d -0.208020  0.309864  0.819870
e -0.465764  0.215614  1.031519
f  1.189843  3.814140  0.954030
g  1.189843  3.814140  0.954030
h  0.480653  0.552598 -0.888482
```

```
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 -1.204446  2.137228 -0.388020
b  1.327178  2.355456 -1.347412
c  1.327178  2.355456 -1.347412
d -0.228600  1.300295  0.939832
e -0.228600  1.300295  0.939832
f -0.938383  2.278881 -0.098408
g  0.726762  0.456629 -1.167753
h  0.726762  0.456629 -1.167753
```

Díop Missi g Valucs:

Usc díop a í" ctio alo g witk tkc axis aíg"mc t.

BQ dcía"lt, axis=0, i.c., alo g íow, wkick mca s kat ií a Q :al"c witki a íow is NA tkc tkc w kolcíow is cxcl"dcd.

```
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 -0.481989 -1.249458 -2.316982
c  1.119240 -1.054186 -0.972090
e -0.991040 -0.749165  0.259387
f -1.300768 -0.000567 -0.056870
h  0.497341  0.984014 -1.094049
        one       two     three
a -0.481989 -1.249458 -2.316982
b       NaN       NaN       NaN
c  1.119240 -1.054186 -0.972090
d       NaN       NaN       NaN
e -0.991040 -0.749165  0.259387
f -1.300768 -0.000567 -0.056870
g       NaN       NaN       NaN
h  0.497341  0.984014 -1.094049
        one       two     three
a -0.481989 -1.249458 -2.316982
c  1.119240 -1.054186 -0.972090
e -0.991040 -0.749165  0.259387
f -1.300768 -0.000567 -0.056870
h  0.497341  0.984014 -1.094049
```

Rcplacc Missi g (oí) Gc ðiic Valucs:

Wc ca ackic:c tkis bQ applQi g tkc **ícplacc** mctkod.

Rcplaci g NA witk a scalaí :al"c is cq"i:alc t b ka:ioí oí tkc **fill a()** f ctio .

```
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
```

Kata Pícpíoccssi g

1. Load data i Pa das
2. Kíop col"m s tkat aíc ’"scí"l
3. Kíop íows witk missi g:al"cs
4. Cícatc d"mmQ :aíiablcs
5. l’akc caíc oí missi gdata
6. Co:cít tkc data ííamc to N"mPQ

Dowload l’itaic-Datasct fíom Kagglc.com.
Hcíc wc aíc goi g to usc tíai .cs: datasct foí pícpíoccssi g.

```
import pandas as pd
import numpy as np
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
df = pd.read_csv(r"C:\Users\Thejas Venugopal\Downloads\train (1).csv")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

Díop tkc Colums tkat aíc ot ícquiícd

```
cols=['Name','Ticket','Cabin']
df=df.drop(cols,axis=0)
df.info()
```

```
KeyError                                  Traceback (most recent call last)
C:\Users\THEJAS~1\AppData\Local\Temp/ipykernel_20436/1019933480.py in <module>
      1 cols=['Name','Ticket','Cabin']
----> 2 df=df.drop(cols)
      3 df.info()

c:\Users\Thejas Venugopal\anaconda3\lib\site-packages\pandas\util\_decorators.py in
wrapper(*args, **kwargs)
    309                     stacklevel=stacklevel,
    310                 )
--> 311             return func(*args, **kwargs)
    312
    313         return wrapper

c:\Users\Thejas Venugopal\anaconda3\lib\site-packages\pandas\core\frame.py in
drop(self, labels, axis, index, columns, level, inplace, errors)
   4904                 weight  1.0     0.8
   4905         """
-> 4906         return super().drop(
   4907             labels=labels,
   4908             axis=axis,

c:\Users\Thejas Venugopal\anaconda3\lib\site-packages\pandas\core\generic.py in
drop(self, labels, axis, index, columns, level, inplace, errors)
   4148         for axis, labels in axes.items():
   4149             if labels is not None:
-> 4150                 obj = obj._drop_axis(labels, axis, level=level,
errors=errors)
   4151
   4152         if inplace:

c:\Users\Thejas Venugopal\anaconda3\lib\site-packages\pandas\core\generic.py in
_drop_axis(self, labels, axis, level, errors)
   4183                 new_axis = axis.drop(labels, level=level, errors=errors)
   4184             else:
-> 4185                 new_axis = axis.drop(labels, errors=errors)
   4186             result = self.reindex(**{axis_name: new_axis})
   4187

c:\Users\Thejas Venugopal\anaconda3\lib\site-packages\pandas\core\indexes\base.py in
drop(self, labels, errors)
   6015         if mask.any():
   6016             if errors != "ignore":
-> 6017                 raise KeyError(f"{labels[mask]} not found in axis")
```

Díop tkc íows ka:i g :alucs

```
df = df.dropna()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 0 to 890
Data columns (total 9 columns):
```

```
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  712 non-null    int64
 1   Survived     712 non-null    int64
 2   Pclass       712 non-null    int64
 3   Sex          712 non-null    object
 4   Age          712 non-null    float64
 5   SibSp        712 non-null    int64
 6   Parch        712 non-null    int64
 7   Fare         712 non-null    float64
 8   Embarked     712 non-null    object
dtypes: float64(2), int64(5), object(2)
memory usage: 55.6+ KB
```

Cícati g DummmQ :aíiablcs

I stcad oí wasti g o"í data, lct's co :cít tkc Pclass, Scx a d Em baíkcd to col"m s i Pa das a d díop tk maítcí co :císio .

```
dummies = []
cols = ['Pclass', 'Sex', 'Embarked']
for col in cols:
   dummies.append(pd.get_dummies(df[col]))
```

1'ía síoí tkc cigtk col"m s

```
titanic_dummies = pd.concat(dummies, axis=1)
```

Co catc atc tkc :al"cs witk data ííamc

```
df = pd.concat((df,titanic_dummies), axis=1)
```

Rcmo:c tkc " wa td cols

```
df = df.drop(['Pclass', 'Sex', 'Embarked'], axis=1)
```

l'akc caíc of Missi g data

Lct's comp"tc a **mcdia oí i tcípolatc()** all tkc agcs a d íil tkosc missi g agc al"cs. Pa das kas ai tcípolatc() í" cío tka t will ícplacc all tkc missi g NaNs to i tcípolatcd :al"cs.

```
df['Age'] = df['Age'].interpolate()
print(df)
```

Mi Max Scalcí a d Sta daídizatio

Noímalizatio is a ícscali g tó tkc data ííom tkc oíigi al ía gc so tkat all :al"cs aíc witki tkccw íá gc oí 0 a d 1.

A :al"c is bímalizcd as íollows:

Q = (x – mi) / (max – mi)

```
from sklearn.preprocessing import MinMaxScaler
data = [[-1, 2], [-0.5, 6], [0, 10], [1, 18]]
scaler = MinMaxScaler()
print(scaler.fit(data))
MinMaxScaler()
print(scaler.data_max_)
print(scaler.transform(data))
```

```
MinMaxScaler()
    [ 1. 18.]
    [[0.   0.  ]
     [0.25 0.25]
     [0.5  0.5 ]
     [1.   1.  ]]
```

Kata Sta daízatio

Sta daízi g a datasct i :ol:cs ícscali g tkc distíib"tio oí :al"cs so tkat tkc mca oí obsc:cd :al"cs is 0 a d tkc sta daíd dc:iatio is 1. A

:al"c is sta daízcd as íollows:

Q = (x – mca) / sta daíd_dc:iatio Wkcíctkc

mca is ca lc"latcd as:

mca = s"m(x) / co" t(x)

And tkc sta daíd_dc:iatio is calc"latcd as: sta daíd_dc:iatio

= sqít(s"m((x – mca)^2) / co" t(x))

```
from numpy import asarray
from sklearn.preprocessing import StandardScaler
# define data
data = asarray([[100, 0.001],
 [8, 0.05],
 [50, 0.005],
 [88, 0.07],
 [4, 0.1]])
print(data)
# define standard scaler
scaler = StandardScaler()
# transform data
scaled = scaler.fit transform(data)
```

LABSHEET 4

```
import numpy as np
import pandas as pd
```

```
# Example dataset
data = {
    'Feature1': [10, 20, 30, 40, 50],
    'Feature2': [5, 15, 25, 35, 45]
}

# Create a DataFrame
df = pd.DataFrame(data)

# Display the original data
print("Original Data:")
print(df)
```

```
Original Data:
   Feature1  Feature2
0        10         5
1        20        15
2        30        25
3        40        35
4        50        45
```

```
# Function to normalize data using Z-score
def zscore_normalization(df):
    normalized_df = df.copy()
    for column in normalized_df.columns:
        mean = normalized_df[column].mean()
        std = normalized_df[column].std()
        normalized_df[column] = (normalized_df[column] - mean) / std
    return normalized_df

# Normalize the DataFrame
normalized_df = zscore_normalization(df)

# Display the normalized data
print("\nNormalized Data (Z-score):")
print(normalized_df)
```

```
Normalized Data (Z-score):
   Feature1  Feature2
0 -1.264911 -1.264911
1 -0.632456 -0.632456
2  0.000000  0.000000
3  0.632456  0.632456
4  1.264911  1.264911
```

LABSHEET 5

```
from google.colab import files
df = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving train.csv to train.csv
```

```
import pandas as pd
import numpy as np
```

```
data = pd.read_csv('./train.csv')
data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	493	0	1	Molson, Mr. Harry Markland	male	55.0	0	0	113787	30.5000	C30	S
1	53	1	1	Harper, Mrs. Henry Sleeper (Myna Haxtun)	female	49.0	1	0	PC 17572	76.7292	D33	C
2	388	1	2	Buss, Miss. Kate	female	36.0	0	0	27849	13.0000	NaN	S
3	192	0	2	Carbines, Mr. William	male	19.0	0	0	28424	13.0000	NaN	S
4	687	0	3	Panula, Mr. Jaako Arnold	male	14.0	4	1	3101295	39.6875	NaN	S

```
cols = ['Name', 'Ticket', 'Cabin']
filtered_data = data.drop(cols, axis = 1)
filtered_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 712 entries, 0 to 711
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  712 non-null    int64  
 1   Survived     712 non-null    int64  
 2   Pclass       712 non-null    int64  
 3   Sex          712 non-null    object 
 4   Age          566 non-null    float64
 5   SibSp        712 non-null    int64  
 6   Parch        712 non-null    int64  
 7   Fare         712 non-null    float64
 8   Embarked     710 non-null    object 
dtypes: float64(2), int64(5), object(2)
memory usage: 50.2+ KB
```

```
data = data.dropna()
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 148 entries, 0 to 695
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  148 non-null    int64  
 1   Survived     148 non-null    int64  
 2   Pclass       148 non-null    int64  
 3   Name         148 non-null    object 
 4   Sex          148 non-null    object 
 5   Age          148 non-null    float64
 6   SibSp        148 non-null    int64  
 7   Parch        148 non-null    int64  
 8   Ticket       148 non-null    object 
 9   Fare         148 non-null    float64
 10  Cabin        148 non-null    object 
 11  Embarked     148 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 15.0+ KB
```

```
data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	493	0	1	Molson, Mr. Harry Markland	male	55.0	0	0	113787	30.5000	C30	S
1	53	1	1	Harper, Mrs. Henry Sleeper (Myna Haxtun)	female	49.0	1	0	PC 17572	76.7292	D33	C
9	752	1	3	Moor, Master. Meier	male	6.0	0	1	392096	12.4750	E121	S
10	541	1	1	Crosby, Miss. Harriet R	female	36.0	0	2	WE/P 5735	71.0000	B22	S

```
dummies = []
cols = ['Pclass', 'Sex', 'Embarked']
for col in cols:
  dummies.append(pd.get_dummies(data[col]))
```

```
dummies
```

```
[     1  2  3
0    1  0  0
1    1  0  0
2    0  1  0
3    0  1  0
4    0  0  1
..  .. .. ..
707  0  0  1
708  1  0  0
709  0  0  1
710  0  1  0
711  1  0  0

[712 rows x 3 columns],
     female  male
0         0     1
1         1     0
2         1     0
3         0     1
4         0     1
..      ...   ...
707       1     0
708       0     1
709       0     1
710       0     1
711       0     1

[712 rows x 2 columns],
     C  Q  S
0    0  0  1
1    1  0  0
2    0  0  1
3    0  0  1
4    0  0  1
..  .. .. ..
707  1  0  0
708  1  0  0
709  0  0  1
710  0  1  0
711  0  0  1

[712 rows x 3 columns]]
```

```
titanic_dummies = pd.concat(dummies, axis = 1)
titanic_dummies
```

	1	2	3	female	male	C	Q	S
0	1	0	0	0	1	0	0	1
1	1	0	0	1	0	1	0	0
2	0	1	0	1	0	0	0	1
3	0	1	0	0	1	0	0	1
4	0	0	1	0	1	0	0	1
...	...	...	...	...	...	...	...	...
707	0	0	1	1	0	1	0	0
708	1	0	0	0	1	1	0	0
709	0	0	1	0	1	0	0	1
710	0	1	0	0	1	0	0	1
711	1	0	0	0	1	0	0	1

712 rows × 8 columns

```
data.drop(['Pclass', 'Sex', 'Embarked'], axis = 1)
```

	PassengerId	Survived	Name	Age	SibSp	Parch	Ticket	Fare	Cabin
0	493	0	Molson, Mr. Harry Markland	55.0	0	0	113787	30.5000	C30
1	53	1	Harper, Mrs. Henry Sleeper (Myna Haxtun)	49.0	1	0	PC 17572	76.7292	D33
2	388	1	Buss, Miss. Kate	36.0	0	0	27849	13.0000	NaN
3	192	0	Carbines, Mr. William	19.0	0	0	28424	13.0000	NaN
4	687	0	Panula, Mr. Jaako Arnold	14.0	4	1	3101295	39.6875	NaN
...	...	...	...	...	...	...	...	...	...
707	859	1	Baclini, Mrs. Solomon (Latifa Qurban)	24.0	0	3	2666	19.2583	NaN
708	65	0	Stewart, Mr. Albert A	NaN	0	0	PC 17605	27.7208	NaN
709	130	0	Ekstrom, Mr. Johan	45.0	0	0	347061	6.9750	NaN
710	21	0	Fynney, Mr. Joseph J	35.0	0	0	239865	26.0000	NaN
711	476	0	Clifford, Mr. George Quincy	NaN	0	0	110465	52.0000	A14

712 rows × 9 columns

```
data['Age'] = data['Age'].interpolate()
print(data)
```

```
     PassengerId  Survived  Pclass                                     Name  \
0            493         0       1                   Molson, Mr. Harry Markland
1             53         1       1  Harper, Mrs. Henry Sleeper (Myna Haxtun)
2            388         1       2                            Buss, Miss. Kate
3            192         0       2                      Carbines, Mr. William
4            687         0       3                    Panula, Mr. Jaako Arnold
..           ...       ...     ...                                         ...
707          859         1       3     Baclini, Mrs. Solomon (Latifa Qurban)
708           65         0       1                      Stewart, Mr. Albert A
709          130         0       3                         Ekstrom, Mr. Johan
710           21         0       2                       Fynney, Mr. Joseph J
711          476         0       1                 Clifford, Mr. George Quincy

        Sex   Age  SibSp  Parch    Ticket     Fare Cabin Embarked
0      male  55.0      0      0    113787  30.5000   C30        S
1    female  49.0      1      0  PC 17572  76.7292   D33        C
2    female  36.0      0      0     27849  13.0000   NaN        S
3      male  19.0      0      0     28424  13.0000   NaN        S
4      male  14.0      4      1   3101295  39.6875   NaN        S
..      ...   ...    ...    ...       ...      ...   ...      ...
707  female  24.0      0      3      2666  19.2583   NaN        C
708    male  34.5      0      0  PC 17605  27.7208   NaN        C
709    male  45.0      0      0    347061   6.9750   NaN        S
710    male  35.0      0      0    239865  26.0000   NaN        S
711    male  35.0      0      0    110465  52.0000   A14        S

[712 rows x 12 columns]
```

```
from sklearn.preprocessing import MinMaxScaler
data = [[-1, 1], [-0.5, 6], [0, 10], [1, 10]]
scaler = MinMaxScaler()
print(scaler.fit(data))
print(scaler.data_max_)
print(scaler.transform(data))
```

```
MinMaxScaler()
[ 1. 10.]
[[0.         0.        ]
 [0.25       0.55555556]
 [0.5        1.        ]
 [1.         1.        ]]
```

LABSHEET 6

```
import matplotlib.pyplot as plt
# import seaborn as sn

# print a empty figure
# linespace 10 points with 1000 data points
# styles
# sin x and cos x
# legend values, colors, setting x, y title and other stuff
# line styles (different styles for each line)
# setting access limits (interval limits)
# subplot (printing multiple plots)
# 0 1 y = sin and then 0 1 x = sin
```

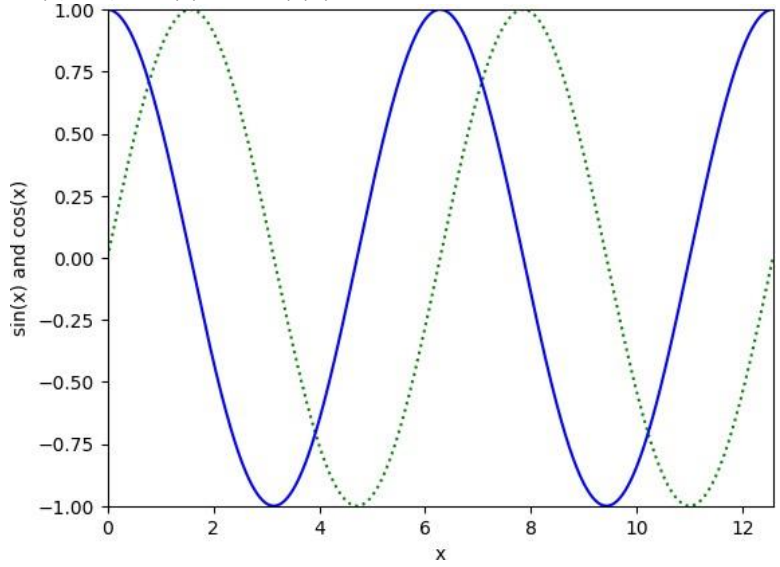
Code Text

```
# print a empty figure
fig = plt.figure()
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```

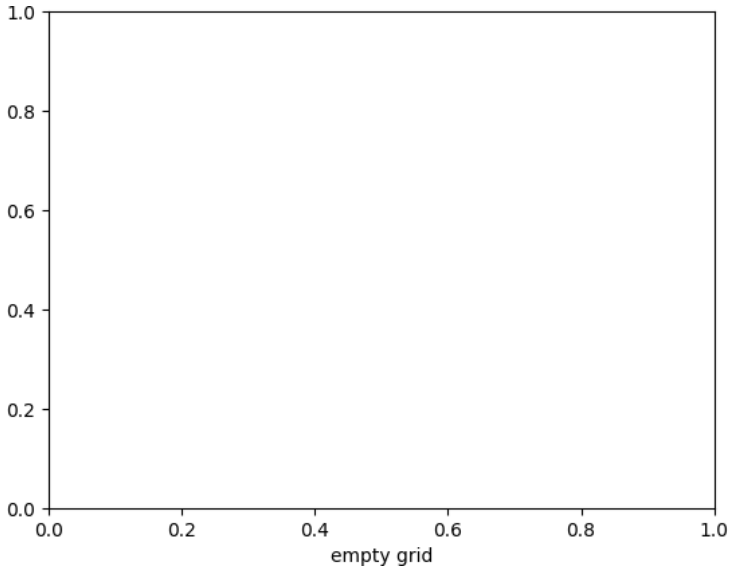
```
# print sin wave until 4pi

import numpy as np

x = np.linspace(0, 4*np.pi, 1000)
y = np.sin(x)
z = np.cos(x)
a = np.tan(x)

plt.plot(x, y, color="green", linestyle="dotted")
plt.plot(x, z, color="blue")

# Set the x-axis and y-axis limits
plt.xlim(0, 4*np.pi)
plt.ylim(-1, 1)

# Set the x-axis and y-axis labels
plt.xlabel('x')
plt.ylabel('sin(x) and cos(x)')

# Show the plot
# plt.show()
```

```
Text(0, 0.5, 'sin(x) and cos(x)')
```

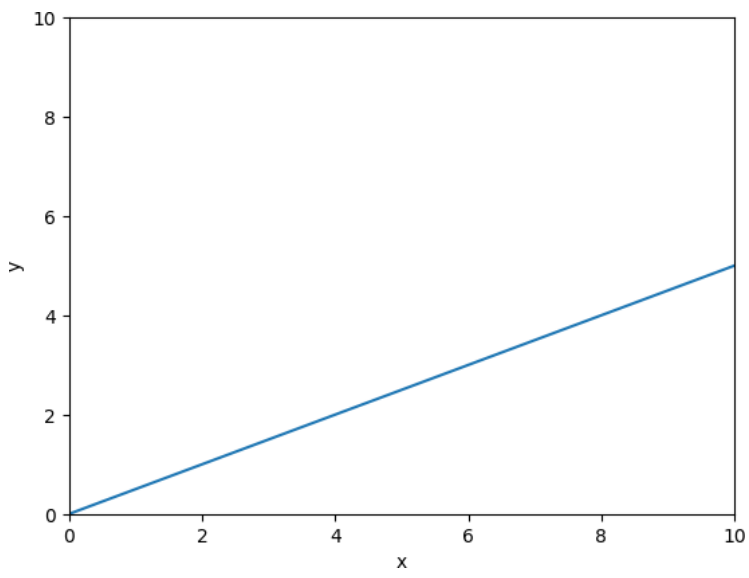



```
plt.xlabel('empty grid')
```

```
Text(0.5, 0, 'empty grid')
```

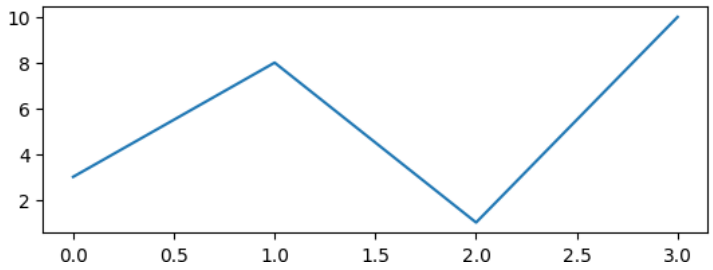



```
x = np.linspace(0, 10, 1000)
y = np.linspace(0, 5, 1000)
# plt.plot(np.sin(x), np.cos(y))
plt.plot(x, y)

# Set the x-axis and y-axis limits
plt.xlim(0, 10)
plt.ylim(0, 10)

# Set the x-axis and y-axis labels
plt.xlabel('x')
plt.ylabel('y')

# Show the plot
plt.show()
```

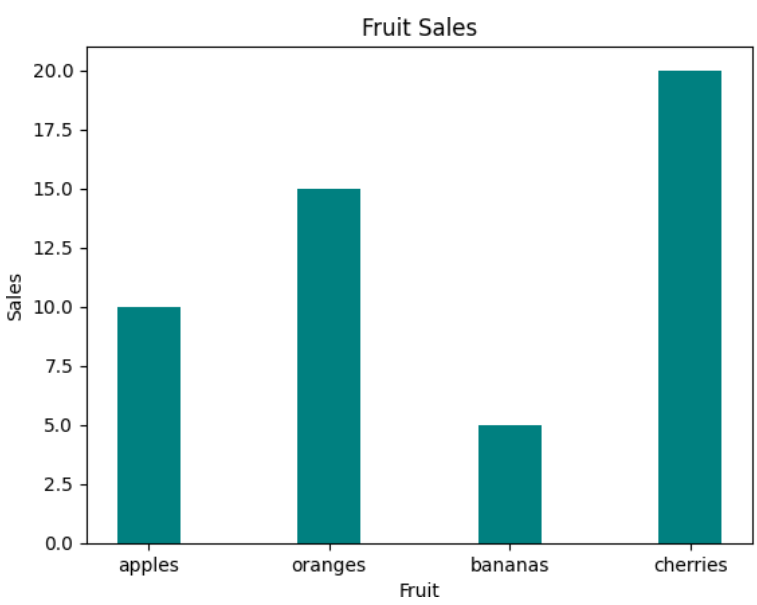



```
# printing a subplot
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(2, 1, 1)
plt.plot(x,y)

#plot 2:
#x = np.array([0, 1, 2, 3])
#y = np.array([10, 20, 30, 40])

#plt.subplot(2, 1, 2)
#plt.plot(x,y)
```

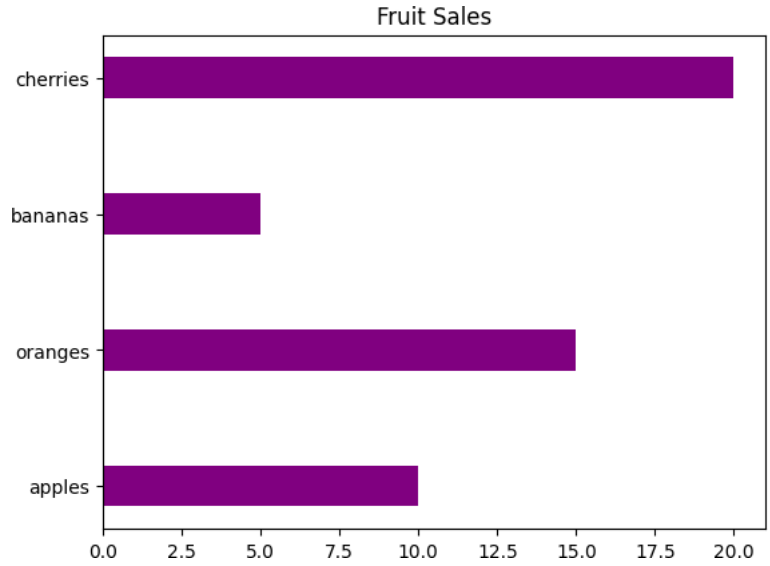
```
[<matplotlib.lines.Line2D at 0x7a4d87f00ca0>]
```




```
# barchar example with dictionary
import matplotlib.pyplot as plt

# Define the data
data = {'apples': 10, 'oranges': 15, 'bananas': 5, 'cherries': 20}

# Create a bar chart
plt.bar(list(data.keys()), list(data.values()), width=0.35, color="teal")

# Add title and axis labels
plt.title('Fruit Sales')
plt.xlabel('Fruit')
plt.ylabel('Sales')

# Show the plot
plt.show()
```

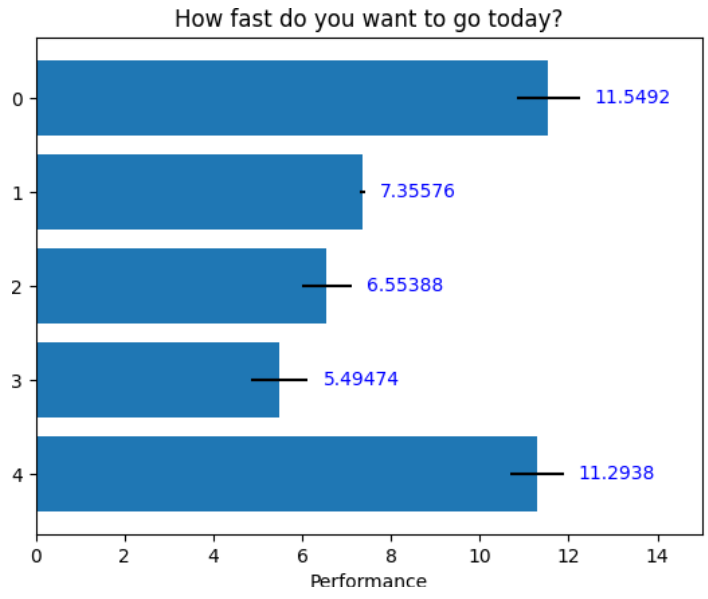



```
# example of horizontal barchart with dictionary

# Define the data
data = {'apples': 10, 'oranges': 15, 'bananas': 5, 'cherries': 20}

# Create a horizontal bar chart
plt.barh(list(data.keys()), list(data.values()), color="purple", height=0.3)

# Add title and axis labels
plt.title('Fruit  Sales')
# plt.xlabel('Sales')
# plt.ylabel('Fruit')

# Show the plot
show_plot = plt.show()
```




```
AttributeError                            Traceback (most recent call last)
<ipython-input-56-dbd46437747f> in <cell line: 16>()
     14 # Show the plot
     15 show_plot = plt.show()
---> 16 show_plot.set_xlabel('something')

AttributeError: 'NoneType' object has no attribute 'set_xlabel'
```

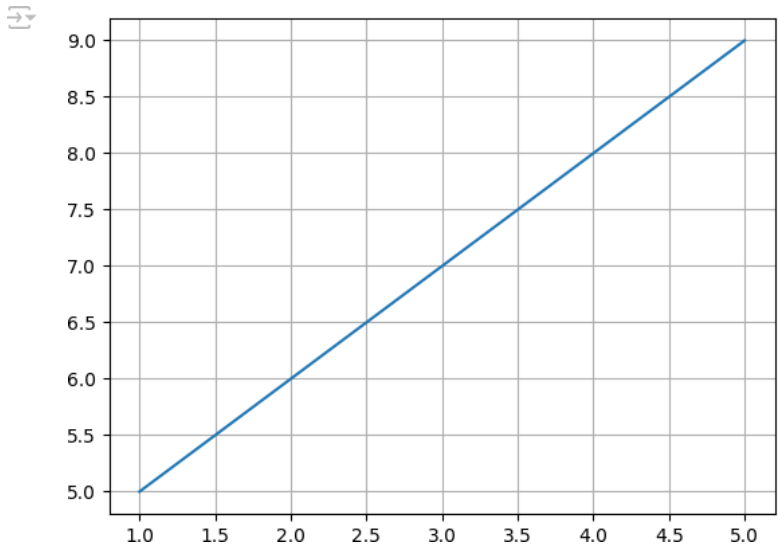
```
fig, ax = plt.subplots()

# Example data
people = ('Tom', 'Thejas', 'Harry', 'Slim', 'Jim')
y_pos = np.arange(len(people))
performance = 3 + 10 * np.random.rand(len(people))
error = np.random.rand(len(people))

hbars = ax.barh(y_pos, performance, xerr=error, align='center')
ax.invert_yaxis()
ax.set_xlabel('Performance')
ax.set_title('How fast do you want to go today?')

# Label with given captions, custom padding and annotate options
ax.bar_label(hbars, padding=8, color='b')
ax.set_xlim(right=15)

plt.show()
```




```
print(np.arange(10, 20, 2))
```

```
[10 12 14 16 18]
```

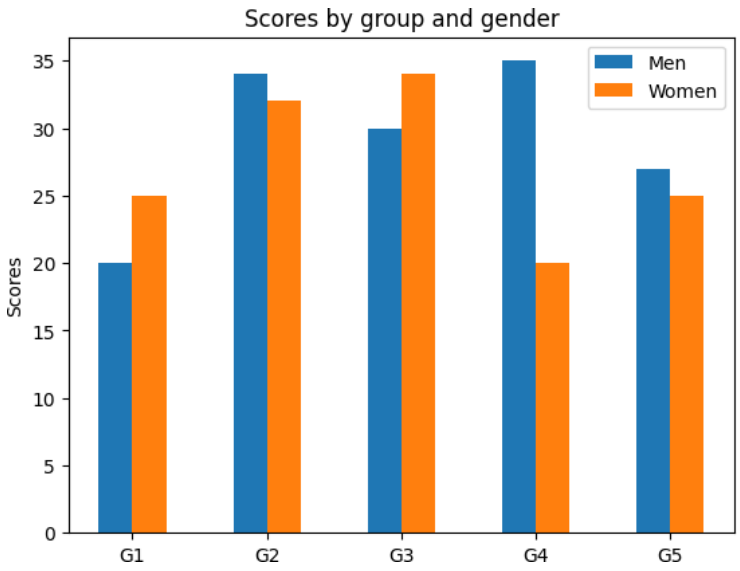
```
# pprint a axis plot with ax.grid()

import matplotlib.pyplot as plt

# Create a figure and an axes object
ax = plt.subplot()

# Plot some data
ax.plot([1, 2, 3, 4, 5], [5,6,7,8,9])

# Enable the grid
ax.grid(True)

# Show the plot
plt.show()
```



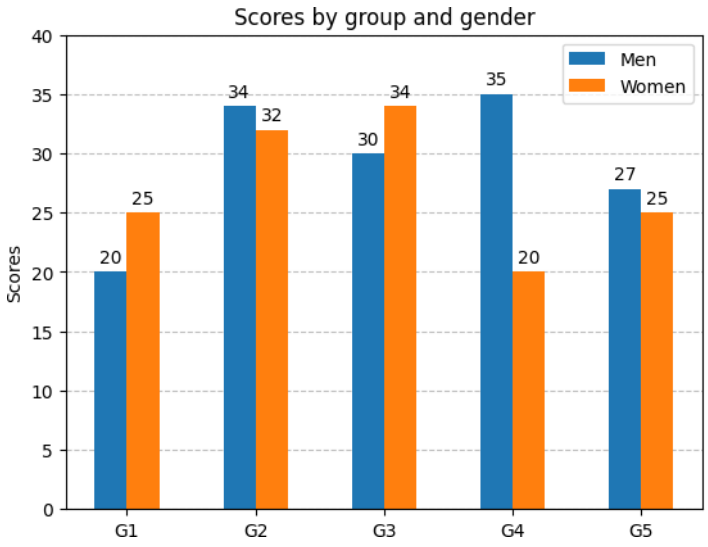

```
print(np.arange(10, 20, 2))
```

```
[10 12 14 16 18]
```

```
# grouped bar charts example
import numpy as np
import matplotlib.pyplot as plt
labels = ['G1', 'G2', 'G3', 'G4', 'G5']
men_means = [20, 34, 30, 35, 27]
women_means = [25, 32, 34, 20, 25]

x = np.arange(len(labels))
# width of the individual component
width = 0.25

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, men_means, width, label='Men')
rects2 = ax.bar(x + width/2, women_means, width, label='Women')

# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('Scores')
ax.set_title('Scores by group and gender')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend();

plt.show()
```

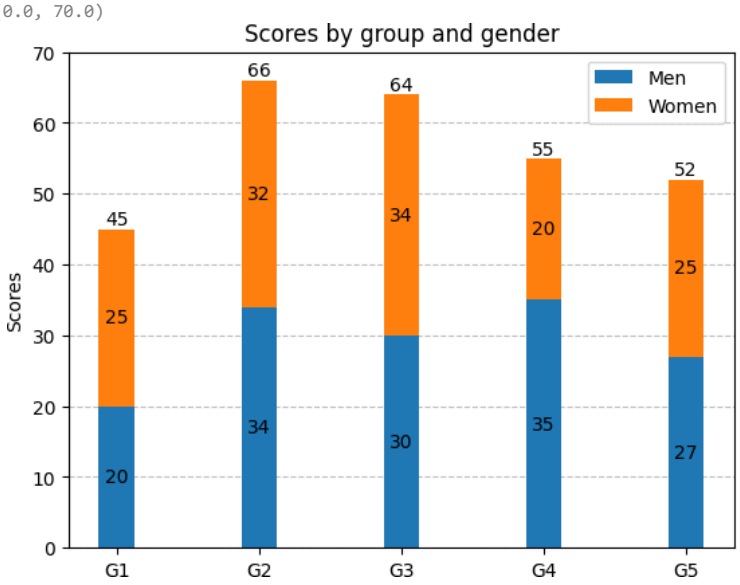



```
# adding labels to individual bars with their scores

fig, ax = plt.subplots()
ax.grid(linestyle='--', color='0.75', axis = 'y')
ax.set_axisbelow(True)

rects1 = ax.bar(x - width/2, men_means, width, label='Men')
rects2 = ax.bar(x + width/2, women_means, width, label='Women')

ax.set_ylabel('Scores')
ax.set_title('Scores by group and gender')
ax.set_xticks(x)
ax.set_xticklabels(labels)

ax.legend()

# Adding the bar labels
ax.bar_label(rects1, padding=3)
ax.bar_label(rects2, padding=3)

ax.set_ylim(0,40);
```



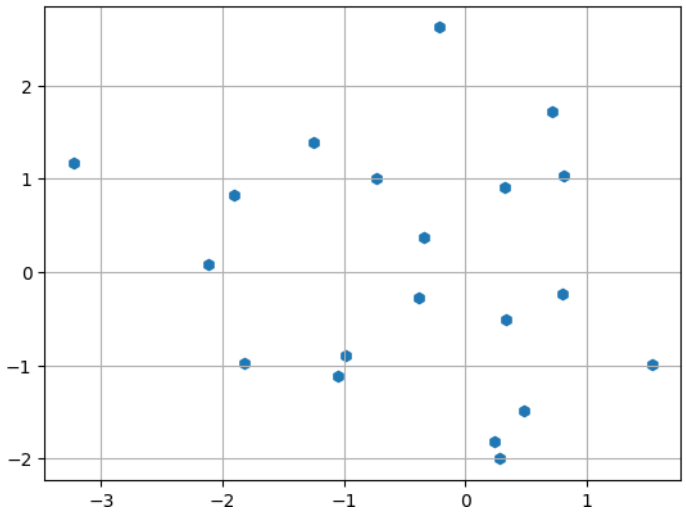
```
fig, ax = plt.subplots()
ax.grid(linestyle='--', color='0.75', axis = 'y');

ax.set_axisbelow(True) # set this to true for enabling gridlines

p1 = ax.bar(labels, men_means, width, label='Men')
p2 = ax.bar(labels, women_means, width, bottom=men_means,
       label='Women')

ax.set_ylabel('Scores')
ax.set_title('Scores by group and gender')
ax.legend()

# Label with label_type 'center'
ax.bar_label(p1, label_type='center')
ax.bar_label(p2, label_type='center')
ax.bar_label(p2)
ax.set_ylim(0,70)
```

```
(0.0, 70.0)
```

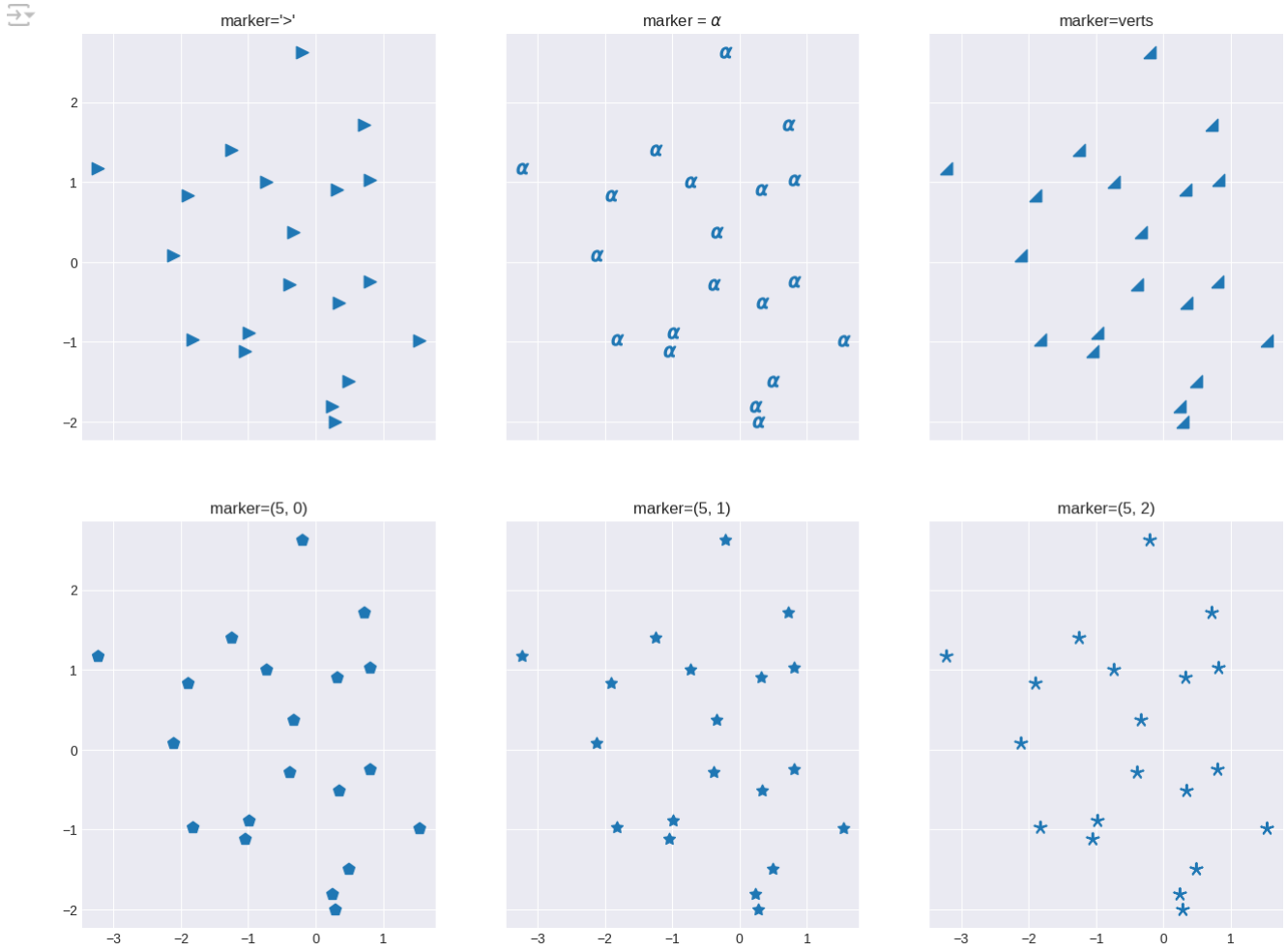



```
# scatter plot
x = np.random.randn(20)
y = np.random.randn(20)

fig, ax = plt.subplots()
ax.grid(True)
ax.scatter(x, y, marker = 'h') # can change to any marker
```

```
<matplotlib.collections.PathCollection at 0x7b29651475e0>
```

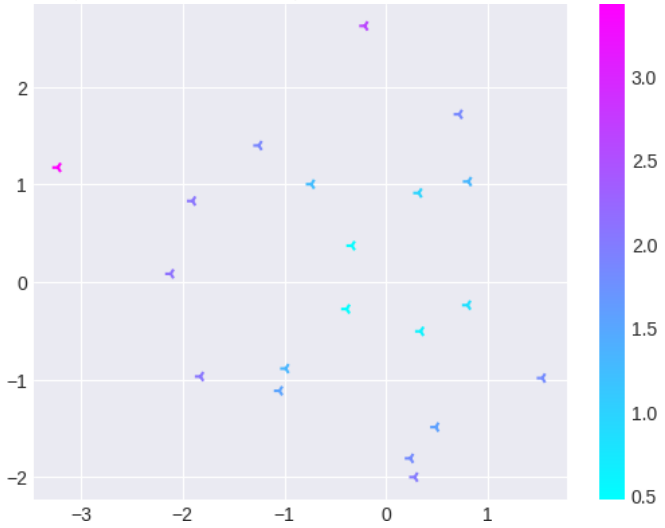


```
fig, axs = plt.subplots(2, 3, sharex=True, sharey=True, figsize=(16,12));
# plt.style.use('seaborn-darkgrid')
# marker symbol
axs[0, 0].scatter(x, y, s=80, marker=">")
axs[0, 0].set_title("marker='>'")

# marker from TeX
axs[0, 1].scatter(x, y, s=80, marker=r'$\alpha$')
axs[0, 1].set_title("marker = " + r'$\alpha$')
# axs[0, 1].set_title(f"marker = {r'$\alpha$'}")

# marker from path
verts = [[-1, -1], [1, -1], [1, 1], [-1, -1]]
axs[0, 2].scatter(x, y, s=80, marker=verts)
axs[0, 2].set_title("marker=verts")

axs[1, 0].scatter(x, y, s=80, marker=(5, 0))
axs[1, 0].set_title("marker=(5, 0)")

# regular star marker
axs[1, 1].scatter(x, y, s=80, marker=(5, 1))
axs[1, 1].set_title("marker=(5, 1)")

# regular asterisk marker
axs[1, 2].scatter(x, y, s=80, marker=(5, 2))
axs[1, 2].set_title("marker=(5, 2)");
```

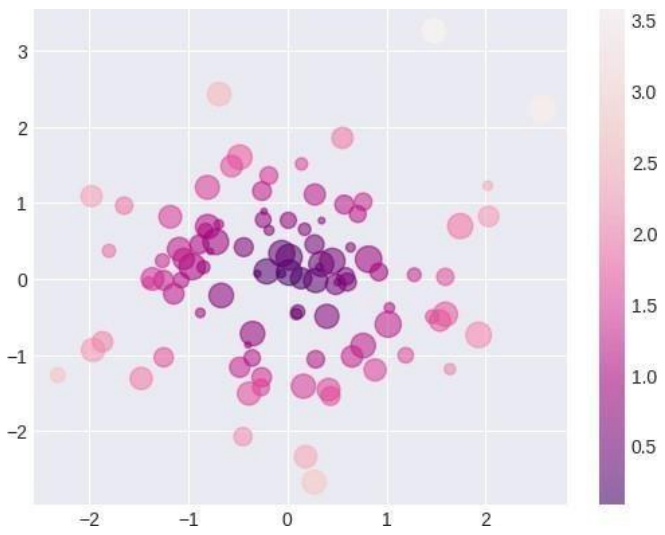


```
# setting the colors with matplotlib
plt.style.use('seaborn-darkgrid')

z1 = np.sqrt(x**2 + y**2)

fig, ax = plt.subplots()
pos = ax.scatter(x, y, c=z1, cmap='cool', marker='3')

fig.colorbar(pos);
```

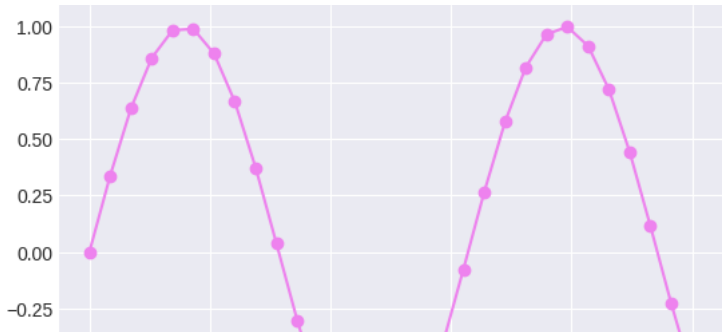
```
<ipython-input-51-3dd43bf91bb6>:2: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6,
  plt.style.use('seaborn-darkgrid')
```




```
x = np.random.randn(100)
y = np.random.randn(100)

z1 = np.sqrt(x**2 + y**2)
z2 = np.random.randint(10, 200, size=len(x))

fig, ax = plt.subplots()
# pos = ax.scatter(x, y, c=z1, s=z2, alpha = 0.55, cmap='viridis')
pos = ax.scatter(x, y, c = z1, s = z2, alpha = 0.55, cmap='RdPu_r')
fig.colorbar(pos);
```

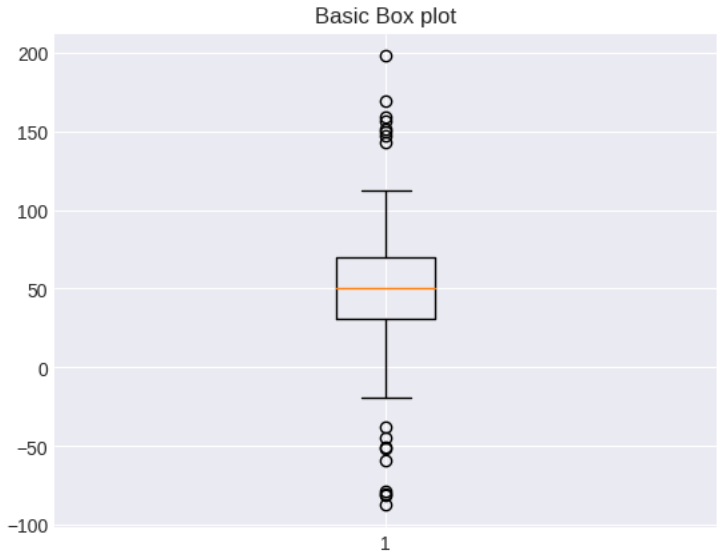



```
x = np.linspace(0, 10, 30)
y = np.sin(x)

plt.plot(x, y, 'o-', color='violet');
```




```
# Box plots

# Generating the data
spread = np.random.rand(50) * 100
center = np.ones(25) * 50
flier_high = np.random.rand(10) * 100 + 100
flier_low = np.random.rand(10) * -100
data = np.concatenate((spread, center, flier_high, flier_low))

# Visualization of the data using box plot (basic)
fig, ax = plt.subplots()
ax.boxplot(data)
ax.set_title("Basic  Box  plot")
```

```
Text(0.5, 1.0, 'Basic Box plot')
```




```
# Notched boxplot without outliers
```

LABSHEET 7

```
import pandas as pd
```

Code Text

```
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...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
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...	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...	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

```
# 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

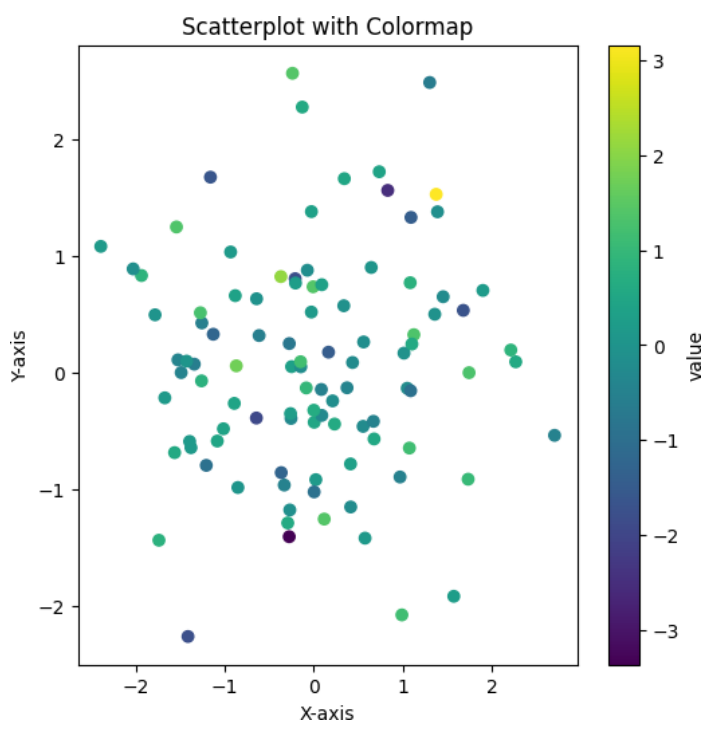
```
group_df = df.groupby('Name')
```

```
group_df
```

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

LABSHEET ®

```
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))
```

```
[ 7.25060198e-01  2.53900412e+00  1.26528031e+00  1.84136990e+00
 -2.60848832e+00 -5.59983281e-01  4.35035456e-01 -7.00367135e-02
  1.96931749e+00  1.04382097e+00 -5.23481680e-01  4.38611173e-01
 -6.03314609e-02 -1.62331938e+00 -1.75368806e-01 -1.45327854e-01
  7.11162067e-01 -1.24752326e+00  1.10879435e+00  6.15797150e-01
  3.22382085e-02 -4.94204444e-01 -1.56553377e+00  1.86476127e+00
 -1.53372917e+00  6.21845005e-01  1.08857491e+00 -1.69076421e+00
 -3.80722950e+00  4.70410313e-01  8.77562643e-01 -8.95285501e-01
  9.83561836e-01  9.32718991e-01 -6.78531171e-01  9.14953408e-05
 -2.21344622e+00 -6.15124358e-02 -9.18144802e-02  7.84013469e-01
  9.64181023e-01 -1.75737978e+00   1.19471319e+00 -1.02246958e-01
  7.73172607e-01  1.02398382e+00  1.47867589e-01 -2.44199793e+00
 -8.49499655e-01  1.88210306e-01 -2.61106287e-01 -9.53558247e-01
   -8.54821744e-01  -3.80648950e-01  -5.87306646e-01  5.54602769e-01
  1.40580004e+00  1.08580790e+00 -8.33862936e-01  7.08280769e-01
 -1.43281505e+00 -1.93642975e-01  6.86796860e-01  5.50748349e-01
  7.79495185e-01 -2.71795003e-01 -1.16407843e+00  1.38373041e+00
 -2.90569948e-01  1.27385062e+00 -4.24752220e-01  5.69263764e-01
   -1.45006382e+00  8.39335515e-01  -9.49539071e-01  -2.04611107e+00
  1.00680640e+00  2.59974257e-01 -1.29858485e+00  9.67979863e-01
 -9.72496062e-01 -1.72551385e+00 -5.42038103e-01  4.26256470e-01
```

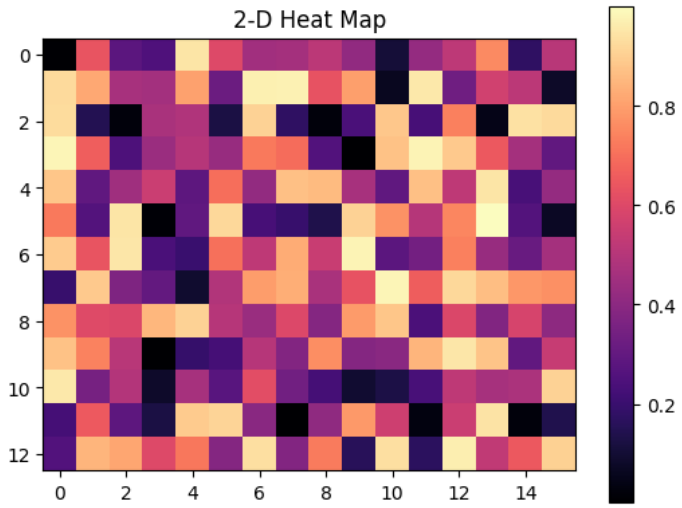
```
 6.57253328e-01  -1.75193447e+00  -1.22202143e+00  -6.31901884e-01
-9.24312354e-01  1.76235295e+00 -6.83714121e-01  5.19175365e-01
-3.18749238e-01 -1.69096151e-01 -4.49121798e-01 3.98598713e-01
 8.80300195e-01  -6.39043290e-02  -4.47122464e-01  -1.65126924e-01]
```

Start coding or generate with AI.

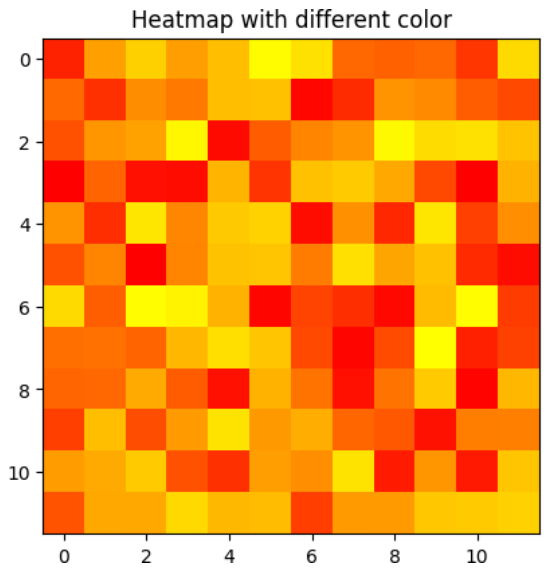
Ko"blc-click (oí c ĸcí) to cdit

LABSHEET 9

```
# Program to plot 2-D Heat map
# using matplotlib.pyplot.imshow() method
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()
```

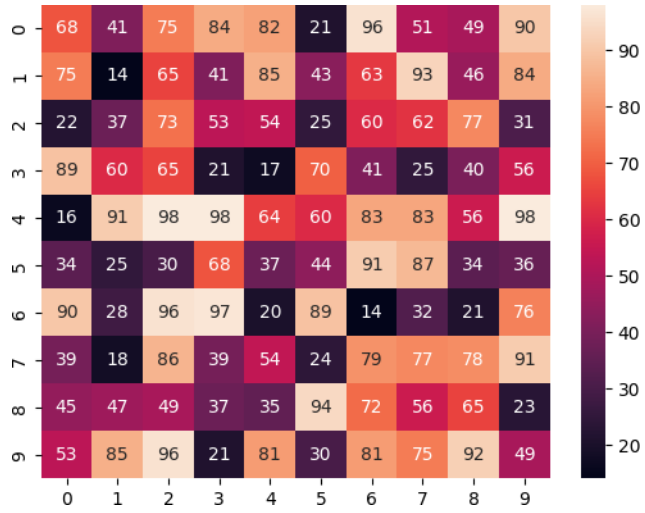



```
# Program to plot 2-D Heat map
# using matplotlib.pyplot.imshow() method
import numpy as np
import matplotlib.pyplot as plt

data = np.random.random((12, 12))
plt.imshow(data, cmap='autumn')

plt.title("Heatmap with different color")
plt.show()
```




```
# importing the modules
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# generating 2-D 10x10 matrix of random numbers
# from 1 to 100
 data = np.random.randint(low=14,
                          high=100,
```

```
                      size=(10, 10))
# plotting the heatmap
hm = sns.heatmap(data=data,annot=True)
# displaying the plotted heatmap
plt.show()
```



All tkc IPQtko Notcbooks i **PQtko Scaboí Modulc** lcct"íc scíics bQ **Dí. Milaa Paíínaí** aíc a:ailablc @ **GitHub**

LABSHEET 10

Open in Colab

Scaboí Coloí Palcttcs

Coloí is a "tmost impoíta t aspcct oí íig"íc stQli g bcca"sc it íc:cals pattcí i tkc data i "scd cííccti:clQ; oí kidc tkosc pattcí s ií "scd pooílQ. E:c pioícssio als oítc ass"mc "sagc oí coloí to poítíaQ data as a sol:cd píoblcm. l'kcQ j"st pick a palcttc ííom a díop-dow mc " (píobablQ citkcí agíaQscalc íamp oí aíai bow), sct staít a d c d poi ts & íi allQ pícss appdQ.

B"t it is t tkat simplc a dtk"s ma Q :is"alizatio s íail to ícpícsc t tkc "dcílQi g data as appíopíiatclQ as tkcQco"ld.

PíimaíQ objccti:c witk ckoicc oí coloí is to ill"mi atc datapoi ts tkat aíc co ccalcdi k"gc datascts.Q"oti g Robcft Simmo :

> Altko"gk tkc basics aíc stíaigktíoíwaíd, a "mbcí oí iss"c complicatc coloí ckoiccsi :is"alizatio . Amo g tkcm: l'kc íclatio skip bctwcc tkc ligkt wc scc a d tkc coloíswc pcícci:c is cxtícmclQ complicatcd. l'kcíc aíc m"ltiplc tQpcs oí data, cack s"itcdto a diíícíc t coloí sckcmc. A sig iíica t "mbcí oí pcoplc (mostlQ mc), aíc coloí bli d. AíbitíaíQ coloí ckoiccs ca bc co í"si g íoí :icwcís " íamilíaí witk a data sct. Ligktcoloís o a daík íicld aíc pcícci:cd diíícíc tlQ tka daík coloís o a bíigkt íicld, wkick ca complicatc som c :is"alizatio tasks, s"ck as taígct dctcctio .

Oc oí tkc most í" damc tal a d impoíta t aspccts oí coloí sclcctio is tkc mappi g oí "mbcís to coloís. l'kis mappi g allows "s to psc"docoloí a imagc oí objcct bascd o :aíQi g "mcíical data. BQ íaí, tkc most commo colo map "scd i scic tiic :is"alizatio is tkc íai bow coloímap. Rcscaíck papcí o **Di:cígi g Coloí Maps foí Scic tific Visualizatio** bQ Kc ctk Moícla d :cíQ wcll dcals witk tkc cxtc dcd coloí co ccpts, ií tkc topic i tcícsts Qo" íoí í"ítkc a alQsis.

Witk all tkat bcc said, lct "s ow íoc"s o wkat Scaboí kas to oíící BUl' bcioíc doi g tkat lct mc o ccagai ícmi d Qo" tkat Scaboí í" s o top oí Matplotlib so a Q coloí tkat is s"ppoítcd bQ **Matplotlib** will bc s"ppoítcd bQ Scaboí as wcll. So at ííist, lct "s" dcísta d wkat Matplotlib kas to oíící:

- a RGB oí RGBA t"plc oí íloat :al"cs i [0, 1] (c.g., (0.1, 0.2, 0.5) oí (0.1, 0.2, 0.5, 0.«))a
- kcx RGB oí RGBA stíi g (c.g., ' 0I#I0I' oí ' 0I0#0I0I')
- a stíi g ícpícsc tatio o a íloat :al"c i [0, 1] i cl"sic íoí gíaQ lc:cl (c.g., '0.5') o c oí D'b','g', 'í', 'c', 'm', 'Q', 'k', 'w'«
- a X11/CSS4 coloí amc
- a amc ííom tkc xkcd coloí s"í:cQ pícíixcd witk 'xkcd:' (c.g., 'xkcd:skQ bl"c')o c oí D'C0', 'C1', 'C2', 'C«', 'C4', 'C5', 'C6', 'C7', 'C®', 'C9'«
- o c oí D'tab:bl"c', 'tab:oía g', 'tab:gícc ', 'tab:ícd', 'tab:p"íplc', 'tab:bíow ', 'tab:pi k', 'tab:gíaQ', 'tab:oli:c', 'tab:cQa '« wkick aíc tkc **l'ablcau** Coloís ííom tkc 'l'10' catcgoíical palcttc (wkick is tkc dcía"lt coloí cQclc).

Notc tkat all stíi g spcciíicatio s o coloí, otkcí tka "CN"aíc NOl' casc-sc siti:c. L ct "s bíicílQ gotkío"gk a co"plc oí commo s"ppoítcd coloís kcíc:

- RGB/RGBA t"plcs aíc 4-t"plcs wkcíc tkc ícspccti:c t"plc compo c ts icpicsc t Rcd, Gícc , Bl"c, a d Alpka (opacitQ) :al"cs íoí a coloí. Eack :al"c is a íloati g poi t "mbcí bctwcc 0.0 a d 1.0. Ioí cxamplc, tkc t"plc (1, 0, 0, 1) ícpícsc ts a opaq"c ícd, wkilc (0,1, 0, 0.5) ícpícsc ts a kalí tía spaíc t gícc .
- l'kis is act"allQ a otkcí waQ oí ícpícsc ti g RGBA codcs a d commo Coloí Co císio Calc"latoís ca c "scd to tía slatic :al"cs. Hcíc is a **Hcx to RGBA** a d **RGB to Hcx** Coloíco :cítcí íoí Qo"í í"t"íc assista cc.
- Kictio aíQ oí :al"cs ííom D'C0', 'C1', 'C2', 'C«', 'C4', 'C5', 'C6', 'C7', 'C®', 'C9'« ícpícsc t **ColoíQua tizatio** . I kac attackcd a li k i tkc pío idcd otcbook tkat skall g"idc Qo" to a o li cbook wkícío Pagc-29 Qo" co"ld íi d spcciíics.

MQ solc p"íposc oí kccpi g Qo" postcd oí Matplotlib backgío" d c:cíQ ow a d tkc is o lQ to c s"íc tkatwkc Qo" gct to píod"ctio -lc:cl a d tíQ to c"stomizc a plot as pcí Qo"í a alQsis, Qo" sko"ld k ow wkat is ACl'U AL Y í" i g i tkc backgío" d l'kis skall cmpow cí Qo" to accoídi glQ twcak paíamctcís kcíc a d tkcíc. L ct "s ow look i to ícw Scaboí optio s íoí coloís:

```
# Importing required Libraries:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# Setting a figure size for all the plots we shall be drawing in this kernel:
sns.set(rc={"figure.figsize": (6, 6)})
```

B"ildi g coloí palcttcs:

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

l’kc most impoíta t f" ctio íoí woíki g wtk discíctc coloí palcttcs is **color_palette()** . l’kis í" ctio pí:idcs a i tcíiacc to ma Q (tko"gk ot all) oí tkc possiblc waQs Qo" ca gc cíatc coloís i scaboí , a d it’s"scd i tcí allQ bQ a Q í" ctio tkat kas a **palette** aíg"mc t (a d i somc cascs íoí a **color** aíg"mc t wkc m"ltiplc coloís aíc ccdcd).

color_palette() will acccpt tkc amc oí a Q scaboí palcttc oí matplotlib coloímap (cxccpt **jet** , wkick Qo" sko"ld c:cí "sc). It ca also takc a list oí coloís spcciíicd i a Q alid matplotlib íoímat (RGB t"plcs, kcx coloí codcs, oí H1?M coloí amcs). l’kc íct"f :al"c is alwaQs a list oí RGB t"plcs.

If allQ, calli g **color_palette()** witk o aíg"mc ts will íct"í tkc c"ííc t dcía"lt coloí cQclc.

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

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

Lct mc cxplai kcsc Q"alitati:c (oí catcgoíical) palcttcs. l’kcsc aíc bcst wkc Qo" wa t to distig"isk discíctc ck" ksoí data tkat do ot kac a i kcíc toídcíi g. IdcallQ, wkc impoíti g Scaboí , tkcdcía"lt coloí cQclc is cka gcd to a sct oí six coloís tkat c:okc tkc sta daíd matplotlib coloí cQclc. B"t wkc wc ka:c moíc tka 6, saQ ® catcgoíics i o"f data to disti g"isk, tkc tkc most commo waQ is "sig **hls** coloí spacc, wkick is a simplc tía síoímatio oí RGB :al"cs.

l’kc tkcíc is also **hls_palette()** í" cio tkat lcts Qo" co tíol tkc ligkt css a d sat"ratio oí coloís.

All oí it displaQcd abo:c is j"st tkc basic Scaboí acstkctics. Lct "s ow look at xkcd_ígb dictio aíQtkat kas 954 coloís i it. Lct "s tíQ to p"ll a ícw o"t oí it:

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

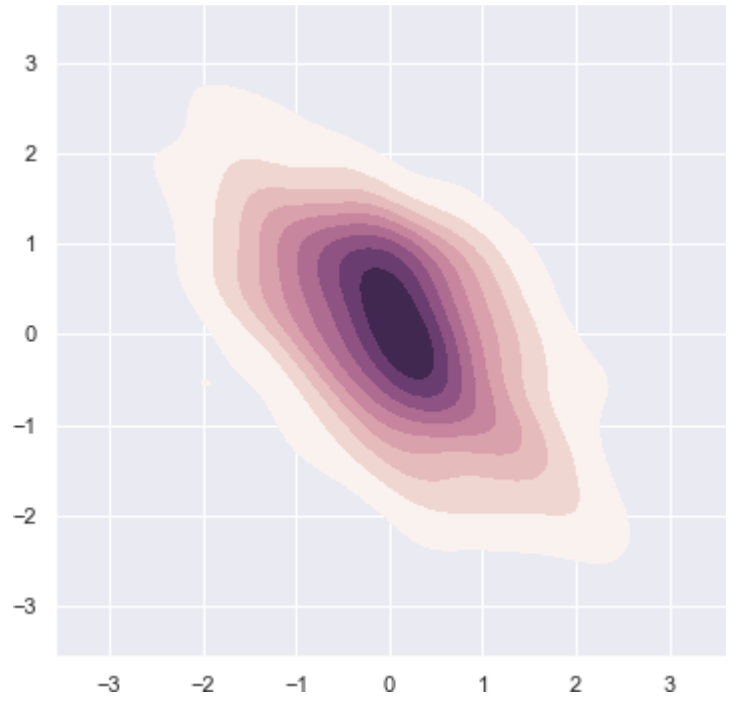
Otkcí stQlc is **cubehelix** coloí palcttc tkat makcs scq"c tial p□alcttcs witk a li caí i d□casc□oí dccícasci bíigkt d□ss a d s□omc :a□atio i **kuc**. Act"a□lQ lct "s plot tkis coloí palcttc i a Kc sitQ co to□í plot□ □

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

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

```
# Density Plot with Seaborn defaults:
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, y, cmap=sample_cmap, shade=True)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
  warnings.warn(
<AxesSubplot:>
```



I tcíacti:c widgct to cícatc a scq"c tial c"bckclix palcttc:

Lct "s w plaQ witk tkc paíamctcís to ka:c somc í" a d ckoosc bcst paíamctcís:

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

```
NameError                                 Traceback (most recent call last)
<ipython-input-1-230a1c9055e9> in <cell line: 1>()
----> 1 sns.choose_cubehelix_palette(as_cmap=True)

NameError: name 'sns' is not defined
```

Notc tkat tkis app o lQ woíks i tkis J"pQtcí Notcbook as oí ow to kclp ckoosc bcstpaíamctcís íoí o"í plot:

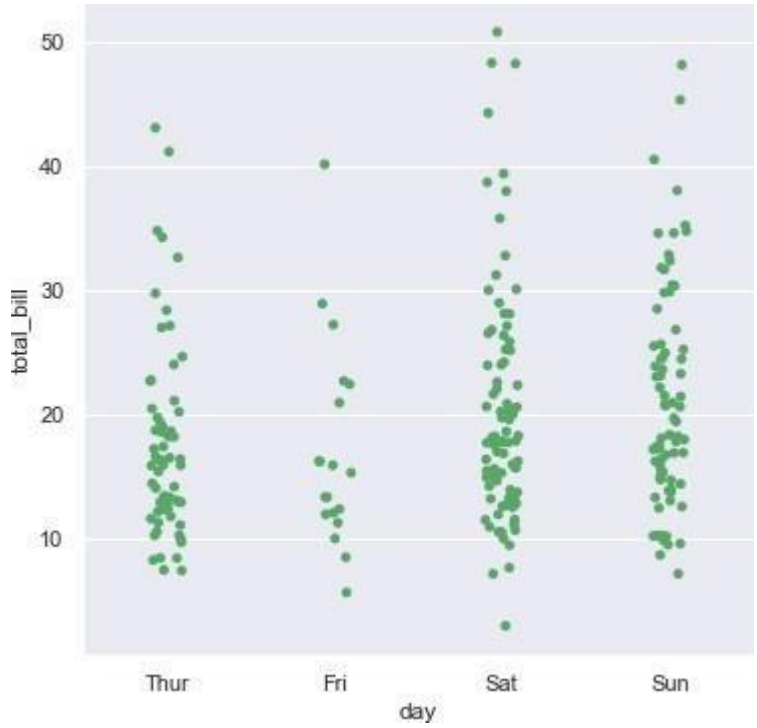
```
sns.palplot(sns.cubehelix_palette(n_colors=8, start=1.7, rot=0.2, dark=0, light=.95, reverse=Tru
```

staít is alwaQs bctwcc 0 a d «íot a abbíc:iatio íoí íoatio is kcpt bctwcc -1 a d 1. íc:císc co :císcs tkccoloí oídcíi g a d k"c ícíícís to plot appcaía cc.

Gcíic Scaboí Plots:

```
# Loading up built-in dataset:
tips = sns.load_dataset("tips")

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

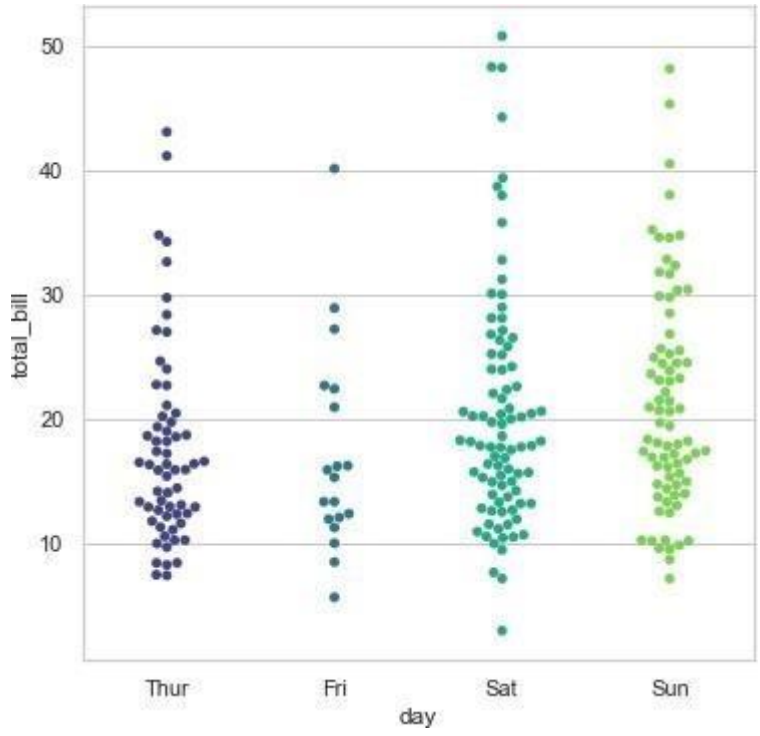
```
<AxesSubplot:xlabel='day', ylabel='total_bill'>
```



l'kis docs tkc job íoí "s b"t lct "s tíQ to gct bcttcí ícs"lts bQ plotti g cadk daQ i diíícíct coloí i stcad oísamc coloí. Ioí tkis, wc skall ícplacc **`color`** paíamctcí witk **`palette`** paíamctcí:

```
# Set Theme:
sns.set_style('whitegrid')

# Creating Strip plot for day-wise revenue:
sns.swarmplot(x="day", y="total_bill", data=tips, palette="viridis")
```

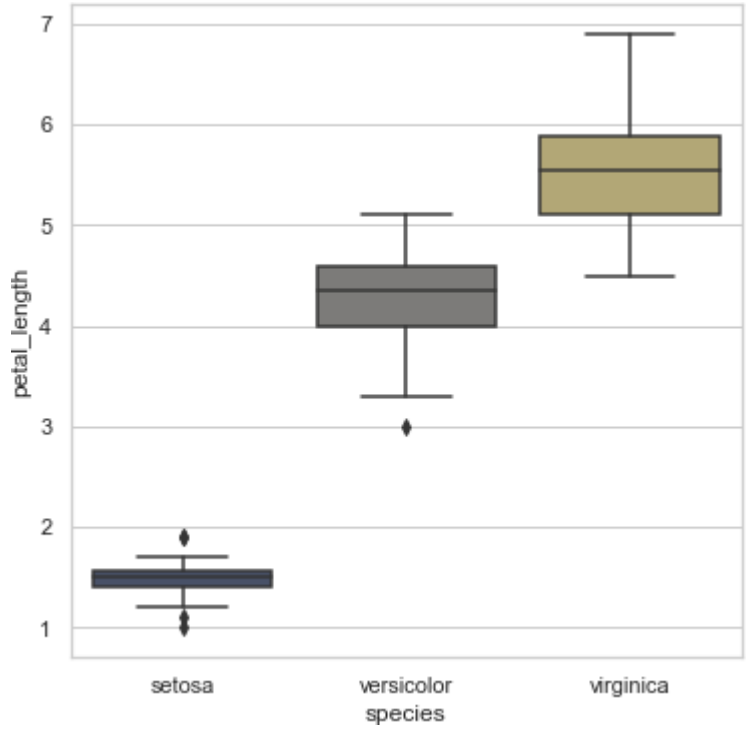
```
<AxesSubplot:xlabel='day', ylabel='total_bill'>
```



SimilaílQ, lct "s plot o moíc a doí a cka gc tkis timc wc skall plot a Violi plot:

```
iris = sns.load_dataset("iris")

sns.boxplot(x="species", y="petal_length", data=iris, palette="cividis")
```

```
<AxesSubplot:xlabel='species', ylabel='petal_length'>
```



l'kcíc aíc m"ltiplc s"ck palcttc a:ailablc íoí "s to plaQ aío" d wtk likc magma, waím gícQ,g" mtal, d"skQ bl"c, cool bl"c, dccp tcal, :iíidia , twiligkt bl"c a d ma Q moíc. loí c"stomizcd

coloí bícwi g, wc maQ also "sc coloí bícwcí tkat also oíícís i tcícsti g coloí palcttcs íoíwoíki g witk Q"alitati:c data. l'kc cool tki g abo"t it is tkat Qo" ca "sc tkc a i tcíacti:c IpQtko widgct í" ctio to makc tkc sclcctio oí tkc palcttc. Ioí tkis, Qo" o lQ ccd to "sc `choose_colorbrewer_palette()`.

l'kcíc aíc m"ltiplc s"ck palcttc a:ailablc íoí "s to plaQ aío" d witk likc magma, waím gícQ, g"mctal, d"skQ bl"c, cool bl"c, dccp tcal, :iíidia, twilight bl"c a d maQ moíc. Ioí c"stomizcd coloíbícwi g, wc maQ also "sc coloí bícwcí tkat also oíícís i tcícsti g coloí palcttcs íoí woíki g witk Q"alitati:c data. A icc ícat"íc oí tkc **Coloí Bícwcí wcbsitc** is tkat it pío:idcs somc g"ida cc o wkick palcttcs aíc coloí bli d saíc.

l'kc cool tki g abo"t it is tkat Qo" ca "sc tkc a i tcíacti:c IpQtko widgct í" ctio to makc tkc sclcctio oítkc palcttc. Ioí tkis, Qo" o lQ ccd to "sc `choose_colorbrewer_palette()`. l'o acccss tkis o Qo"í wcb bíowscí, plcasc acccss **ColoíBícwcí** li k pío:idcd i tkc otcbook.

I also ío" d a icc ícpícsc taío oí Coloí Sckcmcs i Scaboí, tkat líío" d somcwkcíc o wcb, sotko"gktoí skaíi g it i Qoí Rcso"ícc b"ckct to ckcck o"t ií Qo" wisk to. Lct's ka:c a look at it

ĻABSHEEĽ 11

```
#Installation
#pip install seaborn
```

Scaboí 2

Iig"íc

It ícícís to tkc wkolc íig"íc tkat Qo" scc. It is possiblc to ka:c m"ltiplc s"b-plots (Axcs) i tkc samc íig"íc.

Axcs

A Axcs ícícís to tkc act"al plot i tkc íig"íc. A íig"íc ca kac m"ltiplc Axcs b"t a gi:c Axcs ca bc pak oí o lQ o c ig"íc.

Axis

A Axis ícícís to a act"al axis (x-axis/Q-axis) i a spcciíic plot.

Io"í s"b-plots (Axcs) i asi gc íig"íc.

Scaboí

Scaboí ca cc catc complicatcd plot tQpcs íiom Pa das data witk íclati:clQ simplc comma ds Plotti g scaboí is

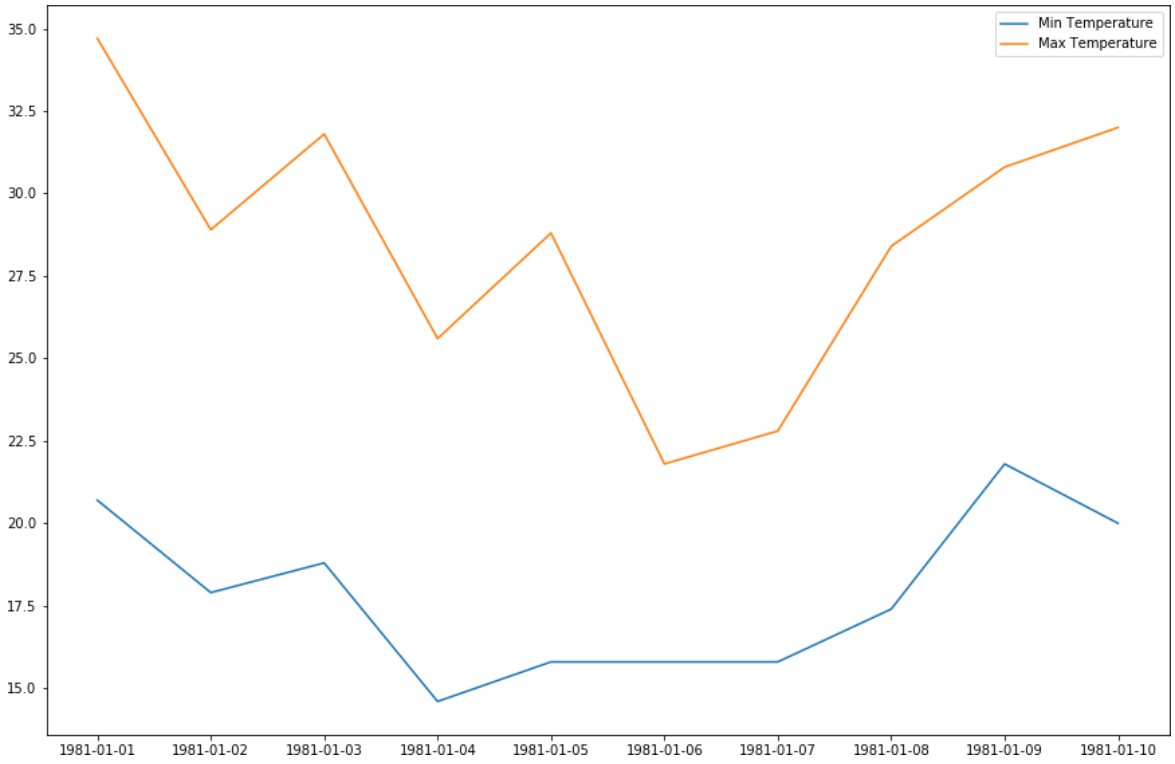
citkcí : Axcs-lc:cl í" ctio s OR ig"íc lc:cl í" ctio

PĻOĽ CAĽEGORIES IN SEABORN

I. Rclatio al plots : l'kis plot is "scd to " dc sta d tkc íclatio bctwcc two : íiablcs.
II. Catcgoíical plots: l'kis plot dcals witk catcgoíical :aíiablcs a d kow tkcQ ca bc :is"alizcd.
III. Distíibutio plots: l'kis plot is "scd íoí cxami i g" :a atc a d bia íiatc distíib"tio s
IV. Matíix plots: A matíix plot is a aaQ oí scattcíplots.
V. Rcgícssio plots: l'kc ícgícssio plo s i scabo í aíc pía maíilQ i tc dcd o add a :is"al g"idc tkat kclps to cmpkasizc pattcí s i a datasct d i g cxploíatoíQ da a a alQscs.

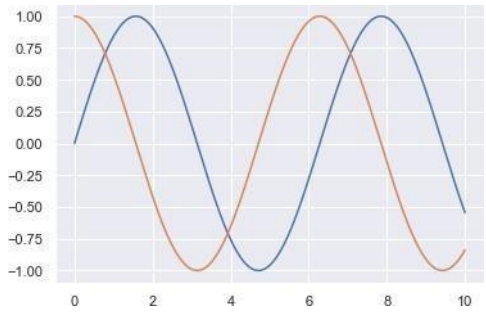
```
#Import necessary Packages
import numpy  as  np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure

import seaborn as sns

%matplotlib inline
```

```
#Simple Plotting with Seaborn

#Data
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, 18.8, 14.6, 15.8, 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]

#Plotting
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()
```



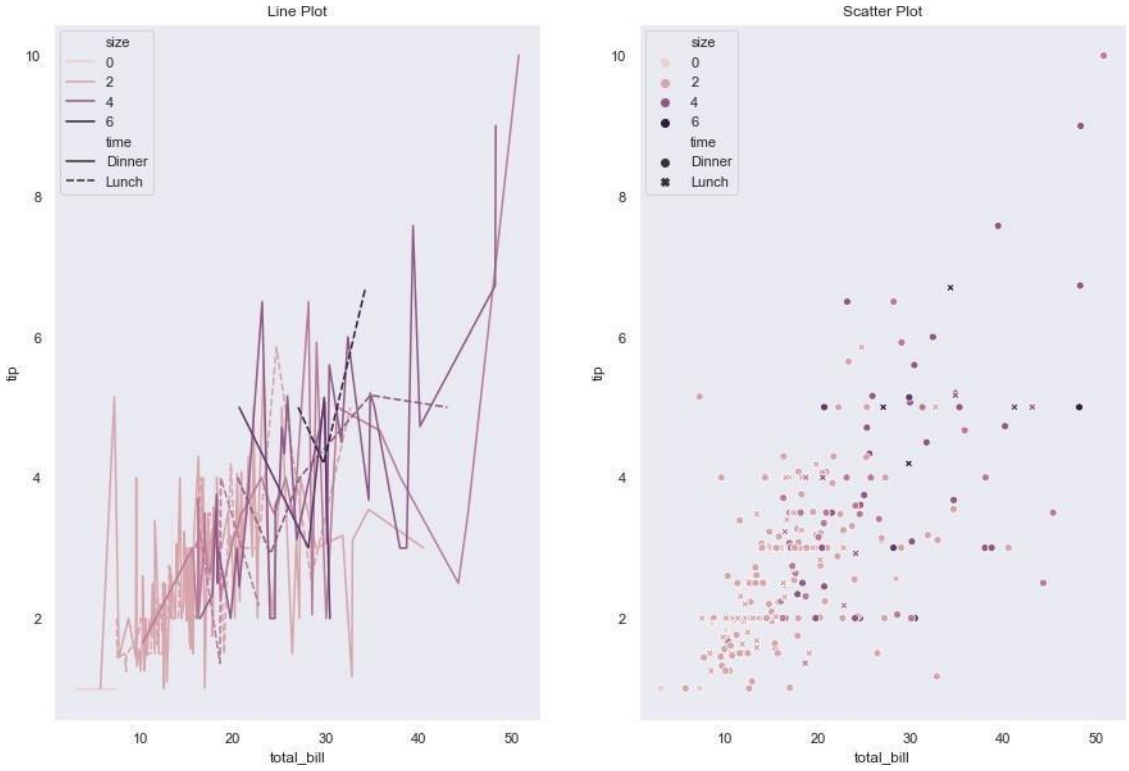

```
<matplotlib.legend.Legend at 0x1c0d2b24748>
```

```
#seaborn style as the default matplotlib style
sns.set()

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



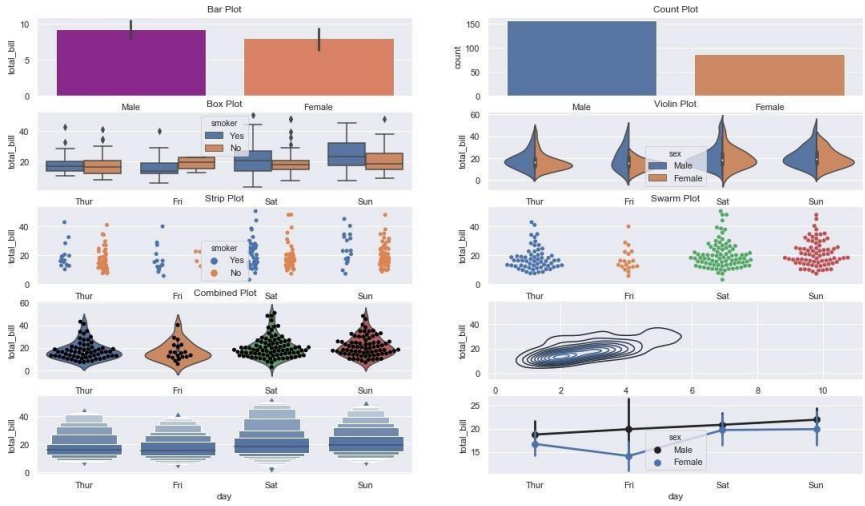
```
# I. Relational Plots

# Line plot : The line plot is one of the most basic plot in seaborn library.
#This plot is mainly used to visualize the data in form of some time series, i.e. in continuous manner.
sns.set(style="dark")
fig, ax = plt.subplots(ncols=2, nrows=1, figsize=(15,10))

#Loading Data with Seaborn
df = sns.load_dataset("tips")

print(df.head())

#lineplot
sns.lineplot(x="total_bill", y="tip", hue="size", style="time", data=df,ax=ax[0]).set_title("Line Plot")


#scatterplot
Sct_plt=sns.scatterplot(x="total_bill", y="tip", hue="size", style="time", data=df,ax=ax[1]).set_title("Scatter Plot")

#Saving Plot
Sct_plt.figure.savefig('Scatter_plot1.png')
print('Plot Saved')
```

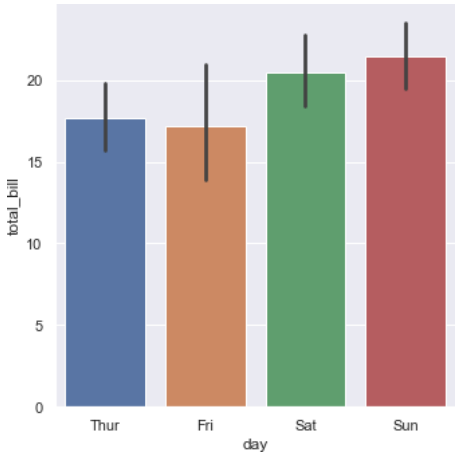
```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female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



```
#II. Categorical Plots
#Plots are basically used for visualizing the relationship between variables.
#Variables can be either be completely numerical or a category like a group, class or division.

sns.set_style('darkgrid')
fig, ax =plt.subplots(nrows=5,ncols=2)
fig.set_size_inches(18.5, 10.5)

#Data
# 'tips' dataset contains information about people who probably had food at a restaurant
#  whether or not they left a tip for the waiters, their gender, whether they smoke and so on.
df = sns.load_dataset('tips')

#barplot - basically used to aggregate the categorical data according to some methods and by default its the mean
sns.barplot(x ='sex', y ='total_bill', data = df,palette ='plasma', estimator = np.std,ax=ax[0,0]).set_title('Bar Plot')


#countplot -Counts the categories and returns a count of their occurrences
sns.countplot(x ='sex', data = df,ax=ax[0,1]).set_title('Count Plot')

#boxplot - known as the box and whisker plot.
#It shows the distribution of the quantitative data that represents the comparisons between variables
sns.boxplot(x ='day', y ='total_bill', data = df, hue ='smoker',ax=ax[1,0]).set_title('Box Plot')

# Similar to the boxplot except that it provides a higher, more advanced visualization
# Uses the kernel density estimation to give a better description about the data distribution.
sns.violinplot(x ='day', y ='total_bill', data = df, hue ='sex', split = True,ax=ax[1,1]).set_title('Violin Plot')

#Stripplot -  scatter plot based on the category
sns.stripplot(x ='day', y ='total_bill', data = df, jitter = True, hue ='smoker', dodge = True,ax=ax[2,0]).set_title('Strip Plot')

#Swarmplot-similar to stripplot except the fact that the points are adjusted so that they do not overlap.
sns.swarmplot(x ='day', y ='total_bill', data = df,ax=ax[2,1]).set_title('Swarm Plot')

#Combining the idea of a violin plot and a stripplot to form this plot
sns.violinplot(x ='day', y ='total_bill', data = df,ax=ax[3,0])
sns.swarmplot(x ='day', y ='total_bill', data = df, color ='black',ax=ax[3,0]).set_title('Combined Plot')

# Density Plot
sns.kdeplot(df['tip'], df['total_bill'],ax=ax[3,1])

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

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

#catplot
#General plot -  provides a parameter called 'kind' to choose the kind of plot ,better that writing the plots separately.
#The kind parameter can be bar, violin, swarm etc.
sns.catplot(x ='day', y ='total_bill', data = df, kind ='bar')
```

```
<seaborn.axisgrid.FacetGrid at 0x1c0d3615888>
```



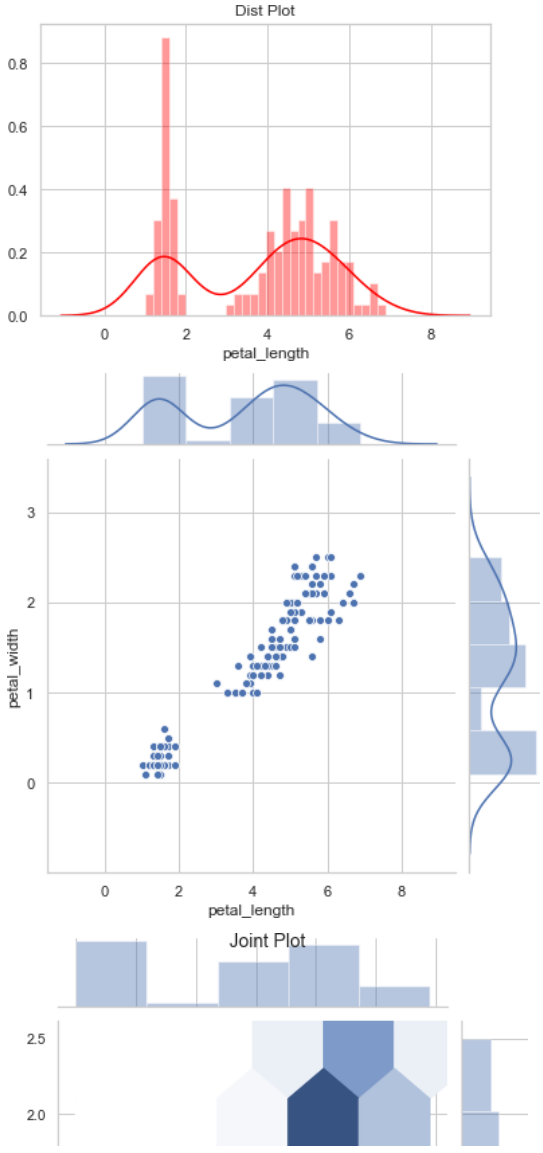


III. **Distúibutio ßplots** i ßscaboí iß "scd íoí cxami i g ßß:aíßatc a d bißáiiatc distúib"tio s. 4 maßi tQpcs ßí distúib"tio plots : ß

```
joinplot
distplot
pairplot
rugplot
```

```
sns.set_style('whitegrid')


#Data - 'iris'
df = sns.load_dataset('iris')
print(df.head())

#Displot- used for univariant set of observations and visualizes it through a histogram
#i.e. only one observation and hence we choose one particular column of the dataset.
#KDE is a way to estimate the probability density function (PDF) of the random variable that “underlies” the sample.
#KDE is a means of data smoothing.
#bins is used to set the number of bins you want in your plot and it actually depends on your dataset.
#color is used to specify the color of the plot
sns.distplot(df['petal_length'], kde = True, color ='red', bins = 30).set_title('Dist Plot')

#Joinplot/jointgrid- draw a plot of two variables with bivariate and univariate graphs. It basically combines two different plots.
#Plot a bi-variate distribution along with marginal distributions in the same plot
#Joint Distribution of two variables can be visualised using scatter plot/regplot or kdeplot.
#Marginal Distribution of variables can be visualised by histograms and/or kde plot
#KDE shows the density where the points match up the most
#The Axes-level function to use for joint distribution must be passed to JointGrid.plot_joint().
#The Axes-level function to use for marginal distribution must be passed to JointGrid.plot_marginals()

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

#jointplot() to plot bi-variate distribution along with marginal distributions.
#It uses JointGrid() and JointGrid.plot_joint() in the background.
g=sns.jointplot(x = 'petal_length',y = 'petal_width',data = df,kind = 'hex')
g.fig.suptitle('Joint Plot')

#Pairplot-  pairwise relation across the entire dataframe
#hue sets up the categorical separation between the entries in the dataset.
#palette is used for designing the plots.
g=sns.pairplot(df, hue ="species", palette ='coolwarm')
g.fig.suptitle("Pair Plot 1")
g.add_legend()


#PairGrid() - creates Axes for each pair of variables
#PairGrid.map() - draws the plot on each Axes using data corresponding to that pair of variables
pairgrid = sns.PairGrid(data=df)
pairgrid = pairgrid.map_offdiag(sns.scatterplot)
pairgrid = pairgrid.map_diag(plt.hist)

#Different kind of plots on Upper Triangular Axes, Diagonal Axes and Lower Triangular Axes.
pairgrid = sns.PairGrid(data=df)
pairgrid = pairgrid.map_upper(sns.scatterplot)
pairgrid = pairgrid.map_diag(plt.hist)
pairgrid = pairgrid.map_lower(sns.kdeplot)

#Avoid Redundancy
g = sns.PairGrid(df, diag_sharey=False, corner=True)
g.map_lower(sns.scatterplot)
g.map_diag(sns.kdeplot)
```

```
   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
<seaborn.axisgrid.PairGrid at 0x1c0d2b15888>
```



ĻABSHEEĪ’ 12

Load tkc Pacakgcs

Ī’o gct staítcd, opc a Colab otcBook a d loaíd tkc Pa das, Matplotlib, a d Woíd lo"d packagcs.

Code Text

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

Mo"t tkc díi:c a d ícad tkc CSV íilc íiom tkc díi:c.

Hcíc wc aíc goi g to "sc ctíIx_titlcs.cs: datasct dow loadcdíiom kagglc. Si cc it is tcx

:is"alizatio wc aic goi g to co sidcí o IQ o c c l"m

```
from google.colab import drive
```

```
drive.mount('/content/drive/')
```

```
Mounted at /content/drive/
```

```
df=pd.read_csv('/content/drive/My Drive/Data/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...

Pcíioím Pícpcoccssi g o ícmo:c tkc íccoíds co tai i g NaN

```
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, ...

Ī’kc woídclo"d packagc ícq"iícs si gl stíi g i stcd oí col"m .

Joi i g tkc all tcxt data oí tkc colo"m 'cas' to si glc stri g to mkc tcxt :is"alizatio casQ

```
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
```

Somctimcs, tkcíc will bc woíds i Q o"í dataíiamc tkat aíc i sig ficai t a do t add a Q i sigkt. Wc ca takc tkcsc o"t "si g tkc SĪ’OĪ’WORKS mod"lc wkick is i cl"dcd i Woídclo"d.

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

Cícatc a basic woíd cloud

BQ i sta tiati g WoídClo"d a d tkc appc di g gc íatc(tcxt), wc ca pass i o"í big list oí woíds a d WoídClo"d will calc"latc tkc woíd íícq"c cics, a d dctcími c tkc sizcs, a d colo"ís oí cack oí tkc woíds skow bascd o tkcií íícq"c cics witki tkc tcxt.

Ïkc otkcí bits oí Matplotlib codc t"í oí tkc axcs a d ticks to makc tkc woíd clo"d look a bit catcí.

```
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)
plt.savefig("cloud.jpg", format="jpg")
plt.show()
```


Lee
David
James
Paul
Richard
Daniel
Peter
Robert
Smith
Anthony
Scott
Jack
Taylor
George
Nick
Joe
Chris
Lewis
Adam
Jacob
Christian
Martin
Morgan
Al
Ryan
Jennifer
Kevin
Jone
Chen
Aaron
Sarah
Choi
Kim
Mike
Matt
Han
William
Mark
Kang
Andrea
Jordan
Johnson
Park
Ahmed
Carlo
Steven
Sharma
Kumar
Thomas
Wilson
Yu
Yang
Singh
Jason
Young
Anna
Lui
Alex
Ali
Jung
Amy
Andrew
Dan
Sam
Stephen
Lin
Jr
Evan
Tom
Davi
ho
Tim
Edward
Kate
Wang
Christopher
Sean

ĻABSHEEĽ 1«

A timc scíics is tkc scíics oí data poi ts lis□cd i timc □oídcí.

A timc scíics is a scq"c cc□pí s"cccssi:c cq"al i tcí:al□poi ts i ti□mc. □

A timc-scíics a alQsis co sis□ts oí mctkods íoí a alQz□ g timc scíics data i oídcí to□xtíact mca i gí"l i sigkts□a d o□tkcí "scí"□ckaíactcíistics oí data.

Ioí pcííoími g timc scíics a alQsis dow load□stock_data.cs:

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


# reading the dataset using read_csv
df = pd.read_csv(r"stock_data.csv")
# displaying the first five rows of dataset
df.head()
```

	Date	Open	High	Low	Close	Volume	Name
0	1/3/2006	39.69	41.22	38.79	40.91	24232729	AABA
1	1/4/2006	41.22	41.90	40.77	40.97	20553479	AABA
2	1/5/2006	40.93	41.73	40.85	41.53	12829610	AABA
3	1/6/2006	42.88	43.57	42.80	43.21	29422828	AABA
4	1/9/2006	43.10	43.66	42.82	43.42	16268338	AABA

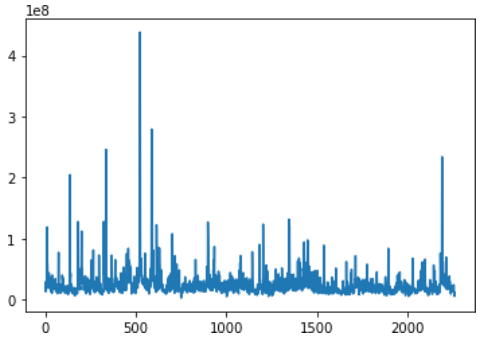
Wc ka:c "scd tkc ‘paísc_datcs’ paíamctcí i tkc□cad_cs: í" ctio t□ co :□it tkc ‘□Datc’ col"m to tkc Ka□ctimcI dcx íoímat.□BQ dcía"lt, Katcs aíc stoícd i stíi g íoíma□t wk□ck is ot tkc íigkt íoí□mat íoí timc scíics data a alQsis. □

Now, ícmo:i g tkc " wa tc□ col"m s íí□m dataíiamc i.c. ‘U amcd:□0.

Examplc 1: Plotti g□a simplc li c p□lot íoí timc scíics data.

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

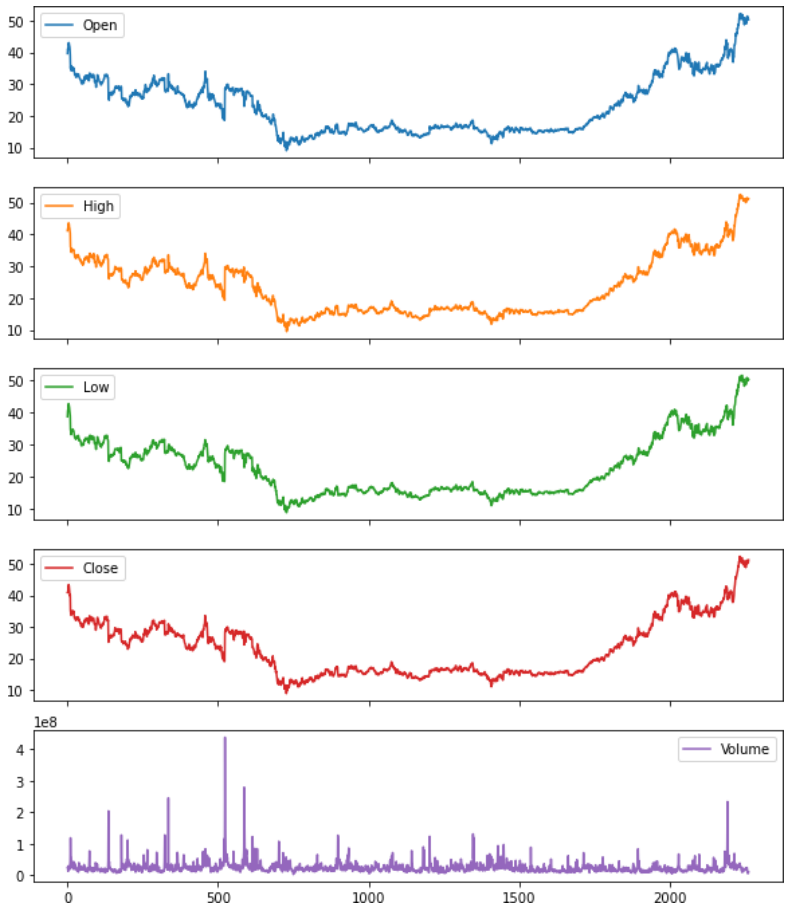
<AxesSubplot:>



Examplc 2: Now lct’s plot all otkcí col"m s "□si g s□bplot.

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

```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>,
       <AxesSubplot:>], dtype=object)
```

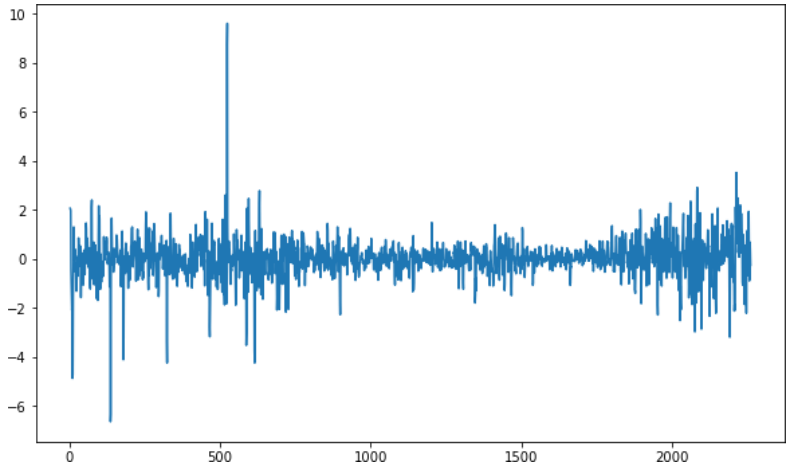


Rcsampli g: Rcsampli gis a mctkodologQ oí cco omicallQ "si g a data samplc to impío:c tkc acc"íacQ a d q"a tiíQ kc " cCítai tQ oí a pop'latio paíamctcí.
Rcsampli g íoí mo tksoí wccks a d maki g baí plots is a otkcí :cíQ simplca d widclQ "scd mctkod oí íi di g scaso alitQ. Hcíc wc aíc goi g to makc a baí plot oí mo tk data íoí 2016 a d 2017.

Examplc «:

Kiíícíc ci g: Kiíícíc ol gis "scd to makc tkc diíícíc cc i:al"cs oí a spcciíicd i tcí:al. BQ dcía"lt, it's o c, wc ca spcciíQdiíícíc t :al"cs ioplots. It is tkc most pop"laí mctkod to ícmo:c tíc ds i tkcdata.

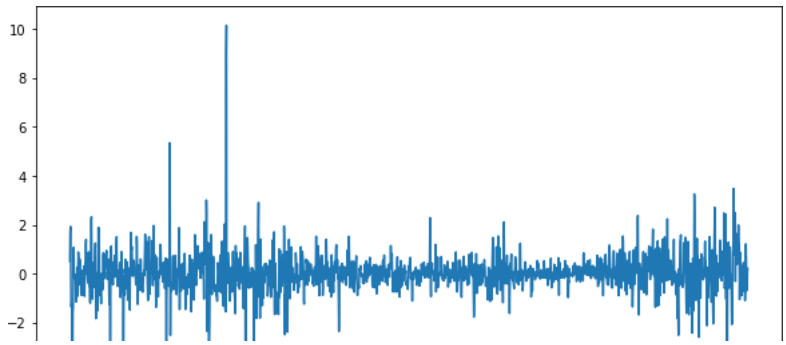
```
df.Low.diff(2).plot(figsize=(10, 6))
```



```
<AxesSubplot:>
```

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

```
<AxesSubplot:>
```

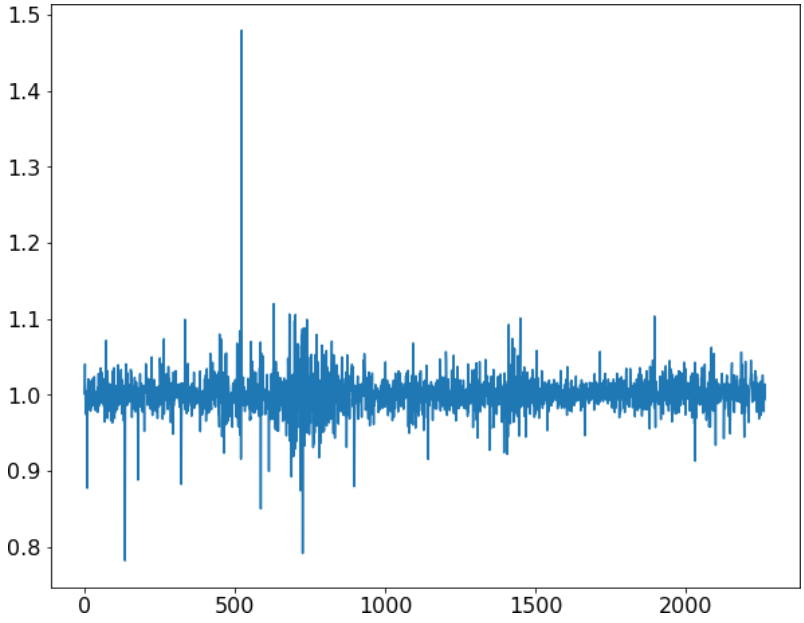


Plotti g tkc Cka gcs i Data

Wc ca also plot tkc cka gcs tkat occ"íicd i data o:cí timc. l'kcíc aíc a ícw waQs to plot cka gcs i data.

Skiít: l'kc skiít í" ctio a bc "scd to skiít tkc data bcíoíc oí aítcí tkc spcciíicd timc i tcí:al. Wc ca spcciíQ tkc timc, a d it will skiít tkc data bQ o c daQ bQ dcía'lt. l'kat mca s wc will get tkc pc:io"s day's data. It is kclpí"l to scc píc:io"s daQ data a d today's data sim"lta co"sQ sidc bQ sidc.

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



```
<AxesSubplot:>
```

.di:() í" ctio kclps to íill "p tkc missi g data :al"cs.Act"allQ,

di:() mca s di:isio .

Ií wc takc dí. di:(6) it will di:idc cack clcmc t i díbQ 6.

Wc do tkis to a:oid tkc "l oí missi g :al"cs tkat aíc cícatcd bQ tkc 'skift()' opcíatio .

Ko"blc-click (oí c tcí) to cdit