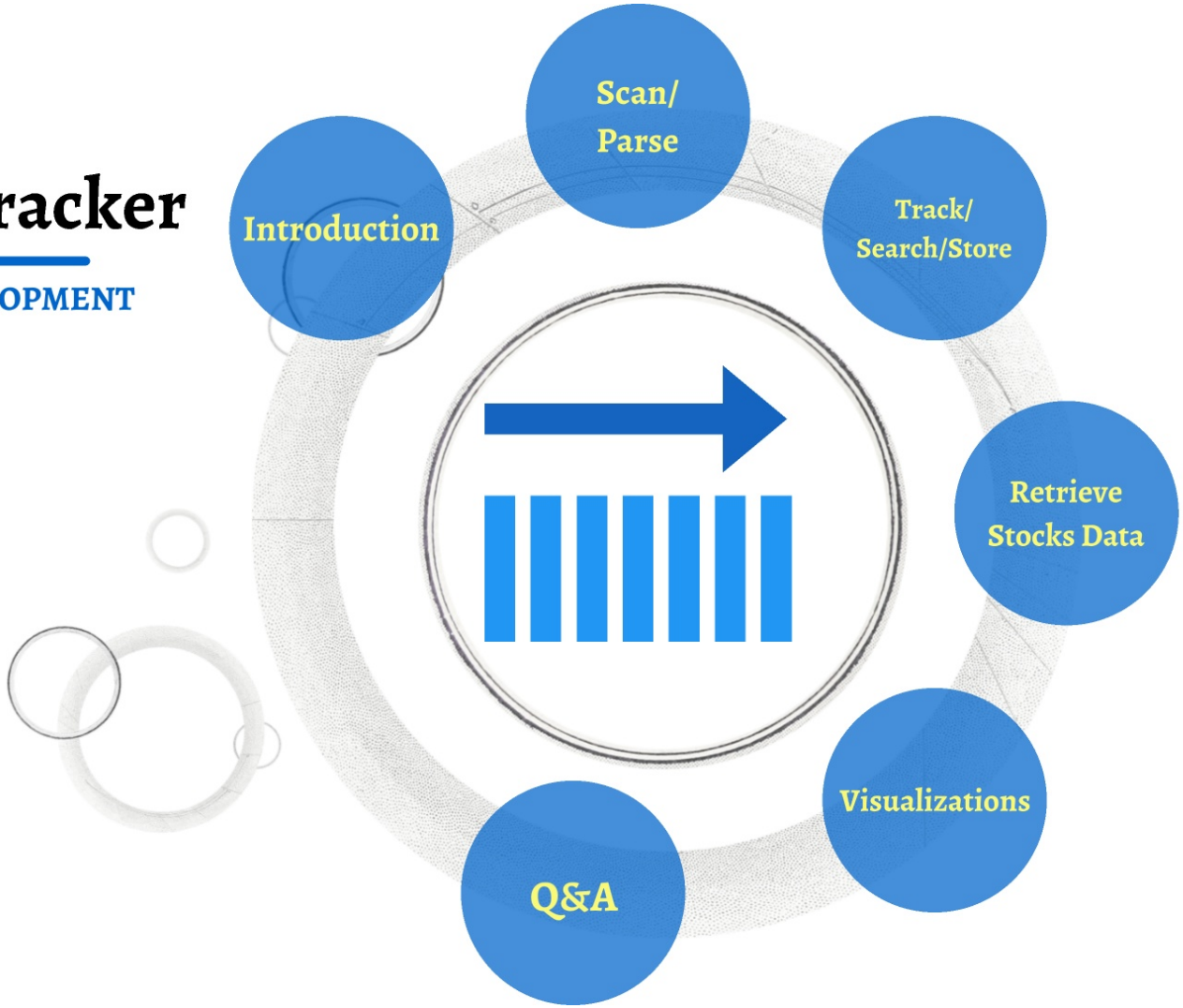
Simple Stocks Tracker

INTRODUCTION TO AI DEVELOPMENT

AIDI-1100-02

Group 7- Fall/2021



Introduction

Goal

Track performance of stocks mentioned in the news recently.

Our Team

Solution Breakdown

Additional Features

Our Team

Project POC:
Raj Dholakia

Project Architect :
Nikolai Melnikov
Praharsh Bhatt

Project Specialist:
Mervat Mustafa
Komal Sodera

Project BA:
Priyankkumar Prakashbhai Patel

Introduction

Goal

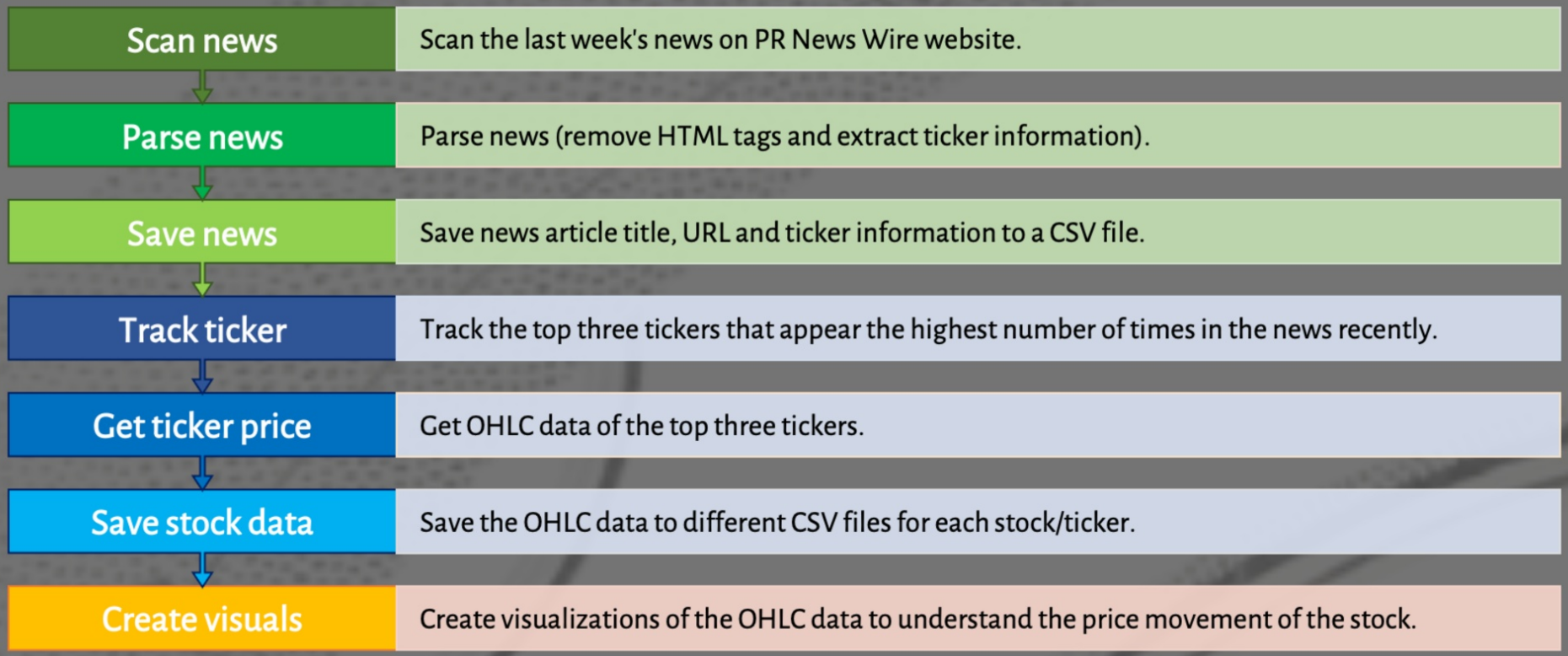
Track performance of stocks mentioned in the news recently.

Our Team

Solution Breakdown

Additional Features

Solution Breakdown



Introduction

Goal

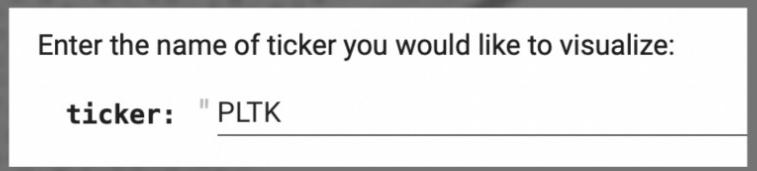
Track performance of stocks mentioned in the news recently.

Our Team

Solution Breakdown

Additional Features

Additional Features



Note:

- All code is written in Python.
- We collaborated on GitHub.
- Google Colab notebook environment used.

Introduction

Goal

Track performance of stocks mentioned in the news recently.

Our Team

Solution Breakdown

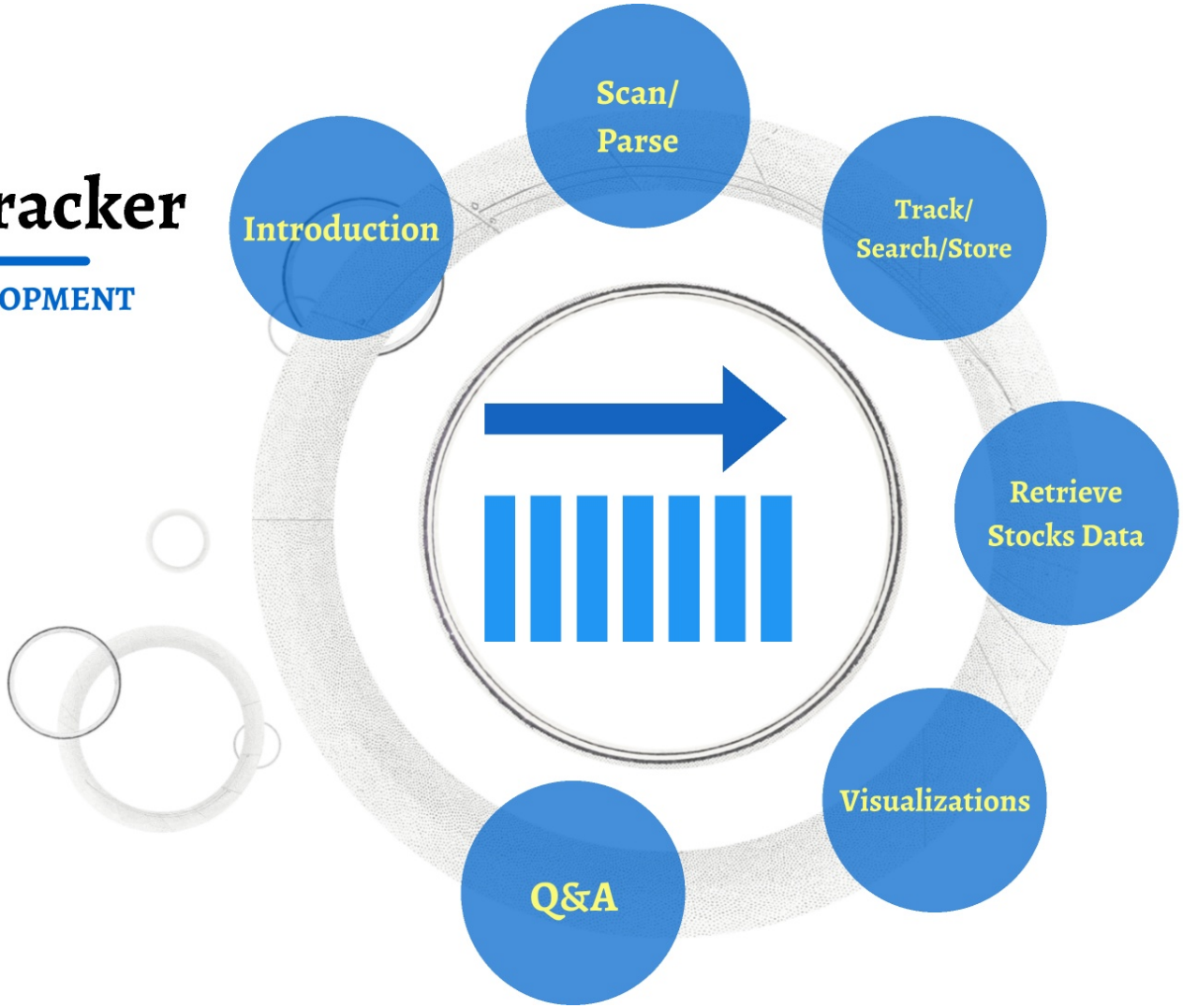
Additional Features

Simple Stocks Tracker

INTRODUCTION TO AI DEVELOPMENT

AIDI-1100-02

Group 7- Fall/2021



Result

PR Newswire
a CISION company

Function **page_parse()** scans PRNewsWire.com and returns URLs of scanned articles.

- Function **url_parse()** parses each article and detects any tickers mentioned in the article.
- Function **run_scanner()** uses the above functions and returns a pandas dataframe containing:
  - Article Date
  - Article Title
  - Ticker
  - Article URL.

```
# Function to parse a particular page for all the news to later parse them for tickers.
# Takes 2 parameters: a number of pages and initial dataset of already saved news.
def page_parse(x, page_session, data=[]):
    page_url = f'https://www.prnewswire.com/news-releases/english-releases/?page={x}&pagesize=100'
    page_request = page_session.get(page_url)
    content = page_request.html.find('div.row.arabiclistingcards')
    for item in tqdm(content, desc='Parsing page...\t', leave=False):
        date = item.find('h3', first=True).text.split('ET')[-2]
        title = item.find('h3', first=True).text.split('ET')[-1]
        article_url = 'https://www.prnewswire.com' + item.find('a.newsreleaseconsolidatelink',
first=True).attrs['href']
        ticker = url_parse(article_url, page_session)
        try:
            dic = {
              'Date': pd.to_datetime(date),
              'Title': title,
              'Ticker': ticker,
              'Article URL': article_url
            }
            data.append(dic)
        except Exception:
            pass

    return data
```

Result

SAN FRANCISCO, Nov. 23, 2021 /PRNewswire/ -- Autodesk, Inc. (NASDAQ: ADSK) today reported financial results for the third quarter of fiscal 2022.

```
<a class="ticket-symbol" data-toggle="modal" href="https://www.prnewswire.com/news-releases/autodesk-inc-announces-fiscal-2022-third-quarter-results-301431093.html#financial-modal">
ADSK</a> == $0
") today reported financial results for the third quarter of fiscal 2022. "
```

Algorithm's performance

```
Loading Pages...        :   0%|          | 0/10 [00:00<?, ?it/s]
Parsing page... :   0%|          | 0/100 [00:00<?, ?it/s]
Parsing page... :   0%|          | 0/100 [00:00<?, ?it/s]
Parsing page... :   0%|          | 0/100 [00:00<?, ?it/s]
Parsing page... :   0%|          | 0/100 [00:00<?, ?it/s]
Parsing page... :   0%|          | 0/100 [00:00<?, ?it/s]
Parsing page... :   0%|          | 0/100 [00:00<?, ?it/s]
Parsing page... :   0%|          | 0/100 [00:00<?, ?it/s]
Parsing page... :   0%|          | 0/100 [00:00<?, ?it/s]
Parsing page... :   0%|          | 0/100 [00:00<?, ?it/s]
Parsing page... :   0%|          | 0/100 [00:00<?, ?it/s]
Time Taken:  170.1s
```

Results dataframe

Date	Title	Ticker	Article URL
2021-11-09 19:48:00	Intercorp Financial Services announces virtua...	IFS	https://www.prnewswire.com/news-releases/inter...
2021-11-09 19:46:00	ICC Holdings, Inc. Reports 2021 Third Quarter...	ICCH	https://www.prnewswire.com/news-releases/icc-h...
2021-11-09 19:00:00	Smith+Nephew establishes its first Medical Ed...	SNN	https://www.prnewswire.com/news-releases/smith...
2021-11-09 18:56:00	Cambridge Bancorp Announces Expansion of Weal...	CATC	https://www.prnewswire.com/news-releases/cambr...
2021-11-09 18:35:00	EQT Ventures and EQT Growth to exit its holdi...	DASH	https://www.prnewswire.com/news-releases/eqt-v...

PR Newswire
a CISION company

Result

Function **page_parse()** scans PRNewsWire.com and returns URLs of scanned articles.

- Function **url_parse()** parses each article and detects any tickers mentioned in the article.
- Function **run_scanner()** uses the above functions and returns a pandas dataframe containing:
  - Article Date
  - Article Title
  - Ticker
  - Article URL.

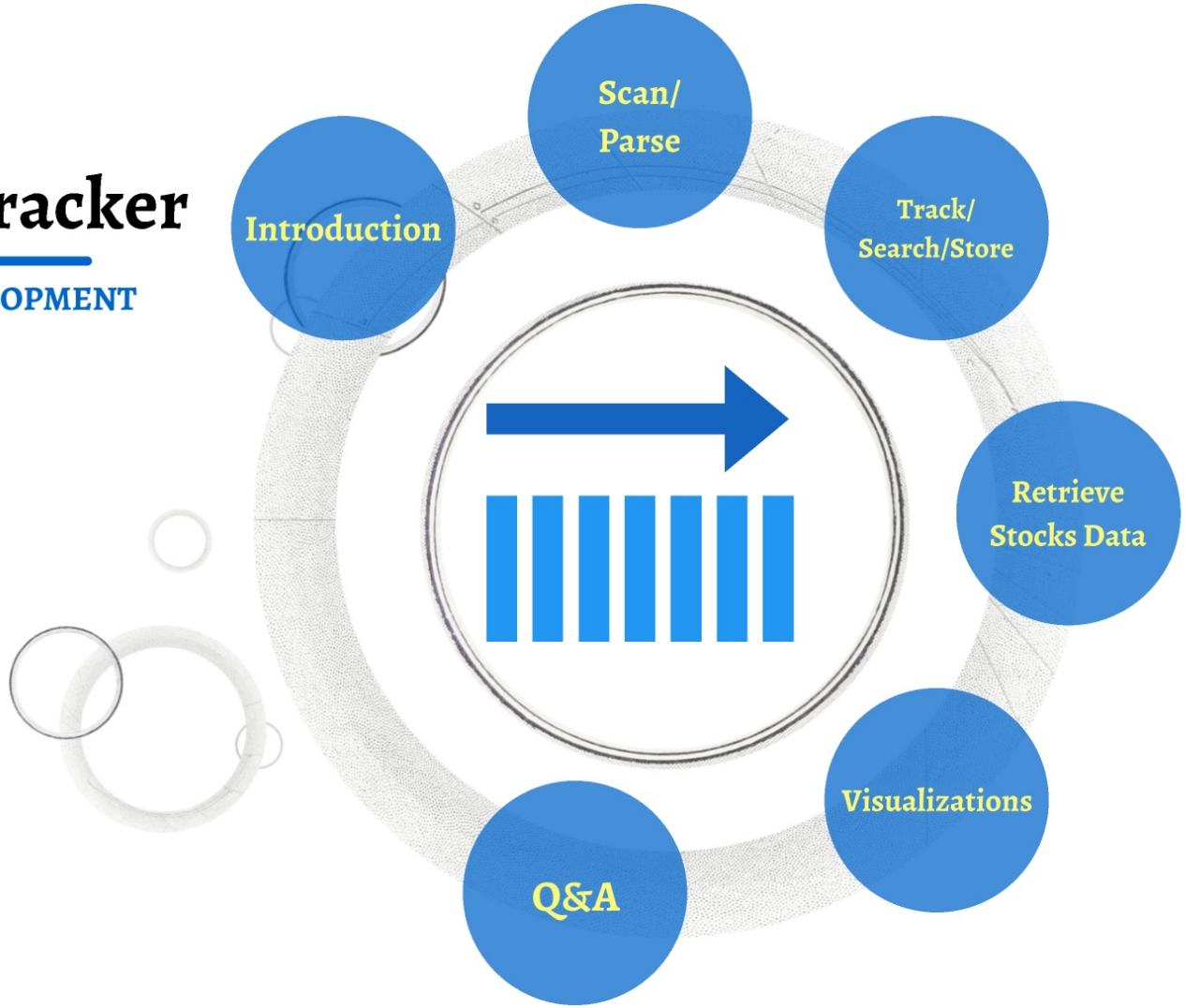
```
# Function to parse a particular page for all the news to later parse them for tickers.
# Takes 2 parameters: a number of pages and initial dataset of already saved news.
def page_parse(x, page_session, data=[]):
    page_url = f'https://www.prnewswire.com/news-releases/english-releases/?page={x}&pagesize=100'
    page_request = page_session.get(page_url)
    content = page_request.html.find('div.row.arabiclistingcards')
    for item in tqdm(content, desc='Parsing page...\t', leave=False):
        date = item.find('h3', first=True).text.split('ET')[-2]
        title = item.find('h3', first=True).text.split('ET')[-1]
        article_url = 'https://www.prnewswire.com' + item.find('a.newsreleaseconsolidatelink',
first=True).attrs['href']
        ticker = url_parse(article_url, page_session)
        try:
            dic = {
              'Date': pd.to_datetime(date),
              'Title': title,
              'Ticker': ticker,
              'Article URL': article_url
            }
            data.append(dic)
        except Exception:
            pass

    return data
```

Simple Stocks Tracker

INTRODUCTION TO AI DEVELOPMENT

AIDI-1100-02

Group 7- Fall/2021



Track/Search/Store

- Get stock data for a ticker
- Iterate through stocks and save stock data in a CSV file.

1

2

Get stock data for a ticker

```
# create filename using the first and last dates.
sorted_df = df.sort_values('Date')

start_date = sorted_df.index[0].date()
end_date = sorted_df.sort_values('Date').index[-1].date()

# concate dates to create name for the CSV file.
filename = start_date.strftime('%d%b%y') + '-to-' + end_date.strftime('%d%b%y') + '.csv'
print(filename)
```

Track/Search/Store

- Get stock data for a ticker
- Iterate through stocks and save stock data in a CSV file.

1

2

Iterate through stocks and save stock data in a CSV file.

```
data_dir = pathlib.Path('/content/datasets/')

if not data_dir.exists():
    os.mkdir(data_dir)

# Save dataframe as a csv for further analysis
df.to_csv(data_dir / filename)
print(f'CSV file saved to: {data_dir / filename}')


#CSV file saved to: /content/datasets/26Nov21-to-30Nov21.csv
```

Track/Search/Store

- Get stock data for a ticker
- Iterate through stocks and save stock data in a CSV file.

1

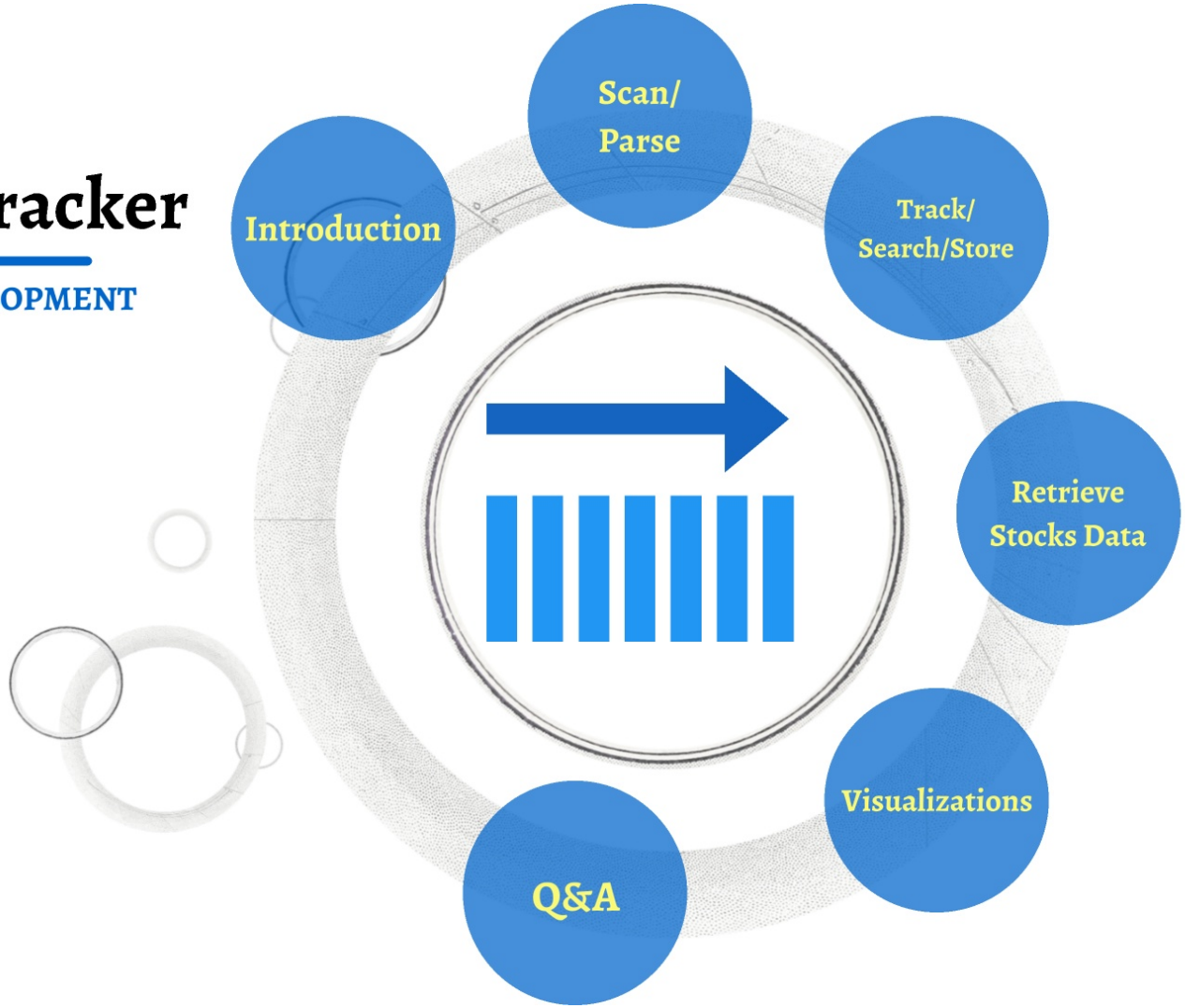
2

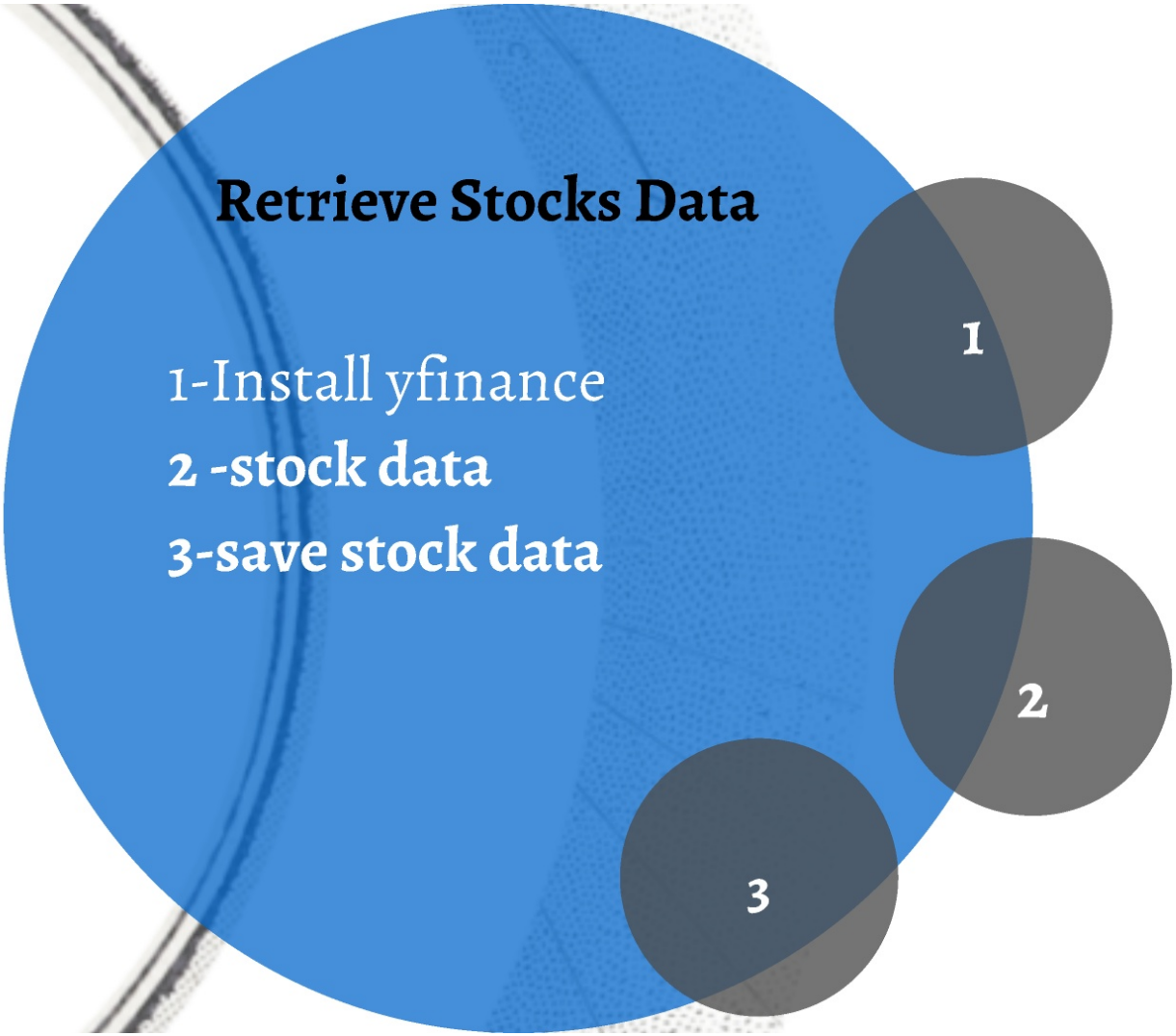
Simple Stocks Tracker

INTRODUCTION TO AI DEVELOPMENT

AIDI-1100-02

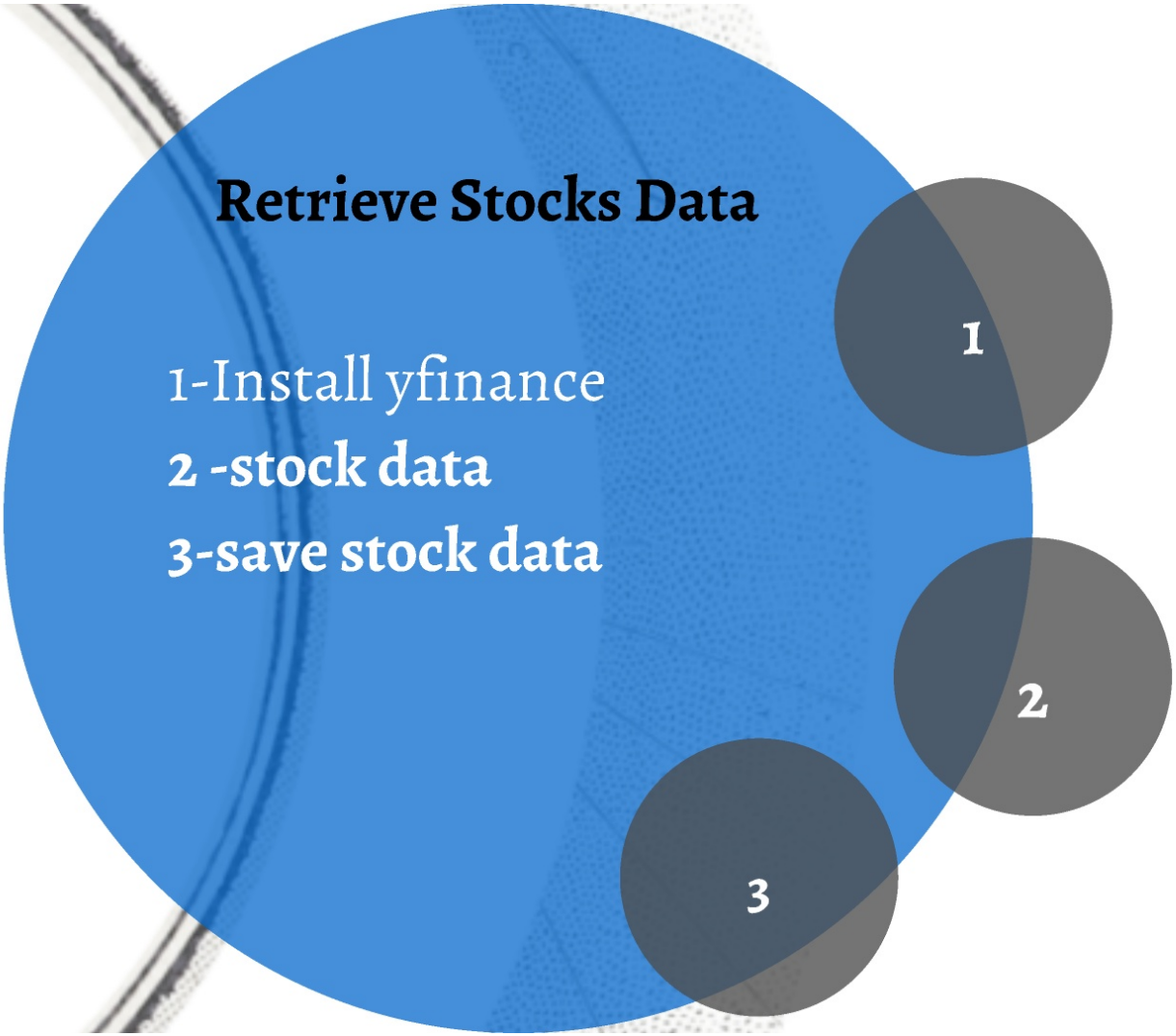
Group 7- Fall/2021




Retrieve Stocks Data
1-Install yfinance
2 -stock data
3-save stock data
1
2
3

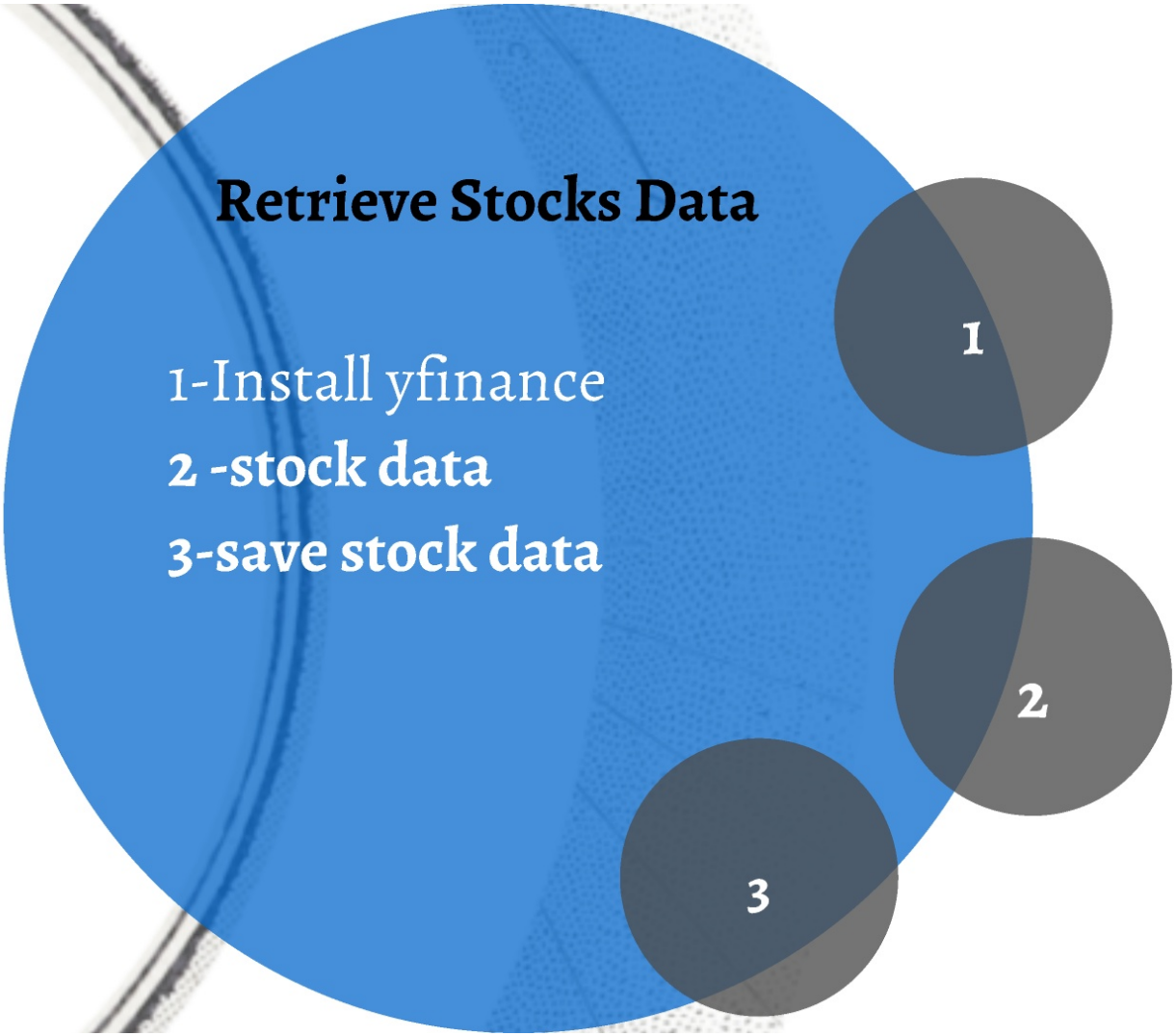
Install yfinance

```
# install and/or load yfinance module
try:
    success_msg = '\n\nYahoo! Finance module loaded.'
    import yfinance as yf
    print(success_msg)
except ModuleNotFoundError as e:
    print('Installing Yahoo! Finance python module...')
    !pip install yfinance
    import yfinance as yf
    print(success_msg)
```


Retrieve Stocks Data
1-Install yfinance
2 -stock data
3-save stock data
1
2
3

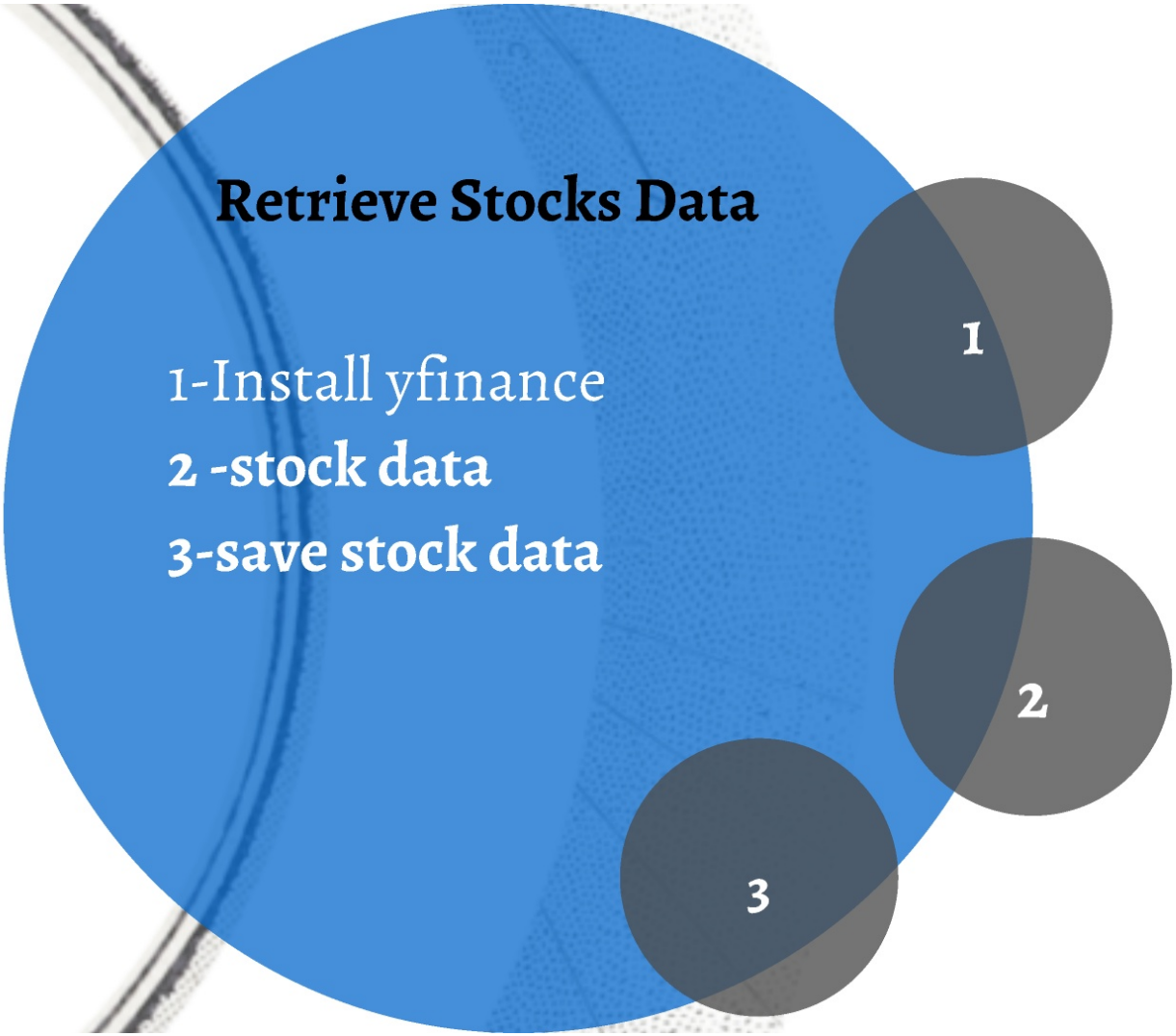
2. Get stock data for a ticker

```python
def get_stock_quotes(ticker: str, period: str = "30d"):

    if type(ticker) is not str:
        return

    # Get the stock data of this ticker
    stock_data = yf.Ticker(ticker)

    # get historical market data of the past 30 days
    return stock_data.history(period=period)
```


Retrieve Stocks Data
1-Install yfinance
2 -stock data
3-save stock data
1
2
3

3. Iterate through stocks and save stock data in CSVs

```
# Create an empty folder
if not output_stock_dump_path.exists():
    os.mkdir(output_stock_dump_path)

stock_names = []

for ticker in tqdm(df.Ticker, desc='Getting Stocks data... \t'):
    # Get the stock performance data for the past 30 days of this stock
    if not ticker in top_tickers2:
        continue

    stock_data = get_stock_quotes(ticker, period)

    if stock_data is None:
        continue

    stock_df = pd.DataFrame(
        {
            'Date': stock_data.Volume.keys(),
            'Open': stock_data.Open.values,
            'Close': stock_data.Close.values,
            'Low': stock_data.Low.values,
            'High': stock_data.High.values,
            'Volume': stock_data.Volume.values,
        }
    )
    stock_names += ticker
    stock_df.to_csv(output_stock_dump_path / (ticker+'.csv'), index=False)

print(f'The data of the following stocks was saved to {output_stock_dump_path}:')
print(f'{top_tickers2}')
```

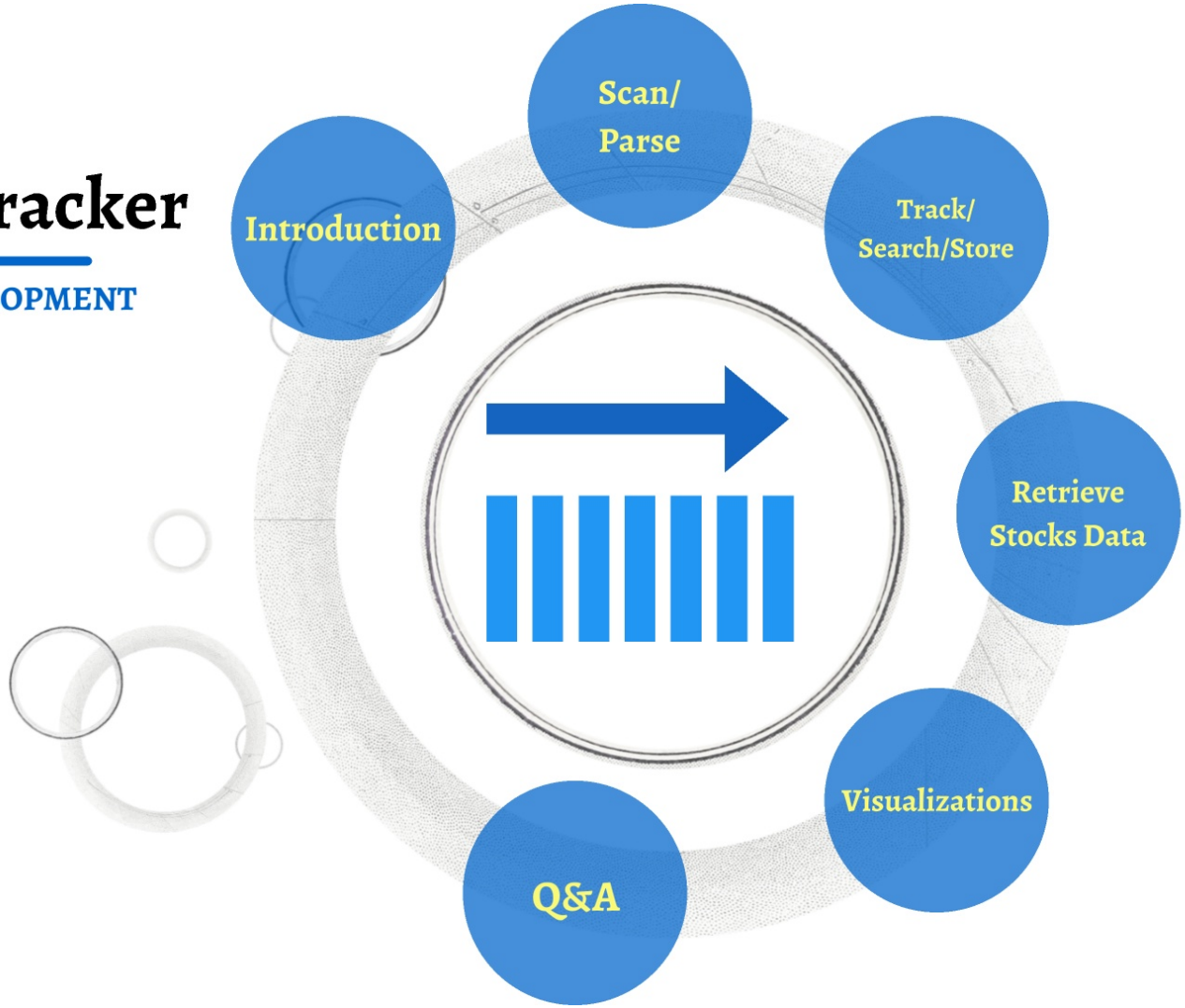

Retrieve Stocks Data
1-Install yfinance
2 -stock data
3-save stock data
1
2
3

Simple Stocks Tracker

INTRODUCTION TO AI DEVELOPMENT

AIDI-1100-02

Group 7- Fall/2021



Visualizations

Date	Date	Open	Close	Low	High	Volume
2021-10-15	2021-10-15	76.809998	77.339996	75.870003	78.735001	20404200
2021-10-18	2021-10-18	78.599998	75.800003	72.610001	79.300003	33053900
2021-10-19	2021-10-19	75.120003	76.430000	73.900002	78.180000	18720700
2021-10-20	2021-10-20	76.570000	75.650002	74.669998	77.519997	15739300
2021-10-21	2021-10-21	74.809998	75.110001	73.889999	75.949997	44450900
2021-10-22	2021-10-22	58.750000	55.139999	55.029999	60.779999	153827500
2021-10-25	2021-10-25	55.959999	54.500000	53.770000	56.150002	62506800
2021-10-26	2021-10-26	55.240002	55.389999	54.880001	57.180000	52905700
2021-10-27	2021-10-27	55.049999	52.020000	51.660000	55.230000	52581300
2021-10-28	2021-10-28	52.680000	54.389999	51.990002	54.959999	34748800
2021-10-29	2021-10-29	53.970001	52.580002	52.209999	53.990002	32465500
2021-11-01	2021-11-01	52.990002	53.980000	52.250000	54.110001	22666700
2021-11-02	2021-11-02	53.570000	52.200001	51.865002	53.660000	24120400
2021-11-03	2021-11-03	51.950001	52.939999	51.470001	52.970001	16319300
2021-11-04	2021-11-04	53.080002	52.259998	51.759998	53.380001	16371700
2021-11-05	2021-11-05	52.150002	53.169998	52.070000	53.570000	17447500
2021-11-08	2021-11-08	53.360001	54.900002	52.849998	55.130001	20154600
2021-11-09	2021-11-09	55.294998	54.820000	54.200001	55.650002	12489000
2021-11-10	2021-11-10	53.910000	52.880001	52.020000	54.169998	14812400
2021-11-11	2021-11-11	53.459999	53.290001	53.200001	54.590000	13322000

Load Data & Select Ticker

Basic Analysis

Visualizations

Load Data & Select Ticker

```
stocks_df = {ticker: pd.read_csv(output_stock_dump_path / (ticker+'.csv'))
             for ticker in top_tickers2}
print("\n".join(list(stocks_df.keys())))
```

```
EBET
CAJ
STWD
```

```
#@markdown Enter the name of ticker you would like to visualize:
ticker = "CAJ" #@param {type:"string"}
#@markdown > **Note**: Choose from one of the tickers listed above this cell.

ticker = ticker.strip().upper()

# check if input is valid
if stocks_df.get(ticker) is None:
    raise ValueError('Please enter a valid Ticker value.')
```

Enter the name of ticker you would like to visualize:

ticker: "CAJ"

> **Note**: Choose from one of the tickers listed above this cell.

Visualizations

Date	Date	Open	Close	Low	High	Volume
2021-10-15	2021-10-15	76.809998	77.339996	75.870003	78.735001	20404200
2021-10-18	2021-10-18	78.599998	75.800003	72.610001	79.300003	33053900
2021-10-19	2021-10-19	75.120003	76.430000	73.900002	78.180000	18720700
2021-10-20	2021-10-20	76.570000	75.650002	74.669998	77.519997	15739300
2021-10-21	2021-10-21	74.809998	75.110001	73.889999	75.949997	44450900
2021-10-22	2021-10-22	58.750000	55.139999	55.029999	60.779999	153827500
2021-10-25	2021-10-25	55.959999	54.500000	53.770000	56.150002	62506800
2021-10-26	2021-10-26	55.240002	55.389999	54.880001	57.180000	52905700
2021-10-27	2021-10-27	55.049999	52.020000	51.660000	55.230000	52581300
2021-10-28	2021-10-28	52.680000	54.389999	51.990002	54.959999	34748800
2021-10-29	2021-10-29	53.970001	52.580002	52.209999	53.990002	32465500
2021-11-01	2021-11-01	52.990002	53.980000	52.250000	54.110001	22666700
2021-11-02	2021-11-02	53.570000	52.200001	51.865002	53.660000	24120400
2021-11-03	2021-11-03	51.950001	52.939999	51.470001	52.970001	16319300
2021-11-04	2021-11-04	53.080002	52.259998	51.759998	53.380001	16371700
2021-11-05	2021-11-05	52.150002	53.169998	52.070000	53.570000	17447500
2021-11-08	2021-11-08	53.360001	54.900002	52.849998	55.130001	20154600
2021-11-09	2021-11-09	55.294998	54.820000	54.200001	55.650002	12489000
2021-11-10	2021-11-10	53.910000	52.880001	52.020000	54.169998	14812400
2021-11-11	2021-11-11	53.459999	53.290001	53.200001	54.590000	13322000

Load Data & Select Ticker

Basic Analysis

Visualizations

Basic Analysis

```
# view top 5 rows of the ticker
print(f'Top 5 rows of {ticker}.\n')
df_to_visual=stocks_df.get(ticker)
df_to_visual.head(5)
```

Top 5 rows of SNAP.

	Date	Open	Close	Low	High	Volume
0	2021-10-15	76.809998	77.339996	75.870003	78.735001	20404200
1	2021-10-18	78.599998	75.800003	72.610001	79.300003	33053900
2	2021-10-19	75.120003	76.430000	73.900002	78.180000	18720700
3	2021-10-20	76.570000	75.650002	74.669998	77.519997	15739300
4	2021-10-21	74.809998	75.110001	73.889999	75.949997	44450900

```
df_to_visual.describe()
```

	Open	Close	Low	High	Volume
count	30.000000	30.000000	30.000000	30.000000	3.000000e+01
mean	75.869467	75.217333	73.629500	77.578334	1.101363e+07
std	16.336616	16.759297	16.028230	16.858242	1.443744e+07
min	53.099998	53.250000	52.570000	55.680000	1.406700e+06
25%	63.460001	62.985000	62.335000	63.809251	4.643800e+06
50%	68.405003	66.940002	65.520000	69.524998	6.680300e+06
75%	91.837502	92.164999	90.162498	95.193750	9.144125e+06
max	102.834999	103.629997	97.650002	104.050003	7.843390e+07

```
# Is there any missing vaue ?
df_to_visual.isnull().sum()
```

```
Date      0
Open      0
Close     0
Low       0
High      0
Volume    0
dtype: int64
```

```
# View data types
df_to_visual.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    30 non-null     object
 1   Open    30 non-null     float64
 2   Close   30 non-null     float64
 3   Low     30 non-null     float64
 4   High    30 non-null     float64
 5   Volume  30 non-null     int64
dtypes: float64(4), int64(1), object(1)
```

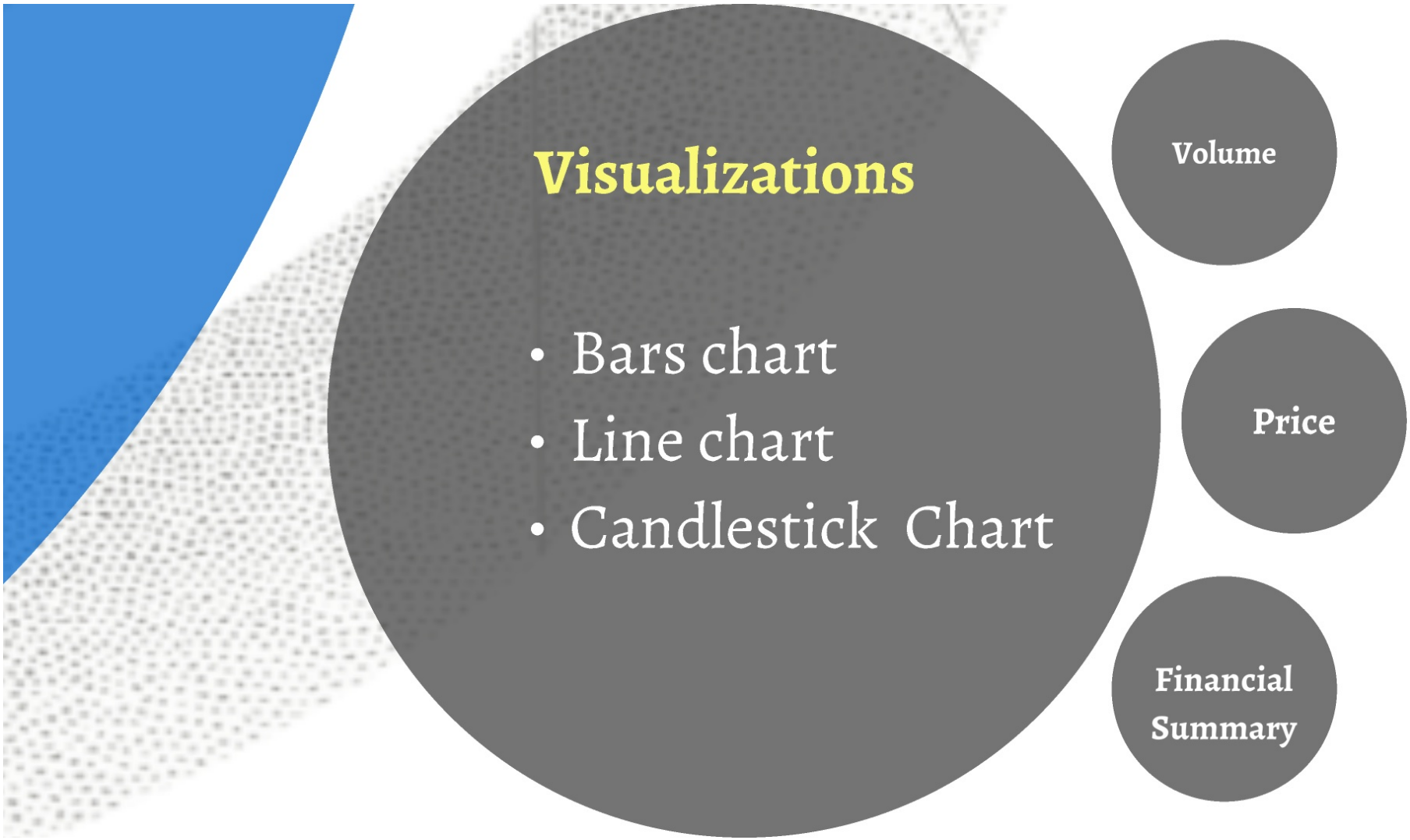
Visualizations

Date	Date	Open	Close	Low	High	Volume
2021-10-15	2021-10-15	76.809998	77.339996	75.870003	78.735001	20404200
2021-10-18	2021-10-18	78.599998	75.800003	72.610001	79.300003	33053900
2021-10-19	2021-10-19	75.120003	76.430000	73.900002	78.180000	18720700
2021-10-20	2021-10-20	76.570000	75.650002	74.669998	77.519997	15739300
2021-10-21	2021-10-21	74.809998	75.110001	73.889999	75.949997	44450900
2021-10-22	2021-10-22	58.750000	55.139999	55.029999	60.779999	153827500
2021-10-25	2021-10-25	55.959999	54.500000	53.770000	56.150002	62506800
2021-10-26	2021-10-26	55.240002	55.389999	54.880001	57.180000	52905700
2021-10-27	2021-10-27	55.049999	52.020000	51.660000	55.230000	52581300
2021-10-28	2021-10-28	52.680000	54.389999	51.990002	54.959999	34748800
2021-10-29	2021-10-29	53.970001	52.580002	52.209999	53.990002	32465500
2021-11-01	2021-11-01	52.990002	53.980000	52.250000	54.110001	22666700
2021-11-02	2021-11-02	53.570000	52.200001	51.865002	53.660000	24120400
2021-11-03	2021-11-03	51.950001	52.939999	51.470001	52.970001	16319300
2021-11-04	2021-11-04	53.080002	52.259998	51.759998	53.380001	16371700
2021-11-05	2021-11-05	52.150002	53.169998	52.070000	53.570000	17447500
2021-11-08	2021-11-08	53.360001	54.900002	52.849998	55.130001	20154600
2021-11-09	2021-11-09	55.294998	54.820000	54.200001	55.650002	12489000
2021-11-10	2021-11-10	53.910000	52.880001	52.020000	54.169998	14812400
2021-11-11	2021-11-11	53.459999	53.290001	53.200001	54.590000	13322000

Load Data & Select Ticker

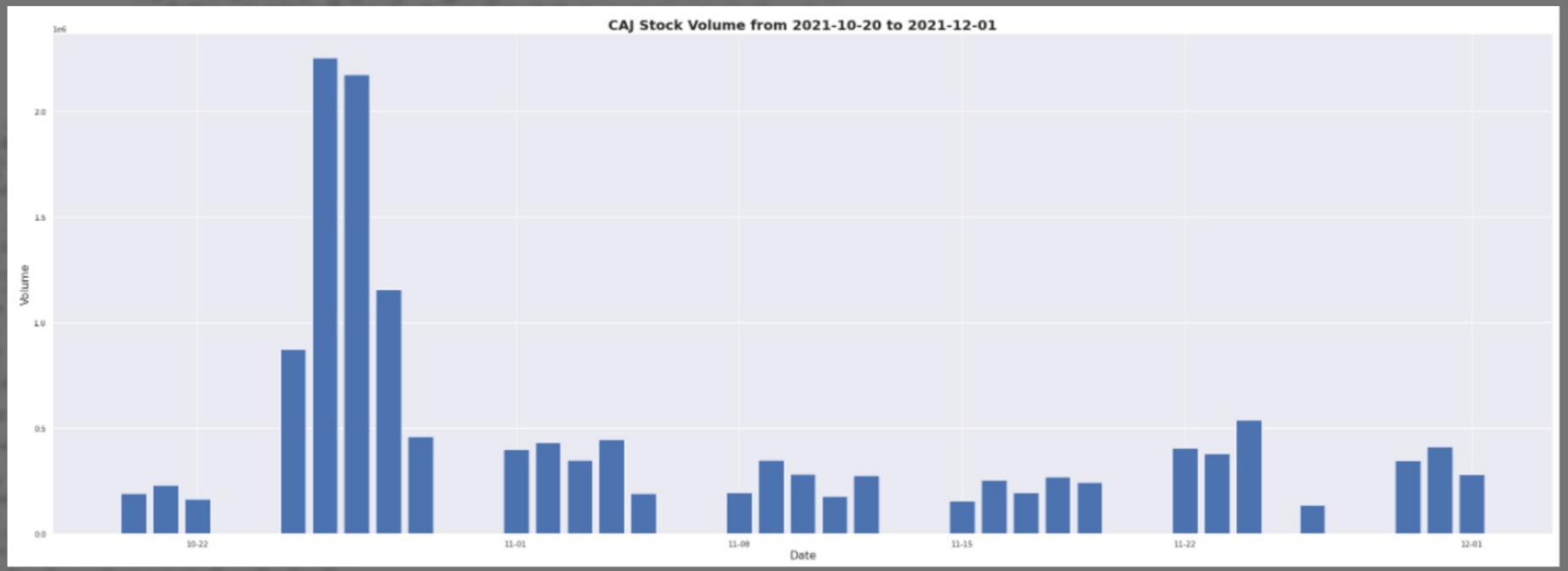
Basic Analysis

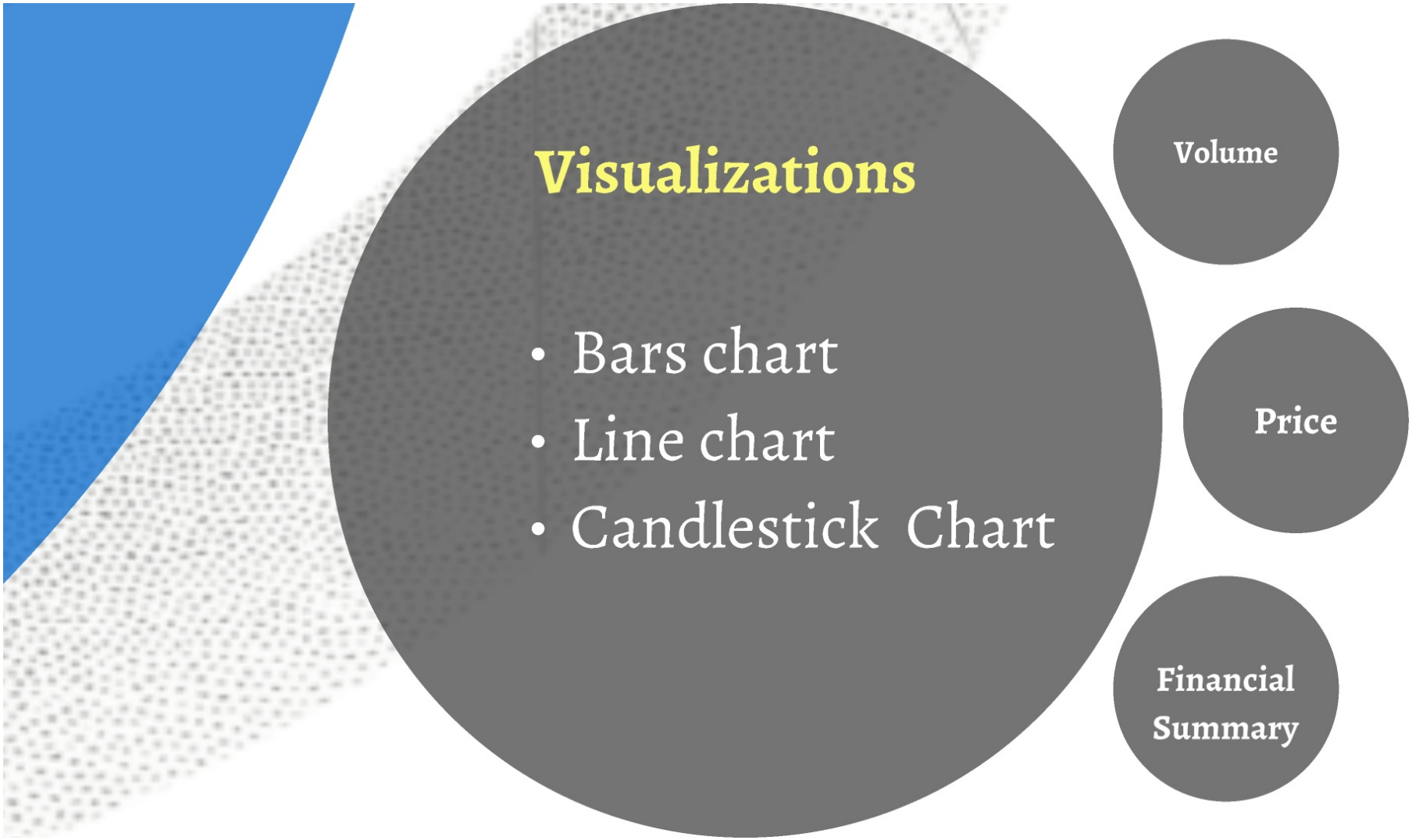
Visualizations


Visualizations
• Bars chart
• Line chart
• Candlestick Chart
Volume
Price
Financial Summary

Volume Vs Time

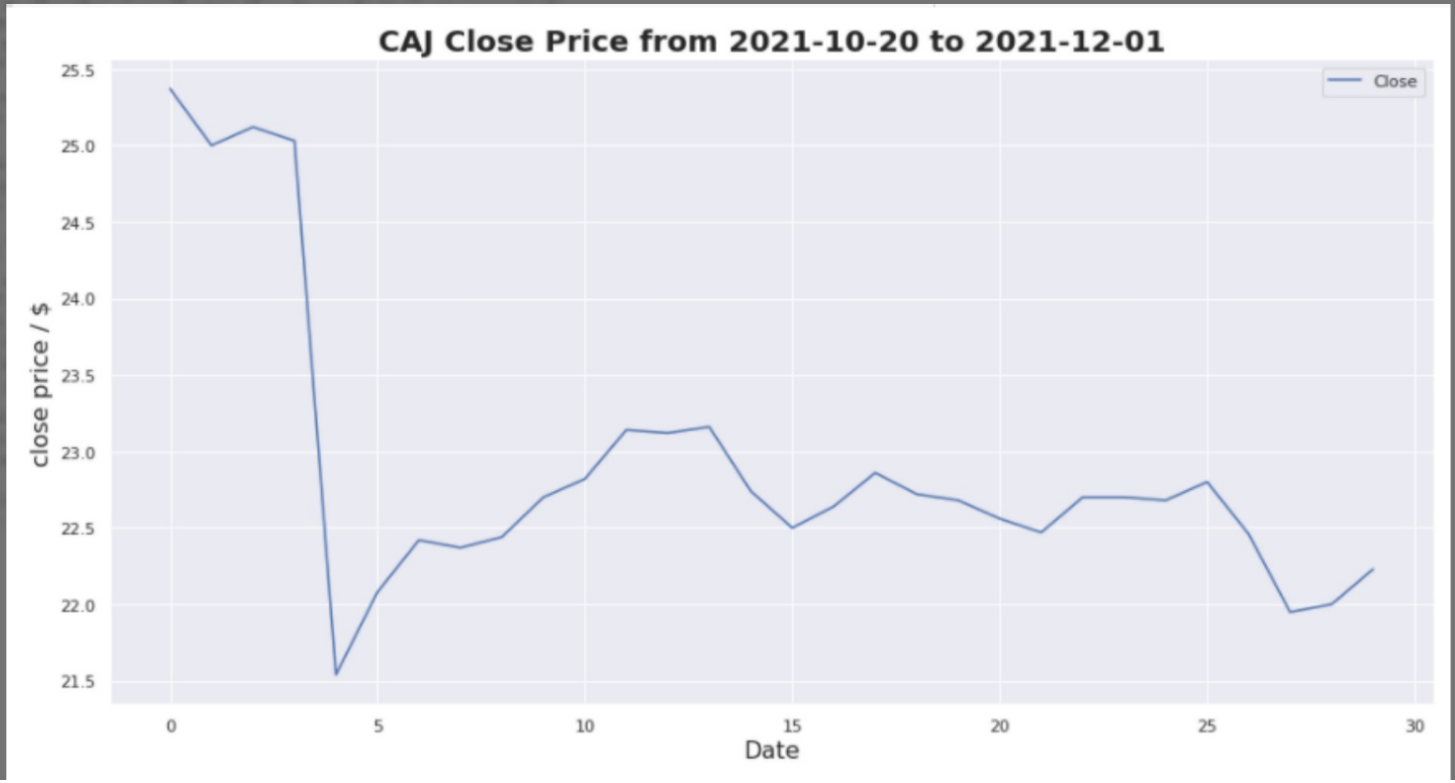
```
# Update date format
import matplotlib.dates as mdates
myFmt = mdates.DateFormatter('%m-%d')
#X axis --> Date
#Y axis --> Volume traded that day
#plt --> imported matplotlib.pyplot as plt
#when we plot a graph using plt, we implicitly create a Figure instance (metaphorically, like a paper
on which we can draw)
#and an Axes inside a Figure object
#add_axes - A parameter of Axes class that has a 4-length sequence of [left, bottom, width, height]
quantities.
#ax.bar() - The bar() function makes a bar plot with the bound rectangle of a particular size w.r.t (x,
height, width, bottom, align)
#plt.xlabel/ylabel - Provides labels to the x and y axes
#plt.title - Provides the title to the figure
#plt.show() - The Plot viewer window is invoked by the this function

current_date = str(df_to_visual['Date'].max().date())
days_before = str(df_to_visual['Date'].min().date())
fig = plt.figure(figsize=(30,10))
ax = fig.add_axes([0,0,1,1])
volum = df_to_visual['Volume']
days = df_to_visual['Date']
ax.bar(days,volum)
ax.xaxis.set_major_formatter(myFmt)
plt.xlabel("Date", fontdict=axis_dict)
plt.ylabel("Volume", fontdict=axis_dict)
plt.title(f"{ticker} Stock Volume from "+ days_before +" to " + current_date,
          fontdict=title_dict)
plt.show()
```

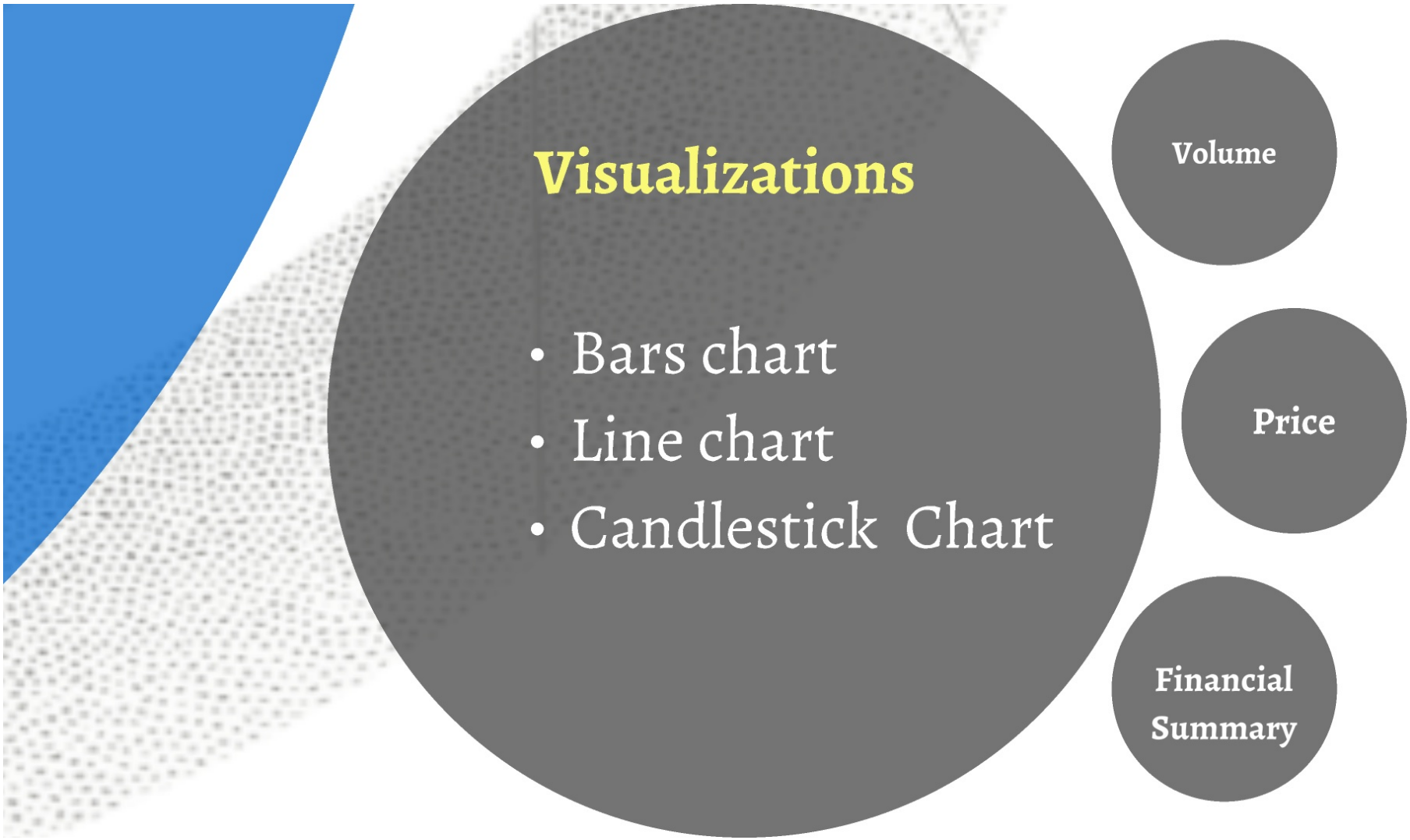

CAJ Stock Volume from 2021-10-20 to 2021-12-01
1e6
2.0
1.5
1.0
0.5
0.0
Volume
10-22
11-01
11-08
11-15
11-22
12-01
Date


Visualizations
• Bars chart
• Line chart
• Candlestick Chart
Volume
Price
Financial Summary

Price Vs Time

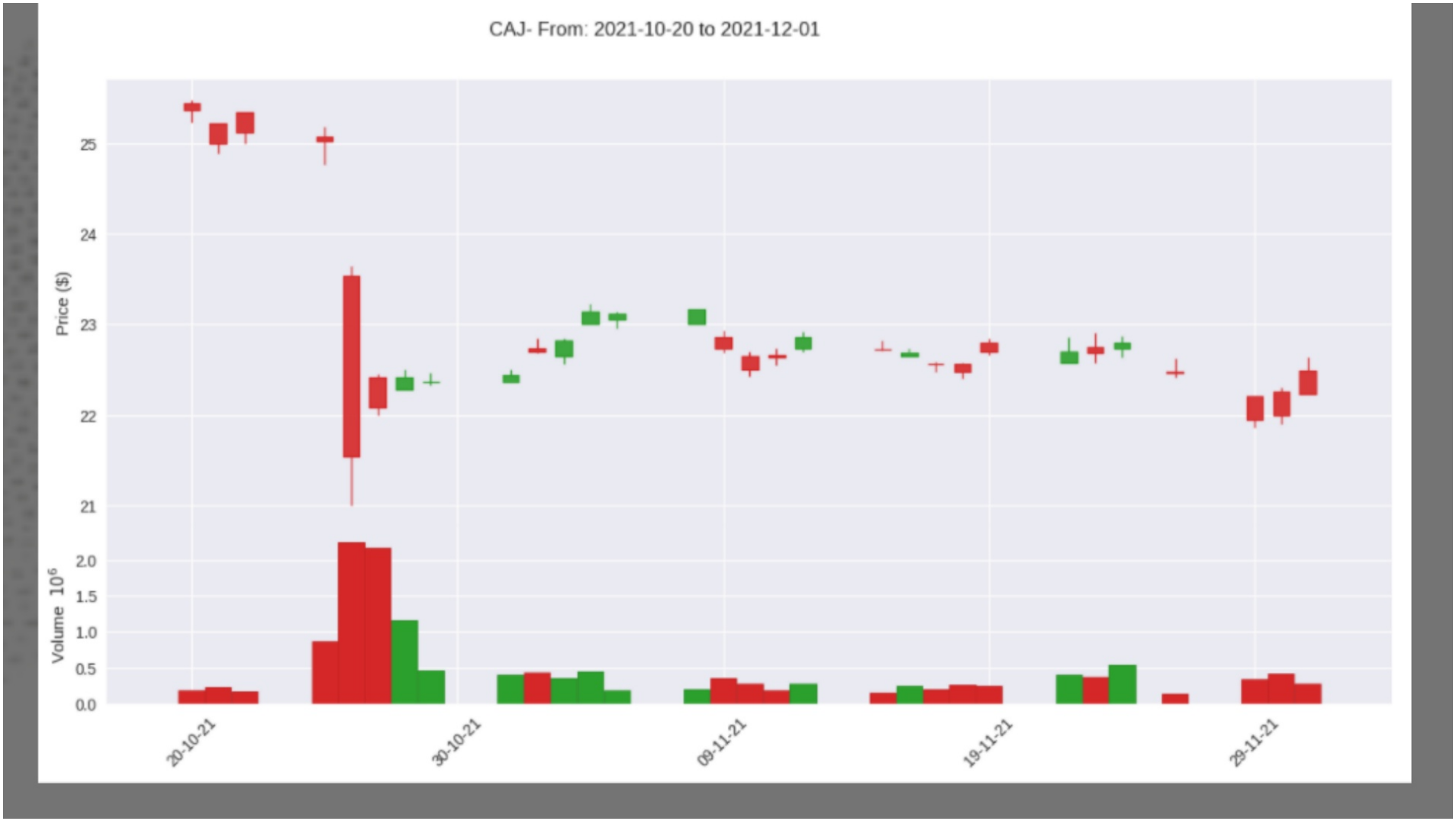
```
df_to_visual[['Close']].plot(label=ticker,figsize=(16,8),title='ClosePrice')
plt.xlabel("Date", fontdict={'size': 16})
plt.ylabel("close price / $", fontdict={'size': 16})
plt.title(f"{ticker} Close Price from "+ days_before +" to " + current_date,
          fontdict={'size': 20, 'weight': 'bold'})
plt.show()
```

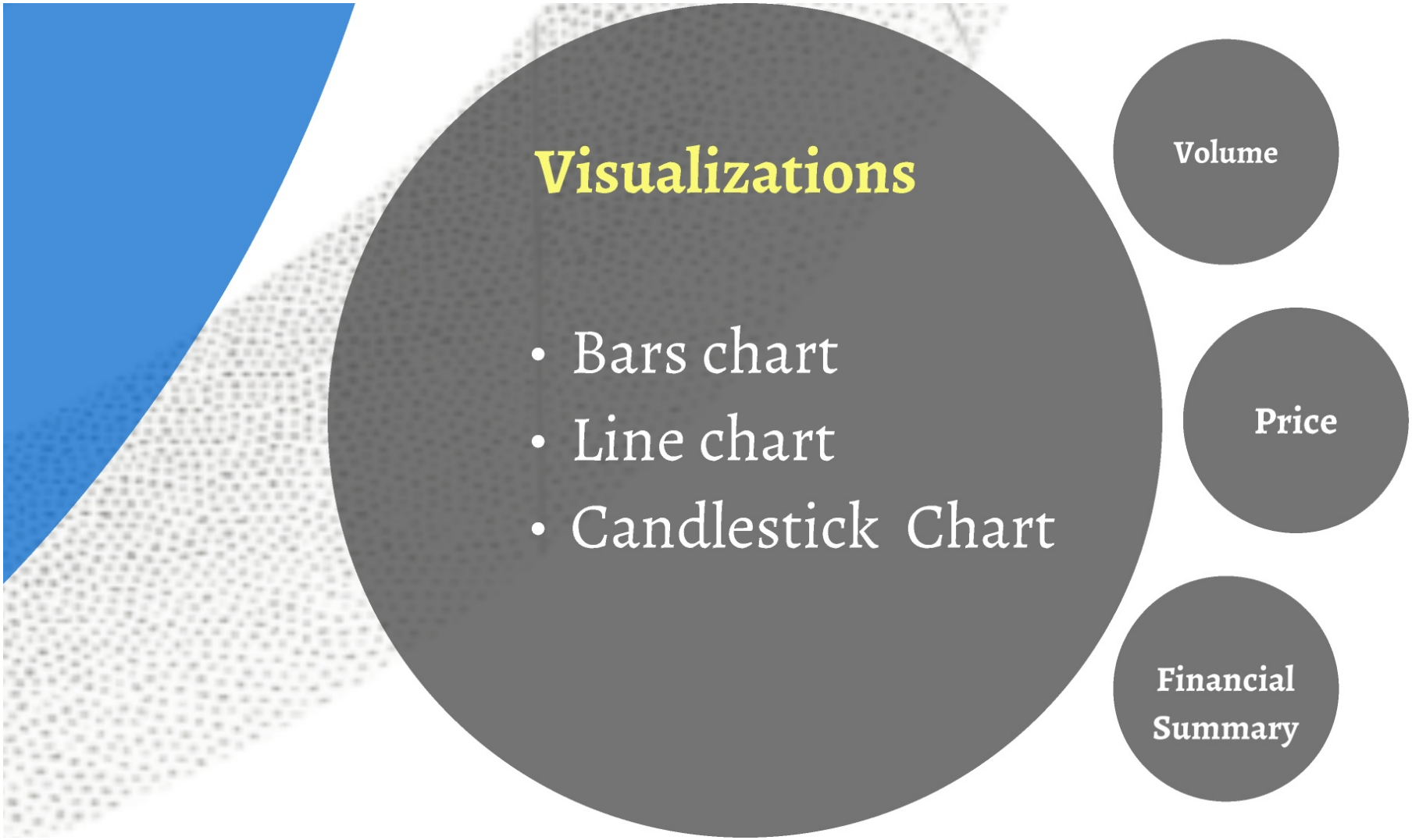



Visualizations
• Bars chart
• Line chart
• Candlestick Chart
Volume
Price
Financial Summary

Financial Summary

```
# plotting the data
# mplfinance library -
# Mplfinance is a dedicated data visualization package from matlplotlib for visualizing financial data.
# It enables visualization of financial data in the form of different types of charts
# like - Candlestick chart, Renko Chart, OHLC chart, Point and Figure chart etc. with very few lines of
code.
# mpf.plot - We can create a chart by simply using the plot function. It defaults to create an OHLC
chart
#            we create a candlestick by adding an extra argument (type='candle') to the plot function.
#            volume=True: adds the trading volume
# mpf.make_marketcolors - Controls the up/down colors of candlesticks, and volume.
#                         Each of the keywords edge, wick, ohlc, and volume can have the following
values:
#                         1. Any single matplotlib color
#                         2. A dict specifying separate colors for 'up' and a 'down'
#                         3. The str 'inherit' indicates that the keyword (edge, wick, ohlc, volume)
should use the same colors
#                            as were specified by the keywords up and down.

import mplfinance as mpf
df_to_visual.index = pd.to_datetime(df_to_visual['Date'])
freq = 'D'
logic = {'Open'  : 'first',
         'High'  : 'max',
         'Low'   : 'min',
         'Close' : 'last',
         'Volume': 'sum'}

dayly_data = df_to_visual.resample(freq).apply(logic)
mc = mpf.make_marketcolors(
    up='tab:green',down='tab:red',
    edge='inherit',
    wick='inherit',
    volume='inherit'
)
style = mpf.make_mpf_style(base_mpl_style="seaborn", marketcolors=mc)
mpf.plot(dayly_data,
         type='candle',
         volume=True,
         style=style,
         title=ticker+'- From: '+ days_before +" to " + current_date ,
         ylabel='Price ($)',
         datetime_format='%d-%m-%y',figsize=(16,8))
```


CAJ- From: 2021-10-20 to 2021-12-01
Price ($)
25
24
23
22
21
Volume 10^6
2.0
1.5
1.0
0.5
0.0
20-10-21
30-10-21
09-11-21
19-11-21
29-11-21


Visualizations
• Bars chart
• Line chart
• Candlestick Chart
Volume
Price
Financial Summary

Visualizations

Date	Date	Open	Close	Low	High	Volume
2021-10-15	2021-10-15	76.809998	77.339996	75.870003	78.735001	20404200
2021-10-18	2021-10-18	78.599998	75.800003	72.610001	79.300003	33053900
2021-10-19	2021-10-19	75.120003	76.430000	73.900002	78.180000	18720700
2021-10-20	2021-10-20	76.570000	75.650002	74.669998	77.519997	15739300
2021-10-21	2021-10-21	74.809998	75.110001	73.889999	75.949997	44450900
2021-10-22	2021-10-22	58.750000	55.139999	55.029999	60.779999	153827500
2021-10-25	2021-10-25	55.959999	54.500000	53.770000	56.150002	62506800
2021-10-26	2021-10-26	55.240002	55.389999	54.880001	57.180000	52905700
2021-10-27	2021-10-27	55.049999	52.020000	51.660000	55.230000	52581300
2021-10-28	2021-10-28	52.680000	54.389999	51.990002	54.959999	34748800
2021-10-29	2021-10-29	53.970001	52.580002	52.209999	53.990002	32465500
2021-11-01	2021-11-01	52.990002	53.980000	52.250000	54.110001	22666700
2021-11-02	2021-11-02	53.570000	52.200001	51.865002	53.660000	24120400
2021-11-03	2021-11-03	51.950001	52.939999	51.470001	52.970001	16319300
2021-11-04	2021-11-04	53.080002	52.259998	51.759998	53.380001	16371700
2021-11-05	2021-11-05	52.150002	53.169998	52.070000	53.570000	17447500
2021-11-08	2021-11-08	53.360001	54.900002	52.849998	55.130001	20154600
2021-11-09	2021-11-09	55.294998	54.820000	54.200001	55.650002	12489000
2021-11-10	2021-11-10	53.910000	52.880001	52.020000	54.169998	14812400
2021-11-11	2021-11-11	53.459999	53.290001	53.200001	54.590000	13322000

Load Data & Select Ticker

Basic Analysis

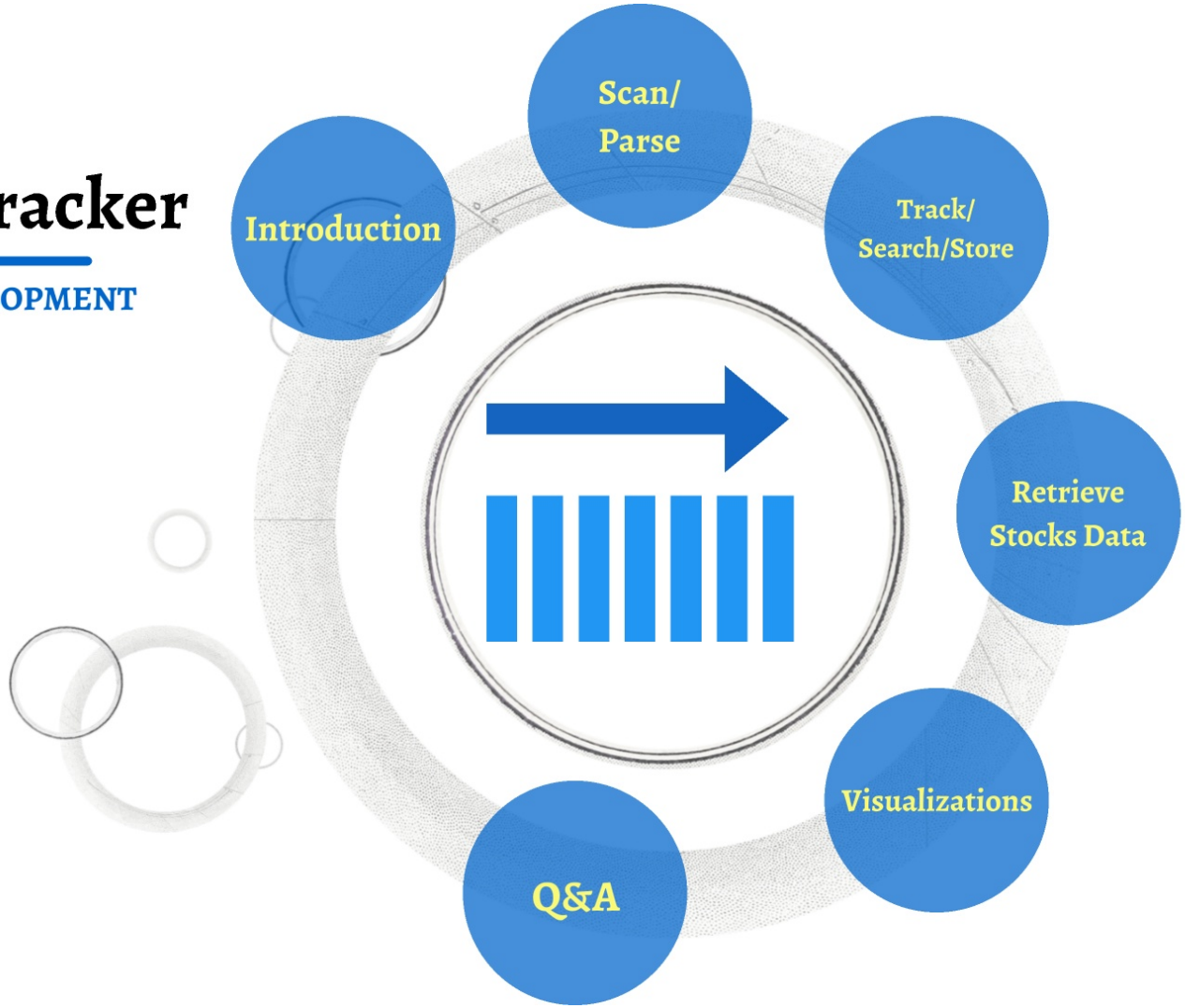
Visualizations

Simple Stocks Tracker

INTRODUCTION TO AI DEVELOPMENT

AIDI-1100-02

Group 7- Fall/2021



Q & A

Source Code:

https://colab.research.google.com/drive/1SukqnEIWs9oRBLht_tB8ZEzp02QODIt3#scrollTo=qR_UVBUQr3WD

Simple Stocks Tracker

INTRODUCTION TO AI DEVELOPMENT

AIDI-1100-02

Group 7- Fall/2021