Data Visualisation

Instructions

Do the following tasks:

1. Create a DataFrame for a **sample_csv_file.csv** file which is available in your github repository: If it is not uploaded in your github repository you can use the following link.

https://raw.githubusercontent.com/jiss-sngce/CO1/main/sample_csv_file.csv

2. Display the first 5 rows of the DataFrame.
3. Display the last 5 rows of the DataFrame.
4. Find the number of rows and columns.
5. Check for the missing values in the DataFrame.
6. Create a scatter plot and a line plot between:
    - `y` and `x1`
    - `y` and `x2`
    - `y` and `x3`
    - `y` and `x4`
    - `y` and `2 * x1 + 3 * x2 + 4 * x3 + 5 * x4`

    such that the `y` values are plotted on the horizontal axis and `x1, x2, x3, 4` values are plotted on the vertical axis.

---

1. Create a DataFrame

Import the required modules. Create a DataFrame for the dataset and store it in the `df` variable.

```
#Create a DataFrame for the dataset and store it in the df variable.
import pandas as pd
df=pd.read_csv('https://raw.githubusercontent.com/jiss-sngce/CO1/main/sample_csv_file.c
df.head()

```

	y	x1	x2	x3	x4
0	1	1	1	0.000000	0.841471
1	2	2	4	0.693147	0.909297
2	3	3	9	1.098612	0.141120

2. Display The First 5 Rows

Here you have to display the first five rows of the `df` DataFrame.

```
1 # Display the first 5 rows using the 'head()' function
2 df.head()
3
```

	y	x1	x2	x3	x4
0	1	1	1	0.000000	0.841471
1	2	2	4	0.693147	0.909297
2	3	3	9	1.098612	0.141120
3	4	4	16	1.386294	-0.756802
4	5	5	25	1.609438	-0.958924

---

3. Display The Last 5 Rows

Here you have to display the last five rows of the `df` DataFrame.

```
1 # Display the last 5 rows using the 'tail()' function
2 df.tail()
3
```

	y	x1	x2	x3	x4
494	495	495	245025	6.204558	-0.980234
495	496	496	246016	6.206576	-0.363143
496	497	497	247009	6.208590	0.587819
497	498	498	248004	6.210600	0.998344
498	499	499	249001	6.212606	0.490995

```
1 df.tail(12)
```

	y	x1	x2	x3	x4
487	488	488	238144	6.190315	-0.868981
488	489	489	239121	6.192362	-0.885911
489	490	490	240100	6.194405	-0.088339
490	491	491	241081	6.196444	0.790452
491	492	492	242064	6.198479	0.942504
492	493	493	243049	6.200509	0.228023
493	494	494	244036	6.202536	-0.696102
494	495	495	245025	6.204558	-0.980234
495	496	496	246016	6.206576	-0.363143
496	497	497	247009	6.208590	0.587819
497	498	498	248004	6.210600	0.998344
498	499	499	249001	6.212606	0.490995

4. Display Number Of Rows & Columns

Now, display the number of rows and columns that are present in the `df` DataFrame.

```
# Display the number of rows and columns using the 'shape' keyword
df.shape
```

```
(499, 5)
```

```
df.shape[0]
```

```
499
```

```
df.shape[1]
```

```
5
```

---

5. Check For The Missing Values

Check whether the `df` DataFrame contains the missing values.

```
# Check for the missing values using the 'isnull()' function.
df.isnull()
```

	y	x1	x2	x3	x4
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
494	False	False	False	False	False
495	False	False	False	False	False
496	False	False	False	False	False
497	False	False	False	False	False
498	False	False	False	False	False

499 rows × 5 columns

```
df.isnull().sum()
```

```
y     0
x1    0
x2    0
x3    0
x4    0
dtype: int64
```

Hint: Use the `sum()` function on top of the `isnull()` to find the total number of `True` values in each column.

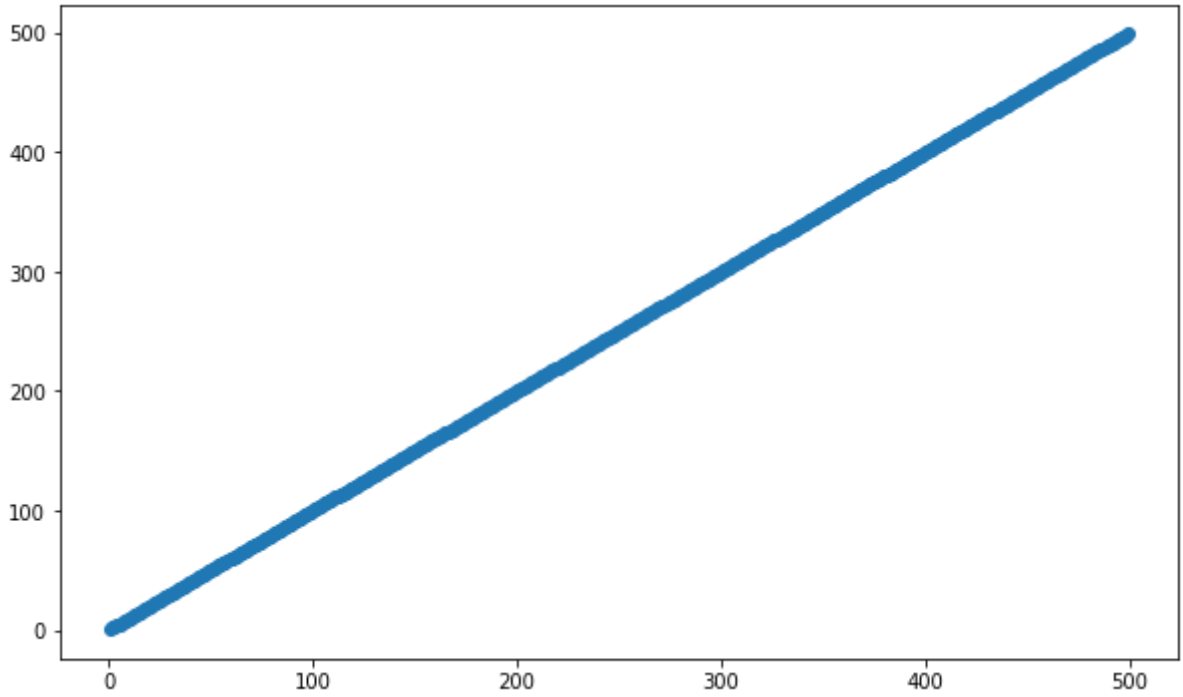
---

6. Scatter & Line Plots

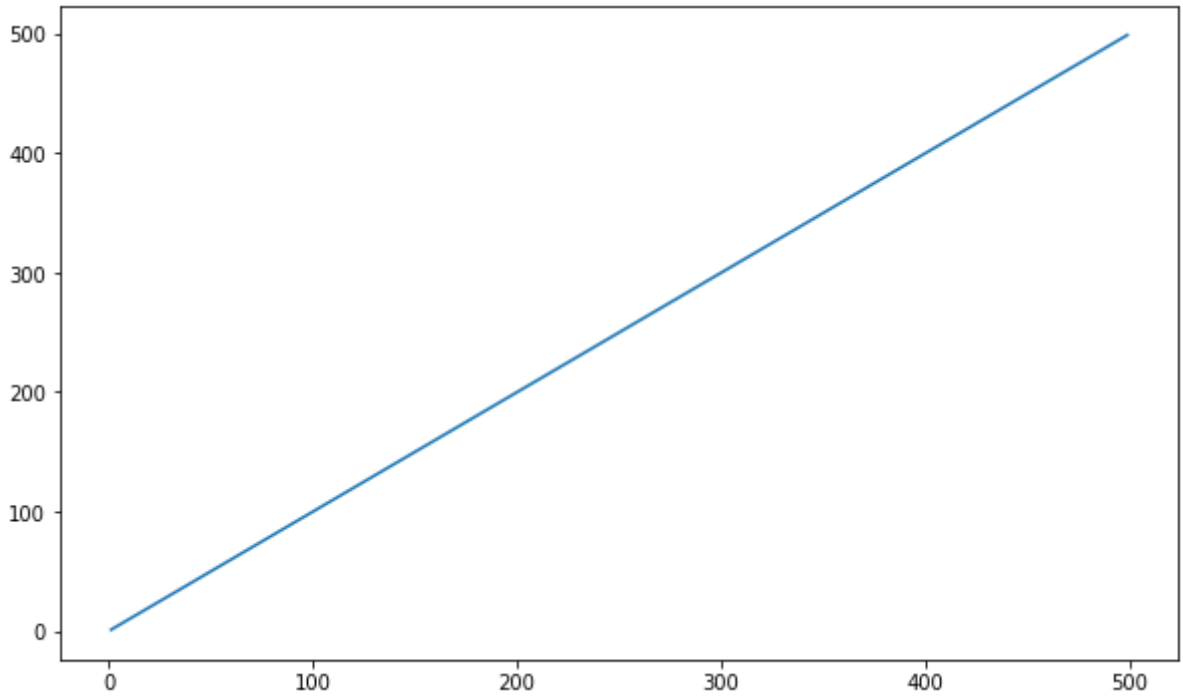
Now you have to create the scatter plots and line plots between the following columns:

1. `y` and `x1`
2. `y` and `x2`
3. `y` and `x3`
4. `y` and `x4`
5. `y1` and `2 * x1 + 3 * x2 + 4 * x3 + 5 * x4`

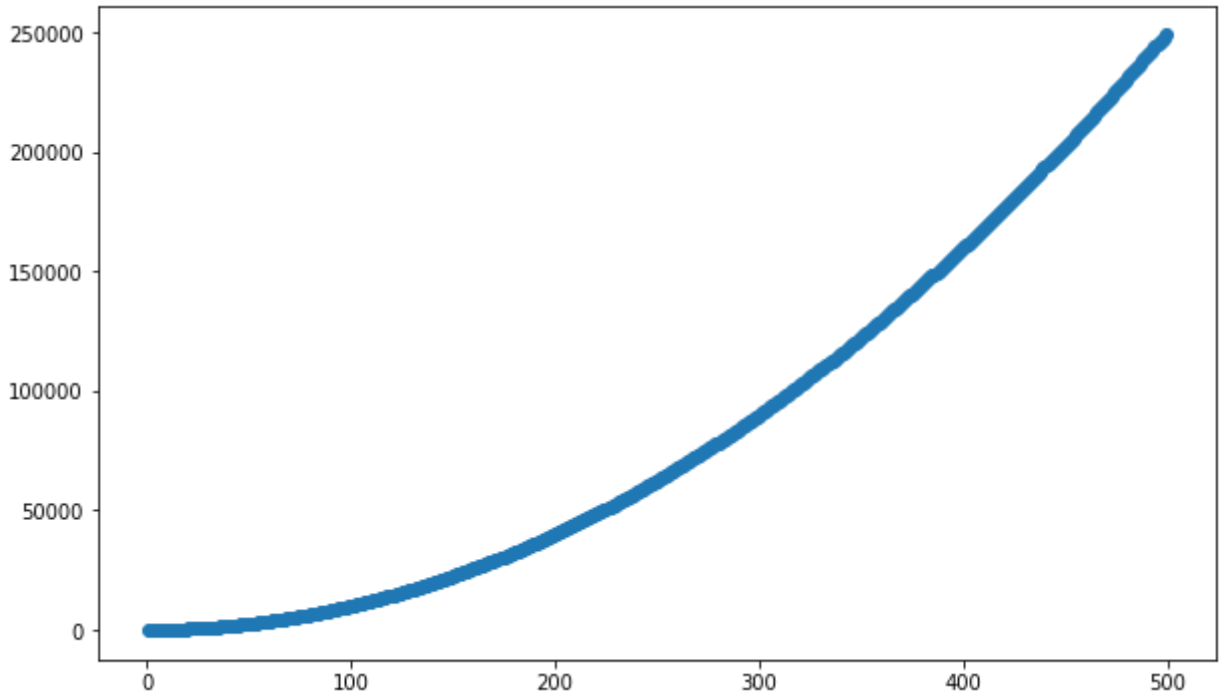
```
# Scatter plot between y and x1.
# Pass 'df['y']' and 'df['x1']' inside the 'scatter()' function of the 'matplotlib.pypl
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
plt.scatter(df['y'], df['x1'])
plt.show()

```



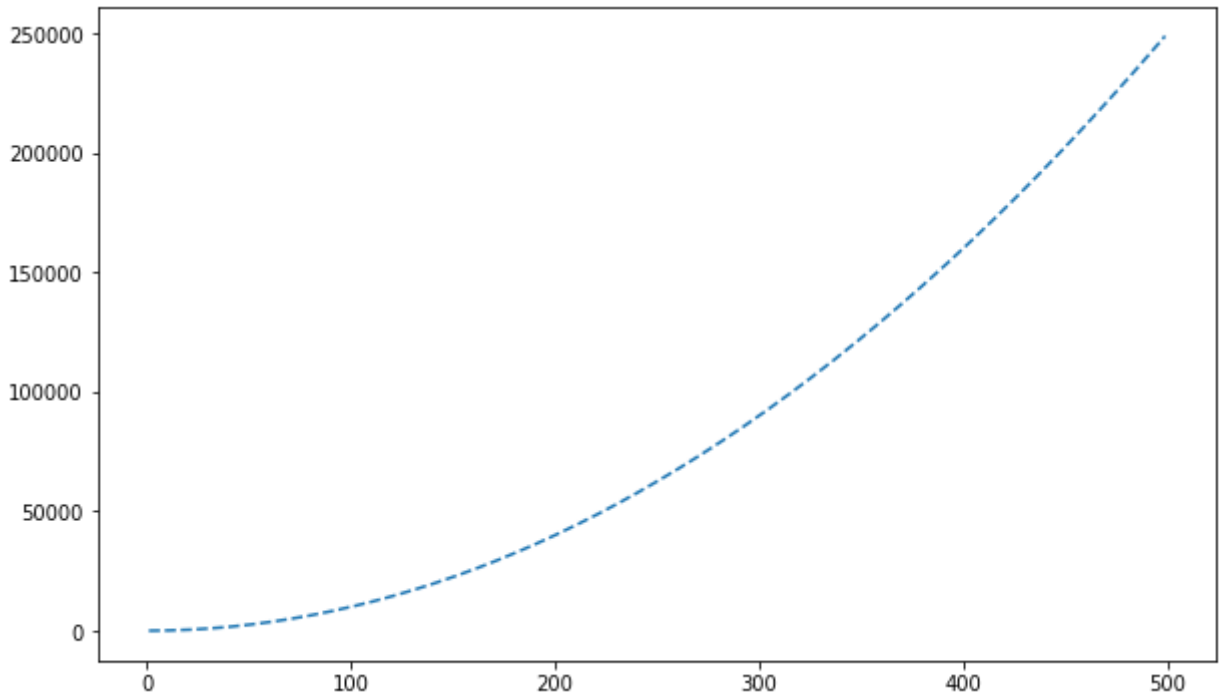

```
# Line plot between y and x1.
# Pass 'df['y']' and 'df['x1']' inside the 'plot()' function of the 'matplotlib.pyplot'
plt.figure(figsize=(10, 6))
plt.plot(df['y'], df['x1'])
plt.show()
```



```
1 # Scatter plot between y and x2.
2 plt.figure(figsize=(10, 6))
3 plt.scatter(df['y'], df['x2'])
4 plt.show()
5
6
```

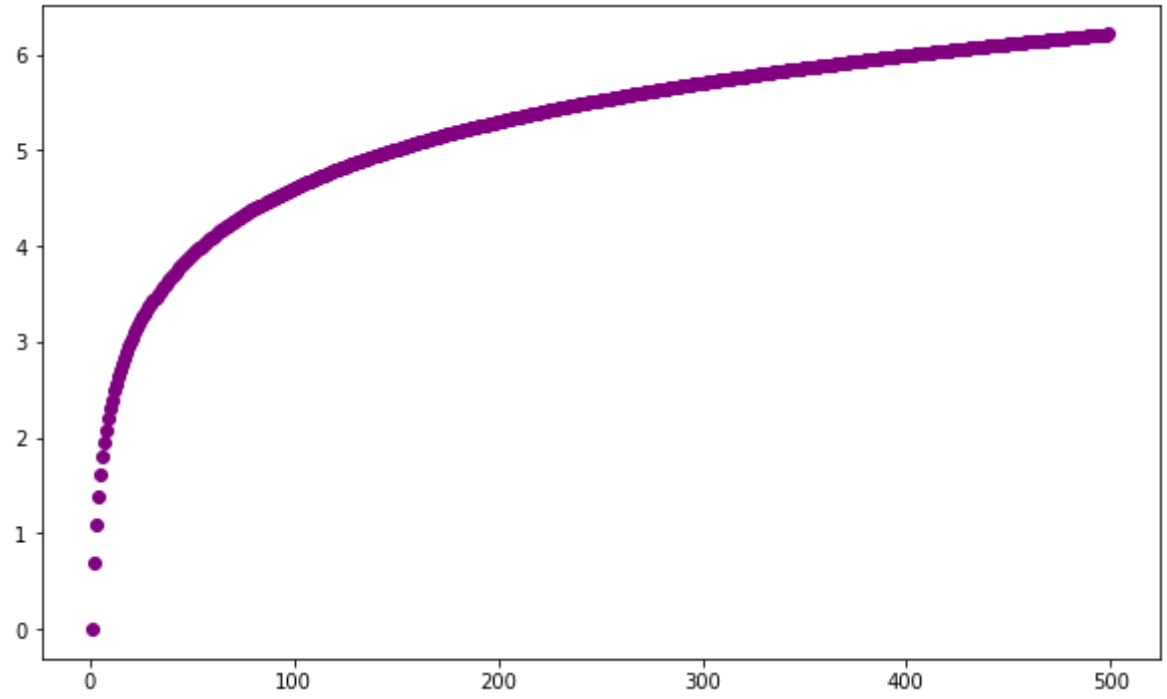



```
1 # Line plot between y and x2.
2 plt.figure(figsize=(10, 6))
3 plt.plot(df['y'], df['x2'],linestyle='--')
4 plt.show()
5
```

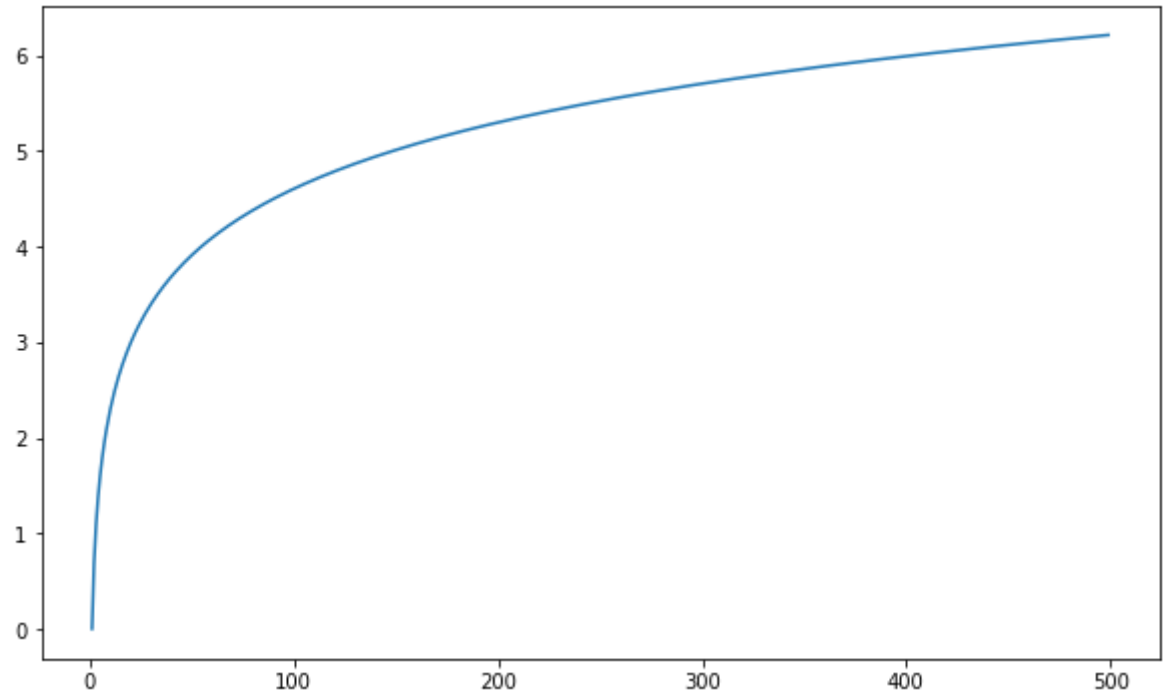



```
1 # Scatter plot between y and x3.
```

```
2 plt.figure(figsize=(10, 6))
3 plt.scatter(df['y'], df['x3'],color='purple')
4 plt.show()
5
```

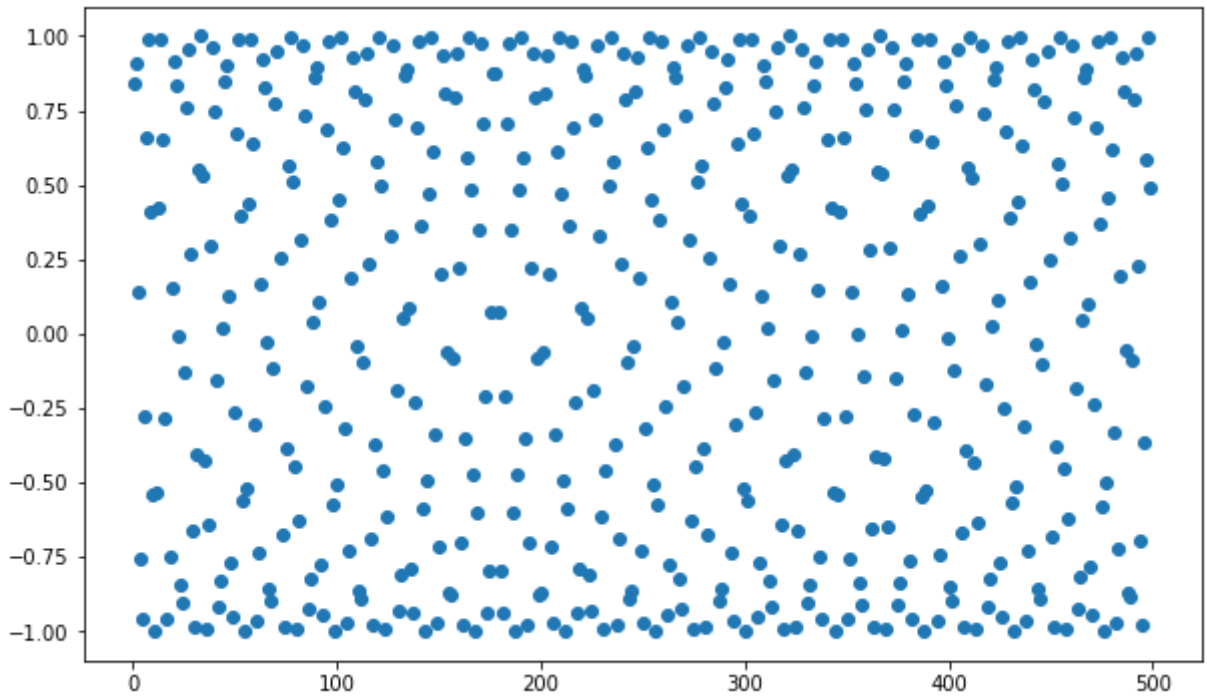



```
1 # Line plot between y and x3.
2 plt.figure(figsize=(10, 6))
3 plt.plot(df['y'], df['x3'])
4 plt.show()
5
```

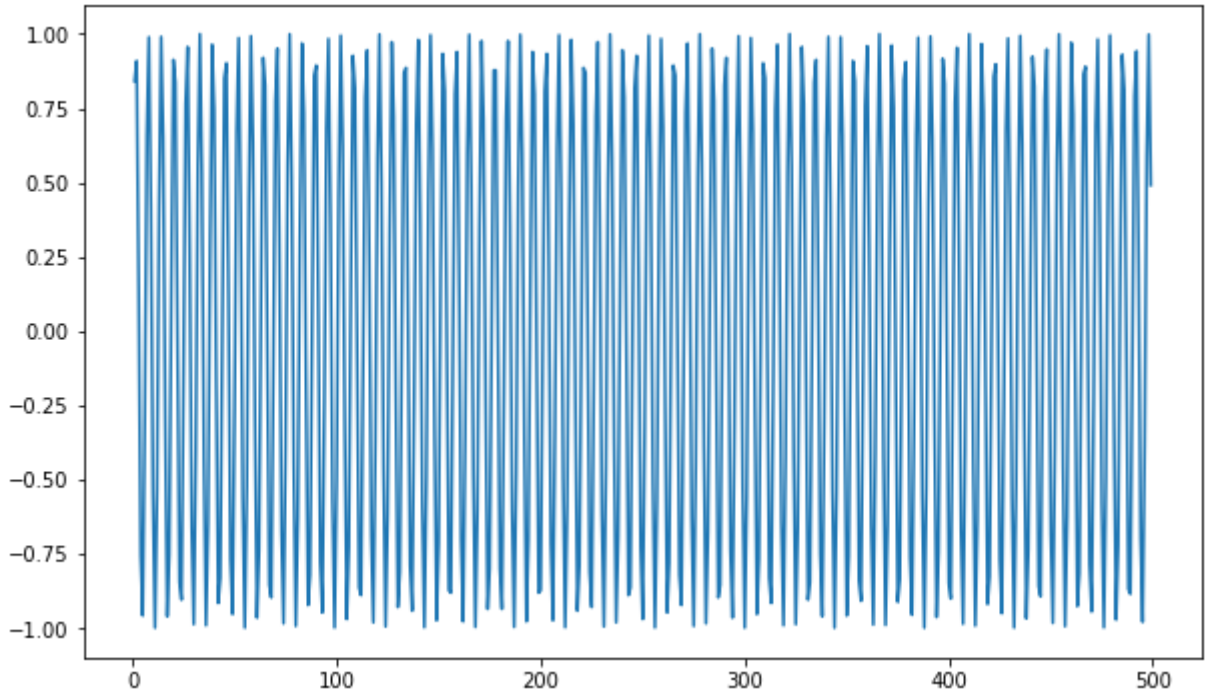



```
1 # Scatter plot between y and x4.
2 plt.figure(figsize=(10, 6))
3 plt.scatter(df['y'], df['x4'])
4 plt.show()
```

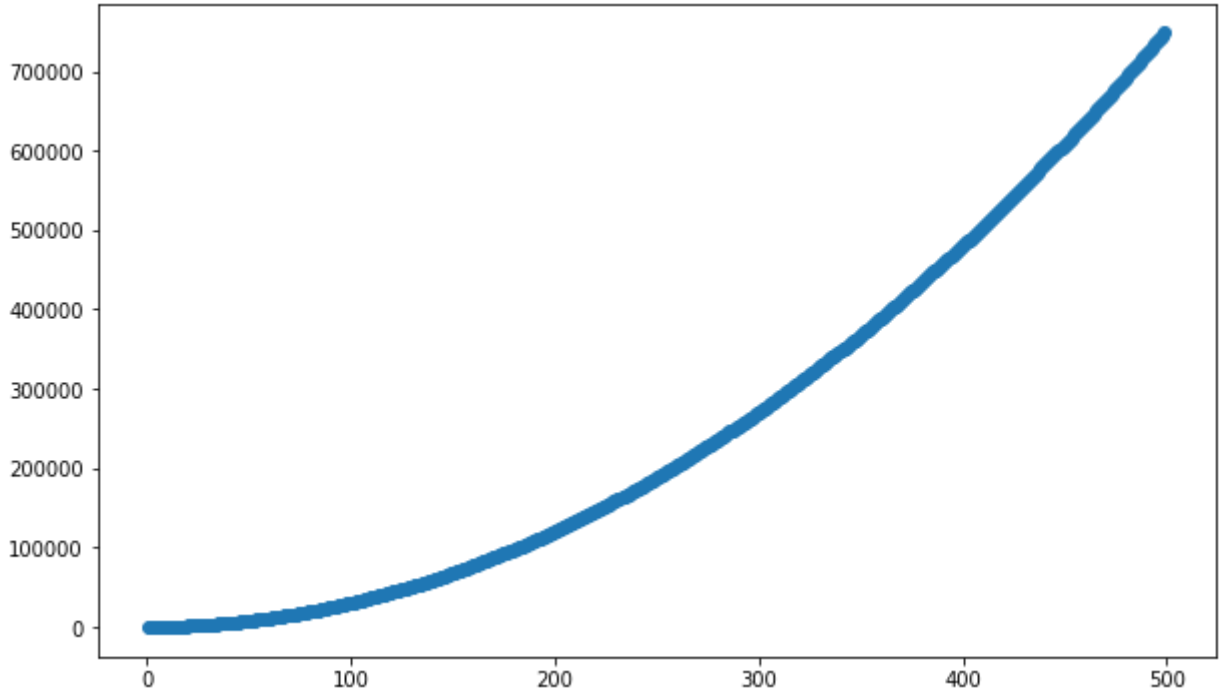
```
plt.show()
5
```




```
1 # Line plot between y and x4.
2 plt.figure(figsize=(10, 6))
3 plt.plot(df['y'], df['x4'])
4 plt.show()
5
```




```
1 # Scatter plot between y and 2 * x1 + 3 * x2 + 4 * x3 + 5 * x4.
2 plt.figure(figsize=(10, 6))
3 plt.scatter(df['y'],(2*df['x1']+3*df['x2']+4*df['x3']+5*df['x4']))
4 plt.show()
5
```




```
1 # Line plot between y and 2 * x1 + 3 * x2 + 4 * x3 + 5 * x4.
2 plt.figure(figsize=(10, 6))
3 plt.plot(df['y'],(2*df['x1']+3*df['x2']+4*df['x3']+5*df['x4']))
4 plt.show()
5
```