Matplotlib & Seaborn - 1

🎯 Session Objectives:

- ✅ Understand what data visualization is and why it matters
- ✅ Use Matplotlib to plot different types of charts
- ✅ Customize plots with markers, colors, linewidth, and line styles
- ✅ Integrate Matplotlib with NumPy and Pandas
- ✅ Understand why Seaborn is important in visualization
- ✅ Recognize common Seaborn plot types

```
# sort_values (by = <column_name> ,  ascending = [True/False])
Customers.sort_values(by = 'state' , ascending = False) # Categorical Columns -> Alphabetical Orders[Z->A]
```

t_name	last_name	email	gender	street_number	street_address	street_suffix	city	state	postcode
lendrick	Gowthrop	Unknown	Male	64	Almo	Avenue	Springfield	Wisconsin	45505
ıtonetta	Leftwich	aleftwichdb@cafepress.com	Female	25	Vermont	Road	Milwaukee	Wisconsin	53234
Berty	Meert	bmeertjr@hatena.ne.jp	Male	511	Lakeland	Drive	Milwaukee	Wisconsin	53215
Elle	Coultish	ecoultish8n@techcrunch.com	Female	88	Westerfield	Pass	Milwaukee	Wisconsin	53234
Paulie	Gadault	pgadaultql@posterous.com	Female	2061	Aberg	Street	Milwaukee	Wisconsin	53285
...	...	...	...	...	...	...	...	...	...
Christal	Goadby	cgoadbyo4@clickbank.net	Female	53	Fremont	Plaza	Tulsa	Alabama	74126
Wendel	Ormesher	Unknown	Male	81	Veith	Place	Mobile	Alabama	36605
Gaelan	Bonas	gbonash3@bigcartel.com	Male	36	Menomonie	Trail	Birmingham	Alabama	35215
Shaine	Lumsdall	slumsdall6n@digg.com	Male	80	Miller	Junction	Montgomery	Alabama	36104
Elliott	Marjanovic	emarjanovichg@mayoclinic.com	Male	17	Eagle Crest	Place	Birmingham	Alabama	0

ımns

```
Products.sort_values(by = 'cost' , ascending = False)
```

	id	product	cost	company
33	34	Muffin Hinge Container 6	99.54	Skyble
13	14	Cookies Oatmeal Raisin	99.40	Dynabox
5	6	Wine - White, Riesling, Semi - Dry	99.22	Livepath
51	52	Soup - Knorr, Country Bean	98.74	Eimbee
34	35	Pie Box - Cello Window 2.5	95.68	Linkbuzz
31	32	Container Clear 8 Oz	92.33	Muxo
20	21	Bagel - Whole White Sesame	90.48	Realcube
9	10	Sambuca - Ramazzotti	88.99	Livepath
41	42	Sauce - Demi Glace	87.51	Wordware

```
Purchases.sort_values(by = ['amount' , 'paid'] , ascending = False) # Both Columns follow high to low
```

	id	purch_date	customer_num	product_num	amount	paid
211	212	2019-01-06	918	34	20	1990.80
2279	2280	2019-03-13	685	34	20	1990.80
4846	4847	2019-05-13	993	34	20	1990.80
2924	2925	2019-03-24	619	14	20	1988.00
3222	3223	2019-03-27	747	14	20	1988.00
...	...	...	...	...	...	...
4678	4679	2019-04-28	917	55	1	8.55
5851	5852	2019-06-17	758	55	1	8.55
2435	2436	2019-03-16	977	1	1	6.36
5424	5425	2019-05-26	957	1	1	5.79
3190	3191	2019-03-27	784	1	1	3.63

6000 rows × 6 columns

```
Purchases.sort_values(by = ['amount' , 'paid'] , ascending = [True,False]) # Amount[Low to High] & Paid[High to
```

	id	purch_date	customer_num	product_num	amount	paid
1301	1302	2019-07-06	450	34	1	99.54
5070	5071	2019-05-18	317	34	1	99.54
5771	5772	2019-06-15	138	14	1	99.40
331	332	2019-02-05	418	6	1	99.22
641	642	2019-03-06	519	52	1	98.74
...	...	...	...	...	...	...
3388	3389	2019-03-29	155	1	20	127.20
3849	3850	2019-04-16	759	1	20	127.20
5849	5850	2019-06-17	113	1	20	127.20
5254	5255	2019-05-22	652	1	20	101.80
2422	2423	2019-03-16	690	1	20	63.60

6000 rows × 6 columns

```
Purchases['year'] = Purchases['purch_date'].dt.year
Purchases['month'] = Purchases['purch_date'].dt.month
Purchases['item_price'] = Purchases['paid'] / Purchases['amount']
Purchases
```

	id	purch_date	customer_num	product_num	amount	paid	year	month	Revenue	item_price
0	1	2019-01-03	823	27	12	568.92	2019	1	6827.04	47.41
1	2	2019-01-03	606	28	14	395.36	2019	1	5535.04	28.24
2	3	2019-01-03	955	9	17	510.17	2019	1	8672.89	30.01
3	4	2019-01-03	577	19	3	68.49	2019	1	205.47	22.83
4	5	2019-01-03	429	8	18	759.42	2019	1	13669.56	42.19
...	...	...	...	...	...	...	...	...	...	...
5995	5996	2019-06-20	893	33	5	411.10	2019	6	2055.50	82.22
5996	5997	2019-06-20	566	23	11	178.97	2019	6	1968.67	16.27
5997	5998	2019-06-20	114	19	9	205.47	2019	6	1849.23	22.83
5998	5999	2019-06-20	404	11	20	429.40	2019	6	8588.00	21.47
5999	6000	2019-06-20	88	57	4	274.52	2019	6	1098.08	68.63

6000 rows × 10 columns

```
Purchases.drop(columns = ['Revenue'] , inplace = True)
Purchases
```

	id	purch_date	customer_num	product_num	amount	paid	year	month	item_price
0	1	2019-01-03	823	27	12	568.92	2019	1	47.41
1	2	2019-01-03	606	28	14	395.36	2019	1	28.24
2	3	2019-01-03	955	9	17	510.17	2019	1	30.01
3	4	2019-01-03	577	19	3	68.49	2019	1	22.83
4	5	2019-01-03	429	8	18	759.42	2019	1	42.19
...	...	...	...	...	...	...	...	...	...
5995	5996	2019-06-20	893	33	5	411.10	2019	6	82.22
5996	5997	2019-06-20	566	23	11	178.97	2019	6	16.27
5997	5998	2019-06-20	114	19	9	205.47	2019	6	22.83
5998	5999	2019-06-20	404	11	20	429.40	2019	6	21.47
5999	6000	2019-06-20	88	57	4	274.52	2019	6	68.63

6000 rows × 9 columns

```
# idxmax() -> Sort the DataFrame with respect to Index [High-Low]
Products.idxmax() # 59
```

```
id          59
product      5
cost        33
company     22
dtype: int64
```

```
Products[Products['id'] == Products.idxmax()['id'] + 1 ]
```

	id	product	cost	company
59	60	Table Cloth 90x90 Colour	41.22	Quinu

```
Products.loc[[Products.idxmax()['product']]]
```

	id	product	cost	company
5	6	Wine - White, Riesling, Semi - Dry	99.22	Livepath

```
Products.iloc[[Products.idxmax()['product']]]
```

	id	product	cost	company
5	6	Wine - White, Riesling, Semi - Dry	99.22	Livepath

```
Products.loc[[Products.idxmax()['id']]]
```

	id	product	cost	company
59	60	Table Cloth 90x90 Colour	41.22	Quinu

```
# GroupBy -> Splitting + Aggregation
sorted_products = Products.sort_values(by = ['cost'] , ascending = False)
sorted_products
```

	id	product	cost	company
33	34	Muffin Hinge Container 6	99.54	Skyble
13	14	Cookies Oatmeal Raisin	99.40	Dynabox
5	6	Wine - White, Riesling, Semi - Dry	99.22	Livepath
51	52	Soup - Knorr, Country Bean	98.74	Eimbee
34	35	Pie Box - Cello Window 2.5	95.68	Linkbuzz
31	32	Container Clear 8 Oz	92.33	Muxo
20	21	Bagel - Whole White Sesame	90.48	Realcube
9	10	Sambuca - Ramazzotti	88.99	Livepath
41	42	Sauce - Demi Glace	87.51	Wordware

```
sorted_products.groupby('company')['cost'].mean()
```

```
company
Aibox          56.330000
Babbleopia     63.980000
Brainsphere    22.830000
Browsezoom     41.050000
Cogibox        51.135000
Digitube        8.550000
Dynabox        99.400000
Dynazzy        85.740000
Eare           87.390000
Eazzy          74.280000
Edgeclub       40.690000
Eimbee         98.740000
Fanoodle       68.630000
Flipstorm      37.790000
Fliptune       75.320000
```

```
sorted_products.groupby('company')['cost'].mean().reset_index() # DataFrame
```

	company	cost
0	Aibox	56.330000
1	Babbleopia	63.980000
2	Brainsphere	22.830000
3	Browsezoom	41.050000
4	Cogibox	51.135000
5	Digitube	8.550000
6	Dynabox	99.400000
7	Dynazzy	85.740000
8	Eare	87.390000

```
# Further if required, Make them sort
sorted_products.groupby('company')['cost'].mean().reset_index().sort_values(by = 'cost', ascending = False)
```

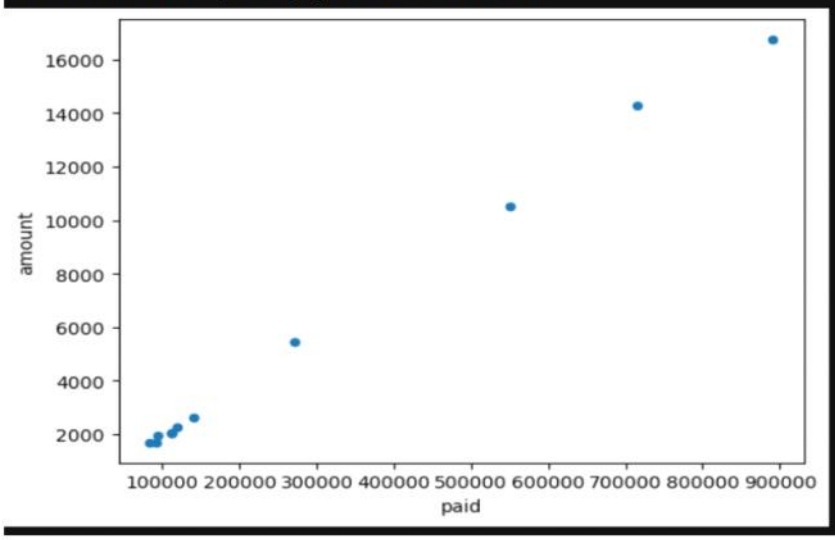
	company	cost
39	Skyble	99.540000
6	Dynabox	99.400000
11	Eimbee	98.740000
18	Linkbuzz	95.680000
19	Livepath	94.105000
23	Muxo	92.330000
46	Wordware	87.510000
8	Eare	87.390000
42	Trunyx	87.050000
7	Dynazzy	85.740000
45	Voonder	82.220000
34	Realcube	75.370000
14	Fliptune	75.320000
9	Eazzy	74.280000

```
# Trend Axis [Time Intelligence] -> Date Columns
Purchases.groupby('month')[['paid','amount']].sum()
```

month	paid	amount
1	139986.42	2643
2	83532.80	1713
3	890751.05	16752
4	715885.30	14268
5	550416.29	10537
6	271329.22	5457
7	112608.31	2063
8	118668.27	2291

```
Purchases.groupby('month')[['paid','amount']].sum().plot.scatter('paid' , 'amount' )
```

```
<Axes: xlabel='paid', ylabel='amount'>
```




```
Purchases.set_index('purch_date')
```

purch_date	id	customer_num	product_num	amount	paid	year	month	item_price
2019-01-03	1	823	27	12	568.92	2019	1	47.41
2019-01-03	2	606	28	14	395.36	2019	1	28.24
2019-01-03	3	955	9	17	510.17	2019	1	30.01
2019-01-03	4	577	19	3	68.49	2019	1	22.83
2019-01-03	5	429	8	18	759.42	2019	1	42.19
...	...	...	...	...	...	...	...	...
2019-06-20	5996	893	33	5	411.10	2019	6	82.22
2019-06-20	5997	566	23	11	178.97	2019	6	16.27
2019-06-20	5998	114	19	9	205.47	2019	6	22.83
2019-06-20	5999	404	11	20	429.40	2019	6	21.47
2019-06-20	6000	88	57	4	274.52	2019	6	68.63

6000 rows × 8 columns

```python
# .isin()
company_list = ['Ntag', 'Skipfire' , 'Vitz' , 'Realcube' , 'Zoombox']
Products[Products['company'].isin(company_list)]
```

	id	product	cost	company
0	1	Liners - Baking Cups	6.36	Skipfire
2	3	Bar Bran Honey Nut	65.40	Ntag
14	15	Cookies Oatmeal Raisin	14.13	Vitz
15	16	Sausage - Chorizo	55.45	Vitz
20	21	Bagel - Whole White Sesame	90.48	Realcube
21	22	Scotch - Queen Anne	60.26	Realcube
22	23	Puree - Blackcurrant	16.27	Zoombox
23	24	Bread - Bagels, Plain	31.26	Vitz
30	31	Wine - Carmenere Casillero Del	55.77	Ntag

```python
Purchases[Purchases['amount'].between(11,21)]
```

	id	purch_date	customer_num	product_num	amount	paid	year	month	item_price
0	1	2019-01-03	823	27	12	568.92	2019	1	47.41
1	2	2019-01-03	606	28	14	395.36	2019	1	28.24
2	3	2019-01-03	955	9	17	510.17	2019	1	30.01
4	5	2019-01-03	429	8	18	759.42	2019	1	42.19
5	6	2019-01-03	275	36	18	176.76	2019	1	9.82
...	...	...	...	...	...	...	...	...	...
5990	5991	2019-06-20	632	2	15	1286.10	2019	6	85.74
5992	5993	2019-06-20	159	18	17	957.61	2019	6	56.33
5994	5995	2019-06-20	840	23	18	292.86	2019	6	16.27
5996	5997	2019-06-20	566	23	11	178.97	2019	6	16.27
5998	5999	2019-06-20	404	11	20	429.40	2019	6	21.47

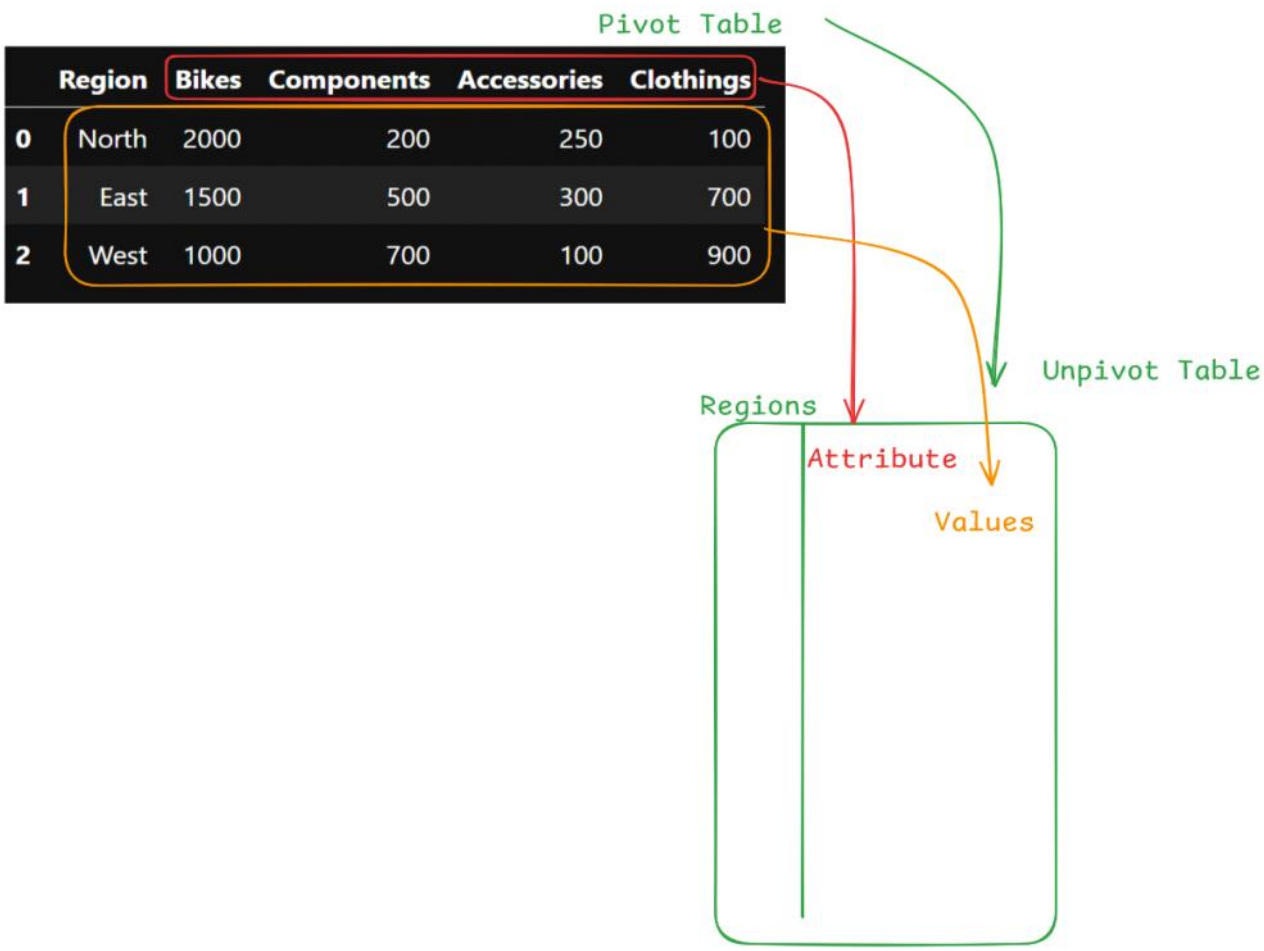
3023 rows × 9 columns

```python
# Regions Vs Products Sales
data = {
    'Region' : ['North','East','East','West','West','North','East','East','West','North'],
    'Products': ['Bikes','Clothing','Accessories','Components','Bikes','Clothing','Accessories','Components',
                 'Clothing','Accessories'],
    'Revenue' : [2000,1000,1500,1000,2500,5000,1500,2000,1000,7000]
}
df = pd.DataFrame(data)
df
```

	Region	Products	Revenue
0	North	Bikes	2000
1	East	Clothing	1000
2	East	Accessories	1500
3	West	Components	1000
4	West	Bikes	2500
5	North	Clothing	5000
6	East	Accessories	1500
7	East	Components	2000
8	West	Clothing	1000
9	North	Accessories	7000

pandas.pivot_table

```
pandas.pivot_table(data, values=None, index=None, columns=None,
aggfunc='mean', fill_value=None, margins=False, dropna=True,
margins_name='All', observed=<no_default>, sort=True)    [source]
```



pandas.melt

```
pandas.melt(frame, id_vars=None, value_vars=None, var_name=None,
value_name='value', col_level=None, ignore_index=True)    [source]
```

```
pivot_table = df.pivot_table(
    values = 'Revenue',
    index = 'Region',
    columns = 'Products',
    aggfunc='sum'
)
pivot_table
```

Products	Accessories	Bikes	Clothing	Components
Region				
East	3000.0	NaN	1000.0	2000.0
North	7000.0	2000.0	5000.0	NaN
West	NaN	2500.0	1000.0	1000.0

```
pivot_table = df.pivot_table(
    values = 'Revenue',
    index = 'Products',
    columns = 'Region',
    aggfunc='sum'
)
pivot_table
```

Region	East	North	West
Products			
Accessories	3000.0	7000.0	NaN
Bikes	NaN	2000.0	2500.0
Clothing	1000.0	5000.0	1000.0
Components	2000.0	NaN	1000.0

```
pivot_table = df.pivot_table(
    values = 'Revenue',
    index = 'Products',
    columns = 'Region',
    aggfunc='sum',
    fill_value=0
)
pivot_table
```

Region	East	North	West
Products			
Accessories	3000	7000	0
Bikes	0	2000	2500
Clothing	1000	5000	1000
Components	2000	0	1000

```
type(pivot_table)
```

```
pandas.core.frame.DataFrame
```

```
filtered_pivot = pivot_table[pivot_table.sum(axis=1) >= 7000]
filtered_pivot
```

Region	East	North	West
Products			
Accessories	3000	7000	0
Clothing	1000	5000	1000

```
pivot_table = pd.pivot_table(
    data = df,
    values = 'Revenue',
    index = 'Products',
    columns = 'Region',
    aggfunc='sum',
    fill_value=0
)
pivot_table
```

Region	East	North	West
Products			
Accessories	3000	7000	0
Bikes	0	2000	2500
Clothing	1000	5000	1000
Components	2000	0	1000

```python
pivot_table = pd.pivot_table(
    data = df,
    values = 'Revenue',
    index = ['Products','Region'],
    aggfunc='sum',
    fill_value=0
)
pivot_table
```

		Revenue
Products	Region	
Accessories	East	3000
	North	7000
Bikes	North	2000
	West	2500
Clothing	East	1000
	North	5000
	West	1000
Components	East	2000
	West	1000

```python
# Performing Pivot Table Concepts on Original Table [Products]
Product_pivot = pd.pivot_table(
    data = Products,
    values = ['cost' , 'id'],
    index = ['company'],
    aggfunc = {'cost' : 'sum' , 'id' : 'count' }
)
Product_pivot
```

	cost	id
company		
Aibox	56.33	1
Babbleopia	63.98	1
Brainsphere	22.83	1
Browsezoom	82.10	2
Cogibox	102.27	2
Digitube	8.55	1
Dynabox	99.40	1
Dynazzy	85.74	1

```python
Product_pivot.sort_values(by = 'id' , ascending =False)
```

	cost	id
company		
Skinder	159.85	3
Vitz	100.84	3
Ntag	121.17	2
Browsezoom	82.10	2
Cogibox	102.27	2
Livepath	188.21	2
Realcube	150.74	2
Thoughtstorm	107.11	2
Photofeed	42.56	1
Photojam	68.16	1
Photolist	59.23	1
Quatz	13.83	1
Quinu	41.22	1

```python
Purchases_pivot = pd.pivot_table(
    data = Purchases,
    values = ['amount' , 'item_price'],
    index = ['year' , 'month'],
    aggfunc = {'amount' : 'sum' , 'item_price' : 'sum'}
)
Purchases_pivot
```

		amount	item_price
year	month		
2019	1	2643	12814.79
	2	1713	8252.11
	3	16752	84916.25
	4	14268	66590.05
	5	10537	52289.29
	6	5457	26174.50
	7	2063	9841.91
	8	2291	11252.76
	9	1962	9373.93
	10	1695	8488.43
	11	2044	10353.10
	12	2032	10455.69

```
# CrossTab
# Regions Vs Products Sales
data = {
    'Region' : ['North','East','East','West','West','North','East','East','West','North'],
    'Products': ['Bikes','Clothing','Accessories','Components','Bikes','Clothing','Accessories','Components',
                 'Clothing','Accessories'],
    'Revenue' : [2000,1000,1500,1000,2500,5000,1500,2000,1000,7000]
}
df = pd.DataFrame(data)
df
```

	Region	Products	Revenue
0	North	Bikes	2000
1	East	Clothing	1000
2	East	Accessories	1500
3	West	Components	1000
4	West	Bikes	2500
5	North	Clothing	5000
6	East	Accessories	1500
7	East	Components	2000
8	West	Clothing	1000
9	North	Accessories	7000

```
cross_tab = pd.crosstab(df['Region'] , df['Products'])
cross_tab
```

Products	Accessories	Bikes	Clothing	Components
Region				
East	2	0	1	1
North	1	1	1	0
West	0	1	1	1

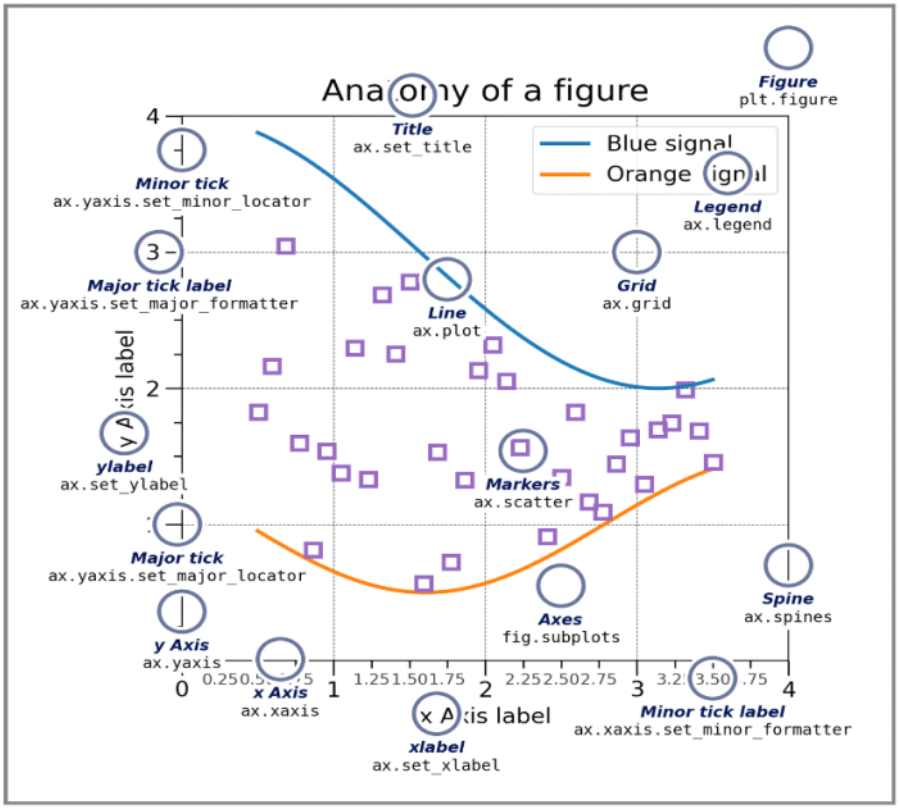
```
# Pivot Table -> Unpivot ['melt']
data = {
    'Region' : ['North','East','West'],
    'Bikes' : [2000,1500,1000],
    'Components' : [200,500,700],
    'Accessories' : [250,300,100],
    'Clothings' : [100,700,900]
}
df = pd.DataFrame(data)

df
```

	Region	Bikes	Components	Accessories	Clothings
0	North	2000	200	250	100
1	East	1500	500	300	700
2	West	1000	700	100	900

```python
# Unpivot Table
melted_df = pd.melt(
    df,
    id_vars = 'Region',
    value_vars = ['Bikes','Components','Accessories','Clothings'],
    var_name = 'Products',
    value_name = 'Total Sales'
)
melted_df
```

	Region	Products	Total Sales
0	North	Bikes	2000
1	East	Bikes	1500
2	West	Bikes	1000
3	North	Components	200
4	East	Components	500
5	West	Components	700
6	North	Accessories	250
7	East	Accessories	300
8	West	Accessories	100
9	North	Clothings	100
10	East	Clothings	700
11	West	Clothings	900

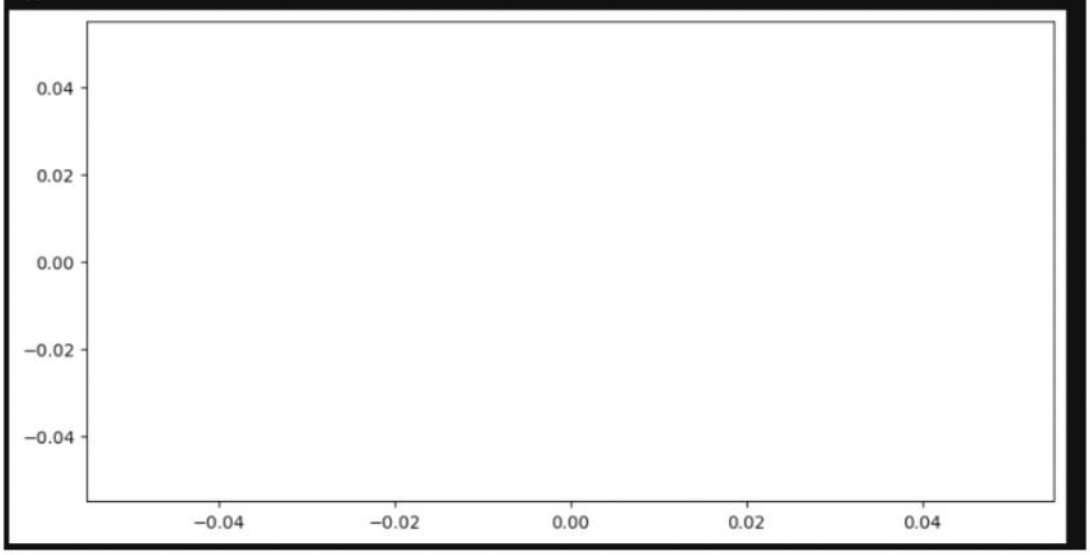
Matplotlib



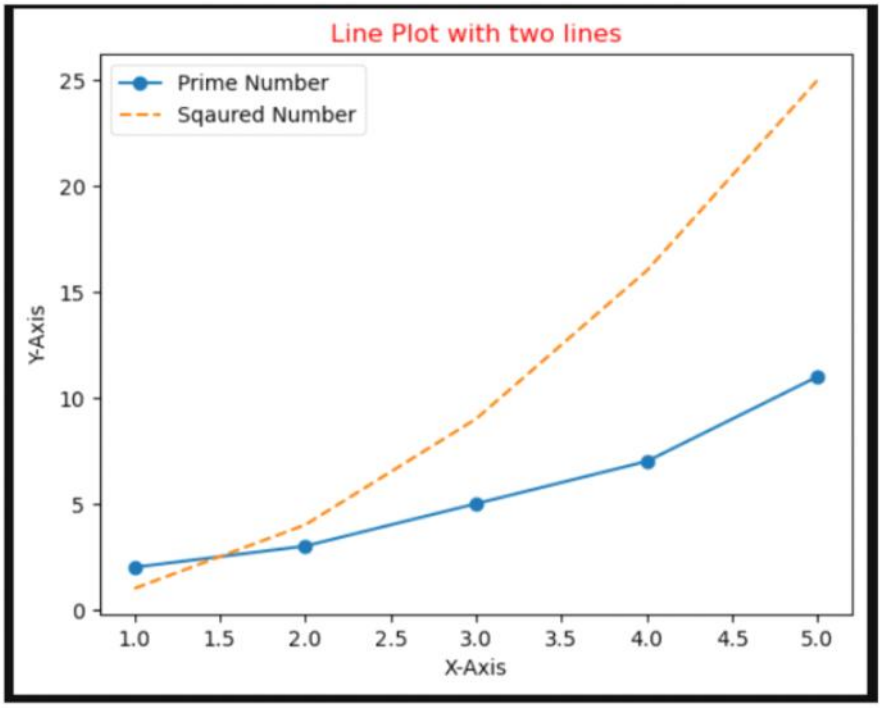
```
import matplotlib.pyplot as plt
```

```
plt.figure(figsize = (10,5)) # (width , height) inches
plt.plot()
```

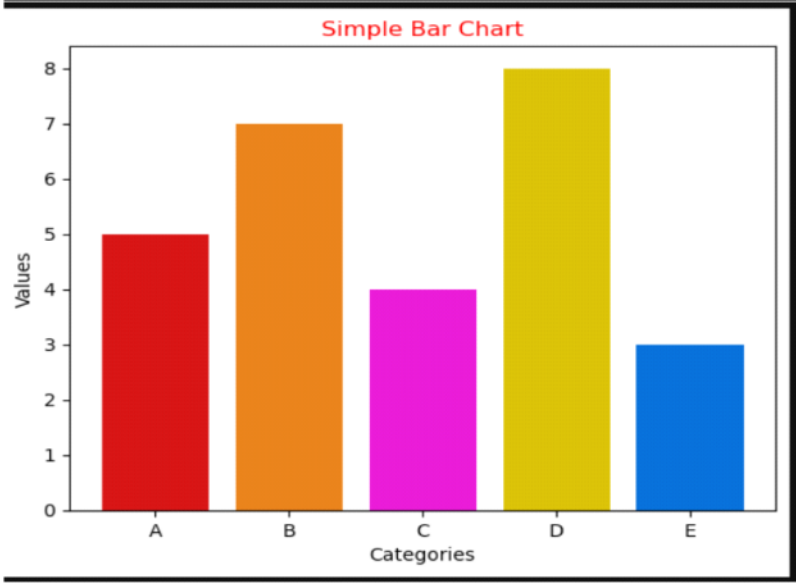
```
[]
```



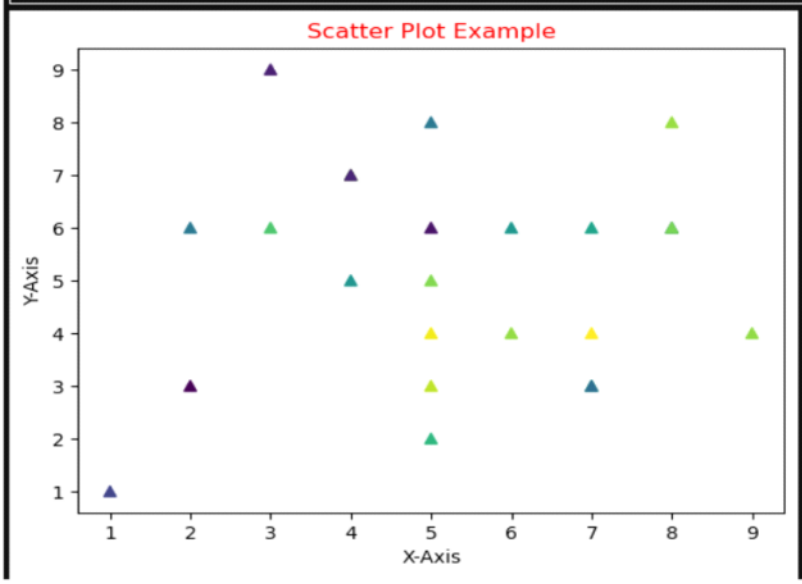

```
# Line Plots with multiple lines
import matplotlib.pyplot as plt

x = [1,2,3,4,5]
y1 = [2,3,5,7,11] # Prime Number
y2 = [1,4,9,16,25] # Squared Number

plt.plot(x, y1, label = 'Prime Number' , marker = 'o')
plt.plot(x, y2, label = 'Sqaured Number' , linestyle = '--')

plt.title("Line Plot with two lines" , color = 'red')
plt.xlabel('X-Axis')
plt.ylabel('Y-Axis')
plt.legend()
plt.show()
```



```python
# Bar Chart
# Data [Categories VS Continuous]
categories = ['A','B','C','D','E']
values = [5,7,4,8,3]
# colors = ['red','orange','pink','yellow','blue']
colors = ['#D31616','#E4801C','#E41CD3','#D6BE08','#086FD6']
plt.bar(categories , values, color = colors)
plt.title("Simple Bar Chart" , color = 'red')
plt.xlabel('Categories')
plt.ylabel('Values')
plt.show()
```




```python
# Scatter Plot [Continuous VS Continuous]
x = [5,2,5,1,7,6,4,8,7,5,9,4,2,3,5,7,8,5,3,4,5,6,7,8]
y = [4,3,2,1,4,6,7,8,6,3,4,5,6,6,6,3,6,8,9,7,5,4,3,6]

colors = np.random.rand(24)
plt.scatter(x,y, c=colors, cmap='viridis',  marker = '^')

plt.title("Scatter Plot Example" , color = 'red')
plt.xlabel('X-Axis')
plt.ylabel('Y-Axis')
plt.show()
```



```python
# Pie Chart
sizes = [25,15,20,30,10]
labels = ['Apple','Mango','Orange','Litchi','Papaya']
colors = ['#D31616','#E4801C','#E41CD3','#D6BE08','#086FD6']

plt.pie(sizes , labels = labels , colors = colors , explode = (0,0,0,0.1,0), autopct='%1.1f%%')
plt.title("Fruits Distribution")
plt.show()
```