Как еще лучше парсить сайты

Александр Ольферук,
наставник

Яндекс Практикум

Цели
на консультацию

Первая часть – 30 минут

- Зачем, почему, какие сайты есть

Перерыв – 10 минут

Вторая часть – 50 минут

- Как реквестить и парсить?
- Мастер-класс

Зачем
парсить сайты?

Вы же аналитики, что за вопрос!

- Выкачать и посмотреть на данные в pandas иногда гораздо удобнее, чем ковыряться на сайте
- Возможность извлечь мета-данные: например, количество комментариев, их сентимент и т.д.
- Это вас развивает, как программиста
- Это офигенно!

Какие сайты парсить?

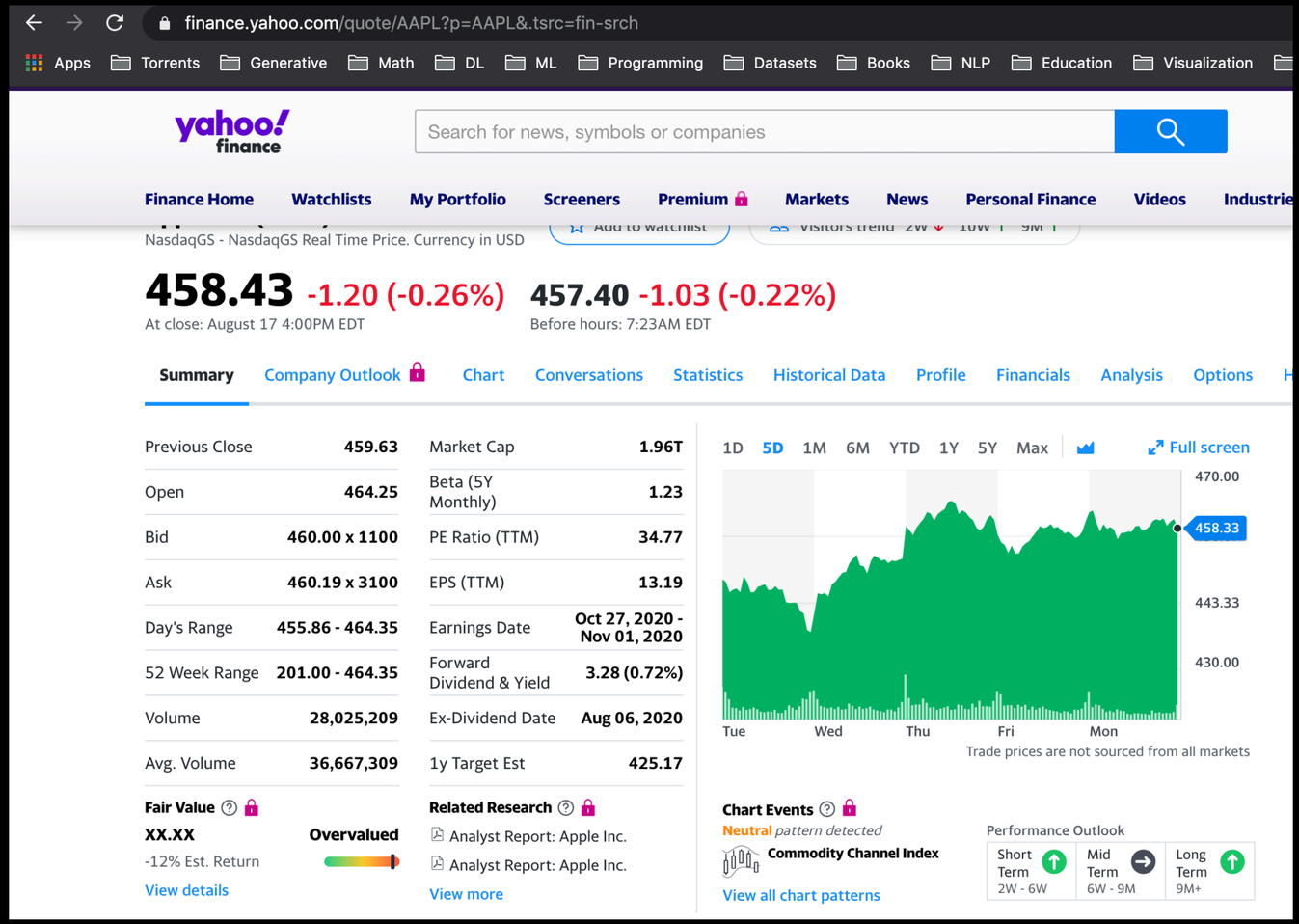
Убедитесь, что сайт не позволяет скачать архив

Убедитесь, что сайт не раздает API

API (программный интерфейс приложения) — описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой.

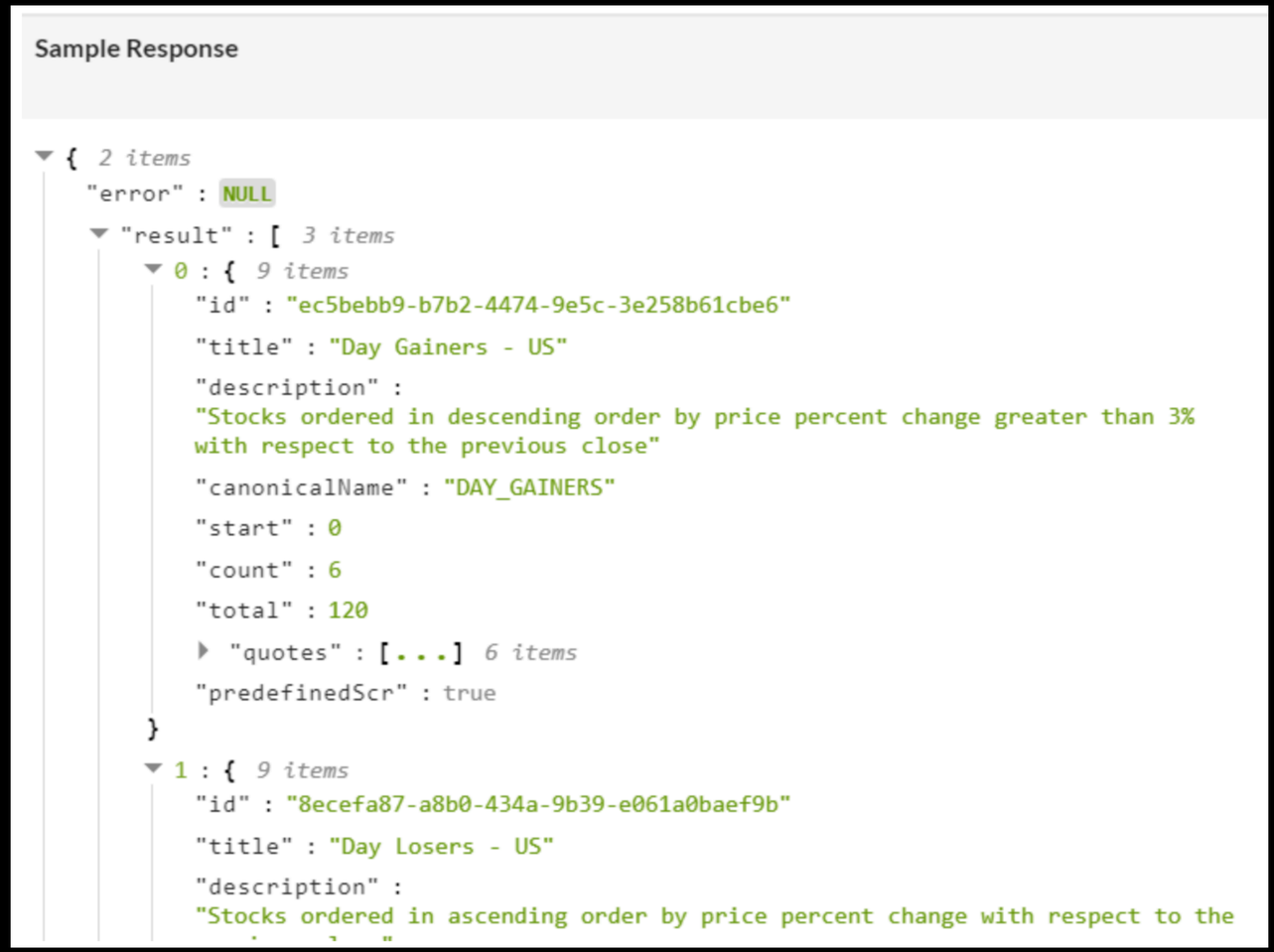
Убедитесь, что сайт не раздает API

Вместо того, чтобы парсить это...



Убедитесь, что сайт не раздает API

...попробуем загуглить “yahoo finance api” и найдем это:



Хороший совет №1

Гуглим *“<название сайта> python api”* или *“<название сайта> api”*,

прежде чем начинать погружение :)

Какие сайты бывают?

Структура представления данных

Пример

Статичная страница:

https://boardgameslv.com/2020/07/24/all-time-top-10-1/

Страница с закрытым API:

https://boardgamegeek.com/

Страница с открытым API:

https://www.boardgameatlas.com/api/docs

Особенности данных, передаваемых через API

В половине случаев это сформированные сервером куски html,
которые потом встраиваются на нужные места с помощью jQuery

Тут нужен BeautifulSoup

В другой половине случаев это JSON, а рендерит его клиент

А тут нужен пакет json

Хороший совет №2

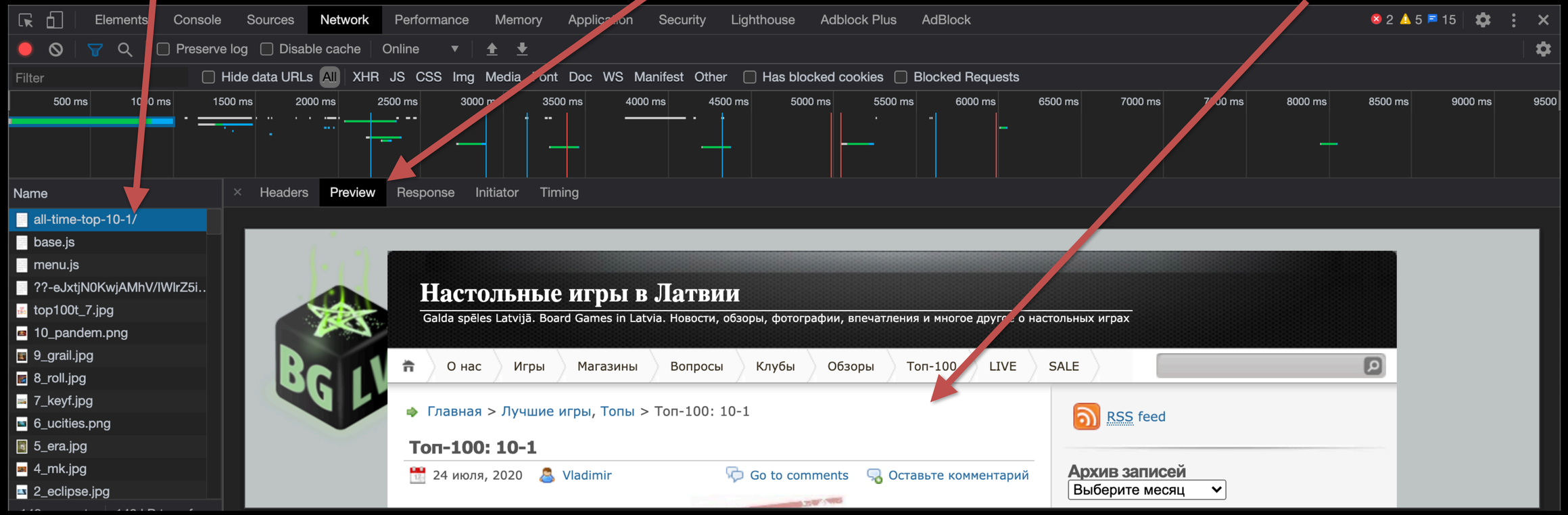
Консоль разработчика - твой бро: смотри, с помощью каких запросов сформирована страница, и ты поймешь, с чем имеешь дело

Хороший совет №2

1. Смотрим первый запрос

2. Смотрим Preview

3. Видим готовую страницу - делаем вывод, что страница статичная



Приоритеты

Сначала API, если его нет, то scraping.

Перерыв!

10 минут

А как реквестить?

Структура запроса

Запрос складывается из:

- URL
- типа запроса (**get**, **post**, put, delete...)
- параметров
- заголовков (хэдэры, “headers”)
- куки (“cookies”)

Структура запроса

Запрос складывается из:

- URL
- типа запроса (**get**, **post**, put, delete...)
- параметров
- заголовков (хэдэры, “headers”)
- куки (“cookies”)

по своей сути куки - тоже хэдэры

GET или POST?

При получении страницы обычно пользуются **GET**

При отправке данных (например, при регистрации - **POST**)

Но это не значит, что нельзя параметры передавать в **GET**...

И не значит, что обязательно передавать их в **POST**...

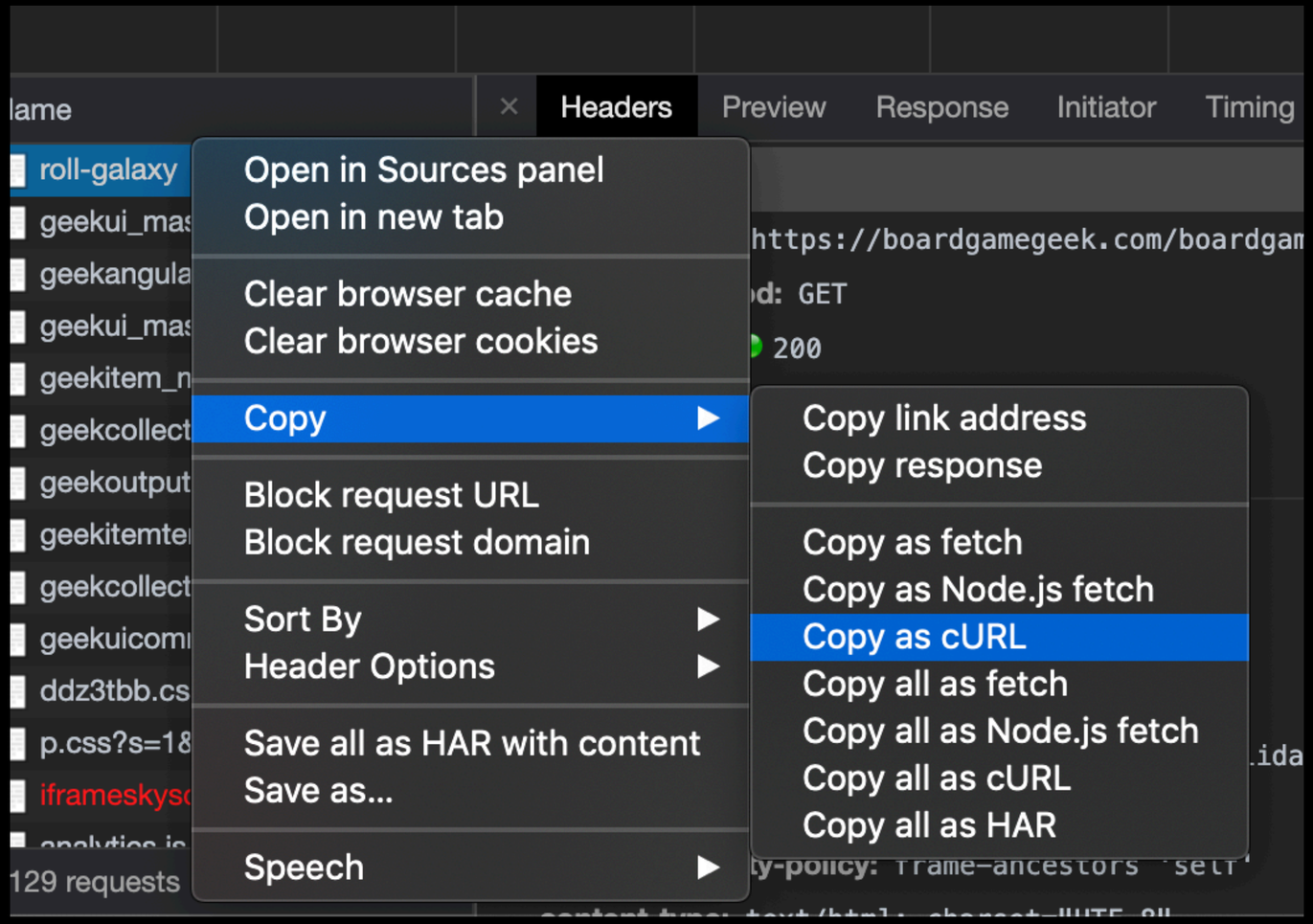
Словом, все размыто.

Смотри в консоль разработчика, чтобы сделать все правильно!

Кто понял жизнь - тот не спешит

```
1  import time
2
3  for i in range(100):
4      time.sleep(1)
5      requests.get('your web site')
```

cURL to Python Requests



cURL to Python Requests

https://curl.trillworks.com/

Маскируемся

```
1  from fake_useragent import UserAgent
2
3  ua = UserAgent()
4
5  ua.random
```

```
'Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.15 (KHTML, like Gecko) Chrome/24.0.1295.0 Safari/537.15'
```

Что можно сделать лучше?

- Сохранять прогресс, чтобы при краше не начинать все заново
- Кэшировать картинки и/или запросы
- Делать запросы многопоточно, если это позволяет сервер

Что можно сделать лучше?

- Сохранять прогресс, чтобы при краше не начинать все заново
- Кэшировать картинки и/или запросы
- Делать запросы многопоточно, если это позволяет сервер

ВЫХОД ЕСТЬ! ЕСЛИ С УТРА ВЫПИТЬ КАПЛЮ ПРОСТОГО, СОВЕТСКОГО....

ОЛЬФЕРУК БЫЛ В ЯРОСТИ КОГДА УЗНАЛ!!!

САЙТЫ ПАРСЯТСЯ САМИ СОБОЙ!!!

ШОК!!!

А как парсить?

select vs find_all

Парсим страницу http://books.toscrape.com/, 3-звездочных книг там 3

select

```
In [39]: 1 len(soup.select('p.Three'))
Out[39]: 3

In [49]: 1 len(soup.select('p.Three.star-rating'))
Out[49]: 3

In [50]: 1 len(soup.select('p.star-rating.Three'))
Out[50]: 3
```

select vs find_all

find_all

Weird 🤪

```
In [53]: 1 len(soup.find_all('p.Three'))  # css selectors won't work
Out[53]: 0

In [41]: 1 len(soup.find_all('p.star-rating'))
Out[41]: 0

In [42]: 1 len(soup.find_all('p', class_='star-rating'))  # only like this
Out[42]: 20

In [54]: 1 len(soup.find_all('p', class_='Three'))
Out[54]: 3

In [55]: 1 len(soup.find_all('p', class_='star-rating Three')) # only works if the order is exactly the same
Out[55]: 3

In [56]: 1 len(soup.find_all('p', class_='Three star-rating'))
Out[56]: 0
```

select vs find_all

Парсим страницу

http://books.toscrape.com/catalogue/tipping-the-velvet_999/index.html,

h2-заголовков там 3

```
1  soup.select('h2', text='Product Description')  # can't do
```

```
[<h2>Product Description</h2>,
 <h2>Product Information</h2>,
 <h2>Products you recently viewed</h2>]
```

select vs find_all

```
1  soup.find_all('h2', text='Product Description')  # you can pass filtering text
```

```
[<h2>Product Description</h2>]
```

```
1  soup.find_all('h2', text=lambda s: 'tion' in s)  # lambdas also work
```

```
[<h2>Product Description</h2>, <h2>Product Information</h2>]
```

CSS-селекторы

https://www.w3schools.com/cssref/css_selectors.asp

Самые важные CSS-селекторы

“div a” ищет все a в div, не важно, как глубоко, главное, что div - родитель a

“div > a” ищет все a **непосредственно** в div, то есть a - ребенок div

“a#user_id” ищет все ссылки с id=“user_id”

“a.navigation” ищет все ссылки с class=“navigation”

Домашка

- Читать документацию requests
- Читать документацию BeautifulSoup4
- Читать про CSS-селекторы
- Что я тебе говорил насчет ложиться спать до 12?!
- Тренироваться можно здесь:
  - http://books.toscrape.com/
  - http://quotes.toscrape.com/
- Читать:
  - 30 страниц – Brian Mulloy – Web API Design. Crafting Interfaces that Developers Love
  - 65 страниц – Web API Design: The Missing Link. Best Practices for Crafting Interfaces that Developers Love

Как еще лучше парсить сайты

Александр Ольферук,
наставник

Яндекс Практикум