Macroeconomic Data using DB.NOMICS

September 1, 2020

Table of Contents

by MachinaFantasma | Twitter

1 Purpose

In this notebook, we do the following:

1. Introduce you to the `Pandas` library for data management and visualization.
2. Exploit a convenient database provider called `DB.NOMICS` and their Python Application Programming Interface (API) for easily fetching publicly available data.
3. Show how to do some basic operations with `Pandas`, including plotting simple time series data (with `Matplotlib`).

2 DB.NOMICS and Python/Pandas Interface

We'll use data provided through a third-party database aggregation service called DB.Nomics. They also provide a convenient `Python` library called `dbnomics`.

Ensure that you have this library installed.

[1]:
```
import dbnomics as db
from dbnomics import fetch_series, fetch_series_by_api_link
```

We'll use the `Pandas` library to manipulate and analyze the data we import into dataframes.

We will also need a plotting library for visualize graphs and charts. Here I choose to use `matplotlib`.

[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

3 How to find and fetch the data we want?

1. Search on the "Providers" page of DB.Nomics.
   - We see there are many agencies that participate in this database.
   - In this example, we are interested in data from the International Monetary Fund (IMF).
2. Use keyword search to discover the data series we want.
3. Note/copy the data series' `dbnomics` *application programming interface* (API) *link name*.
   - For example, the real GDP series in constant international dollars for Korea (KOR) has an *API link name* of `IMF/WEO/KOR.NGDPRPPPPC.purchasing_power_parity_2011_international_dollar`
4. Once we know which data series we want to download, we will use the `fetch_series` function to download this, or to download multiple series at once.
   - If there are many series you're interested in getting in one go, then add these to the "Cart" in DB.Nomics and navigate to their "Cart" page. At the top right, you can download all these API link names all at once using the "Download" -> "Copy API Link - JSON" button/drop-down menu command.

3.1 Fetching data from Cart by API Link

In this example below, we use the `fetch_series_by_api_link` function to download many series from the "Cart". * This is provided by `dbnomics`. * See previous comments.

This automatically organizes the data into a `Pandas` dataframe, or `df` in short:

[3]:
```
# Fetch data using API link from DB.NOMICS "Cart"
df_raw = fetch_series_by_api_link(
    "https://api.db.nomics.world/v22/series?observations=1&series_ids=IMF/WEO/
↪KOR.NGDPRPPPPC.purchasing_power_parity_2011_international_dollar,IMF/WEO/SGP.
↪NGDPRPPPPC.purchasing_power_parity_2011_international_dollar,IMF/WEO/PHL.
↪NGDPRPPPPC.purchasing_power_parity_2011_international_dollar,IMF/WEO/MYS.
↪NGDPRPPPPC.purchasing_power_parity_2011_international_dollar,IMF/WEO/USA.
↪NGDPRPPPPC.purchasing_power_parity_2011_international_dollar,IMF/WEO/AUS.
↪NGDPRPPPPC.purchasing_power_parity_2011_international_dollar,IMF/WEO/DEU.
↪NGDPRPPPPC.purchasing_power_parity_2011_international_dollar,IMF/WEO/JPN.
↪NGDPRPPPPC.purchasing_power_parity_2011_international_dollar,IMF/WEO/TWN.
↪NGDPRPPPPC.purchasing_power_parity_2011_international_dollar,IMF/WEO/CHN.
↪NGDPRPPPPC.purchasing_power_parity_2011_international_dollar"
    )
```

Let's display the first rows of this `DataFrame` by using the `head` method, each row representing an observation of the time series:

[4]:
```
df_raw.head()
```

[4]:
```
  @frequency provider_code dataset_code      dataset_name  \
0     annual           IMF          WEO  WEO by countries
1     annual           IMF          WEO  WEO by countries
2     annual           IMF          WEO  WEO by countries
3     annual           IMF          WEO  WEO by countries
4     annual           IMF          WEO  WEO by countries

                                      series_code  \
0  KOR.NGDPRPPPPC.purchasing_power_parity_2011_in…
1  KOR.NGDPRPPPPC.purchasing_power_parity_2011_in…
2  KOR.NGDPRPPPPC.purchasing_power_parity_2011_in…
3  KOR.NGDPRPPPPC.purchasing_power_parity_2011_in…
4  KOR.NGDPRPPPPC.purchasing_power_parity_2011_in…

                                      series_name original_period  \
0  Korea - Gross domestic product per capita, con…            1980
1  Korea - Gross domestic product per capita, con…            1981
2  Korea - Gross domestic product per capita, con…            1982
3  Korea - Gross domestic product per capita, con…            1983
4  Korea - Gross domestic product per capita, con…            1984

      period  original_value     value weo-country weo-subject  \
0 1980-01-01        5200.031  5200.031         KOR  NGDPRPPPPC
1 1981-01-01        5487.139  5487.139         KOR  NGDPRPPPPC
2 1982-01-01        5849.547  5849.547         KOR  NGDPRPPPPC
3 1983-01-01        6527.211  6527.211         KOR  NGDPRPPPPC
4 1984-01-01        7120.422  7120.422         KOR  NGDPRPPPPC

                                          unit WEO Country  \
0  purchasing_power_parity_2011_international_dollar       Korea
1  purchasing_power_parity_2011_international_dollar       Korea
2  purchasing_power_parity_2011_international_dollar       Korea
3  purchasing_power_parity_2011_international_dollar       Korea
4  purchasing_power_parity_2011_international_dollar       Korea

                                      WEO Subject  \
0  Gross domestic product per capita, constant pr…
1  Gross domestic product per capita, constant pr…
2  Gross domestic product per capita, constant pr…
3  Gross domestic product per capita, constant pr…
4  Gross domestic product per capita, constant pr…

                                          Unit
```

```
0  Purchasing power parity; 2011 international do…
1  Purchasing power parity; 2011 international do…
2  Purchasing power parity; 2011 international do…
3  Purchasing power parity; 2011 international do…
4  Purchasing power parity; 2011 international do…
```

3.2 Fetching data individually

Alternatively, if you're patient and want to see what you import, then you can use `fetch_series` to fetch the same data as the last method.

[5]:
```
df_raw2 = fetch_series([
                    'IMF/WEO/KOR.NGDPRPPPPC.
↪purchasing_power_parity_2011_international_dollar',
                    'IMF/WEO/SGP.NGDPRPPPPC.
↪purchasing_power_parity_2011_international_dollar',
                    'IMF/WEO/PHL.NGDPRPPPPC.
↪purchasing_power_parity_2011_international_dollar',
                    'IMF/WEO/MYS.NGDPRPPPPC.
↪purchasing_power_parity_2011_international_dollar',
                    'IMF/WEO/USA.NGDPRPPPPC.
↪purchasing_power_parity_2011_international_dollar',
                    'IMF/WEO/AUS.NGDPRPPPPC.
↪purchasing_power_parity_2011_international_dollar',
                    'IMF/WEO/DEU.NGDPRPPPPC.
↪purchasing_power_parity_2011_international_dollar',
                    'IMF/WEO/JPN.NGDPRPPPPC.
↪purchasing_power_parity_2011_international_dollar',
                    'IMF/WEO/TWN.NGDPRPPPPC.
↪purchasing_power_parity_2011_international_dollar',
                    'IMF/WEO/CHN.NGDPRPPPPC.
↪purchasing_power_parity_2011_international_dollar',
                   ])
```

[6]:
```
df_raw2.head()
```

[6]:
```
   @frequency provider_code dataset_code      dataset_name  \
0      annual           IMF          WEO  WEO by countries
1      annual           IMF          WEO  WEO by countries
2      annual           IMF          WEO  WEO by countries
3      annual           IMF          WEO  WEO by countries
4      annual           IMF          WEO  WEO by countries

                                          series_code  \
0  KOR.NGDPRPPPPC.purchasing_power_parity_2011_in…
1  KOR.NGDPRPPPPC.purchasing_power_parity_2011_in…
```

```
2  KOR.NGDPRPPPPC.purchasing_power_parity_2011_in…
3  KOR.NGDPRPPPPC.purchasing_power_parity_2011_in…
4  KOR.NGDPRPPPPC.purchasing_power_parity_2011_in…

                                      series_name original_period  \
0  Korea - Gross domestic product per capita, con…            1980
1  Korea - Gross domestic product per capita, con…            1981
2  Korea - Gross domestic product per capita, con…            1982
3  Korea - Gross domestic product per capita, con…            1983
4  Korea - Gross domestic product per capita, con…            1984

      period  original_value     value weo-country weo-subject  \
0 1980-01-01        5200.031  5200.031         KOR  NGDPRPPPPC
1 1981-01-01        5487.139  5487.139         KOR  NGDPRPPPPC
2 1982-01-01        5849.547  5849.547         KOR  NGDPRPPPPC
3 1983-01-01        6527.211  6527.211         KOR  NGDPRPPPPC
4 1984-01-01        7120.422  7120.422         KOR  NGDPRPPPPC

                                           unit WEO Country  \
0  purchasing_power_parity_2011_international_dollar       Korea
1  purchasing_power_parity_2011_international_dollar       Korea
2  purchasing_power_parity_2011_international_dollar       Korea
3  purchasing_power_parity_2011_international_dollar       Korea
4  purchasing_power_parity_2011_international_dollar       Korea

                                      WEO Subject  \
0  Gross domestic product per capita, constant pr…
1  Gross domestic product per capita, constant pr…
2  Gross domestic product per capita, constant pr…
3  Gross domestic product per capita, constant pr…
4  Gross domestic product per capita, constant pr…

                                           Unit
0  Purchasing power parity; 2011 international do…
1  Purchasing power parity; 2011 international do…
2  Purchasing power parity; 2011 international do…
3  Purchasing power parity; 2011 international do…
4  Purchasing power parity; 2011 international do…
```

3.3 Reshaping dataframe

The default dataframe `df_raw` (or `df_raw2`) was organized by default.

- Data is often stored in so-called "stacked" or "record" format.

We want to reshape the dataframe into an orientation that is easier to read. Let's just focus on `df` from earlier.

- Read more here on what you can do in terms of reshaping dataframes.

[7]:
```
df = df_raw.pivot(
                    index="original_period",
                    columns="weo-country",
                    values="original_value"
                  )
```

Warning:

- In this simple example, we are only dealing with the one data type: Real GDP per person.
- So I directly relabelled the column titles by `weo-country` names.
- If you are dealing with more dimensions to your dataframe then you'll need to be more careful! See more here.

Let see this reshaped dataframe:

[8]:
```
df
```

[8]:

weo-country	AUS	CHN	DEU	JPN	KOR
original_period					
1980	24402.740	718.568	26198.777	20769.324	5200.031
1981	25008.726	744.898	26178.349	21486.502	5487.139
1982	24615.690	799.303	25990.568	22043.038	5849.547
1983	24191.201	873.987	26487.013	22662.793	6527.211
1984	25408.361	993.818	27345.986	23529.343	7120.422
1985	26418.451	1112.074	28014.622	24606.934	7597.069
1986	26665.732	1192.394	28672.099	25290.420	8366.135
1987	27533.975	1310.055	29088.147	26359.833	9316.923
1988	28201.940	1434.134	29992.106	28029.506	10324.483
1989	29071.601	1472.119	30857.364	29276.487	10941.503
1990	29107.619	1507.568	32009.721	30606.860	11897.221
1991	28466.994	1626.574	33183.152	31527.689	12999.745
1992	28900.974	1836.177	33600.527	31682.939	13659.854
1993	29754.101	2067.654	33089.104	31417.106	14447.607
1994	30905.534	2310.462	33797.603	31641.206	15619.719
1995	31387.438	2535.854	34250.725	32425.933	16943.274
1996	32317.419	2758.030	34462.475	33358.034	18057.165
1997	33461.991	2981.633	35060.591	33636.777	18948.031
1998	34675.845	3184.960	35796.485	33168.315	17782.517
1999	35808.561	3402.250	36482.048	33022.455	19653.277
2000	36468.738	3664.333	37525.784	33874.974	21119.341
2001	36943.493	3942.990	38131.248	33932.099	21974.887
2002	38056.600	4275.986	38026.104	33900.665	23536.057
2003	38636.064	4676.183	37770.670	34355.617	24151.298
2004	39849.643	5120.754	38262.014	35085.569	25306.041
2005	40460.543	5669.225	38597.366	35664.003	26340.460
2006	40963.985	6356.768	40150.352	36172.055	27581.869

2007	41962.720	7225.381	41437.349	36767.201	29034.430
2008	42163.408	7882.513	41955.675	36383.679	29682.987
2009	42191.741	8581.698	39704.532	34451.232	29764.695
2010	42600.518	9442.823	41468.201	35883.030	31632.138
2011	43098.706	10290.469	43095.965	35775.296	32546.754
2012	43976.171	11048.557	43198.953	36389.622	33153.946
2013	44203.390	11851.874	43266.734	37181.400	34047.826
2014	44714.029	12651.050	44042.441	37383.292	34918.163
2015	45156.956	13457.072	44423.479	37882.777	35710.257
2016	45644.770	14279.285	45049.034	38118.739	36616.990
2017	46022.614	15163.301	45987.537	38922.396	37620.920
2018	46543.938	16097.764	46549.522	39316.947	38467.013
2019	46601.002	17027.479	46765.482	39763.142	39059.699
2020	46910.058	17963.339	47344.212	40085.994	39764.342
2021	47368.825	18972.324	48055.193	40420.033	40691.487
2022	47886.352	20007.671	48733.266	40798.183	41699.381
2023	48396.625	21087.497	49378.144	41187.864	42723.827
2024	48897.415	22212.925	50005.011	41593.440	43791.582

weo-country	MYS	PHL	SGP	TWN	USA
original_period					
1980	7792.239	4389.861	20932.437	8062.540	29135.978
1981	8132.231	4428.330	22107.684	8479.933	29577.300
1982	8400.872	4476.556	22661.098	8732.327	28767.383
1983	8709.122	4448.346	24282.294	9382.795	29813.851
1984	9156.515	4022.880	25922.569	10174.725	31692.731
1985	8792.696	3638.909	25725.990	10528.760	32722.011
1986	8656.637	3673.865	26096.054	11623.696	33548.731
1987	8891.310	3738.799	28482.292	12956.791	34400.814
1988	9538.321	3902.871	30896.373	13834.839	35513.421
1989	10156.633	4048.499	33050.389	14894.372	36472.583
1990	10611.321	4076.491	34911.759	15546.490	36750.105
1991	11449.078	3960.626	36201.908	16678.877	36225.752
1992	12126.274	3883.085	37463.030	17891.227	37007.317
1993	12963.213	3874.205	40714.240	18932.766	37533.546
1994	13777.590	3952.223	43835.808	20176.270	38574.626
1995	14736.726	4042.189	45586.338	21307.318	39144.206
1996	15797.614	4070.816	47040.957	22446.538	40150.403
1997	16529.366	4186.980	49272.307	23580.578	41437.687
1998	14926.152	4061.745	46580.977	24365.377	42792.733
1999	15442.993	4098.281	48855.283	25809.216	44317.452
2000	16363.857	4277.398	52356.669	27239.092	45639.656
2001	16020.013	4313.263	50418.448	26741.174	45623.492
2002	16471.468	4376.410	51916.190	28087.456	45966.325
2003	17016.925	4504.164	55077.181	29136.441	46840.039
2004	17760.717	4709.214	59732.992	30916.390	48179.412
2005	18241.829	4835.821	62638.987	32474.474	49412.962

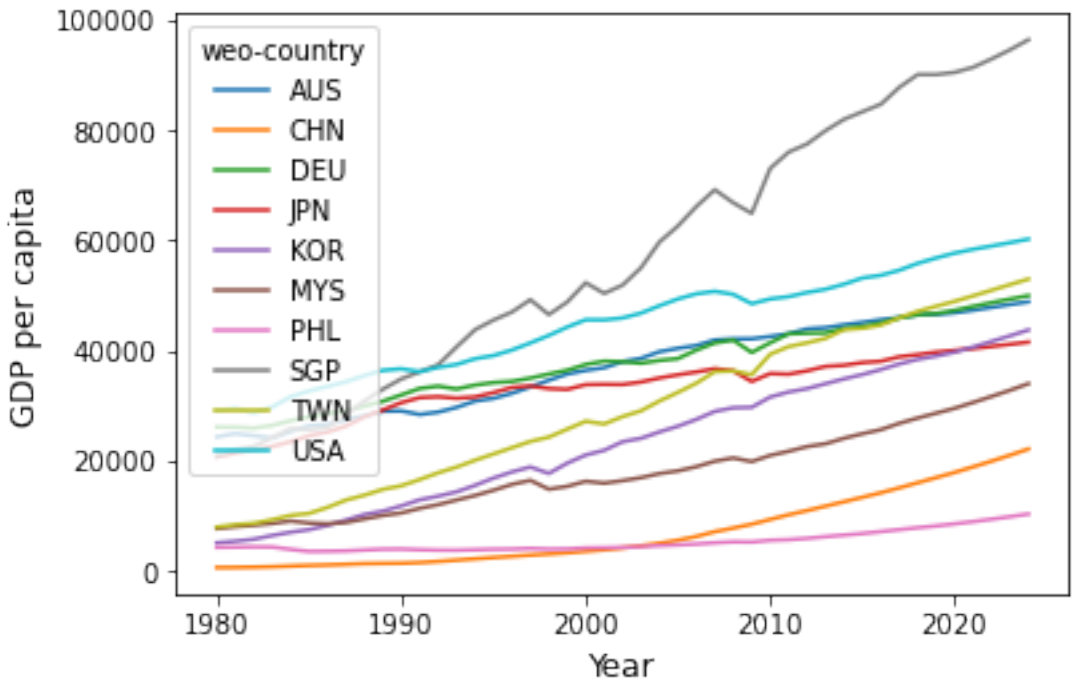
2006	19005.981	4989.294	66176.094	34141.512	50345.008
2007	20003.010	5215.082	69202.697	36236.865	50784.459
2008	20589.705	5387.319	66842.187	36367.406	50246.591
2009	19929.964	5354.592	64934.835	35669.557	48548.842
2010	21050.538	5664.932	73060.985	39389.435	49413.689
2011	21803.797	5773.739	76034.334	40777.485	49825.501
2012	22648.131	6011.558	77492.674	41456.757	50585.879
2013	23158.914	6325.439	79919.307	42265.232	51164.969
2014	24154.422	6600.551	81965.406	43851.727	52080.286
2015	24975.454	6885.166	83341.621	44095.599	53209.057
2016	25717.849	7239.392	84704.328	44671.478	53696.480
2017	26864.109	7599.147	87760.417	45985.546	54614.014
2018	27822.984	7946.378	90091.465	47161.391	55864.784
2019	28705.922	8268.327	90080.158	48084.877	56844.313
2020	29590.722	8641.177	90474.196	48977.491	57719.933
2021	30649.139	9048.868	91392.019	49965.647	58412.121
2022	31730.698	9482.478	92909.797	50984.763	59004.145
2023	32856.754	9941.497	94568.709	52006.275	59614.320
2024	34053.455	10420.907	96398.631	53012.747	60254.524

4 Visualize data

[9]:
```
# There is a current bug in MATPLOTLIB when used with Pandas
# I'll disable the related annoying warning for now
# See Open Issue: https://github.com/pandas-dev/pandas/issues/35684
import warnings
warnings.filterwarnings('ignore')
```

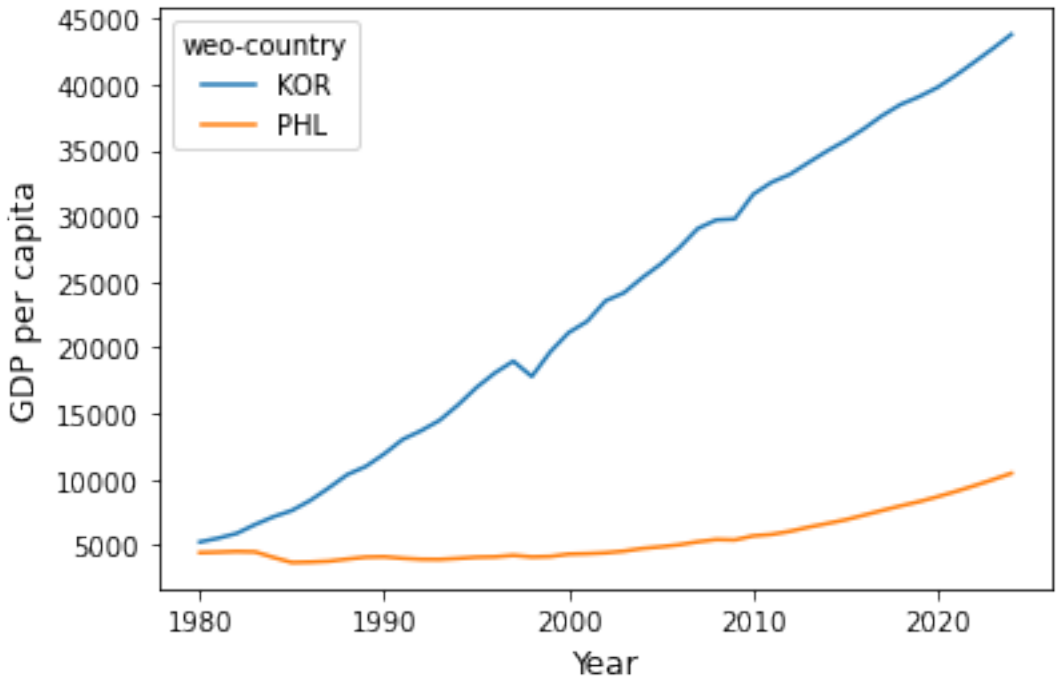
Let's visualize all the data we have in `df`:

[10]:
```
ax = df.plot(kind='line')
ax.set_xlabel('Year', fontsize=12)
ax.set_ylabel('GDP per capita', fontsize=12)
plt.show()
```



You can also cherry pick which one you want to plot:

[11]:
```
# Korea vs Phillipines ...
ax = df[['KOR','PHL']].plot(kind='line')
ax.set_xlabel('Year', fontsize=12)
ax.set_ylabel('GDP per capita', fontsize=12)
plt.show()
```



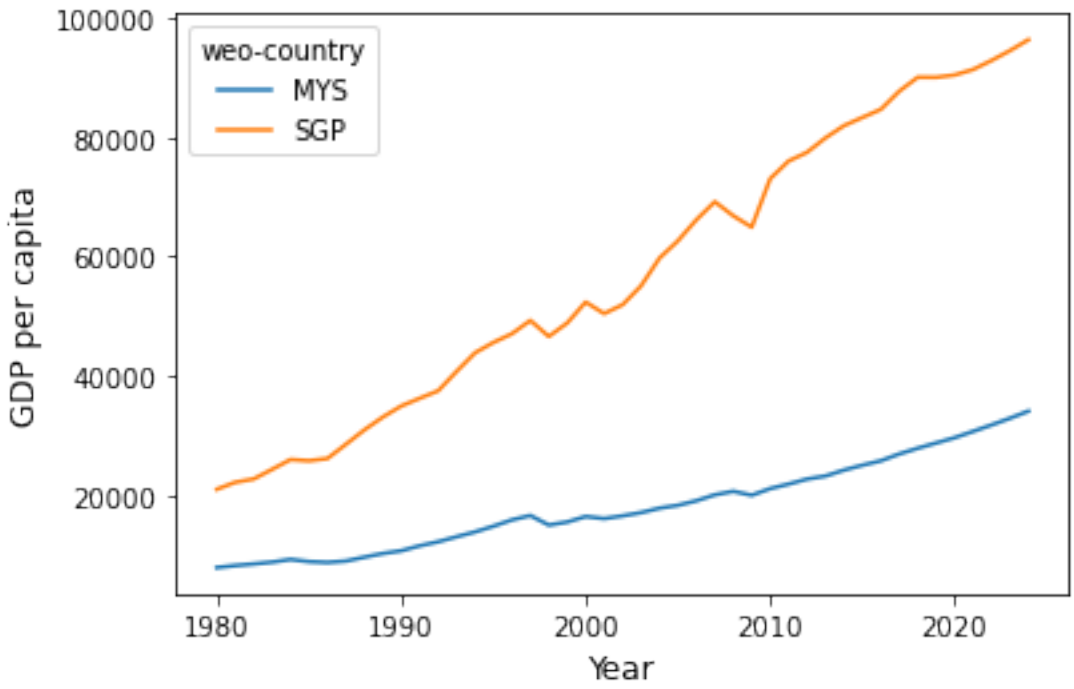
Pause and think …

What can we say about the trajectories of GDP per capita, when we compare

- Korea and the Phillipines?

Hint: Observe that in the initial observations, they are quite "close". What happened over time? What do you think might have *caused* their different paths over time?

[12]:
```
# Malaysia vs Singapore ...
ax = df[['MYS','SGP']].plot(kind='line')
ax.set_xlabel('Year', fontsize=12)
ax.set_ylabel('GDP per capita', fontsize=12)
plt.show()
```



Pause and think …

What can we say about the trajectories of GDP per capita, when we compare

- Malaysia and Singapore?

Hint: Observe that in the initial observations, they are quite "close". What happened over time? What do you think might have *caused* their different paths over time?