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

```
import numpy as np
```

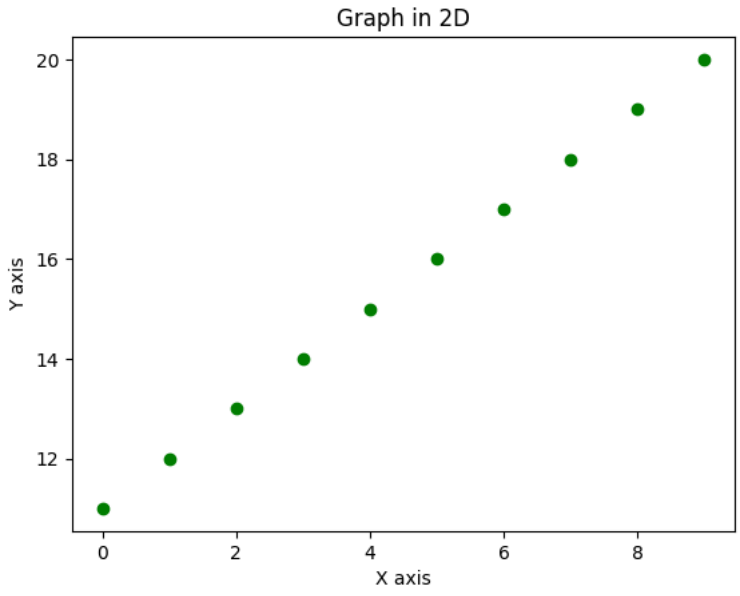
```
## Simple Examples
x=np.arange(0,10)
y=np.arange(11,21)
```

```
x
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
a=np.arange(40,50)
b=np.arange(50,60)
```

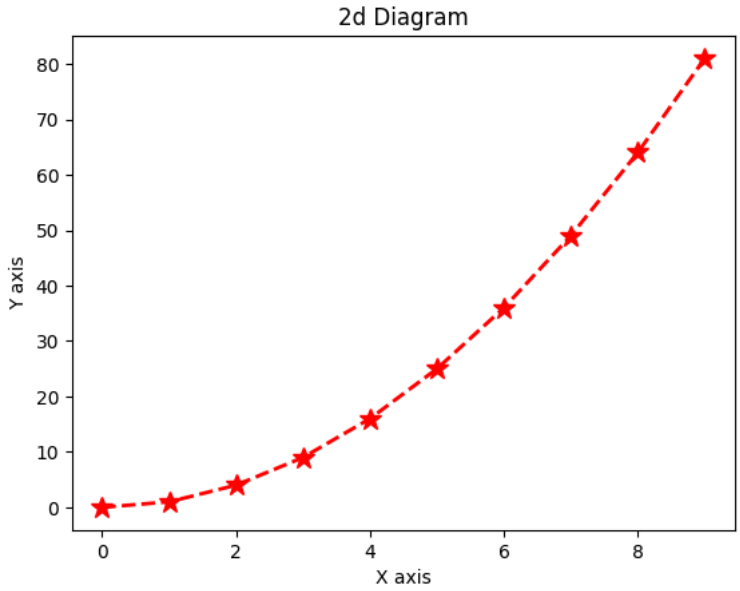
```
##plotting using matplotlib
##plt scatter
plt.scatter(x,y,c='g')
plt.xlabel('X axis')
plt.ylabel('Y axis')
plt.title('Graph in 2D')
plt.savefig('Test.png')
```




```
y=x*x
```

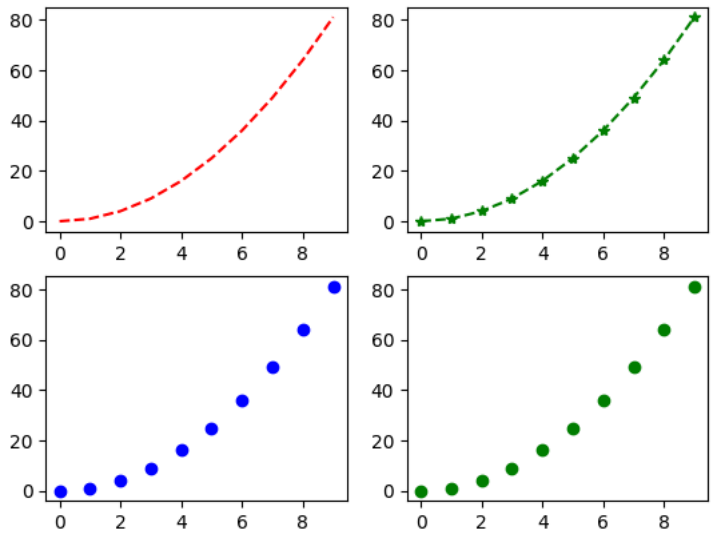
```
## plt plot
plt.plot(x,y,'r*',linestyle='dashed',linewidth=2, markersize=12)
plt.xlabel('X axis')
plt.ylabel('Y axis')
plt.title('2d Diagram')
```

```
Text(0.5, 1.0, '2d Diagram')
```

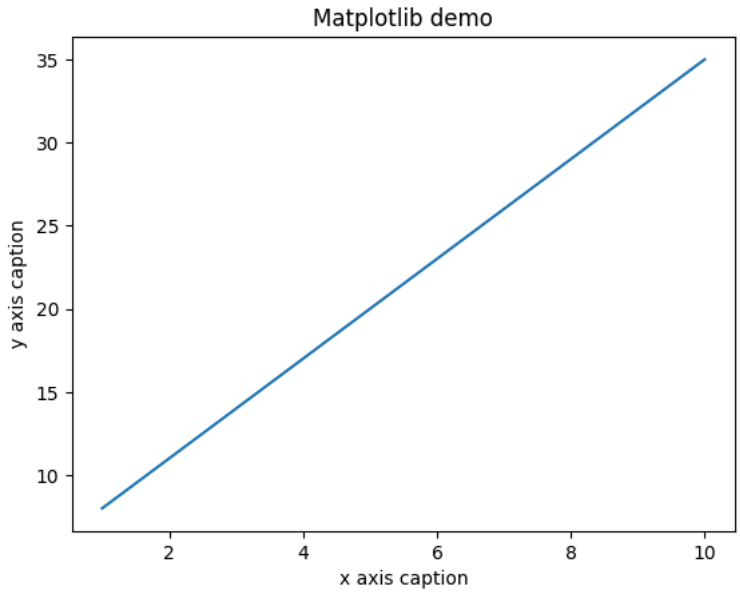



```
## Creating Subplots
plt.subplot(2,2,1)
plt.plot(x,y,'r--')
plt.subplot(2,2,2)
plt.plot(x,y,'g*--')
plt.subplot(2,2,3)
plt.plot(x,y,'bo')
plt.subplot(2,2,4)
plt.plot(x,y,'go')
```

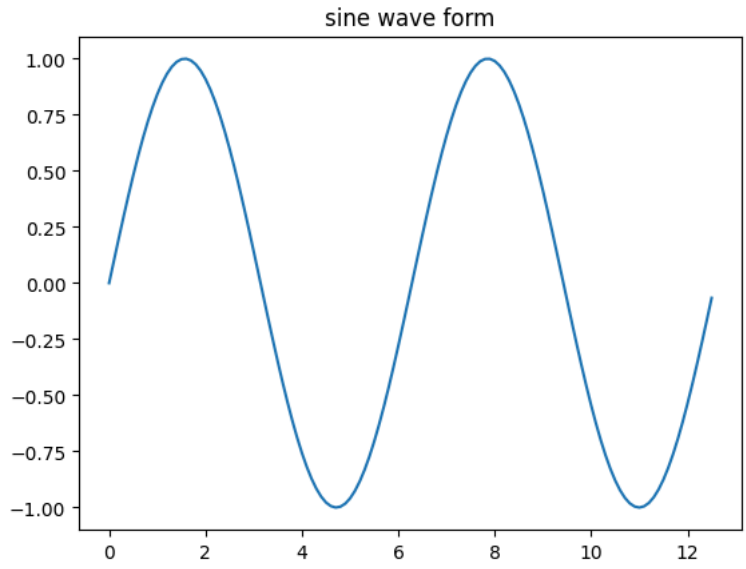
```
[<matplotlib.lines.Line2D at 0x78c1a52c9670>]
```



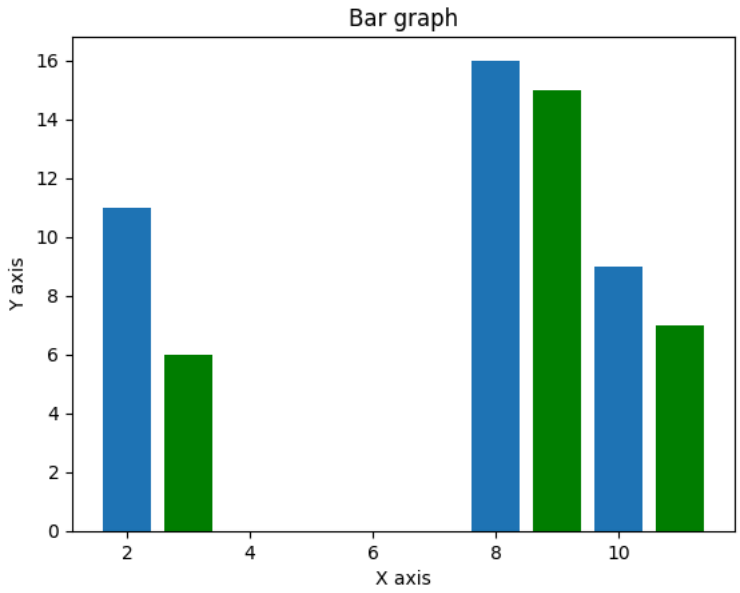

```
x = np.arange(1,11)
y = 3 * x + 5
plt.title("Matplotlib demo")
plt.xlabel("x axis caption")
plt.ylabel("y axis caption")
plt.plot(x,y)
plt.show()
```



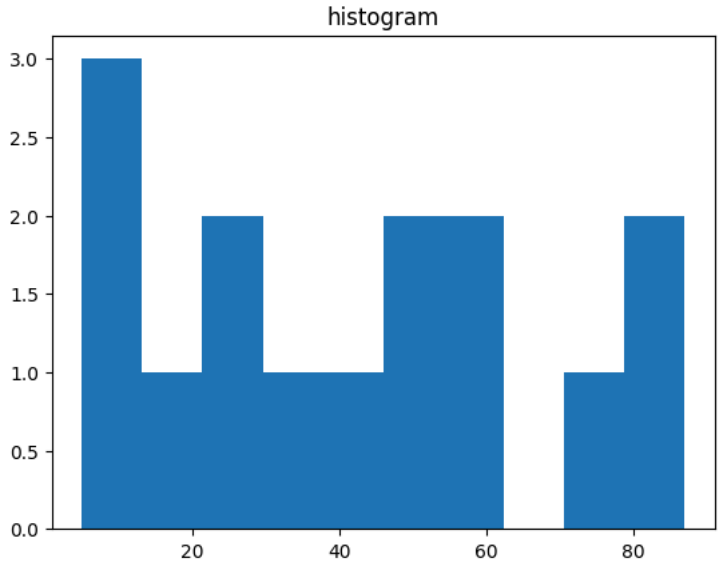

```
# Compute the x and y coordinates for points on a sine curve
x = np.arange(0, 4 * np.pi, 0.1)
y = np.sin(x)
plt.title("sine wave form")
# Plot the points using matplotlib
plt.plot(x, y)
plt.show()
```




```
## Bar plot
x = [2,8,10]
y = [11,16,9]
x2 = [3,9,11]
y2 = [6,15,7]
plt.bar(x, y)
plt.bar(x2, y2, color = 'g')
plt.title('Bar graph')
plt.ylabel('Y axis')
plt.xlabel('X axis')
plt.show()
```




```
a = np.array([22,87,5,43,56,73,55,54,11,20,51,5,79,31,27])
plt.hist(a)
plt.title("histogram")
plt.show()
```




```
Start coding or generate with AI.
```

Meteorite Landings - Histogram

```
# Connect to the drive
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=
```

```
cd/content/gdrive/My Drive
```

```
/content/gdrive/My Drive
```

```
# Below are the activities that have been covered in the previous class.
# 1. Import the necessary libraries for this class and create a DataFrame.
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt

met_df = pd.read_csv('meteorite-landings.csv')
```

```
# 2. Find the number of rows and columns in the DataFrame.
print(met_df.shape)
```

```
(45716, 10)
```

```
# 3. Rows containing the year values less than 860 and greater than 2016.
correct_years_df = met_df[(met_df['year'] >= 860) & (met_df['year'] <= 2016)]
```

```
# 4. Rows having the 'reclong' values greater than or equal to -180 degrees and less t
correct_long_df = correct_years_df[(correct_years_df['reclong'] >= -180) & (correct_ye
```

```
# 5. Rows containing the '0 reclat' and '0 reclong' values from the 'correct_long_df'.
correct_lat_long_df = correct_long_df[~((correct_long_df['reclat'] == 0 ) & (correct_l
```

```
# 6. Indices of the rows having missing mass values.
row_indices = correct_lat_long_df[correct_lat_long_df['mass'].isnull() == True].index
```

```
correct_lat_long_df['mass'].isnull() == True
```

	mass
0	False
1	False
2	False
3	False
4	False
...	...
45711	False
45712	False
45713	False
45714	False
45715	False

32036 rows × 1 columns

dtype: bool

```
# 7. Missing values in the 'mass' column in the 'correct_lat_long_df' DataFrame with m
median_mass = correct_lat_long_df['mass'].median()
correct_lat_long_df.loc[row_indices, 'mass'] = median_mass
```

```
# 8. Convert the 'year' values into an integer type values.
correct_lat_long_df.loc[:, 'year'] = correct_lat_long_df.loc[:, 'year'].astype('int')
```

Activity 1: Count Plot

Recall that in the previous class we couldn't draw count plots for all the meteorites found in a year. We had to divide the years in the 20 to 30 years period and then draw the count plot. The count plot really works well when you want to look at the variation in the count of

a feature across a category. E.g., we observed that the count of `Valid` meteorites was more than the count of the `Relict` meteorites for a year. So, this is where the count plots are really helpful.

However, if you want to observe the count trend in one single graph, then, in that case, it is better to create a histogram.

A histogram is a type of bar graph which plots the data falling in the same interval together. Let's understand a histogram through an example.

Consider the count of the meteorites between the years `1970` & `2000`. Let's create a count plot for this period.

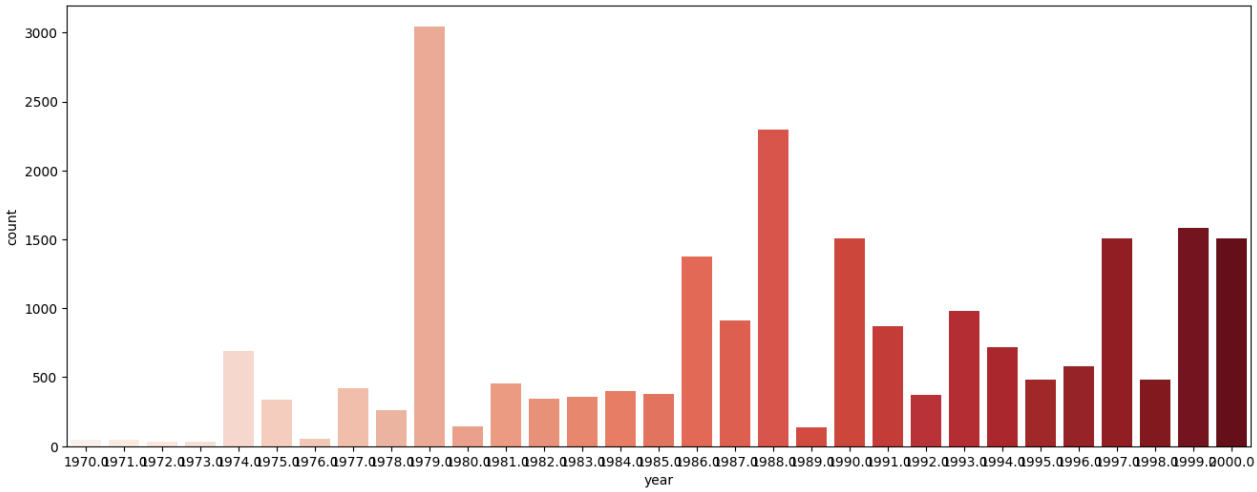
```
# Student Action: Create a count plot for the count of the meteorites between the years
df_2000=correct_lat_long_df[(correct_lat_long_df["year"]>=1970) & (correct_lat_long_df[
df_2000
```

	name	id	nametype	recclass	mass	fall	year	reclat	reclong	GeoLocation
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	1976.0	16.88333	-99.90000	(16.883330, -99.900000)
10	Aioun el Atrouss	423	Valid	Diogenite-pm	1000.0	Fell	1974.0	16.39806	-9.57028	(16.398060, -9.570280)
16	Akyumak	433	Valid	Iron, IVA	50000.0	Fell	1981.0	39.91667	42.81667	(39.916670, 42.816670)
31	Alta'ameem	2284	Valid	LL5	6000.0	Fell	1977.0	35.27333	44.21556	(35.273330, 44.215560)
41	Anlong	2305	Valid	H5	2500.0	Fell	1971.0	25.15000	105.18333	(25.150000, 105.183330)
...	...	...	...	...	...	...	...	...	...	...
45709	Zhongxiang	30406	Valid	Iron	100000.0	Found	1981.0	31.20000	112.50000	(31.200000, 112.500000)
45710	Zillah 001	31355	Valid	L6	1475.0	Found	1990.0	29.03700	17.01850	(29.037000, 17.018500)

Next steps: Generate code with `df_2000` New interactive sheet

```
# Create a DataFrame for the meteorites fallen between 1970 and 2000 including both.
import matplotlib.pyplot as plt
import seaborn as sns
# Create a count plot for the meteorites fallen between 1970 and 2000 including both.
plt.figure(figsize=(16,6))
sns.countplot(x= "year", data=df_2000 ,palette="Reds", )
plt.show()
```

```
/tmp/ipython-input-2548752322.py:6: FutureWarning:

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

  sns.countplot(x= "year", data=df_2000 ,palette="Reds", )
```



Now, suppose we want to plot a bar graph for the count of the number of meteorites fallen in the interval of 5 years between 1970 and 2000. In this case, we will have 6 intervals. They are:

1. 1970 to 1975 (excluding 1975)
2. 1975 to 1980 (excluding 1980)
3. 1980 to 1985 (excluding 1985)
4. 1985 to 1990 (excluding 1990)
5. 1990 to 1995 (excluding 1995)
6. 1995 to 2000 (**including** 2000)

Activity 2: Histogram Using The `hist()` Function

To create such bar graphs, we use a histogram. The bars in the count plot merge together to form one bar in the histogram. So, for the interval:

1. 1970 to 1975 (excluding 1975), the bars in a count plot for the years 1970, 1971, 1972, 1973 and 1974 will merge together to form one bar in a histogram.
2. 1975 to 1980 (excluding 1980), the bars in a count plot for the years 1975, 1976, 1977, 1978 and 1979 will merge together to form one bar in a histogram.
3. 1980 to 1985 (excluding 1985), the bars in a count plot for the years 1980, 1981, 1982, 1983 and 1984 will merge together to form one bar in a histogram.
4. 1985 to 1980 (excluding 1990), the bars in a count plot for the years 1985, 1986, 1987, 1988 and 1989 will merge together to form one bar in a histogram.
5. 1990 to 1995 (excluding 1995), the bars in a count plot for the years 1990, 1991, 1992, 1993 and 1994 will merge together to form one bar in a histogram.
6. 1995 to 2000 (**including** 2000), the bars in a count plot for the years 1995, 1996, 1997, 1998 and 2000 will merge together to form one bar in a histogram.

To create a histogram, you can use the `hist()` function which exists in the `matplotlib.pyplot` library. The `hist()` function requires a one dimensional array/list/series as an input to create a histogram. Let's create a histogram for the Pandas series which contains the year values between 1970 and 2000 including both of them, i.e.,

`correct_lat_long_df.loc[(correct_lat_long_df['year'] >= 1970) & (correct_lat_long_df['year'] < 2001)]`

```
# Teacher Action: Create a Pandas series containing the year values between 1970 and 20
import pandas as pd
series_2000=correct_lat_long_df.loc[(correct_lat_long_df["year"]>=1970) & (correct_lat_
series_2000
```

	year
3	1976.0
10	1974.0
16	1981.0
31	1977.0
41	1971.0
...	...
45709	1981.0
45710	1990.0
45711	1990.0
45712	1999.0
45715	1976.0

22368 rows × 1 columns

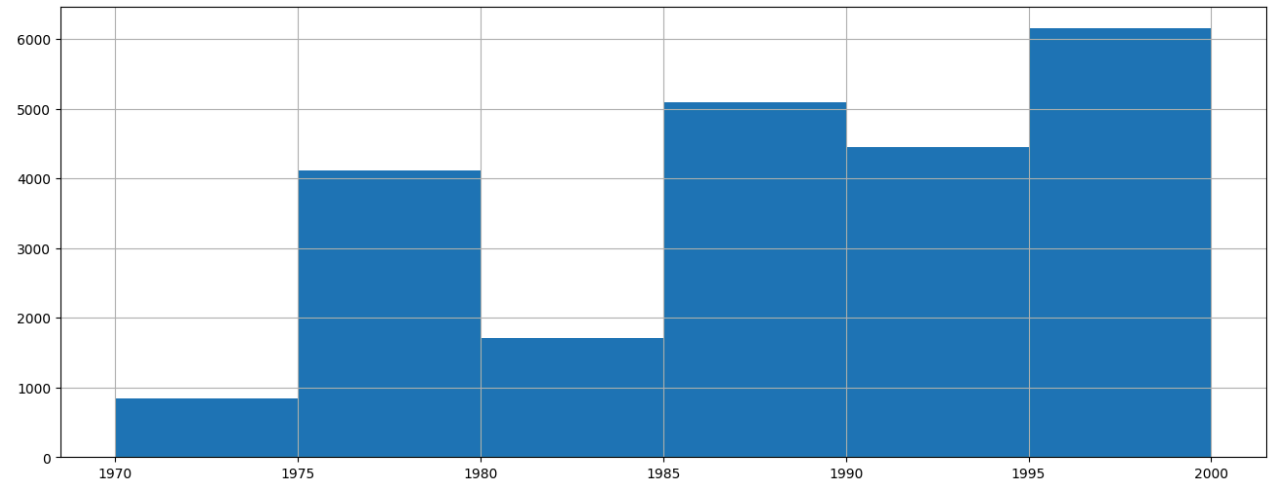
dtype: float64

So, we have sliced the `correct_lat_long_df` DataFrame to obtain a Pandas series countaining the year values between the years 1970 and 2000. **Since for every meteorite, we have recorded its fall year. Hence, the number of year values is equal to the number of meteorites fallen on Earth.**

Now let's create a histogram for the meteorites fallen between 1970 and 2000 using the `hist()` function. We will provide two inputs to the function. They are:

1. The Pandas series `met_bet_1970_and_2000_series` for which the histogram needs to be created.
2. The `bins` value to define the number of bars to be created in the histogram. Here, we are going to create 6 bars, hence we will pass `bins=6` as the second input to the `hist()` function.

```
# Teacher Action: Create a histogram for the Pandas series containing the year values b
plt.figure(figsize=(16,6))
plt.grid()
plt.hist(series_2000,bins=6)
plt.show()
```



```
exo_train_df=pd.read_csv("exoTrain.csv")
exo_train_df
```

	LABEL	FLUX.1	FLUX.2	FLUX.3	FLUX.4	FLUX.5	FLUX.6	FLUX.7	FLUX.8	FLUX.9	...	FLUX.3188	FLUX.3189	FLUX.
0	2	93.85	83.81	20.10	-26.98	-39.56	-124.71	-135.18	-96.27	-79.89	...	-78.07	-102.15	-10
1	2	-38.88	-33.83	-58.54	-40.09	-79.31	-72.81	-86.55	-85.33	-83.97	...	-3.28	-32.21	-3
2	2	532.64	535.92	513.73	496.92	456.45	466.00	464.50	486.39	436.56	...	-71.69	13.31	1
3	2	326.52	347.39	302.35	298.13	317.74	312.70	322.33	311.31	312.42	...	5.71	-3.73	-
4	2	-1107.21	-1112.59	-1118.95	-1095.10	-1057.55	-1034.48	-998.34	-1022.71	-989.57	...	-594.37	-401.66	-40
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
5082	1	-91.91	-92.97	-78.76	-97.33	-68.00	-68.24	-75.48	-49.25	-30.92	...	139.95	147.26	15
5083	1	989.75	891.01	908.53	851.83	755.11	615.78	595.77	458.87	492.84	...	-26.50	-4.84	-7
5084	1	273.39	278.00	261.73	236.99	280.73	264.90	252.92	254.88	237.60	...	-26.82	-53.89	-4
5085	1	3.82	2.09	-3.29	-2.88	1.66	-0.75	3.85	-0.03	3.28	...	10.86	-3.23	-
5086	1	323.28	306.36	293.16	287.67	249.89	218.30	188.86	178.93	118.93	...	71.19	0.97	5

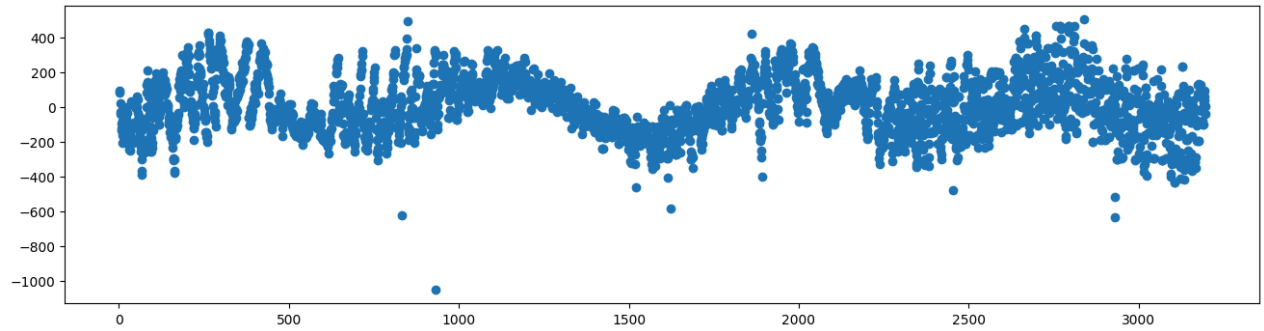
5087 rows × 3198 columns

```
star_0 = exo_train_df.iloc[0, :]
star_0.head()
```

	0
LABEL	2.00
FLUX.1	93.85
FLUX.2	83.81
FLUX.3	20.10
FLUX.4	-26.98

dtype: float64

```
#  Create a scatter plot for 'star_0' Pandas series.
# 1. Import the 'numpy' and 'matplotlib.pyplot' modules.
import matplotlib.pyplot as plt
import numpy as np
# 2. Call the 'figure()' function to resize the plot.
plt.figure(figsize=(16, 4))
# Here, 16 means the plot is 16 units wide and 4 units high. Play with these numbers t

# Call the 'scatter()' function to make a scatter plot between the x and y values.
# The scatter() function requires two inputs: x and y where x is the data to be plotte
# In our case, x is a Pandas series of numbers between 1 and 3197 and y is the 'FLUX'
x_values_star_0 = np.arange(1, 3198)
y_values_star_0 = star_0[1:]

# Here, star_0[1:] is a Pandas series containing all the 'FLUX' values starting from t
# The 'arange(1, 3198)' function from the 'numpy' module will generate numbers from 1

# 3. Call the 'scatter()' function.
plt.scatter(x_values_star_0, y_values_star_0)

# 4. Call the 'show()' function.
plt.show()

# The 'show()' function displays the plot
```




```
# Create a cartogram for the landing sites of the meteorites found in the withered cond
import folium

Valid_df=correct_lat_long_df[(correct_lat_long_df["nametype"]=="Valid") & (correct_lat_
Relict_df=correct_lat_long_df[(correct_lat_long_df["nametype"]=="Relict") & (correct_la

map1 = folium.Map(location=[0, 0], width='90%', height='90%', zoom_start=1)
for i in Relict_df.index:
  folium.Marker(location=[Relict_df.loc[i, 'reclat'], Relict_df.loc[i, 'reclong']],popu
map1
```

In the histogram above, on the horizontal axis, we have the year values and on the vertical axis, we have the count of the meteorites fallen to Earth including both `Fell` & `Found` meteorites. So from the histogram, we can say that

1. just below 1000 meteorites fell on Earth from 1970 to 1975 (excluding 1975)
2. just over 4000 meteorites fell on Earth from 1975 to 1980 (excluding 1980)
3. just below 2000 meteorites fell on Earth from 1980 to 1985 (excluding 1985)
4. just over 5000 meteorites fell on Earth from 1985 to 1990 (excluding 1990)
5. just below 4500 meteorites fell on Earth from 1990 to 1995 (excluding 1995)
6. just over 6000 meteorites fell on Earth from 1995 to 2000 (**including** 2000)

Activity 3: The `sum()` Function^^

We can find the actual number of meteorites fell on Earth in each of the six intervals using the `sum()` function which exists in the Pandas module.

Note: Number of meteorites fell on Earth is equal to the number of year values.

To do this exercise, first, obtain a Pandas series containing the year values for each interval of 5 years using the `loc[]` function. E.g.,

```
correct_lat_long_df.loc[(correct_lat_long_df['year'] >= 1970) & (correct_lat_long_df['ye
```

Then apply the `value_counts()` function followed by the `sum()` function. The `value_counts()` function will return another Pandas series containing the count of year values in the 5 year interval. The `sum()` function will return the sum of the all counts. E.g.,

```
correct_lat_long_df.loc[(correct_lat_long_df['year'] >= 1970) & (correct_lat_long_df['ye
```

```
# Student Action: Find the actual number of meteorites fell on Earth in each of the si
series_2000.value_counts().sum()
```

```
correct_lat_long_df.loc[(correct_lat_long_df['year'] >= 1970) & (correct_lat_long_df['
```

```
correct_lat_long_df.loc[(correct_lat_long_df['year'] >= 1975) & (correct_lat_long_df['
```

```
correct_lat_long_df.loc[(correct_lat_long_df['year'] >= 1980) & (correct_lat_long_df['
```

```
correct_lat_long_df.loc[(correct_lat_long_df['year'] >= 1985) & (correct_lat_long_df['
```

```
correct_lat_long_df.loc[(correct_lat_long_df['year'] >= 1990) & (correct_lat_long_df['
```

```
correct_lat_long_df.loc[(correct_lat_long_df['year'] >= 1995) & (correct_lat_long_df['y
```

Therefore,

- 846 meteorites fell on Earth from 1970 to 1975 (excluding 1975)
- 4116 meteorites fell on Earth from 1975 to 1980 (excluding 1980)
- 1708 meteorites fell on Earth from 1980 to 1985 (excluding 1985)
- 5098 meteorites fell on Earth from 1985 to 1990 (excluding 1990)
- 4448 meteorites fell on Earth from 1990 to 1995 (excluding 1995)
- 6152 meteorites fell on Earth from 1995 to 2000 (**including** 2000)

There's an alternate way to do the above exercise. Create a `while` loop to iterete through each interval of 5 years to count the number of years in each interval.

```
# Teacher Action: Create a 'while' loop to iterete through first 5 intervals to obtain
year=1970
while year<2000:
  print(correct_lat_long_df.loc[(correct_lat_long_df['year'] >= year) & (correct_lat_lo
  year=year+5
```

Activity 4: Histogram Using The `distplot()` Function^

We can also create a histogram using the `distplot()` function from the `seaborn` module. The term `distplot` stands for distribution plot. It also takes a Pandas series / NumPy array / Python list as an input. The other inputs that you can provide are `bins` and `kde` values. The `kde` parameter takes a boolean value, i.e., either `True` or `False`. Let's set the `bins` parameter equal to `6` and the `kde` parameter equal to `False` in the `distplot()` function. We will learn about the `kde` parameter later.

```
# Teacher Action: Create a histogram using the 'displot()' function from the seaborn l
plt.figure(figsize=(16,6))
sns.displot(series_2000)
plt.show()
```

As you can see, we have created the same histogram using the `distplot()` function. **The `distplot()` function is smarter in terms of choosing the appropriate default number of bins**. Hence, it is best to create a histogram using the `distplot()` function over the `hist()` function.

Now, let's create a histogram to see the distribution of the fall of meteorites over the years starting from the year `860`.

```
# Student Action: Create a histogram for the 'year' column in the 'correct_lat_long_df
plt.figure(figsize=(16,6))
sns.displot(correct_lat_long_df["year"])
plt.show()
```

If you look at the graph, the number of meteorites fallen to Earth is almost 0 till the year 1800. After that, there is some increase but it is very little. After 1900, the is a rapid increase in the number of meteorites fallen to Earth.

Let's closely observe the period after 1900 to see the variation in the number of meteorites falling to Earth by creating another histogram.

```
# Student Action: Create a histogram to visualise the number of meteorites fallen afte
df_1990=correct_lat_long_df.loc[correct_lat_long_df["year"]>1990,"year"]
plt.figure(figsize=(16,6))
sns.displot(df_1990)
plt.show()
```

```
correct_lat_long_df
```

If you increase the size of the bins, the width of the bars will decrease whereas if you decrease the size of the bins, the width of the bars will increase.

```
# Student Action: Set the bins size equal to 25 in the previous histogram.
```

```
# Student Action: Set the bins size equal to 100 in the previous histogram.
```

Activity 5: Meteorite Class-wise Histograms^^^

Let's create a meteorite class-wise histogram to observe the trend of the fall of meteorites belonging to a class. But first let's find out the number of unqiue class of meteorites present in the dataset and the count of their falls.

```
# Student Action: Calcuate the number of unique class of meteorites.
recclass_name=correct_lat_long_df["recclass"].unique()
len(recclass_name)
```