

PARSONGS

и немногого **RESEARCH**

Проект по программированию студентки ФиКЛ НИУ ВШЭ
БКЛ-194 Анастасии Чевелевой

ПРЕДЫСТОРИЯ

Изначально хотела сделать проект
по исследованию места названия песни в её тексте.

Но... Для этого нужны тексты. Много текстов. Руками копипастить - пффф.

Нужен классный сайт, где много песен, и П.А.Р.С.Е.Р.

Сайт - <https://genius.com>

Парсер - https://github.com/ancheveleva/project_parsongs (файл
code_PARSONGS.py)

01

PARSONGS. Введение

Общие сведения и
“перед началом работы”

03

RESEARCH. Дополнение

Пример исследования на
базе данных, собранных Им

02

ЭТАПЫ РАБОТЫ ПАРСЕРА

Что говорит Он,
что говорит пользователь,
что делает Он

04

ЗАКЛЮЧЕНИЕ

Потери и поражения,
успехи и достижения



01

PARSONGS.

Введение

Общие сведения и
“перед началом работы”

Парсер получает от пользователя имя исполнителя и возвращает

файлы с текстами песен первого артиста, нашедшегося по запросу пользователя на сайте Genius.

сводную таблицу с метаинформацией по каждой песне каждого исполнителя, которого когда-либо запрашивал пользователь

ПЕРЕД ЗАПУСКОМ ПРОГРАММЫ

CHROME



С другими браузерами
программа
не будет работать :(

CHROMDRIVER

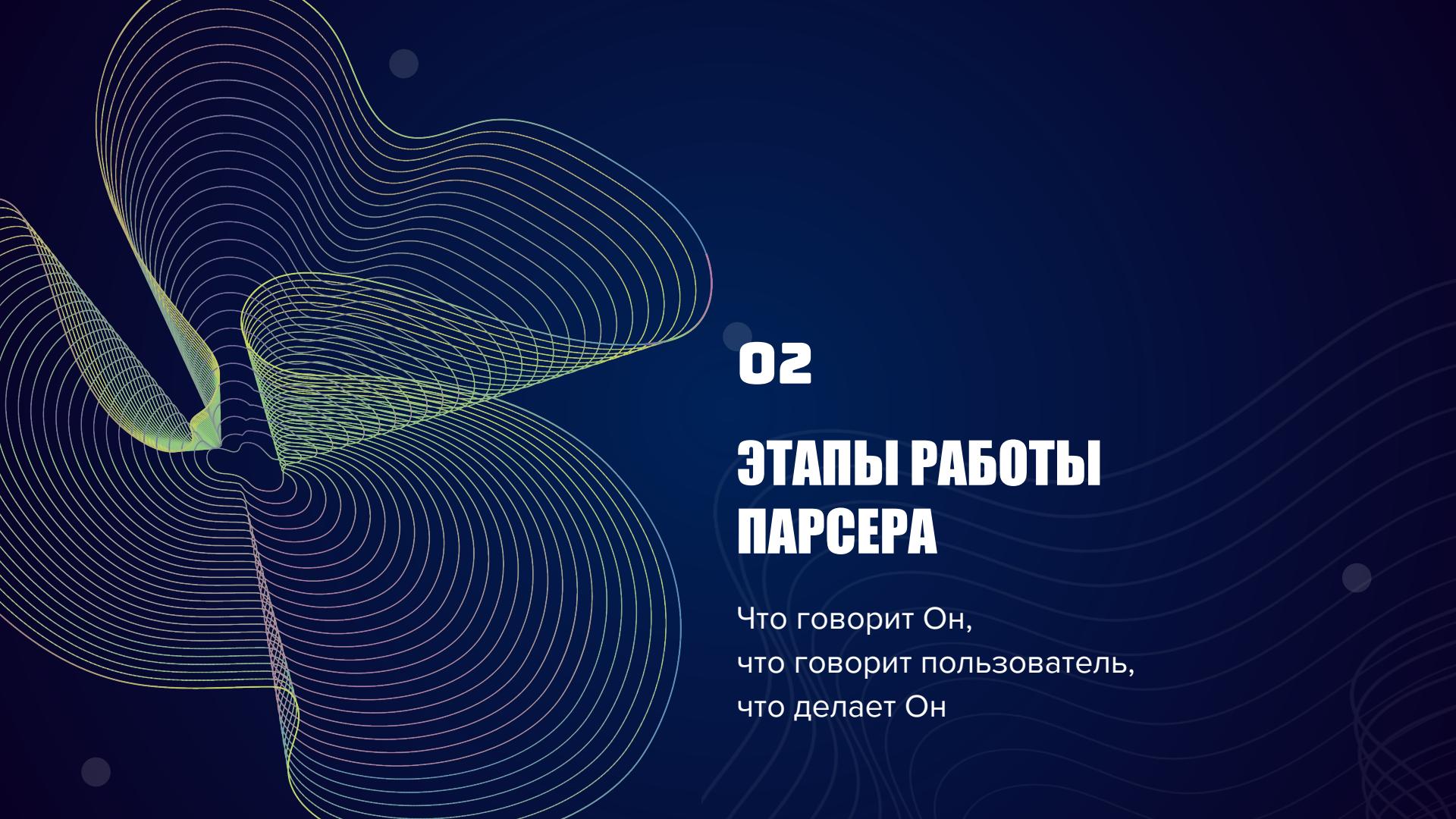


Чтобы диалог с Chrome не ломался,
драйвер должен согласовываться с
ним по версии!

LIBRARIES



Requests
Pyquery
Selenium



02

ЭТАПЫ РАБОТЫ ПАРСЕРА

Что говорит Он,
что говорит пользователь,
что делает Он

ЭТАПЫ РАБОТЫ



1. ПРИВЕТСТВИЕ И ПОИСК ПО САЙТУ

```
#Достаёт с сайта url исполнителя по запросу пользователя
def get_artist_url():
    artist_name_for_search = input('Hi! Write the artist whose songs you want to get: ').lower()
    artist_name_for_search_url = quote(artist_name_for_search)
    artist_search_url = f'{GENIUS_MAIN_PAGE}/search?q={artist_name_for_search_url}'
```

Функция **quote** библиотеки **urllib**. Превращает обычную строку в “формат” URL

● ПРОГРАММА

Hi! Write the artist whose songs you want to get:

● ПОЛЬЗОВАТЕЛЬ

Вводит имя исполнителя. Ограничения на формат нет (преобразование)
Но чем точнее - тем лучше (берёт первый из выдачи поисковика Genius)

Если никого не нашлось Sorry, there is no such artist on Genius
or I just can't search well enough :(и прекратит свою работу.

2. ЗАГРУЗКА В ТАБЛИЦУ

```
dic_song_all = json.loads(dic_song_all_get)
list_song = [str(dic_song_all["response"]["song"]["tracking_data"][2]["value"]), #Artist
             str(dic_song_all["response"]["song"]["title"]), #Title
             str(dic_song_all["response"]["song"]["share_url"]), #URL
             str(dic_song_all["response"]["song"]["tracking_data"][4]["value"]), #Album
             str(dic_song_all["response"]["song"]["release_date"]), #Release Date
             str(dic_song_all["response"]["song"]["recording_location"]), #Recording Location
             str(dic_song_all["response"]["song"]["tracking_data"][21]["value"]), #Lyrics Language
             str(dic_song_all["response"]["song"]["tracking_data"][6]["value"])) #Genre
             ]
list_songs.append(list_song)
```

Программа: чекает, есть ли рабочие папки и таблица. Если нет - **os.makedirs(), csv.writer()**

Чекает, есть ли этот исполнитель в таблице. Если нет - добавляет в таблицу ВСЕ его песни.

Задача - обход сайта каждой песни исполнителя

Genius + json = <3

Чем больше песен, тем дольше работает. Но такое надо сделать с каждым исполнителем один раз!

3. ЧЕК С ПОЛЬЗОВАТЕЛЕМ ПЕРЕД ЗАГРУЗКОЙ

Программа: There are songs of *имя артиста в формате Genius* in the list. If this the right artist, do you want to download any of its songs? Print 'YES' or 'NO':

Пользователь: возмущается, чё так много да-неток??
Ну, мы же в свободной стране живём...

4. СКАЧИВАНИЕ

ИБО ЧЕМ БОЛЬШЕ СИЛА, ТЕМ ДОЛЬШЕ РАБОТАЕТ КОД...

```
if row["Artist"] == artist:  
    lyrics = ""  
    t = 0  
  
    while not lyrics and t != 5: #Это нужно потому, что, кажется, иногда гениус  
        res_link = requests.get(row["URL"]).text  
        lyrics = pq(res_link).find("div.lyrics").text()  
        t += 1  
  
    if not lyrics:  
        failed += 1  
  
    if lyrics:  
        file_name = clean_punc(row["Title"]).lower().replace(' ', '_')  
        with open(f"{PATH_TO_SONGS_FOLDER}/{artist}/{file_name}.txt", "w", encoding="utf-8") as f:  
            f.write(artist + "\n" + row["Title"] + "\n" + lyrics)  
        counter += 1  
  
    if counter == how_many_failed:  
        break
```

...понимает, что его
парсят, и обижается.
Но иногда даже 5 раз
попросить не помогает
:(

5. РАДОСТЬ

Чувак из шаблона тупо
кайфует, что
воспользовался
PARSONGS,
а не какой-то любой [lyricsgenius](#),
где ещё и регаться надо





03

RESEARCH.

Дополнение

Пример исследования на базе данных,
собранных Им

Без претензии на реальное исследование!
Просто ради идеи
Код в репозитории - [code_RESEARCH.py](#)

ЖАНРЫ ИСПОЛНИТЕЛЯ ПО ЧАСТОТЕ

```
#Жанры исполнителя и их частота
def artists_genres(artist_from_the_table):
    artist_genres = []
    with open(PATH_TO_CSV, encoding='utf-8') as f:
        reader = csv.DictReader(f)
        for row in reader:
            if row["Artist"] == artist_from_the_table:
                artist_genres.append(row["Genre"])
    count_genres_list = list(sorted(collections.Counter(artist_genres).items(),
                                    reverse=True, key=lambda kv_pair: kv_pair[1]))
    return count_genres_list
```

Практически, только таблица, без сложных дополнительных инструментов!

ЧИСЛО СЛОВ

Общее число слов всех песен

Исполнителя, которые
скачал пользователь
Тут были рЕгУлярКи



Число "немусорных" слов

Без стоп-слов
Тут было немного либы `nltk`
(см. след. слайд)

Идея: отношение правой к левой штуке (Ratio) должно быть тем выше, чем оригинальнее
тексты исполнителя

На деле: примерно одинаковое число у Muse, The Cure, Imagine Dragons, Thirty Seconds To
Mars (0.45-0.46). Кажется, язык не ломает искусство (или наоборот).

НЕМНОГО NLTK

```
import nltk  
nltk.download('stopwords')  
  
from nltk.corpus import stopwords  
  
STOP_WORDS = set(stopwords.words("english")) #Он странный.  
my_set = {"n't", "'re", "'ve", "'s", "'ll", "'m"} #Не понял  
STOP_WORDS.update(my_set)  
  
without_stop_words_song = [word for word in all_words_song if not word in STOP_WORDS]
```

Список стоп-слов:
есть couldn't, но нет could,
есть can, но нет can't

Для лемматизированного списка слов:
Хорошо работает при указании параметра POS,
но это надо для каждого слова делать

```
with open(f'{PATH_TO_RESEARCH_WORK_FOLDER}/{artist}_meaning_words.txt', 'a',  
         encoding='utf-8') as a:  
    for word in without_stop_words_song:  
        a.write(lemmatizer.lemmatize(word) + '\n')
```

```
nltk.download('wordnet')  
  
from nltk.stem import WordNetLemmatizer  
lemmatizer = WordNetLemmatizer()
```

Топ-10 самых частотных немусорных слов (по их леммам)

Сводный файл по мини-исследованию

Muse

Genres: [('rock', 243), ('pop', 36), ('non-music', 4)]

Total number of words: 40076

Number of 'meningful' words: 17835

Ratio: 0.45

Top-10 'meaningful' words: [('could', 264), ('know', 250),
('can't', 219), ('yeah', 209), ('never', 192), ('want', 191),
('away', 181), ('let', 168), ('wrong', 165)]

The Cure

Genres: [('pop', 352), ('rock', 181)]

Total number of words: 99263

Number of 'meningful' words: 45428

Ratio: 0.46

Top-10 'meaningful' words: [('like', 857), ('never', 812),
('love', 513), ('go', 441), ('time', 430), ('know', 418),
('always', 409), ('oh', 398), ('one', 396)]

Imagine Dragons

Genres: [('rock', 131), ('pop', 81), ('rap', 2), ('non-music', 2), ('r-b', 1)]

Total number of words: 64185

Number of 'meningful' words: 28917

Ratio: 0.45

Top-10 'meaningful' words: [('oh', 867), ('let', 348), ('life', 324), ('never', 316), ('time', 296), ('know', 295), ('love', 274), ('come', 261), ('hey', 251)]

Thirty Seconds to Mars

Genres: [('pop', 64), ('rock', 58), ('rap', 1)]

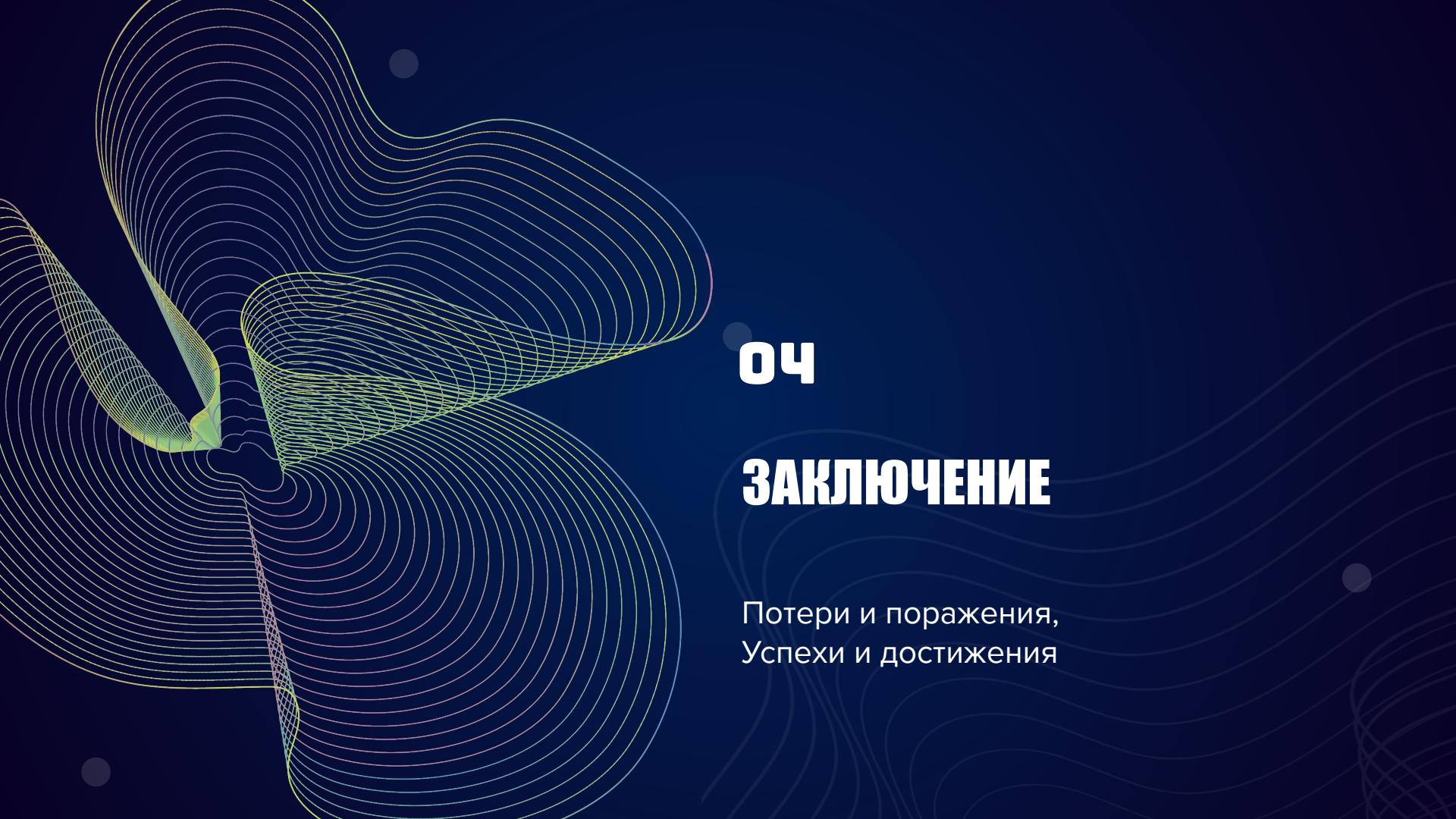
Total number of words: 24623

Number of 'meningful' words: 11231

Ratio: 0.46

Top-10 'meaningful' words: [('oh', 274), ('never', 125), ('love', 125), ('want', 124), ('time', 122), ('one', 113), ('life', 112), ('believe', 111), ('away', 100)]

У всех в топе есть NEVER с:

The background features a dark blue gradient with a complex pattern of thin, light-colored wavy lines. These lines form several large, sweeping curves that overlap each other, creating a sense of depth and motion. Small, semi-transparent grey dots are scattered across the background, some aligned with the wavy lines and others appearing randomly.

04

ЗАКЛЮЧЕНИЕ

Потери и поражения,
Успехи и достижения

- Начальные задумки про упоминание названия не удалось воплотить
- Аккуратная обработка текстов - это сложно
- Текстов, скаченных из интернета - ещё сложнее
- Не уверена в легальности происходящего...

Зато:

- Освежила html
- Прокачала os, csv, json
- Узнала про много важных либ: **request, pyquery, nltk**

СПАСИБО ЗА ВНИМАНИЕ!

Желаю всем
вдохновения и музы
на время сессии и
на всю жизнь!

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik.

Please keep this slide for attribution.