- -> notebooks from this lecture: https://github.com/ine-rmotr-curriculum/freecodecamp-intro-to-pandas
- -> more examples of modifying data
- ***-> create columns which are combinations of columns***
  - -> for example, GDP over capita
  - -> if you have a GDP column and a total capita column, then this one is the quotient of them
  - -> doing this with broadcasting operations
    - these operations are quick because hey use numpy
- ***-> you take a dataframe and extract a column from it***
  - -> you perform operations on that column
  - -> that generates another series, which can then the used as a column in the dataframe
- ***-> reading external data and plotting***
  - -> .read_csv <- this is a pandas function
    - -> importing data from an external source
    - -> we have the price of something, and it after it was increased
    - -> you can use this to inspect the data
    - -> tuning it to get to the right point
  - -> it can be xml, json
  - -> parsing an html page
  - -> in this example, he's imported a csv file and is inspecting the data
  - -> the method will automatically parse the csv
  - -> the process is to start tuning it to get to the right point
- ***-> using the read csv function***
  - -> there are different attributes we can use
    - -> look these up in the documentation, you don't have to remember them
    - -> the first row of the csv is the column names
  - -> the first row of the csv file was the column name, in this case
  - -> he changes the file sand then re-reads it
  - -> pandas is assuming that the first rows of the csv file are the columns
  - -> the csv file doesn't have a collumn name in this case
  - -> he is entering arguments into the .dread csv function, from the documentation
    - -> this is the header argument
  - -> showing you the first rows
  - -> the df.head method
  - -> df.info <- to return information (for example the number of datapoints we have)
    - -> df.head
    - -> df.tail <- for the first or last rows in the data frame
  - -> pd.todatetime <- to turn the column timestamps into an actual date
    - -> to set the index of the set equal to the timeframe
    - -> the timeframes are the dates which we have the bitcoin prices for
    - -> we have the value of bitcoin on the certain dates
    - -> using the .loc method to access the values directly from the indices
- ***-> if we want to turn the process into an automated script***
  - -> reading the csv file, strip the columns, turn them into daytime data and assign them indices
  - -> the read csv method can allow us to automate all of those methods
  - -> customising the behaviour of the code in four lines
  - -> read the csv, don't infer a header
  - -> setting it the column names
  - -> the first column is the index of that data
  - -> parsing it a date
- ***-> pandas plotting***
  - -> you can create plots with pandas
  - -> the plot method

        - -> this uses the matplot lib library
        - -> this uses the plug library
            - -> this is part of the standard pydata stack
        - -> they are graphs of the price of bitcoin over time, in comparison to another cryptocurrency
            - -> you can plot an entire data frame with two series on it
            - -> data cleaning and reading other files and information sources of data
            - -> getting data info the pipeline from Excel and SQL (next)
                - -> pronounced 'sequel'
- ***-> question***

What code would add a "Certificates per month" column to the certificates_earned DataFrame like the one below?

	Certificates	Time (in months)	Certificates per month
Tom	8	16	0.50
Kris	2	5	0.40
Ahmad	5	9	0.56
Beau	6	12	0.50

```
certificates_earned['Certificates'] /
certificates_earned['Time (in months)']

certificates_earned['Certificates per month'] = round(
    certificates_earned['Certificates'] /
    certificates_earned['Time (in months)']
)

certificates_earned['Certificates per month'] = round( <- This one
    certificates_earned['Certificates'] /
    certificates_earned['Time (in months)'], 2
)
```