Useful libraries
Projects

June 9, 2016

Before you start:
- create a suitable directory for this exercise
- download the zipped material from `http://www.physik.uzh.ch/~python/python/lecture7/`

To use git repositories to facilitate the collaborative work, please get the information from Nicola, Roman or Christian. The standard path to the repository is ...

Project 1: NYC Taxi

Targeted libraries: `csv`, `requests`, `datetime`, `bokeh`
New York's TLC (Taxi and Limousin Commission) publishes since 2010 data about every single trip of the yellow cabs – since their initialisation in 2013 also of the green boro taxis. The data is available as CSV files (one per month and vehicle category) from `http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml`. The data contain various information (time and date, length of the trip, pick-up/drop-off locations, payment method, tip). Analyse for example the taxi rides from and to the Bronx Zoo (Entrance coordinates: 40.843° N / 73.877° W) according to the time of the day, the weekdays or seasonal effects. Investigate for example differences in the tipping behaviour between the boroughs or as a function of the time of day and the weekday or as seasonal effects.
Hints: Download only a sample set to test your code and run the code over the whole data set online. If you need geographical information about coordinates (*e.g.* a gazeetter), you can use the `geopy` module.

Project 2: SNB API

Targeted libraries: `pandas`, `requests`, `datetime`, Visualisation
The Swiss National Bank (SNB) offers a large amount of economical and financial information. The corresponding data sheets are downloadable from `https://data.snb.ch/en` in CSV or Excel format. Unfortunately there is no direct API allowing to load the data directly from a Python script. Try to build such an API allowing to retrieve the data specified by the corresponding id code (*cf.* snbnomu for the number of banknotes and coins in circulation) and provide them as a timeseries. You can also build with object-oriented functionalities an SNB data specific container allowing the storing of metadata. On top of that try to use the API to build a data inspector allowing to get a first glance at the data.
Hints: You can find details about how the data can be retrieved as CSV at `https://data.snb.`

`ch/en/help`. The CUBE id is the mentioned id code. In addition You can find details about how the data can be retrieved as CSV

Project 3: Wikipedia-Geo-Crawler

Targeted libraries: `requests`, `BeautifulSoup`, `sqlite3`, `json`
Wikipedia entries – at least in the english edition –have standardised parts containing data in structurised format. The goal of this project is to build a crawler able to retrieve geographical as well as economical and sociological information about locations like cities, states, countries, etc. The data will be then stored in a format that is queriable. As a target information about geographical locations in the US could be used.
Hints: Study the HTML format of corresponding wikipedia entries. Identify from a few sample pages the extractable fields and think about how to store them in an efficient way (trade of between query and strucutre).

Project 4: Poor-man's Bloomberg

Targeted libraries: `csv`, `requests`, `pandas`, `BeautifulSoup`
The idea of this project is to build up a database about stock prices and analyst reviews. There are two subprojects to achieve this: On the one hand download historical stock quotes from *e.g.* Yahoo Finance `http://finance.yahoo.com` and provide them in the Pandas Timeseries format (Pandas has actually a direct API to Yahoo Finance, but as an exercise let's build the functionality from scratch). On the other hand download information about companies with traded stocks from `http://www.finanzen.ch`, especially information about analysts' opinion (*i.e.* targeted prices). This needs to be done by parsing the HTML code of the corresponding pages.
Combine the two samples to rate the qualitiy of the different analysis. Perform a benchmarking of the targeted share prices against the realised share price after a certain period posterior to the publication of the analysis.
Hints: For the Yahoo Finance API call `http://real-chart.finance.yahoo.com/table.csv?<parameter_1>=<value_1>&...&<parameter_n>=<value_n>` where the parameters are a: starting year, b: starting month (January = 0), c: starting day, d: final year, e: final month (January = 0), f: final day, g: frequency (d = day, w = week, m = month), s: ticker symbol. Study the HTML format of the corresponding analyses webpages to find out how to parse them and extract the target price. Focus on a few (3-5) large companies in the beginning, but design your code such that it is scalable and easy to be extended.

Project 5: RSS information feed gatherer

Targeted libraries: `requests`, `feeder`, `BeautifulSoup`, `sqlite`, Visualisation
RSS feed from media corporations allow to gather news items easily. But normally they do not carry the full text of news items. Use the modules to read RSS feed to obtain the metadata and use then the link to the news item to pbtain the full text. Analyse the news items in terms of term frequency or try to cluster the news items topically across the different sources.
Hints: Use a database to track already retrieved items and the discussed HTML parser to retrieve the full text.

Project X: Your own idea

In case you have another idea, feel free to team up and develop it. If you need additional module, you can install them locally with `pip --user`.