```
1 import pandas as pd
2 import numpy as np
3 import json
```

```
1
```

```
1 from google.colab import drive
2 drive.mount('/content/drive')
3
4 %cd "drive/My Drive/DA1"
```

```
Drive already mounted at /content/dri
[Errno 2] No such file or directory:
/content/drive/My Drive/DA1
```

> Load the modules.

```
1 df = pd.read_csv('API_EG.USE.PCAP.KG.OE_DS2_en_csv_v2_4028587.csv',skiprows=4
```

> Read in the data into a pandas dataframe.
>
> ```
> You need to skip 4 lines from
> ```

```
1 df.info(verbose=False)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Columns: 67 entries, Country Name to
dtypes: float64(63), object(4)
memory usage: 139.4+ KB
```

F **Fruzsina Kulcsár**
07:28 Today

We usually put such comments before the code to which it corresponds.

F **Fruzsina Kulcsár**
07:27 Today

I wouldn't set verbose=False, because it's good to check the columns..

F **Fruzsina Kulcsár**
07:27 Today

How do you know t this point?

> Display the usual basic information (datatypes, names, non-null values for columns) about the dataframe.
>
> ```
> Every row represent a county a
> ```

```

```

```
1 pd.set_option("display.max_columns",100)
2 pd.set_option("display.max_rows",50)
3 df
```

Fruzsina Kulcsár
07:32 Today
(edited 07:32 Today)

Did you check the result? It is not what you wanted here, just the df is displayed. The usage of the pd.set_option would look like this:

with pd.option_context("display.max_rows", len(dfmain), "display.max_columns", dfmain.shape[1]):
display(dfmain)

	Country Name	Country Code	Indicator Name	
0	Aruba	ABW	Energy use (kg of oil equivalent per capita)	EG
1	Africa Eastern and Southern	AFE	Energy use (kg of oil equivalent per capita)	EG
2	Afghanistan	AFG	Energy use (kg of oil equivalent per capita)	EG
3	Africa Western and Central	AFW	Energy use (kg of oil equivalent per capita)	EG
4	Angola	AGO	Energy use (kg of oil equivalent per capita)	EG
...	...	...	...	
261	Kosovo	XKX	Energy use (kg of oil equivalent per capita)	EG
262	Yemen, Rep.	YEM	Energy use (kg of oil equivalent per capita)	EG
263	South Africa	ZAF	Energy use (kg of oil equivalent per capita)	EG
264	Zambia	ZMB	Energy use (kg of oil equivalent per capita)	EG
265	Zimbabwe	ZWE	Energy use (kg of oil equivalent per capita)	EG

266 rows × 67 columns

Fruzsina Kulcsár
07:36 Today
(edited 07:38 Today)

Not this column should be renamed, but the column indices' name should get this name.

Fruzsina Kulcsár
07:38 Today

And please display the result all the time. How did you check that the renaming happened otherwise?

> Display the dataframe.
>
> ```
> Because the terminal has limit
> ```

```
```

```
1 df["Indicator Name"].unique() , len(df["Indicator Name"].unique())
```

```
(array(['Energy use (kg of oil
equivalent per capita)'],
dtype=object), 1)
```

> Find out what values occur under "Indicator Name".
>
> ```
> Get the unique value in column
> There is just one unique value
> ```

```
```

```
1 df.rename(columns={"Indicator Name" : df["Indicator Name"][0]},inplace=True)
```

> Set the name of the columns to the only value you found under "Indicator Name" (the columns and their labels remain the same!). (Apply the operation to your dataframe!)

```
dfcode = pd.DataFrame(df,columns=["Country Name","Country Code"])
dfcode
```

	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

> Create a new dataframe dfcode which contains the "Country Code" and "Country Name" columns of your original dataframe. You will continue working with your original dataframe, in what follows, though.

```
df = df.drop(columns=["Country Code","En
```

a

Fruzsina Kulcsár
07:41 Today

I cannot see all the years in your list.
There is a built-in function for this task, the describe function does exactly this. Try the following:

df.describe().T

> Remove the "Country Code", "Indicator Name", "Indicator Code" columns. (Apply the operation to your dataframe!)
>
> ```
> Get rid of unnecessary columns
> ```

```

```

```
1 df.set_index("Country Name",inplace=True)
```

> Make the column "Country Name" the index of your dataframe. (Apply the operation to your dataframe!)
>
> ```
> Since the value of the "Countr
> ```

```

```

F **Fruzsina Kulcsár**
07:45 Today

Again, it would be good to display your results somehow. This way I cannot check the results. Although probably your results are OK, this is not an elegant solution, not automated.

You should apply .describe again for the previous result of (df.describe().T).

F **Fruzsina Kulcsár**
07:48 Today

Good!
But for dynamic programming, try to write df.columns[-1] instead of 2015. Next time probably the last year will be different, and then the dynamic code will work, the manual not.

```
df = df.dropna(axis = "columns", how ='all')
df
```

Country Name	1960	1961	1962	1963	19
Aruba	NaN	NaN	NaN	NaN	Na
Africa Eastern and Southern	NaN	NaN	NaN	NaN	Na
Afghanistan	NaN	NaN	NaN	NaN	Na
Africa Western and Central	NaN	NaN	NaN	NaN	Na
Angola	NaN	NaN	NaN	NaN	Na
...	...	...	...	...	
Kosovo	NaN	NaN	NaN	NaN	Na
Yemen, Rep.	NaN	NaN	NaN	NaN	Na
South Africa	NaN	NaN	NaN	NaN	Na
Zambia	NaN	NaN	NaN	NaN	Na
Zimbabwe	NaN	NaN	NaN	NaN	Na

266 rows × 56 columns

> Get rid of columns which contain no values. (Apply the operation to your dataframe!)

```
There is no numerical values for some of y
```

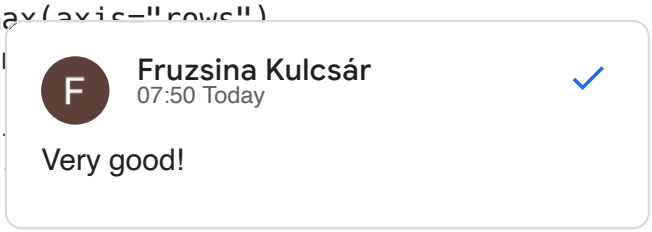
```
df_min_years = df.loc[:,"1960":"2015"].min(axis="rows")
df_max_years = df.loc[:,"1960":"2015"].max(axis="rows")
df_mean_years = df.loc[:,"1960":"2015"].
df_median_years = df.loc[:,"1960":"2015"
df_std_years = df.loc[:,"1960":"2015"].s
df_median_years.sort_values(ascending=Fa
```

```
2015    3746.158627
1970    3551.313507
1969    2982.833004
1968    2802.166121
1967    2575.607415
dtype: float64
```

> Display the basic statistics by year, incl. minimal, maximal, mean, median, standard deviation values. (Make sure you display these values for all the years!)

> ***What can you conclude from the yearly descriptives? Answer in a markdown cell.***
>
> ```
> 2006 has the lower value and 1
> ```

```




```
df_min_country = df.min(axis="columns")
df_max_country = df.max(axis="columns")
df_mean_country = df.mean(axis="columns")
df_median_country = df.median(axis="columns")
df_std_country = df.std(axis="columns")
```

> Display the basic statistics of your previous by country-statistics (i.e., the mean of country means, etc.).

```
If you define axis as rows it calculate sta
```

```

```
df_min = df_min_country.min()
df_max = df_max_country.max()
df_mean = df_mean_country.mean()
df_median = df_median_country.mean()
df_std = df_std_country.std()
```

> Display the basic statistics by country, incl. minimal, maximal, mean, median, standard deviation values. (Make sure you display these values for all the countries!)

```
df.sort_values(by='2015', ascending = Fa
```

Country Name	1960	1961
Iceland	3082.711563	2916.706232
Canada	4251.435911	4307.820754
North America	5516.355617	5494.086457
United States	5641.740755	5612.079503
Luxembourg	10523.406695	10534.018211
Finland	2196.953067	2252.778690
Norway	1906.174930	1937.644745
Australia	3063.554271	3115.787084
Korea, Rep.	NaN	NaN
Sweden	2698.792303	2742.123469
Post-demographic dividend	2812.288505	2848.671622
Belgium	2519.497320	2570.815623
High income	2761.770337	2799.712478
New Zealand	1685.788431	1763.259908
Netherlands	1825.934253	1879.150201

F **Fruzsina Kulcsár**
07:52 Today

Exactly!

F **Fruzsina Kulcsár**
07:57 Today

You didn' find those countries who have None for all the years.

dfmain[dfmain.isna().all(1)].index.tolist()

Estonia	NaN	NaN
OECD members	2666.979333	2702.066693
Czech Republic	NaN	NaN
Germany	1952.588632	1994.324633
Austria	1546.261468	1554.034906
France	1699.250872	1745.201546
Japan	867.203098	971.998193
Euro area	1403.942741	1455.391774
European Union	1487.664770	1541.096121
Slovenia	NaN	NaN
Slovak Republic	NaN	NaN
Switzerland	1398.654836	1454.755116
Ireland	1318.812487	1396.466152
Denmark	1922.973673	2023.308390
Israel	NaN	NaN

> Display the first thirty rows of your original dataframe after sorting by the values for the **last year** in **descending** order. (Do not apply sorting to your original dataframe, just display!)

> ***What are some of the conclusions you can draw from the data you displayed (e.g., about the "countries" in the data)?***

```
Some of counries has small dat
```

```

```

```
df["difference"] = df["2010"].fillna(0) - df["2000"].fillna(0)
df.sort_values(by=['difference'], ascending=False,inplace = False).head(30)
```

Country Name	1960	1961
Trinidad and Tobago	NaN	NaN
Iceland	3082.711563	2916.706232
Oman	NaN	NaN
Saudi Arabia	NaN	NaN
Gabon	NaN	NaN
Montenegro	NaN	NaN
Kazakhstan	NaN	NaN
Caribbean small states	NaN	NaN
Kuwait	NaN	NaN
Brunei Darussalam	NaN	NaN
Turkmenistan	NaN	NaN
Norway	1906.174930	1937.644745

Fruzsina Kulcsár
08:00 Today

You forget to put a negation in the whole thing.

"where the "Country Code" value is NOT in the "iso3" column"

China	NaN	NaN
Korea, Rep.	NaN	NaN
Gibraltar	NaN	NaN
Iran, Islamic Rep.	NaN	NaN
Estonia	NaN	NaN
Late-demographic dividend	NaN	NaN
East Asia & Pacific (IDA & IBRD countries)	NaN	NaN
East Asia & Pacific (excluding high income)	NaN	NaN
Upper middle income	NaN	NaN
East Asia & Pacific	NaN	NaN

> Calculate the difference between the (0-filled) values for the **years 2010 and 2000** by country, sort it in a **descending** order, and display the **first 30 rows**. What are some of the conclusions you could draw from what you see?

```
With fillna(0) command we replace none valu
```

Kosovo	NaN	NaN

```
mean_year = df.loc[:,"1960":"2015"].mean(axis="rows")
max_year = df.loc[:,"1960":"2015"].max(axis="rows")
percentage = (mean_year/max_year)*100
percentage
```

```
1960    22.669063
1961    22.978148
1962    24.208922
1963    25.378166
1964    24.790614
          ...
2011    13.533453
2012    13.845241
2013    13.285970
2014    13.778170
2015    24.189451
Length: 56, dtype: float64
```

> Display the **mean** values by year as a percentage of the **maximal** values for that year (e.g., if the mean value for 1965 were 200, and the maximal for 1965 were 400, then for 1965, you should be displaying 50).

```
df[df.loc[:,"1960":"2015"].isna()].index
```

```
Index(['Aruba', 'Africa Eastern and
Southern', 'Afghanistan',
       'Africa Western and Central',
'Angola', 'Albania', 'Andorra',
       'Arab World', 'United Arab
Emirates', 'Argentina',
       ...
       'Virgin Islands (U.S.)',
'Vietnam', 'Vanuatu', 'World',
'Samoa',
       'Kosovo', 'Yemen, Rep.',
'South Africa', 'Zambia',
'Zimbabwe'],
      dtype='object', name='Country
Name', length=266)
```

F **Fruzsina Kulcsár**
08:14 Today ✓

You had several mistakes and most of them were because of some misunderstanding, please be more careful and ask next time.
Also, try to pay attention to automation (e.g. dynamic programming - using functions and not just checking things manually).
What was an even bigger problem, is that you hardly write any reflection. This is NOT enough! Next time this won't be enough, so please try to interpret the results, and write about them. E.g. check the tendency of the numbers, try to emphasize which country was the best/worth, etc.
And one more thing: Your code was not runned, the results were not displayed. The notebook and the pdf you created should also contain the displayed results next time!!!

> Display the countries (not the subdataframe, just the countries!) where the values for *all* years are missing.

```
1 df.loc[df["2015"]>mean_year["2015"],:]
```

Country Name	1960	1961
Australia	3063.554271	3115.787084
Belgium	2519.497320	2570.815623
Canada	4251.435911	4307.820754
Finland	2196.953067	2252.778690
High income	2761.770337	2799.712478
Iceland	3082.711563	2916.706232
Korea, Rep.	NaN	NaN
Luxembourg	10523.406695	10534.018211
North America	5516.355617	5494.086457
Netherlands	1825.934253	1879.150201
Norway	1906.174930	1937.644745
New Zealand	1685.788431	1763.259908
Post-demographic dividend	2812.288505	2848.671622
Sweden	2698.792303	2742.123469
United States	5641.740755	5612.079503

> Display the subdataframe with the countries whose final year's values are bigger than the **final** year values' mean.

```
with open("capital.json", "r") as file:
    capitaldf = json.load(file)
capitaldf = pd.DataFrame(capitaldf["data"],columns=["name","capital","iso2","
capitaldf
```

	name	capital	iso2	iso3
0	Afghanistan	Kabul	AF	AFG
1	Aland Islands	Mariehamn	AX	ALA
2	Albania	Tirana	AL	ALB
3	Algeria	Algiers	DZ	DZA
4	American Samoa	Pago Pago	AS	ASM
...	...	...	...	...
246	Wallis and Futuna	Mata Utu	WF	WLF
247	Western Sahara	El-Aaiun	EH	ESH
248	Yemen	Sanaa	YE	YEM
249	Zambia	Lusaka	ZM	ZMB

> Load the "capital.json" file, and arrive at a dataframe capitaldf containing the **name, capital, iso2, iso3** information (these three should end up as the columns) for every country in it.

```
dfnocode = dfcode.loc[dfcode["Country Code"].isin(capitaldf["iso3"]),:]
dfnocode
```

	Country Name	Country Code
0	Aruba	ABW
2	Afghanistan	AFG
4	Angola	AGO
5	Albania	ALB
6	Andorra	AND
...	...	...
261	Kosovo	XKX
262	Yemen, Rep.	YEM
263	South Africa	ZAF
264	Zambia	ZMB
265	Zimbabwe	ZWE

216 rows × 2 columns

> Find the subdataframe of the dfcode dataframe you created earlier on where the "Country Code" value is **not** in the "iso3" column of capitaldf. Hint: look up and use the .isin pandas Series method. Store this subdataframe in the variable dfnocode.

> ***Display dfnocode and explain in a markdown cell what you see and what this could be used for.***
>
> ```
> There are the name of each cou
> ```

```
df.loc[df.index.isin(dfnocode["Country Name"]) == False,:]
```

Country Name	1960	1961	
Africa Eastern and Southern	NaN	NaN	
Africa Western and Central	NaN	NaN	
Arab World	NaN	NaN	
Central Europe and the Baltics	NaN	NaN	
Channel Islands	NaN	NaN	
Caribbean small states	NaN	NaN	
East Asia & Pacific (excluding high income)	NaN	NaN	
Early-demographic dividend	NaN	NaN	
East Asia & Pacific	NaN	NaN	
Europe & Central Asia (excluding high income)	NaN	NaN	
Europe & Central Asia	NaN	NaN	
Euro area	1403.942741	1455.391774	1
European Union	1487.664770	1541.096121	1
Fragile and conflict affected situations	NaN	NaN	
High income	2761.770337	2799.712478	2
Heavily indebted poor	NaN	NaN	

countries (HIPC)		

> Now display the subdataframe of your main dataframe with the yearly data where the index (the Country Name) is not in the "Country Name" column of dfnocode.

```
1
```

Latin America & Caribbean (excluding high income)	NaN	NaN
Latin America & Caribbean	NaN	NaN
Least developed countries: UN classification	NaN	NaN
Low income	NaN	NaN
Lower middle income	NaN	NaN
Low & middle income	NaN	NaN
Late-		