Name: Borja, Angelo Louis C.
Section: CPE22S3
Performed on: 03/11/2024
Submitted on: 03/18/2024
Submitted to: Engr . Roman M. Richard

Cleaning Data

Setup

```
import pandas as pd
df = pd.read_csv('/content/drive/MyDrive/nyc_temperatures.csv')
df.head()
```

	date	datatype	station	attributes	value
0	2018-10-01T00:00:00	TAVG	GHCND:USW00014732	H,,S,	21.2
1	2018-10-01T00:00:00	TMAX	GHCND:USW00014732	,,W,2400	25.6
2	2018-10-01T00:00:00	TMIN	GHCND:USW00014732	,,W,2400	18.3
3	2018-10-02T00:00:00	TAVG	GHCND:USW00014732	H,,S,	22.7
4	2018-10-02T00:00:00	TMAX	GHCND:USW00014732	,,W,2400	26.1

Next steps: View recommended plots

Renaming Columns

```
df.columns
```

```
Index(['date', 'datatype', 'station', 'attributes', 'value'], dtype='object')
```

```
# change the column names of 'value' and 'attributes'
df.rename(
    columns = {'value': 'temp_C',
               'attributes' : 'flags'},
    inplace = True)
```

```
df.columns
```

```
Index(['date', 'datatype', 'station', 'flags', 'temp_C'], dtype='object')
```

```
# change all column names into their uppercase version
df.rename(str.upper, axis = 'columns').columns
```

```
Index(['DATE', 'DATATYPE', 'STATION', 'FLAGS', 'TEMP_C'], dtype='object')
```

Type Conversion

```
df.dtypes
```

```
date         object
datatype     object
station      object
flags        object
temp_C      float64
dtype: object
```

```
#change the data type of column 'date' into datetime
df.loc[:,'date'] = pd.to_datetime(df.date)
df.dtypes
```

```
<ipython-input-54-a1d4c92fe6c1>:2: DeprecationWarning: In a future version, `df.iloc[:, i] = newvals` will attempt to set the values in
  df.loc[:,'date'] = pd.to_datetime(df.date)
```

```
date        datetime64[ns]
datatype            object
station             object
flags               object
temp_C             float64
dtype: object
```

```
df.date.describe()
```

```
<ipython-input-55-f7d3fa946723>:1: FutureWarning: Treating datetime data as categorical rather than numeric in `.describe` is deprecate
  df.date.describe()
count                      93
unique                     31
top       2018-10-01 00:00:00
freq                        3
first     2018-10-01 00:00:00
last      2018-10-31 00:00:00
Name: date, dtype: object
```

```
#generates two dates that separated by 1 day that have a timezone of 'EST'
pd.date_range(start='2018-10-25', periods=2, freq='D').tz_localize('EST')
```

```
DatetimeIndex(['2018-10-25 00:00:00-05:00', '2018-10-26 00:00:00-05:00'], dtype='datetime64[ns, EST]', freq=None)
```

```
#sets the 'date' column as an index column, parse it as a date data type and set its timezone into 'EST'
eastern = pd.read_csv(
 '/content/drive/MyDrive/nyc_temperatures.csv', index_col='date', parse_dates=True
).tz_localize('EST')
eastern.head()
```

	datatype	station	attributes	value
date				
2018-10-01 00:00:00-05:00	TAVG	GHCND:USW00014732	H,,S,	21.2
2018-10-01 00:00:00-05:00	TMAX	GHCND:USW00014732	,,W,2400	25.6
2018-10-01 00:00:00-05:00	TMIN	GHCND:USW00014732	,,W,2400	18.3
2018-10-02 00:00:00-05:00	TAVG	GHCND:USW00014732	H,,S,	22.7
2018-10-02 00:00:00-05:00	TMAX	GHCND:USW00014732	,,W,2400	26.1

Next steps: View recommended plots

```
#change the timezone of the index column 'date' into 'UTC'
eastern.tz_convert('UTC').head()
```

	datatype	station	attributes	value
date				
2018-10-01 05:00:00+00:00	TAVG	GHCND:USW00014732	H,,S,	21.2
2018-10-01 05:00:00+00:00	TMAX	GHCND:USW00014732	,,W,2400	25.6
2018-10-01 05:00:00+00:00	TMIN	GHCND:USW00014732	,,W,2400	18.3
2018-10-02 05:00:00+00:00	TAVG	GHCND:USW00014732	H,,S,	22.7
2018-10-02 05:00:00+00:00	TMAX	GHCND:USW00014732	,,W,2400	26.1

```
#change the date format of the index column into 'year-month'
eastern.to_period('M').index
```

```
<ipython-input-59-3831acb8bd11>:2: UserWarning: Converting to PeriodArray/Index representation will drop timezone information.
  eastern.to_period('M').index
PeriodIndex(['2018-10', '2018-10', '2018-10', '2018-10', '2018-10', '2018-10',
             '2018-10', '2018-10', '2018-10', '2018-10', '2018-10', '2018-10',
             '2018-10', '2018-10', '2018-10', '2018-10', '2018-10', '2018-10',
             '2018-10', '2018-10', '2018-10', '2018-10', '2018-10', '2018-10',
             '2018-10', '2018-10', '2018-10', '2018-10', '2018-10', '2018-10',
             '2018-10', '2018-10', '2018-10', '2018-10', '2018-10', '2018-10',
             '2018-10', '2018-10', '2018-10', '2018-10', '2018-10', '2018-10',
             '2018-10', '2018-10', '2018-10', '2018-10', '2018-10', '2018-10',
             '2018-10', '2018-10', '2018-10', '2018-10', '2018-10', '2018-10',
             '2018-10', '2018-10', '2018-10', '2018-10', '2018-10', '2018-10',
             '2018-10', '2018-10', '2018-10', '2018-10', '2018-10', '2018-10',
             '2018-10', '2018-10', '2018-10', '2018-10', '2018-10', '2018-10',
             '2018-10', '2018-10', '2018-10', '2018-10', '2018-10', '2018-10',
             '2018-10', '2018-10', '2018-10', '2018-10', '2018-10', '2018-10',
             '2018-10', '2018-10', '2018-10', '2018-10', '2018-10', '2018-10',
             '2018-10', '2018-10', '2018-10'],
            dtype='period[M]', name='date')
```

```
#change the date format of the index column into 'year-month-day'
eastern.to_period('M').to_timestamp().index
```

```
<ipython-input-60-c96604344ba6>:2: UserWarning: Converting to PeriodArray/Index representation will drop timezone information.
  eastern.to_period('M').to_timestamp().index
DatetimeIndex(['2018-10-01', '2018-10-01', '2018-10-01', '2018-10-01',
               '2018-10-01', '2018-10-01', '2018-10-01', '2018-10-01',
               '2018-10-01', '2018-10-01', '2018-10-01', '2018-10-01',
               '2018-10-01', '2018-10-01', '2018-10-01', '2018-10-01',
               '2018-10-01', '2018-10-01', '2018-10-01', '2018-10-01',
               '2018-10-01', '2018-10-01', '2018-10-01', '2018-10-01',
               '2018-10-01', '2018-10-01', '2018-10-01', '2018-10-01',
               '2018-10-01', '2018-10-01', '2018-10-01', '2018-10-01',
               '2018-10-01', '2018-10-01', '2018-10-01', '2018-10-01',
               '2018-10-01', '2018-10-01', '2018-10-01', '2018-10-01',
               '2018-10-01', '2018-10-01', '2018-10-01', '2018-10-01',
               '2018-10-01', '2018-10-01', '2018-10-01', '2018-10-01',
               '2018-10-01', '2018-10-01', '2018-10-01', '2018-10-01',
               '2018-10-01', '2018-10-01', '2018-10-01', '2018-10-01',
               '2018-10-01', '2018-10-01', '2018-10-01', '2018-10-01',
               '2018-10-01', '2018-10-01', '2018-10-01', '2018-10-01',
               '2018-10-01', '2018-10-01', '2018-10-01', '2018-10-01',
               '2018-10-01', '2018-10-01', '2018-10-01', '2018-10-01',
               '2018-10-01', '2018-10-01', '2018-10-01', '2018-10-01',
               '2018-10-01', '2018-10-01', '2018-10-01', '2018-10-01',
               '2018-10-01', '2018-10-01', '2018-10-01', '2018-10-01',
               '2018-10-01', '2018-10-01', '2018-10-01', '2018-10-01',
               '2018-10-01'],
              dtype='datetime64[ns]', name='date', freq=None)
```

```
# imports a csv into a dataframe while also changing two column names
df = pd.read_csv('/content/drive/MyDrive/nyc_temperatures.csv').rename(
 columns={
 'value' : 'temp_C',
 'attributes' : 'flags'
 }
)
#creates a replicate of df with an additional column 'temp_F'
new_df = df.assign(
 date=pd.to_datetime(df.date),
 temp_F=(df.temp_C * 9/5) + 32
)
new_df.dtypes
```

```
date        datetime64[ns]
datatype            object
station             object
flags               object
temp_C             float64
temp_F             float64
dtype: object
```

```
new_df.head()
```

	date	datatype	station	flags	temp_C	temp_F
0	2018-10-01	TAVG	GHCND:USW00014732	H,,S,	21.2	70.16
1	2018-10-01	TMAX	GHCND:USW00014732	,,W,2400	25.6	78.08
2	2018-10-01	TMIN	GHCND:USW00014732	,,W,2400	18.3	64.94
3	2018-10-02	TAVG	GHCND:USW00014732	H,,S,	22.7	72.86
4	2018-10-02	TMAX	GHCND:USW00014732	,,W,2400	26.1	78.98

Next steps: View recommended plots

```
#assigns three additional columns of dtype int into the dataframe df
df = df.assign(
 date=pd.to_datetime(df.date),
 temp_C_whole=df.temp_C.astype('int'),
 temp_F=(df.temp_C * 9/5) + 32,
 temp_F_whole=lambda x: x.temp_F.astype('int')
)
df.head()
```

	date	datatype	station	flags	temp_C	temp_C_whole	temp_F	temp_F_
0	2018-10-01	TAVG	GHCND:USW00014732	H,,S,	21.2	21	70.16	
1	2018-10-01	TMAX	GHCND:USW00014732	,,W,2400	25.6	25	78.08	
2	2018-10-01	TMIN	GHCND:USW00014732	,,W,2400	18.3	18	64.94	

Next steps: View recommended plots

```
# replicates df and added two additional columns that have a data type of 'category'
df_with_categories = df.assign(
 station=df.station.astype('category'),
 datatype=df.datatype.astype('category')
)
df_with_categories.dtypes
```

```
date            datetime64[ns]
datatype              category
station               category
flags                   object
temp_C                 float64
temp_C_whole             int64
temp_F                 float64
temp_F_whole             int64
dtype: object
```

```
# manually create a user defined object that have three possible values.
pd.Categorical(
 ['med', 'med', 'low', 'high'],
 categories=['low', 'med', 'high'],
 ordered=True
)
```

```
['med', 'med', 'low', 'high']
Categories (3, object): ['low' < 'med' < 'high']
```

Reordering, reindexing, and sorting

```
#sorts the entries in descending order based on their 'temp_C' values
df.sort_values(by='temp_C', ascending=False).head(10)
```

	date	datatype	station	flags	temp_C	temp_C_whole	temp_F	temp_F_whole
19	2018-10-07	TMAX	GHCND:USW00014732	,,W,2400	27.8	27	82.04	82
28	2018-10-10	TMAX	GHCND:USW00014732	,,W,2400	27.8	27	82.04	82
31	2018-10-11	TMAX	GHCND:USW00014732	,,W,2400	26.7	26	80.06	80
4	2018-10-02	TMAX	GHCND:USW00014732	,,W,2400	26.1	26	78.98	78
10	2018-10-04	TMAX	GHCND:USW00014732	,,W,2400	26.1	26	78.98	78
25	2018-10-09	TMAX	GHCND:USW00014732	,,W,2400	25.6	25	78.08	78
1	2018-10-01	TMAX	GHCND:USW00014732	,,W,2400	25.6	25	78.08	78
7	2018-10-03	TMAX	GHCND:USW00014732	,,W,2400	25.0	25	77.00	77
27	2018-10-10	TAVG	GHCND:USW00014732	H,,S,	23.8	23	74.84	74
30	2018-10-11	TAVG	GHCND:USW00014732	H,,S,	23.4	23	74.12	74

```
#sorts the entries in descending order based on their 'temp_C' and 'date' values
df.sort_values(by=['temp_C', 'date'], ascending=False).head(10)
```

	date	datatype	station	flags	temp_C	temp_C_whole	temp_F	temp_F_whole
28	2018-10-10	TMAX	GHCND:USW00014732	,,W,2400	27.8	27	82.04	82
19	2018-10-07	TMAX	GHCND:USW00014732	,,W,2400	27.8	27	82.04	82
31	2018-10-11	TMAX	GHCND:USW00014732	,,W,2400	26.7	26	80.06	80
10	2018-10-04	TMAX	GHCND:USW00014732	,,W,2400	26.1	26	78.98	78
4	2018-10-02	TMAX	GHCND:USW00014732	,,W,2400	26.1	26	78.98	78
25	2018-10-09	TMAX	GHCND:USW00014732	,,W,2400	25.6	25	78.08	78
1	2018-10-01	TMAX	GHCND:USW00014732	,,W,2400	25.6	25	78.08	78
7	2018-10-03	TMAX	GHCND:USW00014732	,,W,2400	25.0	25	77.00	77
27	2018-10-10	TAVG	GHCND:USW00014732	H,,S,	23.8	23	74.84	74
30	2018-10-11	TAVG	GHCND:USW00014732	H,,S,	23.4	23	74.12	74

```
#returns a dataframe with 5 entries, the entries where the top 5 entries with the largest 'temp_C' value
df.nlargest(n=5, columns='temp_C')
```

	date	datatype	station	flags	temp_C	temp_C_whole	temp_F	temp_F_
19	2018-10-07	TMAX	GHCND:USW00014732	,,W,2400	27.8	27	82.04	
28	2018-10-10	TMAX	GHCND:USW00014732	,,W,2400	27.8	27	82.04	
31	2018-10-11	TMAX	GHCND:USW00014732	,,W,2400	26.7	26	80.06	

```
#returns a dataframe with 5 entries, the entries where the top 5 entries with the smallest 'temp_C' value
df.nsmallest(n=5, columns=['temp_C', 'date'])
```

	date	datatype	station	flags	temp_C	temp_C_whole	temp_F	temp_F_
65	2018-10-22	TMIN	GHCND:USW00014732	,,W,2400	5.6	5	42.08	
77	2018-10-26	TMIN	GHCND:USW00014732	,,W,2400	5.6	5	42.08	
62	2018-10-21	TMIN	GHCND:USW00014732	,,W,2400	6.1	6	42.98	

```
#sorts in ascending order the five random entries
df.sample(5, random_state=0).sort_index().index
```

```
Int64Index([2, 13, 16, 30, 55], dtype='int64')
```

```
#sorts the column name in the df dataframe alphabetically
df.sort_index(axis=1).head()
```

	datatype	date	flags	station	temp_C	temp_C_whole	temp_F	temp_F_whole
0	TAVG	2018-10-01	H,,S,	GHCND:USW00014732	21.2	21	70.16	70
1	TMAX	2018-10-01	,,W,2400	GHCND:USW00014732	25.6	25	78.08	78
2	TMIN	2018-10-01	,,W,2400	GHCND:USW00014732	18.3	18	64.94	64
3	TAVG	2018-10-02	H,,S,	GHCND:USW00014732	22.7	22	72.86	72
4	TMAX	2018-10-02	,,W,2400	GHCND:USW00014732	26.1	26	78.98	78

```
#shows all the rows but only their  temp_C,temp_C_whole,temp_F and temp_F_whole values.
df.sort_index(axis=1).head().loc[:,'temp_C':'temp_F_whole']
```

	temp_C	temp_C_whole	temp_F	temp_F_whole
0	21.2	21	70.16	70
1	25.6	25	78.08	78
2	18.3	18	64.94	64
3	22.7	22	72.86	72
4	26.1	26	78.98	78

```
# compares the original dataframe arrangement to the sorted version that was order based on their temp_C values
df.equals(df.sort_values(by='temp_C'))
```

```
False
```

```
#sorted the order of the entries based on their temp_C values then bring it back to the original arrangement by ordereing it by index
df.equals(df.sort_values(by='temp_C').sort_index())
```

```
True
```

```
#selects all the entries that have a 'TAVG' value in their datatype column
#the original index column 'date' will be replaced by a new index column 'index'
#but the date column won't be removed as a column in the dataframe
df[df.datatype == 'TAVG'].head().reset_index()
```

	index	date	datatype	station	flags	temp_C	temp_C_whole	temp_F	tem
0	0	2018-10-01	TAVG	GHCND:USW00014732	H,,S,	21.2	21	70.16	
1	3	2018-10-02	TAVG	GHCND:USW00014732	H,,S,	22.7	22	72.86	
2	6	2018-10-03	TAVG	GHCND:USW00014732	H,,S,	21.8	21	71.24	

```
# set the date column again as the index column of the dataframe
df.set_index("date", inplace=True)
df.head()
```

	datatype	station	flags	temp_C	temp_C_whole	temp_F	temp_F_who
date							
2018-10-01	TAVG	GHCND:USW00014732	H,,S,	21.2	21	70.16	
2018-10-01	TMAX	GHCND:USW00014732	,,W,2400	25.6	25	78.08	
2018-10-01	TMIN	GHCND:USW00014732	,,W,2400	18.3	18	64.94	

Next steps: View recommended plots

```
#grabs all the entries that have a date value that is included from Oct 11, 2018 to
#Oct 12, 2018
df['2018-10-11':'2018-10-12']
```

	datatype	station	flags	temp_C	temp_C_whole	temp_F	temp_F_who
date							
2018-10-11	TAVG	GHCND:USW00014732	H,,S,	23.4	23	74.12	
2018-10-11	TMAX	GHCND:USW00014732	,,W,2400	26.7	26	80.06	
2018-10-11	TMIN	GHCND:USW00014732	,,W,2400	21.7	21	71.06	
2018-	TAVG	GHCND:USW00014732	H,,S,	18.3	18	64.94	

```
#imports the sp500 csv into a dataframe, the 'adj_close' column was removed from the dataframe
sp = pd.read_csv(
 '/content/drive/MyDrive/sp500.csv', index_col='date', parse_dates=True
).drop(columns=['adj_close'])

#creates a new column named 'day_of_week' that shows the day names of the entries
#gathered from the datetime index
sp.head(10).assign(
 day_of_week=lambda x: x.index.day_name()
)
```

	high	low	open	close	volume	day_of_week
date						
2017-01-03	2263.879883	2245.129883	2251.570068	2257.830078	3770530000	Tuesday
2017-01-04	2272.820068	2261.600098	2261.600098	2270.750000	3764890000	Wednesday
2017-01-05	2271.500000	2260.449951	2268.179932	2269.000000	3761820000	Thursday
2017-01-06	2282.100098	2264.060059	2271.139893	2276.979980	3339890000	Friday
2017-01-09	2275.489990	2268.899902	2273.590088	2268.899902	3217610000	Monday
2017-01-10	2279.270020	2265.270020	2269.719971	2268.899902	3638790000	Tuesday
2017-01-11	2275.320068	2260.830078	2268.600098	2275.320068	3620410000	Wednesday

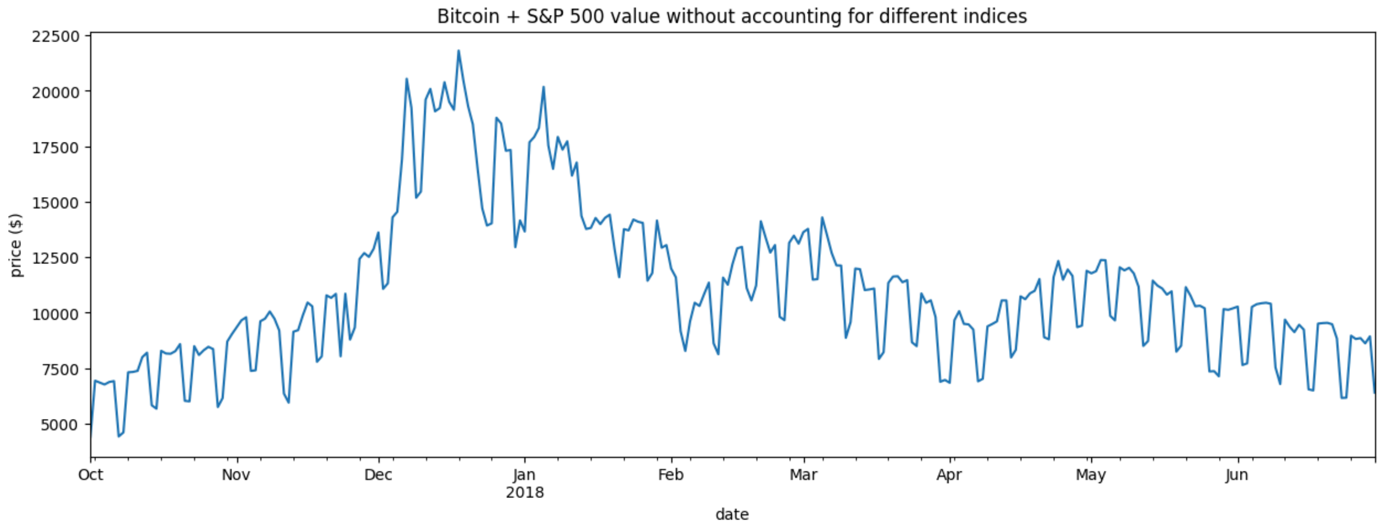
```
# imports the bitcoin.csv and sets the 'date' column as the index, the 'market_cap'
# column was not included in the dataframe
bitcoin = pd.read_csv(
 '/content/drive/MyDrive/bitcoin.csv', index_col='date', parse_dates=True
).drop(columns=['market_cap'])
```

```
# every day's closing price = S&P 500 close + Bitcoin close (same for other metrics)
# merges the two dataframes using the day part of thier 'date' columns as reference
# summed the values of the columns with the same names into one
#added a new column to the first 10 entries of the portfolio named 'day_of_week' and filled it with values
portfolio = pd.concat(
 [sp, bitcoin], sort=False
).groupby(pd.Grouper(freq='D')).sum()
portfolio.head(10).assign(
 day_of_week=lambda x: x.index.day_name()
)
```

	high	low	open	close	volume	day_of_week
date						
2017-01-01	1003.080000	958.700000	963.660000	998.330000	147775008	Sunday
2017-01-02	1031.390000	996.700000	998.620000	1021.750000	222184992	Monday
2017-01-03	3307.959883	3266.729883	3273.170068	3301.670078	3955698000	Tuesday
2017-01-04	3432.240068	3306.000098	3306.000098	3425.480000	4109835984	Wednesday
2017-01-05	3462.600000	3170.869951	3424.909932	3282.380000	4272019008	Thursday
2017-01-06	3328.910098	3148.000059	3285.379893	3179.179980	3691766000	Friday
2017-01-07	908.590000	823.560000	903.490000	908.590000	279550016	Saturday

```
import matplotlib.pyplot as plt # we use this module for plotting
```

```
# will plot the close column of the entries of portfolio
# that is part of the range from the October of 2017 to June of 2018
portfolio['2017-Q4':'2018-Q2'].plot(
 y='close', figsize=(15, 5), legend=False,
 title='Bitcoin + S&P 500 value without accounting for different indices'
) # plot the closing price from Q4 2017 through Q2 2018
plt.ylabel('price ($)') # label the y-axis
plt.show() # show the plot
```



```
# realigns the entries in sp based on the arrangment in bitcoin so both dataframes can match
sp.reindex(bitcoin.index).head(10).assign(
 day_of_week=lambda x: x.index.day_name()
)
```

	high	low	open	close	volume	day_of_week
date						
2017-01-01	NaN	NaN	NaN	NaN	NaN	Sunday
2017-01-02	NaN	NaN	NaN	NaN	NaN	Monday
2017-01-03	2263.879883	2245.129883	2251.570068	2257.830078	3.770530e+09	Tuesday
2017-01-04	2272.820068	2261.600098	2261.600098	2270.750000	3.764890e+09	Wednesday
2017-01-05	2271.500000	2260.449951	2268.179932	2269.000000	3.761820e+09	Thursday
2017-01-06	2282.100098	2264.060059	2271.139893	2276.979980	3.339890e+09	Friday
2017-01-07	NaN	NaN	NaN	NaN	NaN	Saturday
2017-01-08	NaN	NaN	NaN	NaN	NaN	Sunday
2017-01-09	2275.489990	2268.899902	2273.590088	2268.899902	3.217610e+09	Monday
2017-01-10	2279.270020	2265.270020	2269.719971	2268.899902	3.638790e+09	Tuesday

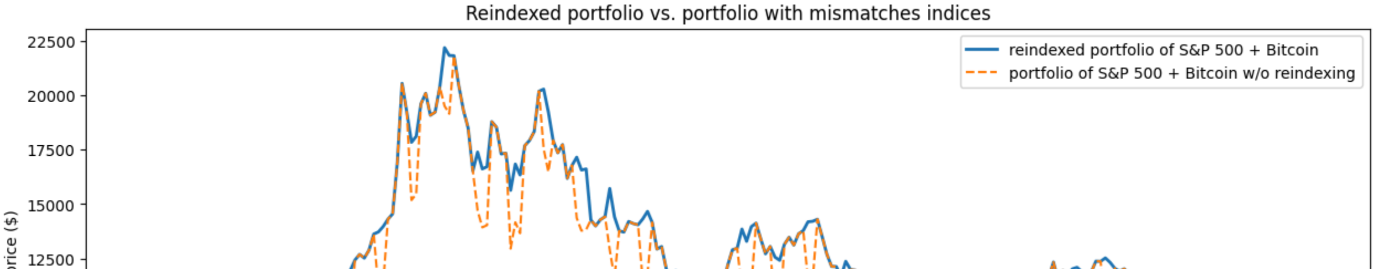
```
# fills the values of the entries that have missing values
# by the values of the entry that was the last day of that trading week
sp.reindex(
 bitcoin.index, method='ffill'
).head(10).assign(
 day_of_week=lambda x: x.index.day_name()
)
```

	high	low	open	close	volume	day_of_week
date						
2017-01-01	NaN	NaN	NaN	NaN	NaN	Sunday
2017-01-02	NaN	NaN	NaN	NaN	NaN	Monday
2017-01-03	2263.879883	2245.129883	2251.570068	2257.830078	3.770530e+09	Tuesday
2017-01-04	2272.820068	2261.600098	2261.600098	2270.750000	3.764890e+09	Wednesday
2017-01-05	2271.500000	2260.449951	2268.179932	2269.000000	3.761820e+09	Thursday
2017-01-06	2282.100098	2264.060059	2271.139893	2276.979980	3.339890e+09	Friday
2017-01-07	2282.100098	2264.060059	2271.139893	2276.979980	3.339890e+09	Saturday
2017-01-08	2282.100098	2264.060059	2271.139893	2276.979980	3.339890e+09	Sunday
2017-01-09	2275.489990	2268.899902	2273.590088	2268.899902	3.217610e+09	Monday
2017-01-10	2279.270020	2265.270020	2269.719971	2268.899902	3.638790e+09	Tuesday

```
import numpy as np
sp_reindexed = sp.reindex(
 bitcoin.index
).assign(
 volume=lambda x: x.volume.fillna(0), # put 0 when market is closed
 close=lambda x: x.close.fillna(method='ffill'), # carry this forward
 # take the closing price if these aren't available
 open=lambda x: np.where(x.open.isnull(), x.close, x.open),
 high=lambda x: np.where(x.high.isnull(), x.close, x.high),
 low=lambda x: np.where(x.low.isnull(), x.close, x.low)
)
sp_reindexed.head(10).assign(
 day_of_week=lambda x: x.index.day_name()
)
```

	high	low	open	close	volume	day_of_week
date						
2017-01-01	NaN	NaN	NaN	NaN	0.000000e+00	Sunday
2017-01-02	NaN	NaN	NaN	NaN	0.000000e+00	Monday
2017-01-03	2263.879883	2245.129883	2251.570068	2257.830078	3.770530e+09	Tuesday
2017-01-04	2272.820068	2261.600098	2261.600098	2270.750000	3.764890e+09	Wednesday
2017-01-05	2271.500000	2260.449951	2268.179932	2269.000000	3.761820e+09	Thursday
2017-01-06	2282.100098	2264.060059	2271.139893	2276.979980	3.339890e+09	Friday
2017-01-07	2276.979980	2276.979980	2276.979980	2276.979980	0.000000e+00	Saturday

```
# every day's closing price = S&P 500 close adjusted for market closure + Bitcoin close (same for other metrics)
fixed_portfolio = pd.concat([sp_reindexed, bitcoin], sort=False).groupby(pd.Grouper(freq='D')).sum()
ax = fixed_portfolio['2017-Q4':'2018-Q2'].plot(
 y='close', label='reindexed portfolio of S&P 500 + Bitcoin', figsize=(15, 5), linewidth=2,
 title='Reindexed portfolio vs. portfolio with mismatches indices'
) # plot the reindexed portfolio's closing price from Q4 2017 through Q2 2018
portfolio['2017-Q4':'2018-Q2'].plot(
 y='close', ax=ax, linestyle='--', label='portfolio of S&P 500 + Bitcoin w/o reindexing'
).set_ylabel('price ($)') # add line for original portfolio for comparison and label y-axis
plt.show() # show the plot
```


Reindexed portfolio vs. portfolio with mismatches indices
reindexed portfolio of S&P 500 + Bitcoin
portfolio of S&P 500 + Bitcoin w/o reindexing
rice ($)