Project Description

To better understand the growth and impact of Bitcoin and other cryptocurrencies you will, in this project, explore the market capitalization of different cryptocurrencies.

Warning: *The cryptocurrency market is exceptionally volatile, and any money you put in might disappear into thin air. Never invest money you can't afford to lose.*

Project Tasks

1. Bitcoin and Cryptocurrencies: Full dataset, filtering, and reproducibility

Load the saved CSV file and select relevant columns.

- Load `datasets/coinmarketcap_06122017.csv` into a DataFrame named `dec6` using `read_csv()` from `pandas`.
- Select the columns `id` and `market_cap_usd` and assign them to `market_cap_raw`.
- Use `count()` to count and print the number of values in `market_cap_raw`.

This project uses **`pandas.DataFrame.plot()`** and the **`Axes`** API in matplotlib extensively, so these are good references to have open in a separate tab.

2. Discard the cryptocurrencies without a market capitalization

Filter out the coins with no known market capitalization.

- `query()` the DataFrame and filter out all the valueless coins and assign the new DataFrame to `cap`.
- Use `count()` again to count and print the number of values in `cap`.

Using **the `query()` method** of a DataFrame is a convenient alternative to using slicing selectors. For example, this:

```
df.query('value > 0')
```

Gives you the same result as this:

```
df[ df['value'] > 0 ]
```

but with less code.

Keep in mind that `query()` uses **numexpr** syntax by default instead of python syntax. It means that this:

```
(condition1 and condition2) or condition3
```

Should be written like this using numexpr:

```
(condition1 & condition2) | condition3
```

3. How big is Bitcoin compared with the rest of the cryptocurrencies?

Visualize the market capitalization of the top 10 cryptocurrencies.

- Select the first 10 coins, set the index to `id`, and assign the resulting DataFrame to `cap10`.
- Calculate the percentage of market capitalization for each coin using `assign()` and assign it to `cap10` again.
- Plot the top 10 coin's `market_cap_perc` in a barplot with the title "Top 10 market capitalization" and assign it to `ax`.
- Using the `ax` object, annotate the y axis with "% of total cap".

Check the pandas docs for **using assign with lambda** for calculating the % market cap. Remember that `.assign` iterates over all rows and creates a new column, but you can plug in numbers external to the DataFrame, for example:

```
cap.market_cap_usd.sum()
```

Also, don't forget to multiply by 100 inside the lambda to turn the resulting proportion into a percentage.

Pandas has an interface for every major plot type, for example `DataFrame.plot.hist()` and `DataFrame.plot.bar()`. For annotating the y axis using the `ax` object you could take a look at the available methods in the **matplotlib docs for the Axes object**.

4. Making the plot easier to read and more informative

Make the plot from the last task more informative with colors and a nice log scale.

- Make a plot like in the last task, but of `market_cap_usd`. Add the given `COLORS` and make the y-axis $\log_{10}$ scaled.
- Again, use the `ax` object to annotate the y axis with "USD".
- Remove the useless label on the x axis.

Scale the y axis using an argument to `.plot.bar()`, so it is only visual. Do not modify the actual value of the column!

5. What is going on?! Volatility in cryptocurrencies

Create a DataFrame that contains volatility information on cryptocurrencies.

- Select the columns `id`, `percent_change_24h`, and `percent_change_7d` from `dec6` and assign the resulting DataFrame to `volatility`.
- Set the index to `id` and drop all rows that contain NaNs.
- Sort `volatility` by `percent_change_24h` in ascending order.
- Print out the `.head()` of `volatility`.

6. Well, we can already see that things are *a bit* crazy

Make a bar plot that shows the biggest gainers and the biggest losers. Finish writing the function that will show the top losers to the left and the top gainers to the right.

- Use `.plot.bar()` to plot the "top losers" from `volatility_series` in `'darkred'` color.
- Set the figure main title using the `fig.suptitle()` method.
- Set the ylabel for the plot on the left using its `Axes` object
- Use `.plot.bar()` again to plot the "top winners" bar chart in `'darkblue'`
- Call the function`top10_subplot` with `volatility.percent_change_24h` and the supplied title.

The function assumes that `volatility_series` is sorted and so `volatility_series[:10]` would pick out the top 10 losers and `volatility_series[-10:]` would pick out the top 10 winners.

In this task, the subplot is already defined for you. To assign a pandas plot to a matplotlib subplot, you need to do the following

```
fig, axes = plt.subplots(...)
#assigns the resulting pandas plot to the first subplot
df1.plot.bar(ax=axes[0])
#assigns the resulting pandas plot to the second subplot
df2.plot.bar(ax=axes[1])
```

7. Ok, those are... interesting. Let's check the weekly Series too.

Call the function you created in the last task above, but with the weekly data.

- Sort `volatility` by `percent_change_7d` in ascending order and assign it to `volatility7d`.
- Call `top10_subplot` with volatility7d and the supplied title.

Keep in mind that our data is not sorted now and that `top10_subplot` assumes the Series is in ascending order.

8. How small is small?

- Use the `.query()` method to select all **large cap** coins in `cap`. That is, coins where `market_cap_usd` is +10 billion USD.
- Assign the resulting DataFrame to `largecaps`.
- Print out `largecaps`.

9. Most coins are tiny

Group *large*, *mid* and *small* cap coins into a group called *biggish* and make a barplot of counts of *biggish*, *micro* and *nano* coins.

- Count how many biggish, micro and nano coins there are using the given function `capcount`.

- Make a list with these 3 numbers and assign it to `values`.
- Make a barplot with `values` and the provided labels.

These are the market cap definitions from Investopedia:

- **Large cap**: +10 billion
- **Mid cap**: 2 billion - 10 billion
- **Small cap**: 300 million - 2 billion
- **Micro cap**: 50 million - 300 million
- **Nano cap**: Below 50 million

As `capcount` uses the `.query()` method the argument to `capcount` should be a string defining a condition for what values to select.

For this final task we will use the matplotlib `bar` interface, instead of `pandas`, as it is more convenient. Check the **matplotlib.pyplot.bar** docs for a reference.