Types of Data

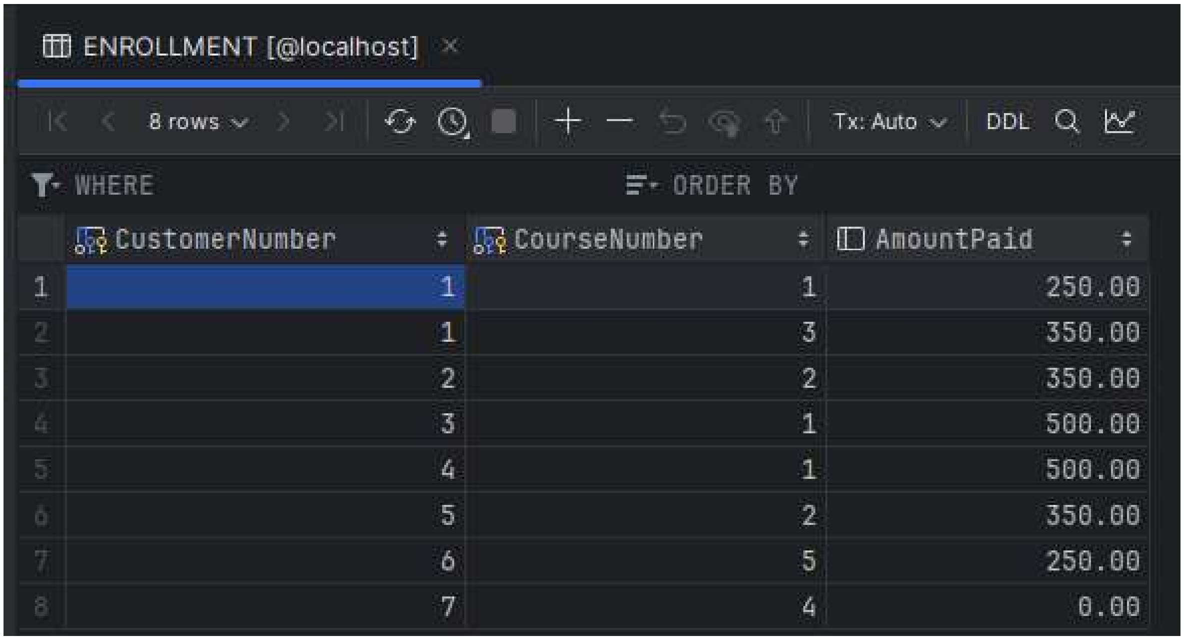
Member

ID	Name	Image
65070501044	PHURIS PIMPAT	
65070501055	SORRAWIT UDOMVITTAYAKRAI	
65070501073	CHAIYAPAT MEEYING	
65070501083	PANURUT SIRINAPAISAN	
65070501092	BHAGYA SARANUNT	

Structured Data

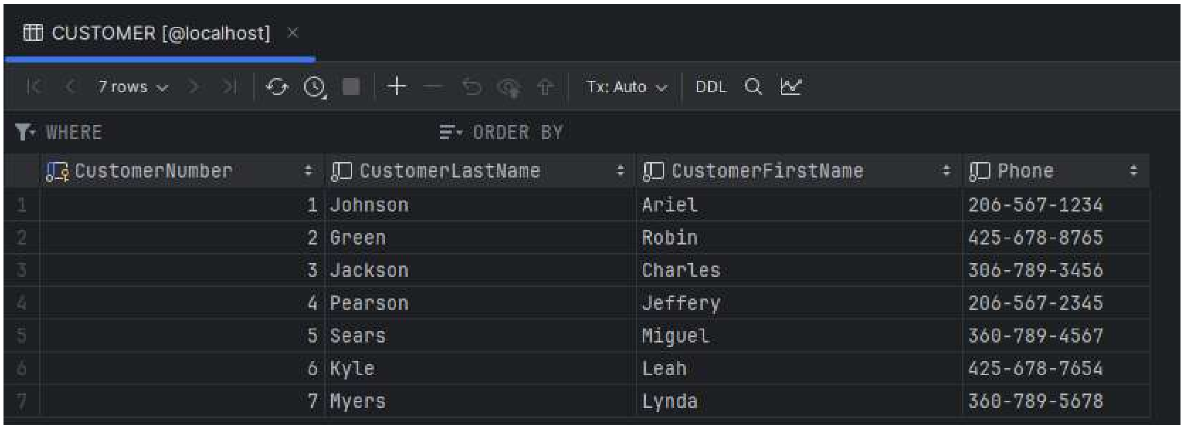
1. Transaction Data

this is relational database from Lab1 of CPE241 Database Systems



	CustomerNumber	CourseNumber	AmountPaid
1	1	1	250.00
2	1	3	350.00
3	2	2	350.00
4	3	1	500.00
5	4	1	500.00
6	5	2	350.00
7	6	5	250.00
8	7	4	0.00

2. Master Table



	CustomerNumber	CustomerLastName	CustomerFirstName	Phone
1	1	Johnson	Ariel	206-567-1234
2	2	Green	Robin	425-678-8765
3	3	Jackson	Charles	306-789-3456
4	4	Pearson	Jeffery	206-567-2345
5	5	Sears	Miguel	360-789-4567
6	6	Kyle	Leah	425-678-7654
7	7	Myers	Lynda	360-789-5678

3. Time Series Data

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

```
!gdown 1VRC_TXzUSj68c5Mf3sy3LVyoDV5T9xtc
```

```
Downloading...
From: https://drive.google.com/uc?id=1VRC_TXzUSj68c5Mf3sy3LVyoDV5T9xtc
To: /content/heart_rate.csv
100% 46.4k/46.4k [00:00<00:00, 81.1MB/s]
```

```
df = pd.read_csv('/content/heart_rate.csv')
df.head()
```

```
        T1       T2       T3       T4
0  84.2697  91.4634  60.4839  59.2885
1  84.2697  91.4634  60.4839  59.2885
2  84.0619  91.1834  60.4606  59.2885
3  85.6542  91.8788  60.3391  58.8973
4  87.2093  91.1772  60.0762  58.4359
```

	T1	T2	T3	T4
0	84.2697	91.4634	60.4839	59.2885
1	84.2697	91.4634	60.4839	59.2885
2	84.0619	91.1834	60.4606	59.2885
3	85.6542	91.8788	60.3391	58.8973
4	87.2093	91.1772	60.0762	58.4359

```
df["T3"].fillna(0,inplace=True)
df["T4"].fillna(0,inplace=True)
df
```

```
            T1       T2       T3       T4
0      84.2697  91.4634  60.4839  59.2885
1      84.2697  91.4634  60.4839  59.2885
2      84.0619  91.1834  60.4606  59.2885
3      85.6542  91.8788  60.3391  58.8973
4      87.2093  91.1772  60.0762  58.4359
...        ...      ...      ...      ...
1795  103.7900  98.6842   0.0000   0.0000
1796  101.6230  98.6842   0.0000   0.0000
1797   99.5679  99.0005   0.0000   0.0000
1798   99.1835  99.3273   0.0000   0.0000
1799   98.8567  99.5205   0.0000   0.0000
```

```
[1800 rows x 4 columns]
```

	T1	T2	T3	T4
0	84.2697	91.4634	60.4839	59.2885
1	84.2697	91.4634	60.4839	59.2885
2	84.0619	91.1834	60.4606	59.2885
3	85.6542	91.8788	60.3391	58.8973
4	87.2093	91.1772	60.0762	58.4359
...	...	...	...	...
1795	103.7900	98.6842	0.0000	0.0000
1796	101.6230	98.6842	0.0000	0.0000
1797	99.5679	99.0005	0.0000	0.0000
1798	99.1835	99.3273	0.0000	0.0000
1799	98.8567	99.5205	0.0000	0.0000

1800 rows × 4 columns

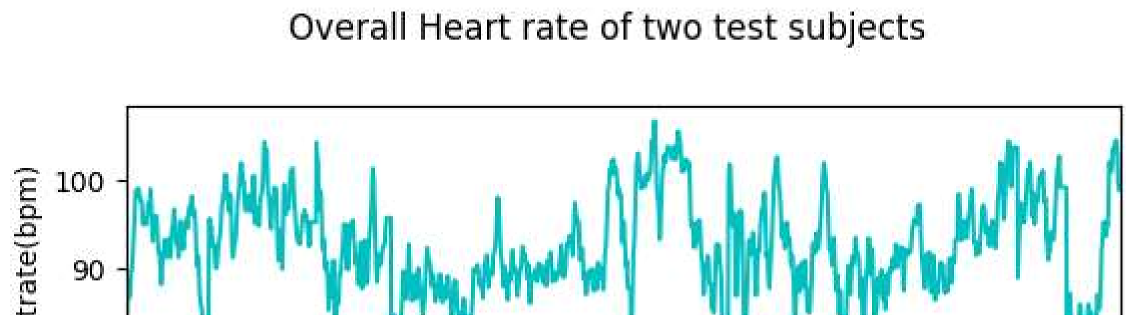
Select the T1 and T2 data and use the matplotlib to some simple visualization

```
testSubject1 = df['T1']
testSubject2 = df['T2']
```

```
fig, (tsj1,tsj2) = plt.subplots(2,1)
fig.suptitle('Overall Heart rate of two test subjects')
tsj1.set_xlim(0.00,1800)
tsj2.set_xlim(0.00,1800)

xlabels = [item.get_text() for item in tsj1.get_xticklabels()]
for i in range(0,len(xlabels)):
  xlabels[i] = str(int(int(xlabels[i])/2))
fig.subplots_adjust(hspace=0.3)
#Figure of Testsubject1 Heartrate
tsj1.plot(testSubject1,'c')
tsj1.set_ylabel('Heartrate(bpm)')
tsj1.set_xlabel('time(sec)')
tsj1.set_xticklabels(xlabels)


#Figure of Testsubject1 Heartrate
tsj2.plot(testSubject2,'y')
tsj2.set_ylabel('Heartrate(bpm)')
tsj2.set_xlabel('time(sec)')
tsj2.set_xticklabels(xlabels)
fig.show()
```

```
<Figure size 640x480 with 2 Axes>
```

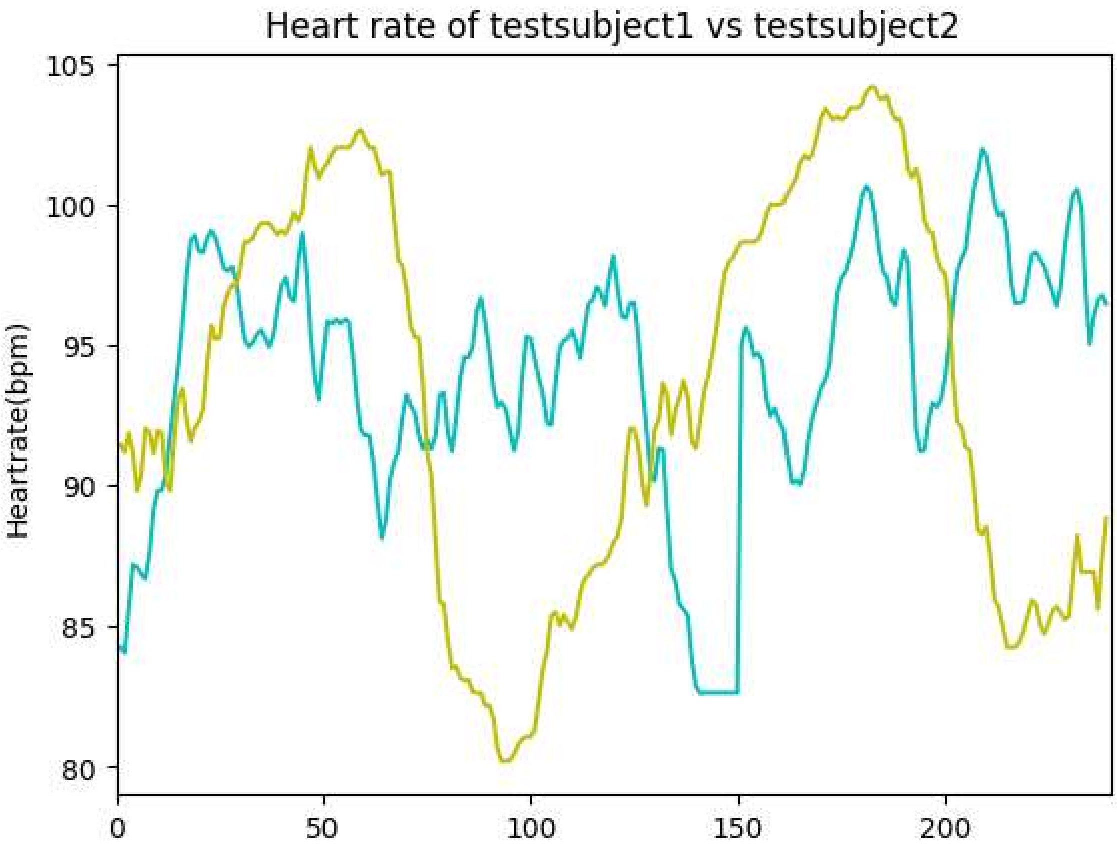
Download



```
plt.xlim(0.00,240)
plt.title('Heart rate of testsubject1 vs testsubject2')
plt.plot(testSubject1[0:240],'c',testSubject2[0:240],'y')
plt.ylabel('Heartrate(bpm)')
plt.show()
```

```
<Figure size 640x480 with 1 Axes>
```

Download



4. Graph/Network

```
# @title Knuth miles Data Sample
import gzip

# Path to your .txt.gz file
file_path = "/content/knuth_miles.txt.gz"

try:
    with gzip.open(file_path, 'rt') as f:
        line_count = 0
        for line in f:
            print(line.strip())
            line_count += 1
            if line_count >= 20:
                break
except FileNotFoundError:
    print("File not found.")
except gzip.BadGzipFile:
    print("Invalid gzip file.")
except Exception as e:
    print("An error occurred:", e)
```

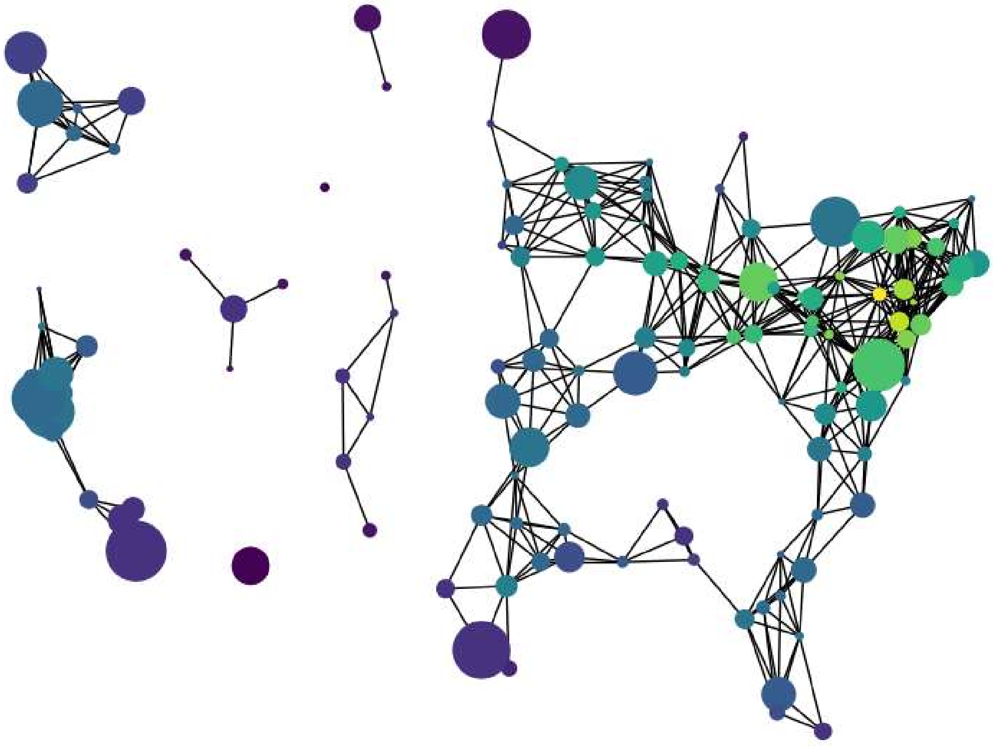
```
* This file miles_dat.txt is part of NetworkX and is distributed
* with the same license as NetworkX.
* Distributed under the terms of the GNU Lesser General Public Licens
* http://www.gnu.org/copyleft/lesser.html
* This file is not part of the Stanford GraphBase; the name has been
* changed to avoid any confusion with files from that collection.
* Original attribution:
* File "miles.dat" from the Stanford GraphBase (C) 1993 Stanford Univ
* Revised mileage data for highways in the United States and Canada,
* This file may be freely copied but please do not change it in any w
* (Checksum parameters 696,295999341)
Youngstown, OH[4110,8065]115436
Yankton, SD[4288,9739]12011
966
Yakima, WA[4660,12051]49826
1513 2410
Worcester, MA[4227,7180]161799
2964 1520 604
Wisconsin Dells, WI[4363,8977]2521
1149 1817 481 595
```

```
# @title The undirected graph of 128 US cities. The cities each have loc
# @markdown Sample Dataset from NetworkX
import gzip
import re

# Ignore any warnings related to downloading shpfiles with cartopy
import warnings

warnings.simplefilter("ignore")

import numpy as np
import matplotlib.pyplot as plt
import networkx as nx


def miles_graph():
    """Return the cites example graph in miles_dat.txt
    from the Stanford GraphBase.
    """
    # open file miles_dat.txt.gz (or miles_dat.txt)

    fh = gzip.open("/content/knuth_miles.txt.gz", "r")

    G = nx.Graph()
    G.position = {}
    G.population = {}

    cities = []
    for line in fh.readlines():
        line = line.decode()
        if line.startswith("*"):  # skip comments
            continue

        numfind = re.compile(r"^\d+")

        if numfind.match(line):  # this line is distances
            dist = line.split()
            for d in dist:
                G.add_edge(city, cities[i], weight=int(d))
                i = i + 1
        else:  # this line is a city, position, population
            i = 1
            (city, coordpop) = line.split("[")
            cities.insert(0, city)
            (coord, pop) = coordpop.split("]")
            (y, x) = coord.split(",")

            G.add_node(city)
            # assign position - Convert string to lat/long
            G.position[city] = (-float(x) / 100, float(y) / 100)
            G.population[city] = float(pop) / 1000
    return G
```

```
G = miles_graph()

print("Loaded miles_dat.txt containing 128 cities.")
print(G)

# make new graph of cites, edge if less than 300 miles between them
H = nx.Graph()
for v in G:
    H.add_node(v)
for u, v, d in G.edges(data=True):
    if d["weight"] < 300:
        H.add_edge(u, v)

# draw with matplotlib/pylab
fig = plt.figure(figsize=(8, 6))

# nodes colored by degree sized by population
node_color = [float(H.degree(v)) for v in H]

# Use cartopy to provide a backdrop for the visualization
try:
    import cartopy.crs as ccrs
    import cartopy.io.shapereader as shpreader

    ax = fig.add_axes([0, 0, 1, 1], projection=ccrs.LambertConformal(),
    ax.set_extent([-125, -66.5, 20, 50], ccrs.Geodetic())
    # Add map of countries & US states as a backdrop
    for shapename in ("admin_1_states_provinces_lakes_shp", "admin_0_cou
        shp = shpreader.natural_earth(
            resolution="110m", category="cultural", name=shapename
        )
        ax.add_geometries(
            shpreader.Reader(shp).geometries(),
            ccrs.PlateCarree(),
            facecolor="none",
            edgecolor="k",
        )
    # NOTE: When using cartopy, use matplotlib directly rather than nx.d
    # to take advantage of the cartopy transforms
    ax.scatter(
        *np.array(list(G.position.values())).T,
        s=[G.population[v] for v in H],
        c=node_color,
        transform=ccrs.PlateCarree(),
        zorder=100,  # Ensure nodes lie on top of edges/state lines
    )
    # Plot edges between the cities
    for edge in H.edges():
        edge_coords = np.array([G.position[v] for v in edge])
        ax.plot(
            edge_coords[:, 0],
            edge_coords[:, 1],
            transform=ccrs.PlateCarree(),
                linewidth=0.75,
                color="k",
            )

except ImportError:
    # If cartopy is unavailable, the backdrop for the plot will be blank
    # though you should still be able to discern the general shape of th
    # from graph nodes and edges!
    nx.draw(
        H,
        G.position,
        node_size=[G.population[v] for v in H],
        node_color=node_color,
        with_labels=False,
    )

plt.show()
```

```
Loaded miles_dat.txt containing 128 cities.
Graph with 128 nodes and 8128 edges


<Figure size 800x600 with 1 Axes>
```

Download



5. CrossTable

```
!gdown 1VVLv4t8jH1DmiTpwP5QxxJgPPZH7QVfN
```

```
Downloading...
From: https://drive.google.com/uc?id=1VVLv4t8jH1DmiTpwP5QxxJgPPZH7QVfN
To: /content/FDI.csv
100% 232k/232k [00:00<00:00, 95.2MB/s]
```

```
import pandas as pd

df = pd.read_csv("/content/FDI.csv")
df.head(5)
```

```
                  Country Name Country Code  \
0                        Aruba          ABW
1  Africa Eastern and Southern          AFE
2                  Afghanistan          AFG
3   Africa Western and Central          AFW
4                       Angola          AGO

                                      Indicator Name     Indicator Co
0  Foreign direct investment, net inflows (BoP, c...  BX.KLT.DINV.CD.
1  Foreign direct investment, net inflows (BoP, c...  BX.KLT.DINV.CD.
2  Foreign direct investment, net inflows (BoP, c...  BX.KLT.DINV.CD.
3  Foreign direct investment, net inflows (BoP, c...  BX.KLT.DINV.CD.
4  Foreign direct investment, net inflows (BoP, c...  BX.KLT.DINV.CD.

   1961  1962  1963  1964  1965  ...          2014          2015  \
0   NaN   NaN   NaN   NaN   NaN  ...  2.506181e+08 -2.877586e+07
1   NaN   NaN   NaN   NaN   NaN  ...  2.768142e+10  2.877423e+10
2   NaN   NaN   NaN   NaN   NaN  ...  4.297526e+07  1.691466e+08
3   NaN   NaN   NaN   NaN   NaN  ...  1.659803e+10  1.564317e+10
```

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	...
0	Aruba	ABW	Foreign direct investment, net inflows (BoP, c...	BX.KLT.DINV.CD.WD	NaN	NaN	NaN	NaN	NaN	NaN	...
1	Africa Eastern and Southern	AFE	Foreign direct investment, net inflows (BoP, c...	BX.KLT.DINV.CD.WD	NaN	NaN	NaN	NaN	NaN	NaN	...
2	Afghanistan	AFG	Foreign direct investment, net inflows	BX.KLT.DINV.CD.WD	NaN	NaN	NaN	NaN	NaN	NaN	...

```python
df.fillna(0, inplace=True)
df
```

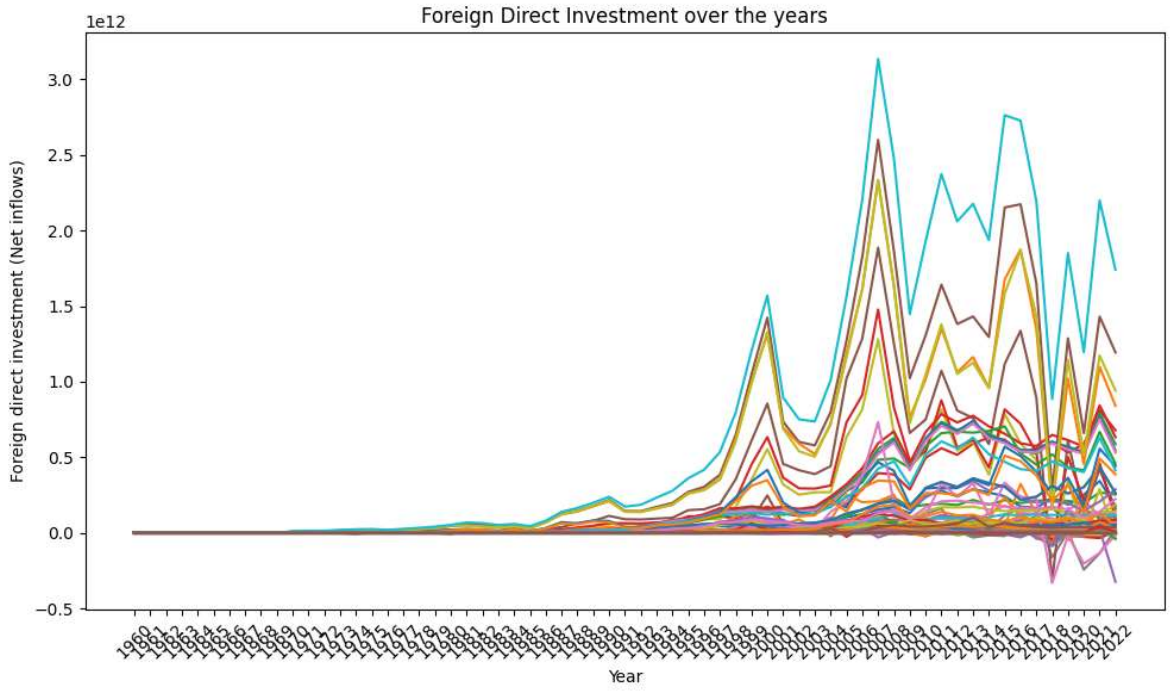
```
                    Country Name Country Code  \
0                          Aruba          ABW
1    Africa Eastern and Southern          AFE
2                    Afghanistan          AFG
3     Africa Western and Central          AFW
4                         Angola          AGO
..                           ...          ...
261                       Kosovo          XKX
262                  Yemen, Rep.          YEM
263                 South Africa          ZAF
264                       Zambia          ZMB
265                     Zimbabwe          ZWE

                                        Indicator Name     Indicator
0    Foreign direct investment, net inflows (BoP, c...   BX.KLT.DINV.C
1    Foreign direct investment, net inflows (BoP, c...   BX.KLT.DINV.C
2    Foreign direct investment, net inflows (BoP, c...   BX.KLT.DINV.C
3    Foreign direct investment, net inflows (BoP, c...   BX.KLT.DINV.C
4    Foreign direct investment, net inflows (BoP, c...   BX.KLT.DINV.C
```

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	...
0	Aruba	ABW	Foreign direct investment, net inflows (BoP, c...	BX.KLT.DINV.CD.WD	0.0	0.0	0.0	0.0	0.0	0.0	...
1	Africa Eastern and Southern	AFE	Foreign direct investment, net inflows (BoP, c...	BX.KLT.DINV.CD.WD	0.0	0.0	0.0	0.0	0.0	0.0	...
2	Afghanistan	AFG	Foreign direct investment, net inflows (BoP, c...	BX.KLT.DINV.CD.WD	0.0	0.0	0.0	0.0	0.0	0.0	...
3	Africa Western and Central	AFW	Foreign direct investment, net inflows (BoP, c...	BX.KLT.DINV.CD.WD	0.0	0.0	0.0	0.0	0.0	0.0	...
4	Angola	AGO	Foreign direct investment, net inflows (BoP, c...	BX.KLT.DINV.CD.WD	0.0	0.0	0.0	0.0	0.0	0.0	...
...	...	...	...	...	...	...	...	...	...	...	...
261	Kosovo	XKX	Foreign direct investment, net inflows (BoP, c...	BX.KLT.DINV.CD.WD	0.0	0.0	0.0	0.0	0.0	0.0	...
262	Yemen, Rep.	YEM	Foreign direct investment, net inflows (BoP, c...	BX.KLT.DINV.CD.WD	0.0	0.0	0.0	0.0	0.0	0.0	...
263	South Africa	ZAF	Foreign direct investment, net inflows (BoP, c...	BX.KLT.DINV.CD.WD	0.0	0.0	0.0	0.0	0.0	0.0	...
			Foreign								

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))

years = [str(year) for year in range(1960, 2023)]
selected_columns = ['Country Name'] + years

data = df[selected_columns]

# Plotting
for index, row in data.iterrows():
    country_name = row['Country Name']
    data = row[years]
    plt.plot(years, data, label=country_name)

plt.xlabel('Year')
plt.ylabel('Foreign direct investment (Net inflows)')
plt.title('Foreign Direct Investment over the years')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
<Figure size 1000x600 with 1 Axes>
```

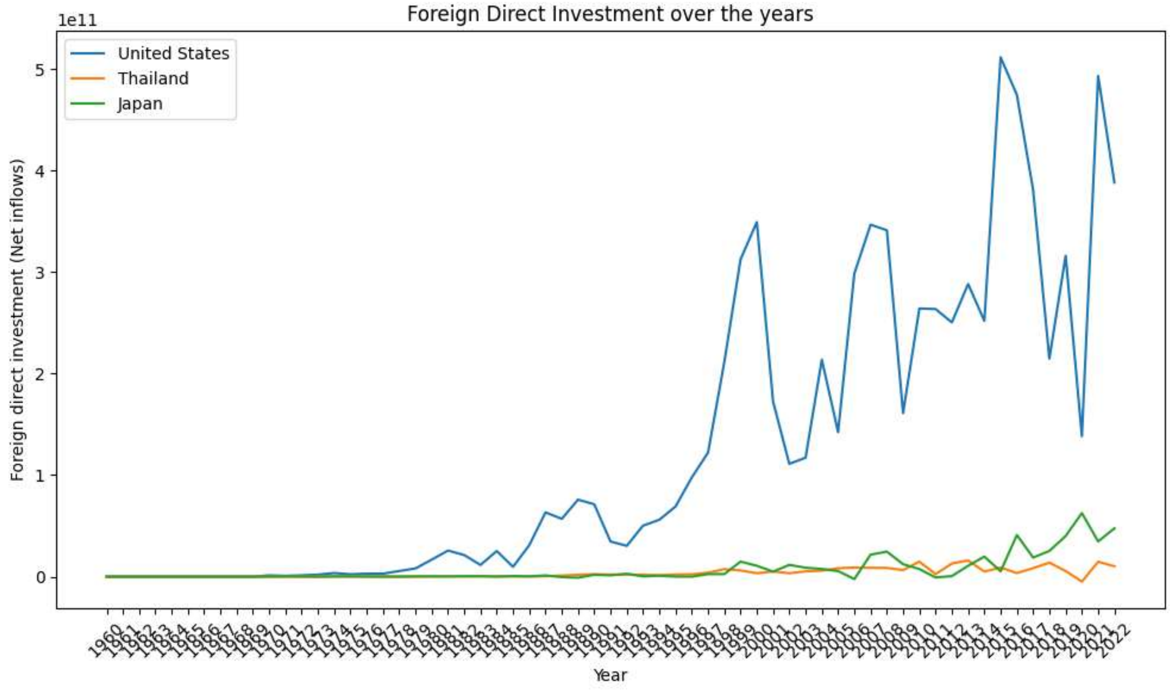
Download



```python
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))

years = [str(year) for year in range(1960, 2023)]
country_name = ['United States','Thailand','Japan']

for country in country_name:
  country_df = df[df['Country Name'] == country]
  data = country_df.loc[:, years].values.flatten()
  plt.plot(years,data,label=country)

plt.xlabel('Year')
plt.ylabel('Foreign direct investment (Net inflows)')
plt.title('Foreign Direct Investment over the years')
plt.xticks(rotation=45)
plt.tight_layout()
plt.legend()
plt.show()
```

```
<Figure size 1000x600 with 1 Axes>
```

Download

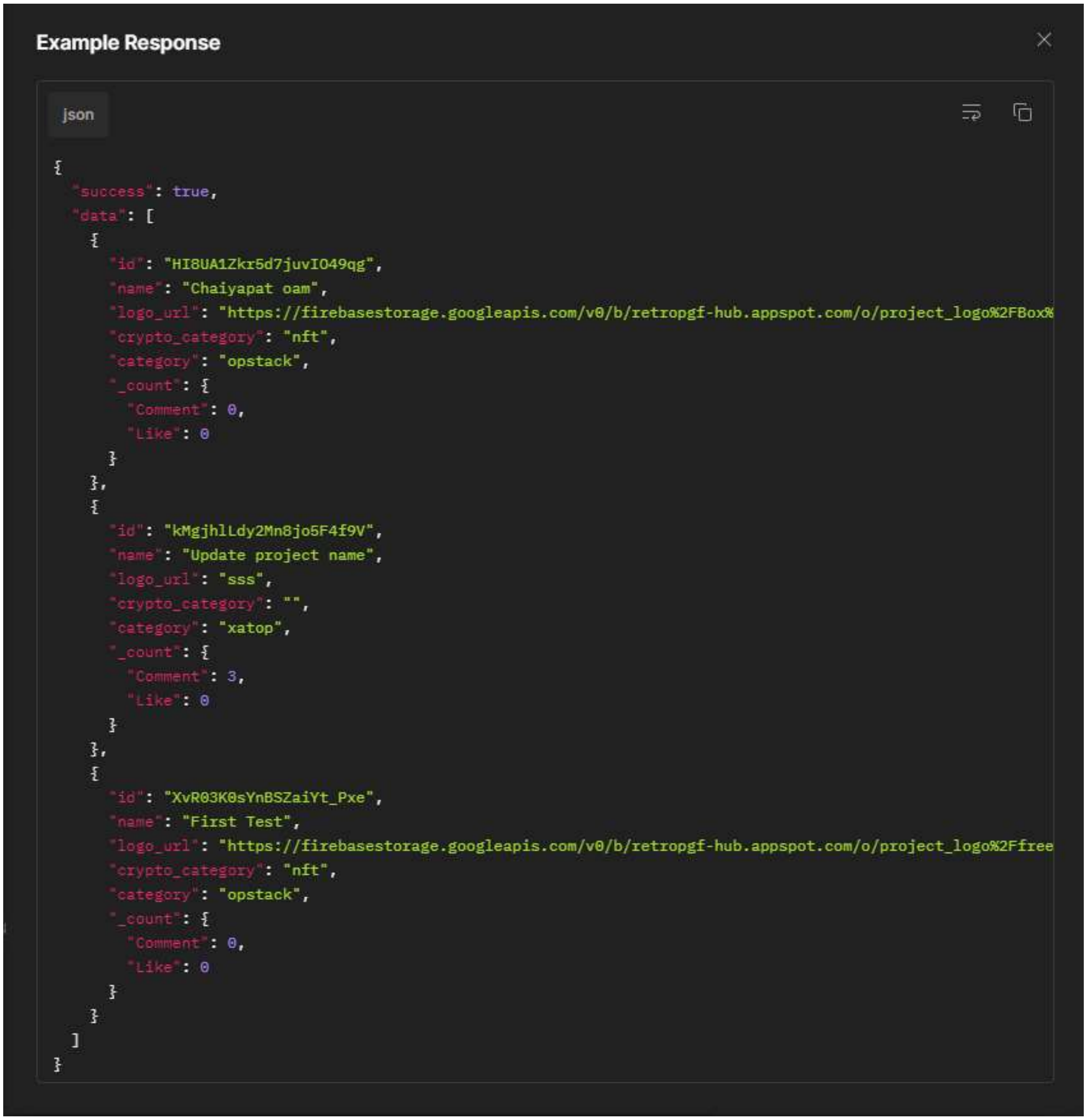


Semi Structured Data

API (Application Programing Interface)

Example: REST API (JSON)

This is my API I wrote it myself link here



```json
{
  "success": true,
  "data": [
    {
      "id": "HI8UA1Zkr5d7juvIO49qg",
      "name": "Chaiyapat oam",
      "logo_url": "https://firebasestorage.googleapis.com/v0/b/retropgf-hub.appspot.com/o/project_logo%2FBox%
      "crypto_category": "nft",
      "category": "opstack",
      "_count": {
        "Comment": 0,
        "Like": 0
      }
    },
    {
      "id": "kMgjhlLdy2Mn8jo5F4f9V",
      "name": "Update project name",
      "logo_url": "sss",
      "crypto_category": "",
      "category": "xatop",
      "_count": {
        "Comment": 3,
        "Like": 0
      }
    },
    {
      "id": "XvR03K0sYnBSZaiYt_Pxe",
      "name": "First Test",
      "logo_url": "https://firebasestorage.googleapis.com/v0/b/retropgf-hub.appspot.com/o/project_logo%2Ffree
      "crypto_category": "nft",
      "category": "opstack",
      "_count": {
        "Comment": 0,
        "Like": 0
      }
    }
  ]
}
```

Unstructured Data

Example: Image file

```
!pip install scikit-image
```

```
Requirement already satisfied: scikit-image in /usr/local/lib/python3.
Requirement already satisfied: numpy>=1.17.0 in /usr/local/lib/python3
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.
Requirement already satisfied: networkx>=2.2 in /usr/local/lib/python3
Requirement already satisfied: pillow!=7.1.0,!=7.1.1,!=8.3.0,>=6.1.0 i
Requirement already satisfied: imageio>=2.4.1 in /usr/local/lib/python
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/p
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/pyt
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/pytho
```

```
!wget https://cdn.discordapp.com/attachments/1186555976673919048/1203725
```

```
--2024-02-04 15:35:23--  https://cdn.discordapp.com/attachments/118655
Resolving cdn.discordapp.com (cdn.discordapp.com)... 162.159.134.233,
Connecting to cdn.discordapp.com (cdn.discordapp.com)|162.159.134.233|
HTTP request sent, awaiting response... 200 OK
Length: 182287 (178K) [image/png]
Saving to: ‘snorlax.png’

snorlax.png         100%[==================>] 178.01K  --.-KB/s    in

2024-02-04 15:35:23 (75.0 MB/s) - ‘snorlax.png’ saved [182287/182287]
```

```python
from skimage.io import imread
img = imread("snorlax.png")
```

```python
from skimage.io import imshow
import matplotlib.pyplot as plt

imshow(img)
plt.axis('off')
plt.show()
```

```
<Figure size 640x480 with 1 Axes>
```

Download

Yeah! I got Kabigon

```
img.shape
```

```
(924, 864, 4)
```

```
img
```

```
array([[[0, 0, 0, 0],
        [0, 0, 0, 0],
        [0, 0, 0, 0],
        ...,
        [0, 0, 0, 0],
        [0, 0, 0, 0],
        [0, 0, 0, 0]],

       [[0, 0, 0, 0],
        [0, 0, 0, 0],
        [0, 0, 0, 0],
        ...,
        [0, 0, 0, 0],
        [0, 0, 0, 0],
        [0, 0, 0, 0]],

       [[0, 0, 0, 0],
        [0, 0, 0, 0],
        [0, 0, 0, 0],
        ...,
```

Why It's all zero ??

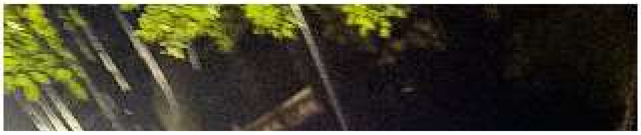
```
# try to load another image from my phone
!wget https://cdn.discordapp.com/attachments/1186555976673919048/1203728
```

```
--2024-02-04 15:49:39--  https://cdn.discordapp.com/attachments/118655
Resolving cdn.discordapp.com (cdn.discordapp.com)... 162.159.134.233,
Connecting to cdn.discordapp.com (cdn.discordapp.com)|162.159.134.233|
HTTP request sent, awaiting response... 200 OK
Length: 3966024 (3.8M) [image/jpeg]
Saving to: ‘image0.jpg.1’

image0.jpg.1        100%[===================>]   3.78M  --.-KB/s    in

2024-02-04 15:49:39 (145 MB/s) - ‘image0.jpg.1’ saved [3966024/3966024
```

```
# load my lovely friend's image
sorn = imread("image0.jpg")
imshow(sorn)
plt.axis('off')
plt.show()
```

```
<Figure size 640x480 with 1 Axes>
```

Download



```
sorn
```

```
array([[[142, 151,  34],
        [132, 141,  24],
        [127, 135,  23],
        ...,
        [ 21,  17,   5],
        [ 23,  19,  10],
        [  6,   2,   0]],

       [[146, 155,  38],
        [139, 148,  31],
        [136, 144,  32],
        ...,
        [ 29,  25,  14],
        [ 35,  31,  22],
        [ 20,  16,   7]],

       [[145, 154,  37],
        [143, 152,  35],
        [143, 151,  39],
        ...,
```

Audio file

```
!pip install librosa
```

```
Requirement already satisfied: librosa in /usr/local/lib/python3.10/d
Requirement already satisfied: audioread>=2.1.9 in /usr/local/lib/pyt
Requirement already satisfied: numpy!=1.22.0,!=1.22.1,!=1.22.2,>=1.20
Requirement already satisfied: scipy>=1.2.0 in /usr/local/lib/python3
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib
Requirement already satisfied: joblib>=0.14 in /usr/local/lib/python3
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/pyt
Requirement already satisfied: numba>=0.51.0 in /usr/local/lib/python
Requirement already satisfied: soundfile>=0.12.1 in /usr/local/lib/py
Requirement already satisfied: pooch>=1.0 in /usr/local/lib/python3.1
Requirement already satisfied: soxr>=0.3.2 in /usr/local/lib/python3.
Requirement already satisfied: typing-extensions>=4.1.1 in /usr/local
Requirement already satisfied: lazy-loader>=0.1 in /usr/local/lib/pyt
Requirement already satisfied: msgpack>=1.0 in /usr/local/lib/python3
Requirement already satisfied: llvmlite<0.42,>=0.41.0dev0 in /usr/loc
Requirement already satisfied: platformdirs>=2.5.0 in /usr/local/lib/
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/pyth
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/pyt
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib
```

```
!wget https://cdn.discordapp.com/attachments/1186555976673919048/1203731
```

```
--2024-02-04 15:59:20--  https://cdn.discordapp.com/attachments/118655
Resolving cdn.discordapp.com (cdn.discordapp.com)... 162.159.133.233,
Connecting to cdn.discordapp.com (cdn.discordapp.com)|162.159.133.233|
HTTP request sent, awaiting response... 200 OK
Length: 13824 (14K) [audio/mpeg]
Saving to: ‘hungry.mp3’

hungry.mp3          100%[===================>]  13.50K  --.-KB/s    in

2024-02-04 15:59:21 (58.3 MB/s) - ‘hungry.mp3’ saved [13824/13824]
```

```
import librosa
s, rate = librosa.load("hungry.mp3", sr=None) # หิวจัง
```

```
s
```

```
array([ 0.0000000e+00,  0.0000000e+00,  0.0000000e+00, ...,
       -2.6863695e-06, -2.0924690e-06,  6.5438962e-07], dtype=float32)
```

```
rate
```

```
48000
```

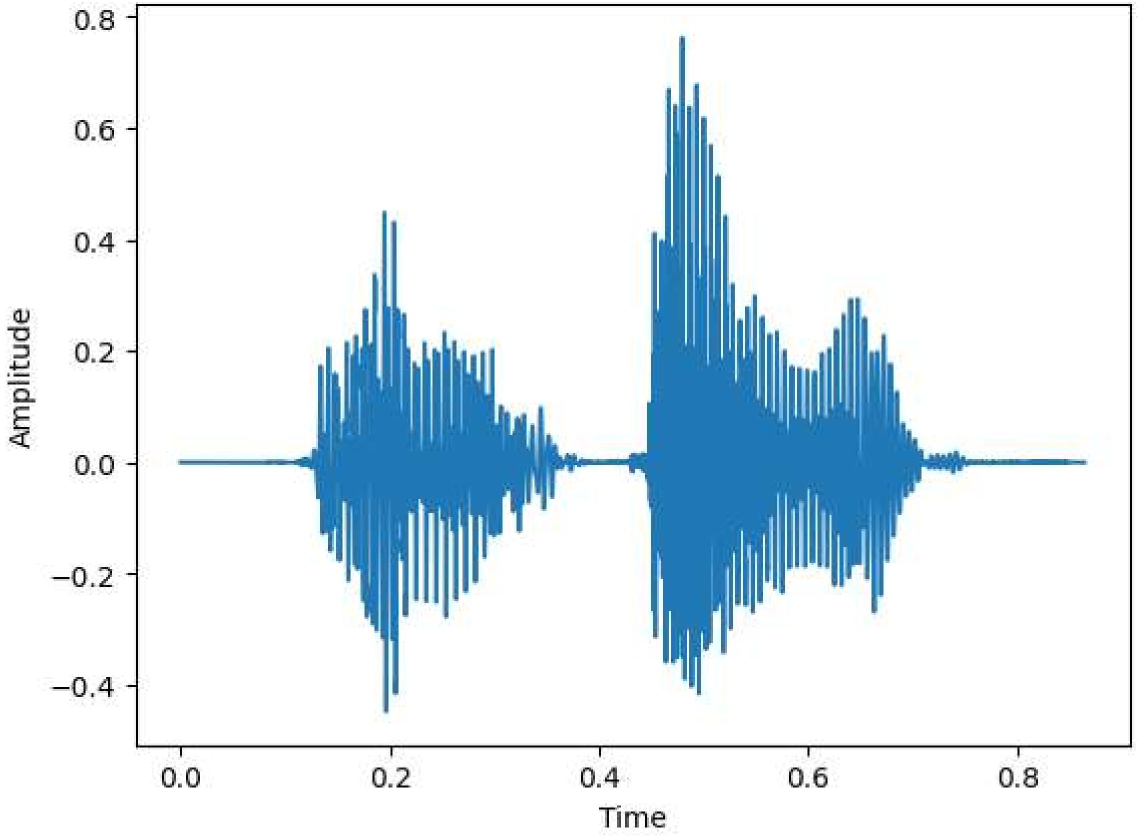
```
# play it !!
from IPython.display import Audio
Audio(data=s, rate=rate) # หิวจัง
```

```
<IPython.lib.display.Audio object>
```

```python
# try to plot
import numpy as np
time = np.linspace(0,len(s)/rate,len(s))
plt.plot(time,s)
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.show()
```

```
<Figure size 640x480 with 1 Axes>
```

Download


Amplitude
Time