Coronavirus World Data Analysis

- Remember to uncomment the line assigning the variable to your answer and don't change the variable or function names.
- Use copies of the original or previous DataFrames to make sure you do not overwrite them by mistake.

First of all, run the following cell to:

- import `pandas` with an alias of `pd`
- read a CSV containing the data to work with
- convert the `date` column to the `datetime` format
- create a DataFrame `df` containing the data for only 1st July 2020
- take a look at the first few rows of the DataFrame

In [ ]:
```
import pandas as pd

data = pd.read_csv('data/owid-covid-data.csv')
data['date'] = pd.to_datetime(data['date'])
df = data[data['date'] == '2020-07-01']

df.head()
```

`df` now has one row of data for each country with data present for July 1st 2020. However, it also has a row with a `location` of `World` which contains aggregated values for all countries.

Q1. Create a new DataFrame which is the same as `df` but with the `World` row removed.

Assign this new DataFrame to the variable `countries`; do not modify `df`.

In [ ]:
```
#countries
```

Q2. Check the shape of your DataFrame to confirm that `countries` has one row fewer than `df`:

In [ ]:
```
#print(df.shape, countries.shape)
```

In [ ]:
```
cols = ['continent', 'location', 'total_deaths_per_million']
```

Q3. Define a DataFrame based on the `countries` DataFrame, but which only contains the columns in `cols` (defined above) and assign this to a variable called `countries_dr`

Order this DataFrame by `total_deaths_per_million`, with the highest numbers at the top.

In [ ]:
```
#countries_dr
```

Q4. Using the `countries` DataFrame we created earlier, find the sum of `total_tests` for countries in `Africa`, assigning the result, *as an integer*, to `africa_tests`.

In [ ]:
```
#africa_tests
```

Q5. How many countries in Africa have no value recorded for the number of `total_tests`? Assign the result to `africa_missing_test_data`.

You may find the pandas `.isna()` method useful.

In [ ]:
```
#africa_missing_test_data
```

Q6. How many countries have a higher value for `total_tests` than the `United Kingdom`? Assign your answer to a variable called `countries_more_tests`.

Remember to work from the `countries` DataFrame rather than `df`. You should avoid modifying any existing DataFrames.

In [ ]:
```
#countries_more_tests
```

Q7. Create a DataFrame called `beds_dr` which is based on the `countries` DataFrame, but contains only the columns `hospital_beds_per_thousand` and `total_deaths_per_million`.

Your answer should only include rows where there are values present in both of these columns. *You may find the `.dropna()` method useful.*

In [ ]:
```

```

In [ ]:
```
#beds_dr
```

Q8. What is the average `total_deaths_per_million` for entries in `beds_dr` where `hospital_beds_per_thousand` is greater than the mean?

Assign the answer to `dr_high_bed_ratio`.

In [ ]:
```
# dr_high_bed_ratio
```

Q9. What is the average `total_deaths_per_million` for entries in `beds_dr` where `hospital_beds_per_thousand` is less than the mean?

Assign the answer to `dr_low_bed_ratio`.

In [ ]:
```
# dr_low_bed_ratio
```

Q10. Create a DataFrame called `no_new_cases` which contains only rows from `countries` with zero `new_cases`.

In [ ]:
```
#no_new_cases
```

Q11. Which country in `no_new_cases` has had the highest number of `total_cases`? Assign your answer to `highest_no_new`.

In [ ]:
```
#highest_no_new
```

Q12. What is the sum of the `population` of all countries which have had zero `total_deaths`?

Assign your answer to `sum_populations_no_deaths`. Your answer should be in millions, rounded to the nearest whole number, and converted to an integer.

In [ ]:
```
#sum_populations_no_deaths
```

Q13. Create a function called `country_metric` which accepts the following three parameters:

- a DataFrame (which can be assumed to be of a similar format to `countries`)
- a location (i.e. a string which will be found in the `location` column of the DataFrame)
- a string (which can be assumed to be a column (other than `location`) which will be found in the DataFrame)

The function should return only the value from the first row for a given `location` and `metric`. *You may find `.iloc[]` useful.*

In [ ]:
```
#def country_metric
```

Q.14 Use your function to collect the value for `Vietnam` for the metric `aged_70_older`, assigning the result to `vietnam_older_70`.

In [ ]:
```
#vietnam_older_70
```

Q.15 Create another function called `countries_average`, which accepts the following three parameters:

- a DataFrame (which can be assumed to be such as `countries`)
- a list of countries (which can be assumed to all be found in the `location` column of the DataFrame)
- a string (which can be assumed to be a column (other than `location`) which will be found in the DataFrame)

The function should return the average value for the given metric for the given list of countries.

In [ ]:
```
#def countries_average()
```

In [ ]:
```
g7 = ['United States', 'Italy', 'Canada', 'Japan', 'United Kingdom', 'Germany', 'Franc
```

Q16. Use your `countries_average` function to find out the average `life_expectancy` of countries in the `g7` list defined above. Assign the result to the variable `g7_avg_life_expectancy`.

In [ ]:
```
#g7_avg_life_expectancy
```

Q.17 Find the country with lowest value for `life_expectancy` in the `countries` DataFrame, and create a string which is formatted as follows:

'{country} has a life expectancy of {diff} years lower than the G7 average.'

Assign your string to the variable `headline` and ensure it is formatted exactly as above, with:

- {country} being replaced by the value in the `location` column of the DataFrame
- {diff} being replaced by a float **rounded to one decimal place**, of the value from the `life_expectancy` column subtracted from `g7_avg_life_expectancy`
- Please note that {diff} should be a positive value

In [ ]:
```
#headline
```