IU Internationale Hochschule

Written Assignment

Web scraping

Data Analyst - Python (B2G-UPS DPDAPE)

Task for Course: DLBDSDQDW01 – Data Quality and Data Wrangling

Date: 14.09.2023

Author: Boieru Mykola

Matriculation number: UPS10618765

Tutor: Sahar Qaadan

Table of content

Introduction

Web scraping is a popular and useful way to collect information from different web sites. Respectively, there are many ways to do it depending on data type, frequency of process and type of website.

On the one hand, web scraping allows to automate manual and routine tasks, but on the other hand, it can be illegal and can violate privacy of people and organizations. Restrictions and policies, that are often pointed by a webpage's owner, must be taken into consideration. Ignoring these private policies can provoke issues. That means this point should be researched in details before the process of web scraping will be started. Rules, that have to be followed to avoid issues, will be also researched in the further section.

It should be also found the most suitable approach to collect data. For that it is needed to analyse type of webpage and type of wanted data. Depending on if websites is static or dynamic, one of different approaches will be chosen and that chose will influence on the whole web scraping process.

After collecting data, it should be researched and found a way to store them. Often data should be stored for further decision because of that it is important to find the best way to do it. In addition, collected data should be visualized because it is worth nothing if it cannot be visualized for further analysis and presentation. Type of visualization format depends on type of data and how it will be used in future. Some data is more understandable in table format and other should be presented in graphs.

The full code of web scraping program is available with the following link:

https://github.com/MykolaBoieru/Data-Quality-and-Data-Wrangling

1. Web scraping and its legal and ethics considerations

Nowadays data has a crucial importance for businesses, because it helps with building a business, making decisions and with analysing competitors and prices, as well as with creating and improving the strategy.

For realisation these goals every company should collect data. Routine tasks can be replaced by web scraping. Web scraping is a process of automatic collecting data from web sites. It is important also to check use policy of the site. Sites can forbid to use web scraping from some reasons. For example, sending to much requests can overload the site or some data can be protected. Because of that, web scraping belong to grey area and web scraper should always check permission for automatic collecting data.

Krotov, V., and Silva, L discussed the problem of legality of web scraping in their research “Legality and ethics of web scraping”. They pointed out the following aspects that have to be considered by web scraping (Krotov, V., & Silva, L.,2018):

1. Consideration of restrictions. To know if there are any restrictions to scrape the data from a web sites, robots.txt file should be read and considered. Restrictions that are in this file have to be followed to avoid gathering forbidden information and to avoid possible issues.
2. Individual privacy and rights of research subjects have to be taken into consideration. Collecting data, it is important not to violate users’ private data. For example, it can happen when collected data belong to personal or sensitive information about users and this data is used against the users. Sometimes users don’t know that their data can be used further, but they expected that data will not be given to third parties. For instance, buying flight ticket, users expect that their private data such as name, age or destination won’t be collected by third party.
3. Avoidance of discrimination. Collected data should not lead to discrimination practices. For example, previous user’s behaviour can be used for targeting services and products, but such data is not allowed to lead to different amount of services or goods because of gender, age, ethnicity etc. Web scraper should avoid such sequences and foresee such probability.
4. Organizational privacy has to be taken into consideration. As well as, users have right to privacy, organizations do the same. Web scraper must not reveal organization operations or confidential information, doing web scraping. For instance, data based on scraping recruitment websites can content information about organization structure of amount of employees.
5. Avoidance of diminishing value the organization. Web scrapers have to consider that a data product created on collected data from a competitor’s website can lead to financial loses of original data owner.
6. Importance of data quality and its impact on decision-making. If a data product can influence on organizations’ decisions, it has to be collected only from trustworthy websites. Sometimes

content created by users can contain false information. Respectively, data based on such content has a low quality and, when it is used for making decisions, can lead to false decisions that can provoke not only financial loses for company, but also reputational loses.

Abovementioned sequences have to be always considered providing web scraping activities and web scrapers that do this, have to understand a responsibility for quality of data and ways they were collected to avoid probable company risks and issues.

In addition to that, it is important to point out that if web scraping will be used on regular basis on concreate website, data owner should be informed, the website should not be overloaded with requests, collected data should be not violate users' privacy and source of data should be trustworthy.

2.Scraping numeric information

For the task were chosen three web pages with numeric information.

The first one contents the amount of total cases of COVID-19, amount of deaths from this disease and amount of recovered people:

https://www.worldometers.info/coronavirus/

The second one counts amount of Wikipedia articles in English, also this web page counts amount of words per article:

https://wikicount.net/

The third web page is financial site that among other thing, shows current price of bitcoin:

https://coinmarketcap.com/currencies/bitcoin/

First of all, it is need to import required packages and libraries. The following lines of code are similar for every scraped web page.

```
import requests
from bs4 import BeautifulSoup
import datetime as dt
```

Now every web page will be scraped separate because code is different for every of them. For getting HTML code of the first website, that contents information about Covid-19 cases, it is needed to use request and BeautifulSoap libraries. To load page and extract its code, the following code can be used:

```
page = requests.get('https://www.worldometers.info/coronavirus/')
soup = BeautifulSoup(page.content, 'html.parser')
```

After analysing the whole HTML code of the web page, it can be found where exactly is located interesting us information. For example, for all cases of Covid-19 in the world it is needed to find the first attribute ‘div’ with a class ‘maincounter-number’, then to find ‘span’-attribute and to get its text. It is needed also to delete all commas of that number to make it suitable for conversion into integer. The number after conversation should be put in ‘total_cases’ variable and converted to integer to enable further analyse.

```
case_one = soup.find_all('div', class_="maincounter-number")[0].find('span').text.replace(",", "")
total_cases = case_one.split()
total_cases = ''.join(total_cases)
total_cases = int(total_cases)
```

To find an amount of all death cases from Covid-19, it can be found the second ‘div’ attribute with the class ‘maincounter-number’. Then it is converted to integer again.

```
deaths = soup.find_all('div', class_="maincounter-number")[1].find('span').text.replace(",", "")
deaths = int(deaths)
```

In the same way can be found amount of recover cases. It is used the third ‘div’ attribute with class ‘maincounter-number’ and converted the text to integer.

```
recovered = soup.find_all('div', class_="maincounter-number")[2].find('span').text.replace(",", "")
recovered = int(recovered)
```

It is needed also the current date. For that can be used datetime package that was imported above. To connect scraped data with a date, the following line of code can be used.

```
insert_datetime = dt.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
```

The basic statistics about Covid is gathered. When needed data will be scraped from all three pages it can be stored in a database. With abovementioned manipulations the data can be scraped on everyday basis.

In the same way can be collected data from the second website. When all required packages are loaded, web page can be analysed and its content can be extracted.

```
page = requests.get('https://wikicount.net/')
soup = BeautifulSoup(page.content, 'html.parser')
```

Then it is needed to find a number of articles, for that it should be found attribute ‘div’ with a class ‘number’, then commas will be deleted and the number of articles will be converted into integer.

```
articles = soup.find('div', class_='number').text.replace(",", "")
articles = int(articles)
```

On the same page can be found a number of words per article. As it is shown below, this cannot be found with a class ‘number’, this time should be found the attribute ‘strong’ deeper in attribute ‘div’. When it is found, it can be converted into suitable format after deleting commas.

```
words = soup.find('div').find('strong').text.replace(",", "")
words = int(words)
```

By analysing collected data, a date of data can be useful too. As it has been done for the first website, in the same way it will done for the next webpages..

```
insert_datetime = dt.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
```

All numeric information was scraped from this web page, so we can go further to the third website to scrape actual prices for bitcoin. To scrape the data from the third page can be used the same libraires that were used before.

```
page = requests.get('https://coinmarketcap.com/currencies/bitcoin/')
soup = BeautifulSoup(page.content, 'html.parser')
```

In this case it is needed to scrape only one number – bitcoin price. As it has been done before with two previous websites, here can be also easily found a data. But in this time it should be deleted dollar symbol to be able to convert the price into float. It won't be converted the price into integer, because it would be useful to have a number after comma for more precision.

```
price = soup.find('div', class_='sc-16891c57-0 hqcKQB flexStart alignBaseline').find(
    'span', class_='sc-16891c57-0 dxubiK base-text').text.replace("$", "").replace(",", "")
price = float(price)
```

It would be also useful to know when exactly bitcoin has such price. For that can be used already familiar code.

```
insert_datetime = dt.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
```

It has been scraped information that was needed. But if it will be done on a daily basis, it has to be stored everywhere.

3. Saving collected data

When a data is collected, it should be stored for further manipulations. For example, it can be used SQL database, but in such case more suitable would be HDF5 files, because it allows to save data from every website separate. Also this format allows to have big data, for example, if data will be collected from those webpages during years. A database or a file, where collected data is stored, can become enormous and it would work very slow. So, HDF5 files are more suitable to work with big amount of data collected from different sources.

HDF5 files have hierarchical structure, that means inside of folders can be also folders or file. This structure is considered as more comfortable to store different files and different type of data. For the purpose of this task it will be created three groups of dataset and when data will be obtained, it will be saved as a new dataset in the same group.

First of all, it is needed to import some libraries and packages as well as earlier created modules.

```
import numpy as np
import corona
import articles
import bitcoin
import h5py
```

The further actions will be dependent on if the HDF5 file has been already created, in other words some part of code will be executed only once. If a file has been just created, there is should be an opportunity to append it, but if the file was created earlier, it has to be an opportunity to read and to write into a datasets. Because of that it should be used a structure ‘try-except’. After trying to create a new datasets, if they were already created, it has to be handled exceptions. Handling exceptions, web scraper can get access to earlier created datasets for storing new data. Then a file is being created and appended with mode ‘a’. It is important also to point out that we will use special type for writing date – string.

```
try:
    filename_hdf = 'collected_data.h5'
    with h5py.File(filename_hdf, 'a') as hdf:
        dt = h5py.special_dtype(vlen=str)
```

On the next step, should be created a group for data, scraped from the first website that contents data about Covid-19. Attributes for the group should be also set.

```
grp_corona = hdf.create_group("coronavirus")

grp_corona.attrs['SOURCE'] = 'Data collected from the website: ' \
                             'https://www.worldometers.info/coronavirus/'
grp_corona.attrs['DATA COLLECTION'] = 'Data collected through web scraping from Aug 24, 2023. '
```

Then datasets are created for this group. It can be done, using a method ‘create_dataset’ and then should be written a name of this dataset and its data, it should be also pointed out that this dataset will be appended with writing ‘None’ in ‘maxshape’. For storing collected data should be written that this data is being stored in a module, that was imported, and stored in variable in that module.

```
corona_datetime = np.array([corona.insert_datetime], dtype=dt)

corona_insert_datetime = grp_corona.create_dataset("insert_datetime", (1,), data=corona_datetime,
                                                   maxshape=(None,))
corona_total_cases = grp_corona.create_dataset("total_cases", (1,), data=corona.total_cases, maxshape=(None,))
corona_deaths = grp_corona.create_dataset("deaths", (1,), data=corona.deaths, maxshape=(None,))
corona_recovered = grp_corona.create_dataset("recovered", (1,), data=corona.recovered, maxshape=(None,))
```

In the same way it has to be done with the articles data. First of all, a group has to be created and attributes for this group have to be set.

```
grp_art = hdf.create_group("english_articles")

grp_art.attrs['SOURCE'] = 'Data on temperature was collected from the website: ' \
                          'https://wikicount.net/'
grp_art.attrs['DATA COLLECTION'] = 'Data collected through web scraping from Aug 24, 2023. '
```

For this group has to be created also datasets and implemented there collected data.

```
art_datetime = np.array([articles.insert_datetime], dtype=dt)
art_insert_datetime = grp_art.create_dataset("insert_datetime", (1,), data=art_datetime, maxshape=(None,))
art_amount = grp_art.create_dataset("amount_of_articles", (1,), data=articles.articles, maxshape=(None,))
art_words = grp_art.create_dataset("words_per_article", (1,), data=articles.words, maxshape=(None,))
```

For the third web page a group is being also created and appended with attributes.

```
grp_bitcoin = hdf.create_group("bitcoin")

grp_bitcoin.attrs['SOURCE'] = 'Data on temperature was collected from the website: ' \
                              'https://coinmarketcap.com/currencies/bitcoin/'
grp_bitcoin.attrs['DATA COLLECTION'] = 'Data collected through web scraping from Aug 24, 2023. '
```

For bitcoin price and date should be also created datasets.

```
btc_datetime = np.array([bitcoin.insert_datetime], dtype=dt)
btc_insert_datetime = grp_bitcoin.create_dataset("insert_datetime", (1,), data=btc_datetime, maxshape=(None,))
btc_price = grp_bitcoin.create_dataset("price", (1,), data=bitcoin.price, maxshape=(None,))
```

If datasets were created earlier, it will provoke an exception, because of that they should be handled. On the picture below, is shown a file that should be opened in ‘r+’ mode, that means that the file will be able to be both read and written.

```
except FileExistsError and ValueError:
    filename_hdf = 'collected_data.h5'
    with h5py.File(filename_hdf, 'r+') as hdf:
        dt = h5py.special_dtype(vlen=str)
```

Then it has to be an access to earlier created datasets to store a new data.

```
corona_deaths = hdf.get("coronavirus/deaths")
corona_recovered = hdf.get("coronavirus/recovered")
corona_insert_datetime = hdf.get("coronavirus/insert_datetime")
corona_total_cases = hdf.get("coronavirus/total_cases")
```

Then it should be added a new values after getting access to the datasets. First of all, it is needed to store a new data, then every dataset should be resized to be able to store not only old values, but also new ones. And then new value can be stored in last created space, using index.

```
corona_datetime = np.array([corona.insert_datetime], dtype=dt)

corona_insert_datetime.resize(corona_insert_datetime.shape[0] + 1, axis=0)
corona_insert_datetime[-1:, ] = corona_datetime

corona_total_cases.resize(corona_total_cases.shape[0] + 1, axis=0)
corona_total_cases[-1:, ] = corona.total_cases

corona_deaths.resize(corona_deaths.shape[0] + 1, axis=0)
corona_deaths[-1:, ] = corona.deaths

corona_recovered.resize(corona_recovered.shape[0] + 1, axis=0)
corona_recovered[-1:, ] = corona.recovered
```

In the same way it can be done with article data. Firstly, it should be got access to datasets.

```
art_amount = hdf.get("english_articles/amount_of_articles")
art_words = hdf.get("english_articles/words_per_article")
art_insert_datetime = hdf.get("english_articles/insert_datetime")
```

Here datasets should be also reshaped to be able to store new data.

```
art_datetime = np.array([articles.insert_datetime], dtype=dt)

art_insert_datetime.resize(art_insert_datetime.shape[0] + 1, axis=0)
art_insert_datetime[-1:, ] = art_datetime

art_amount.resize(art_amount.shape[0] + 1, axis=0)
art_amount[-1:, ] = articles.articles

art_words.resize(art_words.shape[0] + 1, axis=0)
art_words[-1:, ] = articles.words
```

In the same way can be stored new data about bitcoin prices. First of all, getting access to all created datasets.

```
btc_price = hdf.get("bitcoin/price")
btc_insert_datetime = hdf.get("bitcoin/insert_datetime")
```

As it has been already done, new values should be written into datasets, reshaping it.

```
btc_datetime = np.array([bitcoin.insert_datetime], dtype=dt)

btc_insert_datetime.resize(btc_insert_datetime.shape[0] + 1, axis=0)
btc_insert_datetime[-1:, ] = btc_datetime

btc_price.resize(btc_price.shape[0] + 1, axis=0)
btc_price[-1:, ] = bitcoin.price
```

After collecting and saving data in this way during several days, it can be prepared for further analysis with creating tables and graphs.

4. Collected Data Visualization

There are many ways to visualize collected data. An acceptable way is usually dependent on purpose and on type of data. For example, tables are suitable in cases when it is needed an amount of something on certain date and graphs can be useful when trends are more important. For both ways Python can be used. For this task it will be used the following libraries: pandas, numpy, plotly.express and h5py.First of all, it is needed to import them.

```
import pandas as pd
import h5py
import numpy as np
import plotly.express as px
```

Then it is needed to open the file where the data is stored. After that, there is access to the groups and its items.

```
with h5py.File('collected_data.h5', 'r') as hdf:

    base_items = list(hdf.items())
    grp_corona = hdf.get("coronavirus")
    grp_corona_items = list(grp_corona.items())
    grp_art = hdf.get("english_articles")
    grp_art_items = list(grp_art.items())
    grp_btc = hdf.get("bitcoin")
    grp_btc_items = list(grp_btc.items())
```

When getting access to the groups, it is needed also to get access to every single dataset. Firstly, for Covis-19 data. For such type of data table format will be used for visualisation.

```
corona_insert_datetime = hdf.get("coronavirus/insert_datetime")
corona_total_cases = hdf.get("coronavirus/total_cases")
corona_deaths = hdf.get("coronavirus/deaths")
corona_recovered = hdf.get("coronavirus/recovered")
```

For the further process it is needed to read datasets into arrays.

```
corona_insert_datetime = np.array(corona_insert_datetime)
corona_insert_datetime = corona_insert_datetime.astype('datetime64', copy=False)
corona_total_cases = np.array(corona_total_cases)
corona_deaths = np.array(corona_deaths)
corona_recovered = np.array(corona_recovered)
```

For creating table can be used pandas library and its in-built function DataFrame. For that it has to be read arrays into pandas dataframe. It should be given names of variables and names of columns.

```
cor_df_1 = pd.DataFrame(corona_insert_datetime, columns=['insert_datetime'])
cor_df_2 = pd.DataFrame(corona_total_cases, columns=['Amount of cases'])
cor_df_3 = pd.DataFrame(corona_deaths, columns=['Deaths'])
cor_df_4 = pd.DataFrame(corona_recovered, columns=['Recovered'])
cor_df = pd.concat([cor_df_1, cor_df_2, cor_df_3, cor_df_4], axis=1, join="inner")
```

Then the result can be finally seen, using the following code

```
print('COVID-19 DATA')
print(cor_df.to_markdown())
print(' ')
```

And the output looks as it is shown on the following figure.

```
COVID-19 DATA
|    | insert_datetime     |   Amount of cases |   Deaths |   Recovered |
|---:|:--------------------|------------------:|---------:|------------:|
|  0 | 2023-08-24 16:35:22 |         694122809 |  6910119 |   665939559 |
|  1 | 2023-08-25 15:27:29 |         694169485 |  6910402 |   665971109 |
|  2 | 2023-08-28 15:33:03 |         694215733 |  6910780 |   666201391 |
|  3 | 2023-08-29 15:05:39 |         694503032 |  6910964 |   666267958 |
|  4 | 2023-08-30 16:32:18 |         694531662 |  6911205 |   666328783 |
|  5 | 2023-08-31 15:31:22 |         694605890 |  6911454 |   666397860 |
|  6 | 2023-09-01 16:13:24 |         694630525 |  6911550 |   666436019 |
```

Having data in such format facilitates to analyse and to compare collected data.

For visualization data about English articles on Wikipedia will be also used table format. Firstly, it should be got each database.

```
art_articles = hdf.get("english_articles/amount_of_articles")
art_words = hdf.get("english_articles/words_per_article")
art_insert_datetime = hdf.get("english_articles/insert_datetime")
```

Secondly, it is needed to read datasets into arrays as we did it before.

```
art_articles = np.array(art_articles)
art_words = np.array(art_words)
art_insert_datetime = np.array(art_insert_datetime)
art_insert_datetime = art_insert_datetime.astype('datetime64', copy=False)
```

Then, arrays are read into pandas dataframe.

```
art_df_1 = pd.DataFrame(art_insert_datetime, columns=['insert_datetime'])
art_df_2 = pd.DataFrame(art_articles, columns=['Amount of articles'])
art_df_3 = pd.DataFrame(art_words, columns=['Words per article'])
art_df = pd.concat([art_df_1, art_df_2, art_df_3], axis=1, join="inner")
```

Now collected data can be printed, using the following code.

```
print('WIKIPEDIA ARTICLES IN ENGLISH')
print(art_df.to_markdown())
print(' ')
```

The created table shows how amount of articles and amount of words per article have changed during the period.

WIKIPEDIA ARTICLES IN ENGLISH

	insert_datetime	Amount of articles	Words per article
0	2023-08-24 16:35:22	6703021	1230
1	2023-08-25 15:27:30	6703558	2478
2	2023-08-28 15:33:04	6705362	1491
3	2023-08-29 15:05:40	6705720	2537
4	2023-08-30 16:32:18	6706405	1964
5	2023-08-31 15:31:22	6707171	2112
6	2023-09-01 16:13:25	6707774	1788

The difference between numbers in different days shows that amount of articles in English on Wikipedia has increased almost on five thousand during a week.

For visualization bitcoin prices it would be useful create not only a table, but also a graph that shows dynamics or trends how prices changed. In the beginning, it should be also got datasets.

```
btc_price = hdf.get("bitcoin/price")
btc_insert_datetime = hdf.get("bitcoin/insert_datetime")
```

Then, as it has been done twice, it should be read datasets into arrays for creating a table.

```
btc_price = np.array(btc_price)
btc_insert_datetime = np.array(btc_insert_datetime)
btc_insert_datetime = btc_insert_datetime.astype('datetime64', copy=False)
```

And as the last step for creating a table, it should be read arrays into pandas dataframe and then printed.

```
btc_df_1 = pd.DataFrame(btc_insert_datetime, columns=['insert_datetime'])
btc_df_2 = pd.DataFrame(btc_price, columns=['price in $'])

btc_df = pd.concat([btc_df_1, btc_df_2], axis=1, join="inner")

print('BITCOIN PRICES')
print(btc_df.to_markdown())
```

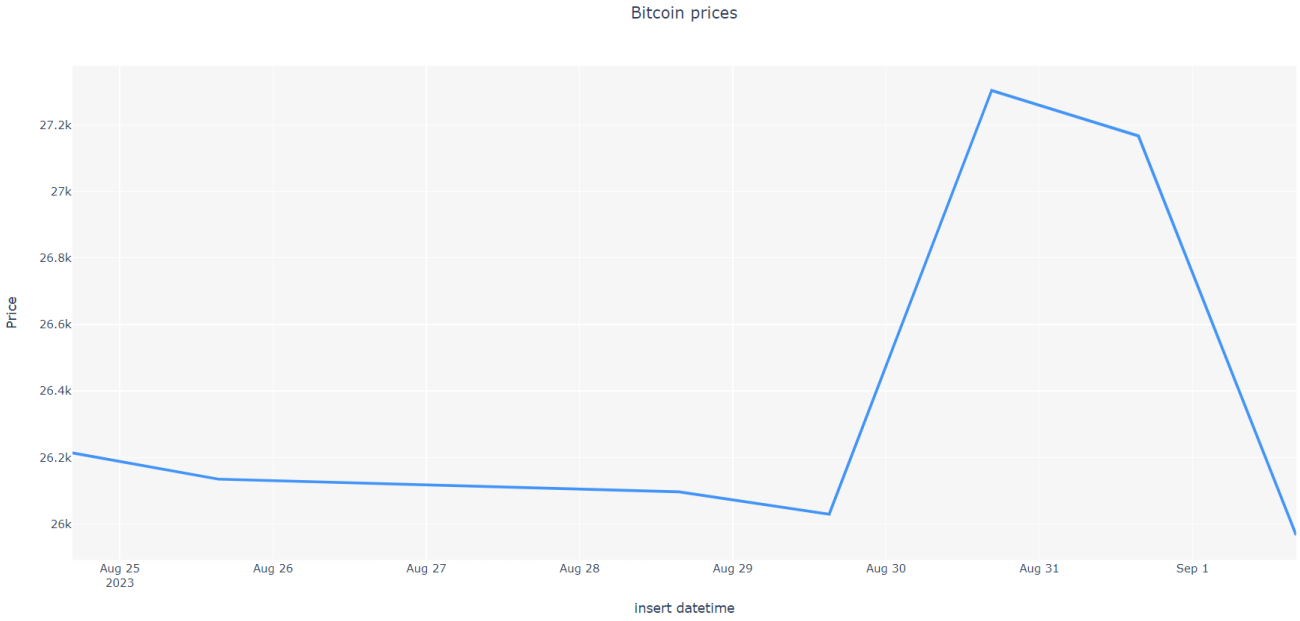
Table that was created is shown on the picture below.

```
BITCOIN PRICES
|    | insert_datetime     |   price in $ |
|---:|:--------------------|-------------:|
|  0 | 2023-08-24 16:35:22 |      26213.3 |
|  1 | 2023-08-25 15:27:30 |      26134.7 |
|  2 | 2023-08-28 15:33:04 |      26096.4 |
|  3 | 2023-08-29 15:05:40 |      26029   |
|  4 | 2023-08-30 16:32:18 |      27303.4 |
|  5 | 2023-08-31 15:31:23 |      27166.8 |
|  6 | 2023-09-01 16:13:25 |      25966.3 |
```

For this case it will be also created a graph in addition to the table. It will allow to see trends how prices have changed. For that plotly.express library will be used that was imported on the beginning. The name of graph is 'Bitcoin prices', it should be also written names for x-and y-axis. The whole code for creating graph, using plotly.express library, is below.

```
bitcoin_fig_temp = px.line(x=btc_df['insert_datetime'], y=btc_df['price in $'],
                           title='Bitcoin prices')
bitcoin_fig_temp.update_layout(title_x=0.5,
                               plot_bgcolor='#f6f6f6',
                               yaxis_title='Price',
                               xaxis_title='insert datetime')
bitcoin_fig_temp.update_traces(line_color="#0096FF", line_width=3)
bitcoin_fig_temp.show()
```

The output of the code will open a browser showing created graph. The output can be seen on the next picture.



Visualization is an important part of data analysis, because of its clarity and readability. Visualization has a big impact on decision-making process, because of that it has to be created reasonably and with understanding of responsibility.

Conclusion

As all three sites content numeric information, the same approach could be used for scraping information from every of them. It had to be used also datetime library to connect a data with a certain date when it was being collected. The three web sites were static and this fact defined the approach that was used.

It could be also created web spiders for automatically scraping data from web pages, but for this task it had to be scraped only during a few days, so it wouldn't have been appropriate.

Collected data should have been stored for further work with it. In real project data can be collected on regular basis during months and every time it can be analysed. In this task the data was saved in HDF5 file because such files allow to store data from different resources in different folders and datasets.

The data was also visualized using some Python libraries suitable for creating tables and graphs. For two groups of data was used a table format for visualization and for the third one was created a graph in addition to a table. In this case there was a price and that means that it can become higher and lower.

It is important to point out that, if there were more difficult and a bigger set of data, it would be needed also to do cleansing from outliers or unsuitable and false data. It is out of the scope of this task, because of that it wasn't researched different approaches and the most suitable one for this type of data.

List of Literature

1. Heydt, M., & Zeng, J. (2018). *Python Web Scraping Cookbook*(1st ed.). Packt Publishing.
2. Krotov, V., & Silva, L. (2018, August). Legality and ethics of web scraping. 24th Americas conference of information systems. Association for Information Systems
3. Matthes, E. Python Crash Course. *A hand-on, project-based introduction to programming*. San Francisco, No starch Press.
4. Shovic, J., & Simpson, A. (2021). *Python-All-in-One For Dummies* (2nd ed.). Wiley.
5. Sweigart, A. (2015). *Automate the boring stuff with python: practical programming for total beginners.* San Francisco, No Starch Press.