

Загружаем необходимые библиотеки

```
Ввод [83]: import numpy as np

import warnings
warnings.filterwarnings('ignore')
warnings.warn('DelftStack')
```

```
Ввод [84]: import matplotlib.pyplot as plt
import pandas as pd
import time
from pathlib import Path
import csv
import openpyxl

import nltk
from nltk.probability import FreqDist
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
nltk.download('punkt')
nltk.download('stopwords')

#!/pip install ipywidgets

[nltk_data] Downloading package punkt to
[nltk_data]   C:\Users\lapte\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]   C:\Users\lapte\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[84]: True

```
Ввод [29]:

Enabling notebook extension jupyter-js-widgets/extension...
- Validating: ok
```

Количество ядер процессора

для каждой машины свои значения

```
Ввод [3]: import multiprocessing as mp  
n_workers = mp.cpu_count()
```

4 workers are available

Загрузим файл и посмотрим на него

Мультипроцессорная обработка данных: Посчитаем количество строк в файле с помощью пандас

```
Ввод [92]: # Вариант с циклом  
dataset_path = Path('parsed_data.csv')  
chunksize = 100  
df_chunks = pd.read_csv(  
    dataset_path,  
    encoding='utf-8',  
    chunksize=chunksize)  
  
start_time = time.time()  
sizes = []  
for chunk in df_chunks:  
    sizes.append(len(chunk))  
print(f'Количество строк {sum(sizes)}')  
print(f'Время выполнения: {round(time.time() - start_time, 2)} секунд.')
```

Количество строк 11375

Время выполнения: 3.05 секунд.

Выведем в консоли размер полученного объекта, названия его столбцов и продолжительность операции чтения файла.

```
Ввод [93]: %%time
df = pd.read_csv('parsed_data.csv', encoding='utf-8')
```

Shape:(11375, 2)

Column Names:

Index(['url', 'content'], dtype='object')

CPU times: total: 2.53 s

Wall time: 2.82 s

Ввод [94]:

Out[94]:

	url	content
0	https://0-100km.ru	NaN
1	https://0-chan.ru	анонимная имиджборда Лого нультиреча вкл луп в...
2	https://0-12.ru	NaN
3	https://0--5.ru	Видионаблюдение Монтаж ремонт обслуживание Вид...
4	https://0-0-1.ru	Главная Корпоративный сайт по продаже компьюте...
5	https://0-base.ru	Без названия Без названия Без названия Поиск л...
6	https://0-nds.ru	Закладки Мед в Заволжск Закладки Мед в Заволжс...
7	https://0-edc.ru	О еде Рецепты блюд и секреты кулинарии Рецепты...
8	https://0--0-0.ru	Автомобильный портал автоновости отзывы автовл...
9	https://0-d.ru	Приветствуем Сайт только что создан Содержимое...

Удалим строки содержащие NaN

```
Ввод [95]: df = df.dropna()  
# Убедимся, что строк содержащих NaN нет  
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
Index: 9919 entries, 1 to 11374  
Data columns (total 2 columns):  
#   Column      Non-Null Count  Dtype  
---  ---  
0    url         9919 non-null    object  
1    content     9919 non-null    object  
dtypes: object(2)  
memory usage: 232.5+ KB
```

Функция возвращающая dataframe с некорректными символами содержащимися в столбце

```
Ввод [96]: def get_illegal_characters(df, column_name, legitimate_characters, encoding='utf8', DEBUG=False):
    # получим все не легитимные символы из датафрейма
    result = df['content'].str.extractall(legitimate_characters)

    # Решим проблему с повторениями.
    result = result[0].unique()

    result = pd.DataFrame(result)

    # Добавим столбец с hex кодировкой символов.
    result['hex_code'] = result[0].str.encode(encoding)

    # Дадим адекватные имена столбцам.
    result.rename(columns={0: 'character'}, inplace=True)

    if (DEBUG):
        print("Найдено", result.shape[0], "уникальных символов не подходящих под заданный шаблон.")
        print("Найдены следующие символы:\n", result)

    return result
    # Шаблон "легитимных символов". Использование этих символов допустимо в русском языке.
    # Естественно символ пробела мы тоже оставим, иначе наши слова "склеятся".
    legitimate_characters = "([а-яёА-ЯЁ ])"

    # Имя столбца, который необходимо проверить
    column_name = 'content'

    bad_symbols = get_illegal_characters(df, column_name, legitimate_characters)

    bad_symbols
```

Out[96]: character hex_code

Создаем функцию, которая очищает текст от ненужных символов

```
Ввод [97]: import re

def clean_text(text):
    template = '|'.join(map(re.escape, bad_symbols['character']))
    # Удаление ненужных символов из строки.
    text = re.sub(template, '', text, flags=re.UNICODE)
    return text
```

Протестируем различные методы очистки слов в тексте

Последовательный метод очистки текста

```
Ввод [10]: #!pip install --upgrade jupyter
```

```
Requirement already satisfied: jupyter in c:\users\lapte\anaconda3\envs\nlp_py3_10_13\lib\site-packages (1.0.0)
Requirement already satisfied: notebook in c:\users\lapte\anaconda3\envs\nlp_py3_10_13\lib\site-packages (from jupyter) (6.5.4)
Requirement already satisfied: qtconsole in c:\users\lapte\anaconda3\envs\nlp_py3_10_13\lib\site-packages (from jupyter) (5.4.4)
Requirement already satisfied: jupyter-console in c:\users\lapte\anaconda3\envs\nlp_py3_10_13\lib\site-packages (from jupyter) (6.6.3)
Requirement already satisfied: nbconvert in c:\users\lapte\anaconda3\envs\nlp_py3_10_13\lib\site-packages (from jupyter) (6.5.4)
Requirement already satisfied: ipykernel in c:\users\lapte\anaconda3\envs\nlp_py3_10_13\lib\site-packages (from jupyter) (6.25.0)
Requirement already satisfied: ipywidgets in c:\users\lapte\anaconda3\envs\nlp_py3_10_13\lib\site-packages (from jupyter) (8.1.1)
Requirement already satisfied: comm>=0.1.1 in c:\users\lapte\anaconda3\envs\nlp_py3_10_13\lib\site-packages (from ipykernel->jupyter) (0.1.4)
Requirement already satisfied: debugpy>=1.6.5 in c:\users\lapte\anaconda3\envs\nlp_py3_10_13\lib\site-packages (from ipykernel->jupyter) (1.6.7)
Requirement already satisfied: ipython>=7.23.1 in c:\users\lapte\anaconda3\envs\nlp_py3_10_13\lib\site-packages (from ipykernel->jupyter) (8.15.0)
```

```
Ввод [98]: from tqdm import tqdm
           from time import sleep

           tqdm.pandas()
           %time df['content'] = df['content'].progress_apply(clean_text)
           sleep(0.01)
```

```
100%|██████████| 9919/9919 [00:25<00:00, 393.67it/s]
```

```
CPU times: total: 24.6 s
```

```
Wall time: 25.2 s
```

Очищаем текст с помощью модуля multiprocessing: не работает в Anaconda

```
Ввод [ ]: %time
           import multiprocessing as mp
           from tqdm.notebook import tqdm

           n_workers = 4 # количество распараллеленых процессов(воркеров)

           # Создание многопроцессорного пула
           p = mp.Pool(n_workers)

           # Примените функцию clean_text к каждому элементу столбца 'content'
           # путем передачи отдельных строк
           text = list(tqdm(p.imap(clean_text, df['content']), total=len(df)))
           p.close()
           p.join()
```

```
CPU times: total: 0 ns
```

```
Wall time: 0 ns
```

```
0%|          | 0/9919 [00:00<?, ?it/s]
```

Анализ текста

```
Ввод [99]: from collections import Counter

text = df['content'].tolist()

# Функция возвращающая общее количество уникальных слов и общее кол-во их вместе взятая
def word_stats(word_counts):
    num_unique = len(word_counts)
    counts = list(word_counts.values())
    return num_unique, counts

# Вычисление частот слов с помощью счетчика
word_counts = Counter(text)

# вызов функции
num_unique, counts = word_stats(word_counts)
print("Число уникальных слов:", num_unique)
print("сумма частот уникальных слов:", sum(counts))
```

Число уникальных слов: 8991

сумма частот уникальных слов: 9919

Популярные слова

```
Ввод [100]: #сохраним данные столбца 'content' с типом str, чтобы можно было работать с nltk
content = ' '.join(df['content']) # Объединить с пробелом в качестве разделителя
text=content.lower() # все с маленькой буквы

print(type(text))
```

<class 'str'>

анонимная имиджборда лого нультиреча вкл луп выкл луп лого нультиреча вкл луп нас не работает главная все до ски тематика аниме радиоэлектроника мех политац тульпафорсинг и саморазвитие зеркало зеркало не работает гла вная метадопка с лучшим обсуждение медиаконтента такого как книги комиксы фильмы ани

Ввод [13]: %%time

```
start_time = time.time()
# токенизируем текст
words = word_tokenize(text)

# Создаем список со стоп словами для русского языка
stop_words = stopwords.words('russian')
stop_words.extend(['это', 'что', 'всё', 'м', 'г', 'п'])

# Удаляем стоп слова
nouns = [word for word in words if word not in stop_words]

# Вычисление частот слов
fdist = FreqDist(nouns)

# Получите 10 популярных слов
most_common_words = fdist.most_common(10)

print(most_common_words)
```

```
[('отдых', 97377), ('руб', 34617), ('сайт', 24983), ('работа', 23591), ('купить', 23536), ('данных', 19696),
('подробнее', 19681), ('поиск', 17285), ('ремонт', 17136), ('доставка', 15209)]
CPU times: total: 2min 6s
Wall time: 2min 8s
```

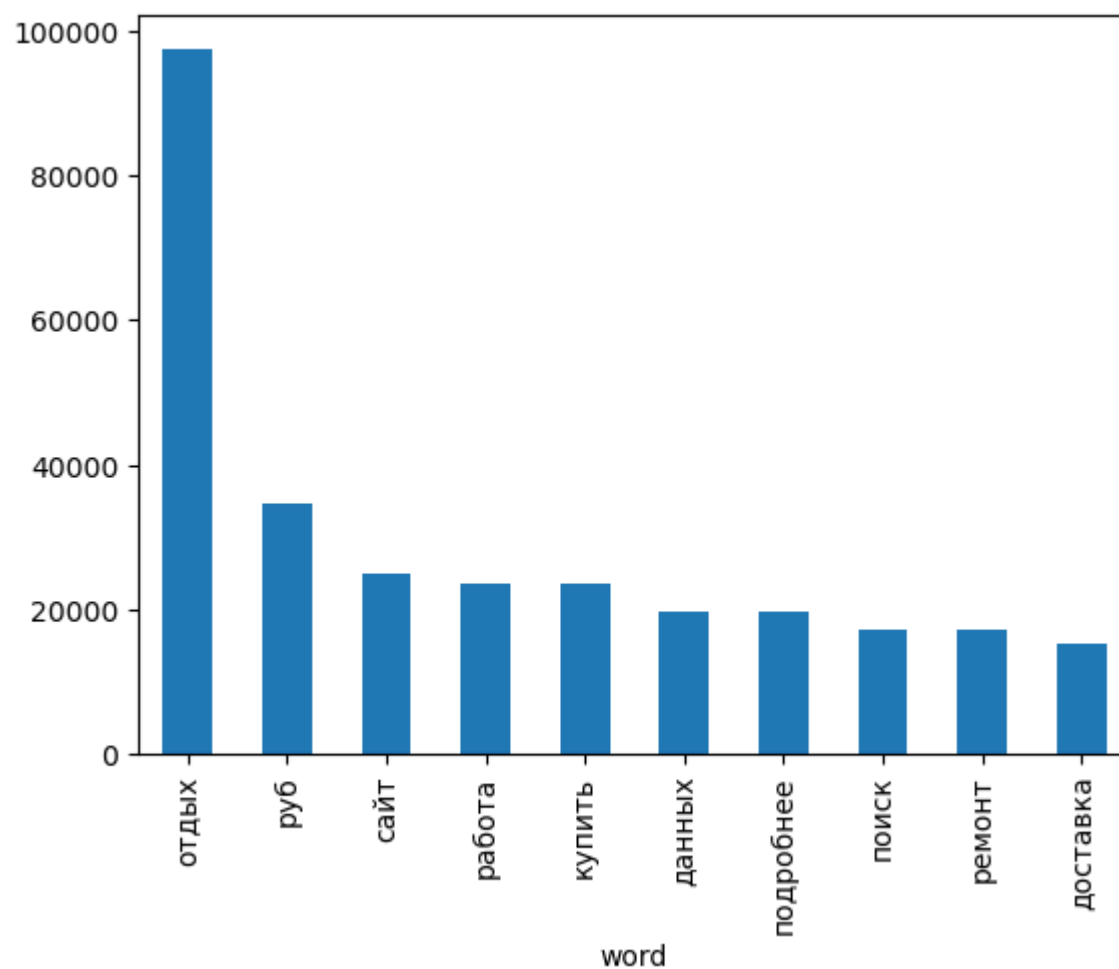
Ввод [18]: *#преобразуем кортеж в таблицу для лучшей визуализации*
df2 = pd.DataFrame(most_common_words, columns=['word', 'count'])

Out[18]:

	word	count
0	отдых	97377
1	руб	34617
2	сайт	24983
3	работа	23591
4	купить	23536
5	данных	19696
6	подробнее	19681
7	поиск	17285
8	ремонт	17136
9	доставка	15209

Нарисуем гистограмму распределения часто встречающихся слов

Ввод [19]:



Визуализация популярности слов в виде облака

Скачиваем необходимые библиотеки

```
Ввод [16]: from wordcloud import WordCloud  
import matplotlib.pyplot as plt  
%matplotlib inline
```

Выведем слова в виде красивой картинки, используя библиотеку WordCloud

```
Ввод [17]: wordcloud = WordCloud(width=800, height=400, background_color='white').generate(text)

plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



Частота вхождения заданных слов в столбце

```
Ввод [79]: print(f"количество сайтов, содержащих слово Контакты: {df.content.str.contains('контакты', na=False).sum()}\n")
print(f"количество сайтов, содержащих слово ИНН: {df.content.str.contains('ИНН', na=False).sum()}\n")
print(f"количество сайтов, содержащих слово телефон: {df.content.str.contains('Телефон', na=False).sum()}\n")
print(f"количество сайтов, содержащих слово адрес: {df.content.str.contains('адрес', na=False).sum()}\n")
print(f"количество сайтов, содержащих слово МФТИ: {df.content.str.contains('МФТИ', na=False).sum()}\n")
df1 = df[df['content'].str.contains("МФТИ")]
```

количество сайтов, содержащих слово Контакты: 889

количество сайтов, содержащих слово ИНН: 810

количество сайтов, содержащих слово телефон: 2773

количество сайтов, содержащих слово адрес: 2571

количество сайтов, содержащих слово МФТИ: 3

	url	content \
1255	https://1-variant.ru (https://1-variant.ru)	Главная Вариант это дефолтный вариант Вариант ...
5655	https://1cov-edu.ru (https://1cov-edu.ru)	Методы решения физико математических задач выс...
9951	https://231-tech.ru (https://231-tech.ru)	Главная Главная страница сайта студенческих се...

	query_count
1255	1
5655	1
9951	1

Частота вхождения заданных словосочетаний

```
Ввод [78]: # Ищем словосочетание в столбце 'content'
result = df[df['content'].str.contains('адрес челябинск', case=False, na=False)]
print(result)
```

#Запись результата в excel, если необходимо

	url \	content	query_count
150	https://007spa.ru (https://007spa.ru)	Спа салон для мужчин Эротического массажа в Че...	1
3671	https://127ds.ru (https://127ds.ru)	МБДОУ Детский сад г Челябинска Муниципальное б...	1
4115	https://147school.ru (https://147school.ru)	Школа Челябинска официальный сайт MAOY COШ Чел...	1
5518	https://1ccrm-tech.ru (https://1ccrm-tech.ru)	С решения для взаимоотношения с клиентами Глав...	1
7386	https://1signal.ru (https://1signal.ru)	Автошкола в Троицке Сигнал Автошкола Сигнал Ав...	1
9630	https://22-volt.ru (https://22-volt.ru)	Газклимат газовые котлы и газовое оборудование...	1
9717	https://2221001.ru (https://2221001.ru)	АвтоАдвокат Услуги автоадвоката в Челябинске п...	1
10734	https://24facecontrol.ru (https://24facecontrol.ru)	ЛайтШоп Главная Интернет магазин заказать купи...	1

```

<!DOCTYPE html>
<html lang="ru">

<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Главная</title>
  <link rel="stylesheet" href="style.css">
</head>

<body>
  <header>
    <div class="center">
      <div class="logo">
        <a
href="https://github.com/BoginskiyVV/webpageDiploma">Github</a>
        <a
href="https://drive.google.com/drive/folders/1ZugqcHTbfqaNu64ULXxcGxFxfEIRDbx
z">GoogleDrive</a>
      </div>
      <div class="menu">
        <a href="contents.html">Содержание</a>
        <a href="introduction.html">Введение</a>
        <a href="chapter1.html">Глава I</a>
        <a href="chapter2.html">Глава II</a>
        <a href="conclusion.html">Заключение</a>
        <a href="appendix.html">Приложения</a>
        <a href="literature.html">Список литературы</a>
        <!-- <a
href="https://github.com/BoginskiyVV/webpageDiploma">Github</a> -->
        <!-- <a
href="https://drive.google.com/drive/folders/1ZugqcHTbfqaNu64ULXxcGxFxfEIRDbx
z">GoogleDrive</a> -->
      </div>
    </div>
  </header>
  <!--Center-->
  <section class="main">
    <div class="center">
      <div class="main_cta">
        <h2>Коллективный дипломный проект на тему:

```

процессе «Применение наиболее значимых компонентов освоенного в

обучения технологического стека для разработки
поисково-аналитического комплекса»

Исполнители:
Афанасьева А.Ю., ВІ аналитик, группа 5295
Богинский В.В., Аналитик больших данных, группа 5321
Евстифеева Ю.В., ВІ аналитик, группа 5295
Лаптева Ф.Р., Аналитик больших данных, группа 5321
Майбродский М.В., Аналитик больших данных, группа

5321

Пиев А.Б., Аналитик больших данных, группа 5321

Афанасьева А.Ю.

ВІ аналитик, группа 5295

Богинский В.В.

Аналитик больших данных, группа 5321

Евстифеева Ю.В.

ВІ аналитик, группа 5295

Лаптева Ф.Р.

Аналитик больших данных, группа 5321

Майбродский М.В.

Аналитик больших данных, группа 5321


```

        </div>
        <div class="diferencial-single" id="single-3">
            <br>
            <h2>Пиев А.Б.</h2>
            <p>Аналитик больших данных, группа 5321</p>
        </div>
    </div>
</div>
</section>
</body>

</html>

<head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Содержание</title>
    <link rel="stylesheet" href="style.css">
</head>

<body>
    <header>
        <div class="center">
            <div class="logo">
                <a href="index.html">На главную</a>
                <!-- <a
href="https://drive.google.com/drive/folders/1ZugqcHTbfqaNu64ULXxcGxFxfEIRDbx
z">GoogleDrive</a> -->
            </div>
            <div class="menu">
                <a href="introduction.html">Введение</a>
                <a href="chapter1.html">Глава I</a>
                <a href="chapter2.html">Глава II</a>
                <a href="conclusion.html">Заключение</a>
                <a href="appendix.html">Приложения</a>
                <a href="literature.html">Список литературы</a>
                <!-- <a
href="https://github.com/BoginskiyVW/webpageDiploma">Репозиторий Github</a>
                <a class="btn-menu" href="#">Ссылка</a> -->
            </div>
        </div>
    </div>
    <!--Center-->

```

```
</header>
<section class="main">
  <div class="center">
    <div class="main_cta">
      <h2>Содержание</h2>
      <p align="justify">
        <br>
        1. Введение.<br>
        <br>
        2. ГЛАВА I. Теоритическая часть.<br>
        2.1. Язык Python для поиска информации в сети Интернет и
парсинга сайтов.<br>
        2.1.1. Общая информация по языку Python и парсингу
сайтов.<br>
        2.1.2. Основы использования библиотеки requests.<br>
        2.1.3. Основы использования библиотеки BeautifulSoup.<br>
        2.1.4. Основы использования библиотеки Scrapy.<br>
        2.1.5. Основы использования библиотеки Selenium.<br>
        2.2. Средства хранения данных при парсинге сайтов.<br>
        2.2.1. Использование формата CSV при парсинге сайтов.<br>
        2.2.2. Использование формата JSON при парсинге
сайтов.<br>
        2.2.3. Использование SQL при парсинге сайтов.<br>
        2.2.4. Использование Hadoop при парсинге сайтов.<br>
        2.3. Использование языка Python для анализа данных.<br>
        2.3.1. Общая информация по возможностям языка Python для
анализа данных.<br>
        2.3.2. Основы использования библиотеки Pandas для анализа
данных.<br>
        2.3.3. Основы использования библиотек Matplotlib и
Seaborn для анализа данных.<br>
        2.4. Использование SQL для анализа данных.<br>
        2.5. Использование BI инструментов для анализа данных и
вывода результатов.<br>
        2.6. dfsgdfg
        <br>
        3. ГЛАВА II. Разработка и тестирование поисково-
аналитического комплекса.<br>
        3.1. Парсер на языке Python с загрузкой данных в SQL
формат.<br>
        3.2. Реализация на Python многопоточной обработки данных
для парсинга сайтов
с последующим анализом в Jupiter notebook.<br>
        3.3. Анализ полученных данных средствами SQL.<br>
```

```

3.4. Анализ и вывод полученных данных с помощью
инструментов BI.<br>
3.5. Поисково-аналитический Telegram bot на Python.<br>
3.6. Вывод результатов проекта с помощью создания сайта
на HTML/CSS.<br>
<br>
4. Заключение.<br>
<br>
5. Приложения.<br>
<br>
6. Список литературы.<br>
</p>

</div>
</div>
</div>
</section>
</body>

</html>
<head>
  <meta charset="UTF-8"/>
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Введение</title>
  <link rel="stylesheet" href="style.css">
</head>

<body>
  <header>
    <div class="center">
      <div class="logo">
        <a href="index.html">На главную</a>
        <!-- <a
href="https://drive.google.com/drive/folders/1ZugqcHTbfqaNu64ULXxcGxFxfEIRDbx
z">GoogleDrive</a> -->
      </div>
      <div class="menu">
        <a href="contents.html">Содержание</a>
        <a href="chapter1.html">Глава I</a>
        <a href="chapter2.html">Глава II</a>
        <a href="conclusion.html">Заключение</a>
        <a href="appendix.html">Приложения</a>
        <a href="literature.html">Список литературы</a>

```

```
<!-- <a
href="https://github.com/BoginskiyVW/webpageDiploma">Репозиторий Github</a>
<a class="btn-menu" href="#">Ссылка</a> -->
</div>
</div>
<!--Center-->
</header>
<section class="main">
<div class="center">
<div class="main_cta">
<h2>Введение</h2>
<p align="justify" style="margin-left: 25px;">
<br>
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<b>1. Введение.</b><br>
<br>
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<b>В настоящее время сеть Интернет
содержит огромное
количество информации, и задача по её поиску, обработке и
анализу становится
все более актуальной. Для разработки эффективных решений
задач такого рода
необходимо использовать соответствующие современные
технологические
инструменты. В данном дипломном проекте рассматривается
применение наиболее
значимых компонентов, освоенных в процессе обучения, для
разработки
поисково-аналитического комплекса.<br>
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<b>В первой главе рассматриваются
основы использования
языка программирования Python для поиска информации в
сети Интернет и парсинга
данных с веб-сайтов. Эта часть дипломного проекта
включает в себя общую информацию
по языку Python и его возможностям, основы использования
библиотеки requests, основы
использования библиотеки BeautifulSoup для парсинга HTML,
основы использования
библиотеки Scrapy, а также основы использования
библиотеки Selenium.<br>
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<b>Также в этой главе
рассматривается использование различных
походов для хранения анных, в том числе таких, как
использование файлов CSV и JSON,
```

возможности взаимодействия Python с SQL, а также средства хранения и обработки больших данных с помощью Hadoop.

Дополнительно проводится обзор возможностей языка Python для анализа данных, включая основы работы с библиотекой Pandas для анализа данных и основы использования библиотек Matplotlib и Seaborn для визуализации данных. Также рассматривается использование SQL для анализа данных и использование BI инструментов для анализа данных и вывода результатов. Завершается глава обсуждением использования стека HTML/CSS для представления результатов проделанной аналитической работы.

Во второй главе описывается процесс разработки и тестирования командой проекта поисково-аналитического комплекса. Разработка поискового механизма производится на языке Python, команда осуществляет выбор и настройку соответствующих инструментов для поиска и парсинга данных. Также в практической плоскости реализуется возможность использования файлов CSV и использование средств SQL для хранения полученных данных, производится анализ полученных данных с помощью языков Python и SQL, а также анализ и вывод полученных данных с помощью доступных инструментов BI. Дополнительно команда представляет использование стека HTML/CSS для отображения результатов выполнения дипломной работы в качестве веб-страницы.

В заключении подводятся итоги проведенной работы, резюмируются результаты и достижения проекта. Оцениваются его преимущества и недостатки, а также даются рекомендации по его дальнейшему развитию и применению. В приложениях представлены дополнительные материалы, которые могут быть полезны для более подробного изучения темы проекта, в том числе исходный код разработанных инструментов, дополнительные схемы и диаграммы, а также результаты дополнительных экспериментов и исследований.

В конце работы приводится список использованной литературы и других источников информации, которые были использованы при разработке проекта. В список входят книги, онлайн-ресурсы и другие источники, которые помогли в освоении и применении технологического стека для разработки поисково-аналитического комплекса.

```
</p>

</div>
</div>
</div>
</section>
</body>

</html>
<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Глава 1</title>
  <link rel="stylesheet" href="style.css">
</head>

<body>
  <header>
    <div class="center">
      <div class="logo">
        <a href="index.html">На главную</a>
        <!-- <a
href="https://drive.google.com/drive/folders/1ZugqcHTbfqaNu64ULXxcGxFxfEIRDbx
z">GoogleDrive</a> -->
      </div>
      <div class="menu">
        <a href="contents.html">Содержание</a>
        <a href="introduction.html">Введение</a>
        <!-- <a href="chapter1.html">Глава I</a> -->
        <a href="chapter2.html">Глава II</a>
        <a href="conclusion.html">Заключение</a>
        <a href="appendix.html">Приложения</a>
        <a href="literature.html">Список литературы</a>
      </div>
    </div>
  </header>
</body>
</html>
```

```
<!-- <a
href="https://github.com/BoginskiyVV/webpageDiploma">Репозиторий Github</a>
<a class="btn-menu" href="#">Ссылка</a> -->
</div>
</div>
<!--Center-->
</header>
<section class="main">
  <div class="center">
    <div class="main_cta">
      <h2>ГЛАВА I</h2>
      <p align="justify">
        <br>
        &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<b>Глава I. Теоретическая
часть.</b><br><br>
        <br>
        &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<b>2.1. Язык Python для
поиска информации в сети Интернет и парсинга
сайтов.</b><br>
        &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<b>2.1.1. Общая информация
по языку Python и парсингу сайтов.</b><br>
        &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<b>В современном информационном обществе
объем данных, представленных в сети
Интернет, увеличивается с каждым днем.
Использование этих данных может стать ценным ресурсом для
решения множества задач, включая
извлечение и дальнейшее
использование информации, анализ данных, машинное
обучение и другие. Большинство данных Интернете,
доступны
в виде веб-страниц, и для их эффективного использования
необходимо уметь собирать и обрабатывать
нужную
информацию.<br>
        &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<b>Парсинг веб-страниц - это процесс
извлечения информации из такой страницы, анализа
ее содержимого
и преобразования в удобный формат для дальнейшей
обработки, хранения и более глубокого анализа.
Python – это
мощный, современный, гибкий, высокоуровневый,
интерпретируемый язык программирования, который
примечателен
```

своей простотой, читаемостью и обширной экосистемой библиотек и инструментов. Python стал одним из наиболее популярных языков программирования для разработки веб-приложений, анализа данных, искусственного интеллекта и других областей.

Язык Python является отличным выбором для работы с данными в Интернете. Благодаря множеству библиотек и инструментов, доступных на этом языке, поиск информации и парсинг веб-страниц доступен для широкого круга как продвинутых, так и начинающих специалистов. Независимо от того, стоит ли просто вопрос получения информации для анализа данных, обновления содержимого сайта или выполнения других задач, Python готов предоставить все необходимые инструменты для успешной реализации поисково-аналитического комплекса или реализации иного проекта.

Основные инструменты для парсинга веб-страниц на языке Python:

- Библиотека requests: используется для отправки HTTP-запросов и получения содержимого страницы.
- Библиотека BeautifulSoup: предоставляет удобные методы для разбора HTML-кода и извлечения данных.
- Библиотека lxml: используется для парсинга XML-документов и работы с XPath-запросами.
- Библиотека Scrapy: предоставляет инструменты для создания веб-скрейперов и парсеров с возможностью обхода и сбора информации с нескольких страниц.
- Библиотека Selenium: мощный и популярный инструмент не только для парсинга сайтов, но и для решения задач в области разработки и тестирования веб-страниц.

2.1.2. Основы использования библиотеки requests.

Библиотека requests является одной из самых популярных библиотек для отправки HTTP-запросов и получения данных из сети в языке программирования Python. Она предоставляет простой и удобный интерфейс для работы с веб-ресурсами, что делает ее идеальным выбором для парсинга сайтов. С ее помощью возможно эффективно и гибко собирать нужную информацию с веб-страниц и использовать ее для различных задач, таких как анализ данных, мониторинг или автоматизация процессов.

Основным способом получения веб-страниц является отправка GET-запросов. Библиотека requests предоставляет функцию get, которая позволяет отправлять GET-запросы на веб-ресурсы и получать содержимое страницы. Возвращаемый объект Response содержит различные данные о запросе и ответе сервера, включая статус-код, заголовки и тело ответа.

Изображение № 1. Пример отправки GET-запроса и вывода содержимого страницы

```

```

В примере, представленном на изображении №1 мы отправляем GET-запрос на страницу `https://example.com` и выводим статус-код ответа и содержимое страницы.

Также часто для получения нужной информации с веб-страницы требуется передать параметры в запросе (например, для поиска). Библиотека requests позволяет передавать параметры запроса в виде словаря с помощью аргумента `params` функции `get`.

Изображение № 2. Пример отправки GET-запроса с аргументом `params`

```

```


для более
гибкого и продвинутого парсинга веб-страниц, которые
будут представлены в практической части работы
и приложениях.

2.1.2. Основы использования библиотеки BeautifulSoup

Библиотека BeautifulSoup предоставляет удобные инструменты для парсинга HTML и XML кода сайтов, облегчая извлечение информации и навигацию по структуре веб-страниц. Использование библиотеки BeautifulSoup значительно упрощает процесс парсинга веб-страниц на языке Python. Благодаря удобным методам и инструментам, предоставляемым этой библиотекой, имеется возможность быстро и эффективно извлечь нужную информацию из HTML кода страницы.

Прежде чем начать использовать BeautifulSoup, необходимо установить ее.

Установка библиотеки производится с помощью следующей команды

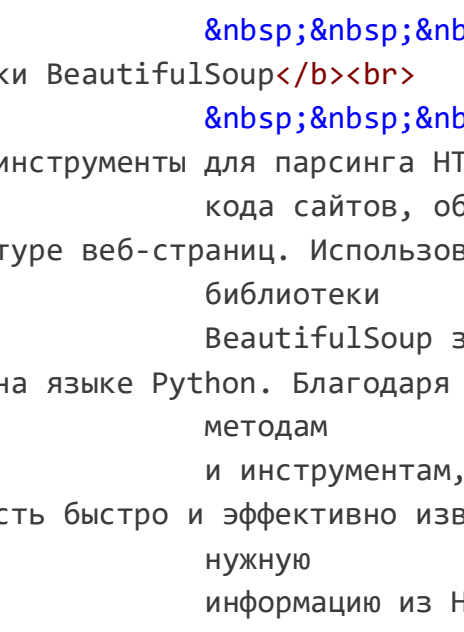
```
pip install beautifulsoup4
```

После установки мы импортируем библиотеку BeautifulSoup в свой код:

```
from bs4 import BeautifulSoup
```

Для парсинга веб-страницы сначала нужно получить HTML код этой страницы. Для этого необходимо использовать описанную ранее библиотеку requests или любой другой способ, который позволяет получить HTML код страницы (Изображение №5).

Изображение № 5



```
from bs4 import BeautifulSoup
url = 'http://example.com'
html = requests.get(url).text
soup = BeautifulSoup(html, 'html.parser')
```

После получения HTML кода страницы, мы можем создать объект BeautifulSoup для

дочерние
элементы. Затем мы ищем все элементы "div" с классом
"container" и выводим их. Наконец, мы находим
следующий
соседний элемент после "header" и выводим его.
Библиотека BeautifulSoup
предоставляет еще множество других методов и
возможностей
для более гибкого и продвинутого парсинга HTML кода,
которые будут представлены в практической части
работы
и приложениях.

2.1.3. Основы использования
библиотеки Scrapy
Scrapy - это мощный и гибкий фреймворк для
создания веб-скрейперов и парсеров на
языке программирования Python. Он предоставляет множество
инструментов и функций для автоматизации
процесса
сбора данных с веб-страниц и обхода нескольких страниц
одновременно. Рассмотрим основы использования
библиотеки Scrapy для парсинга сайтов.
Прежде чем начать использовать Scrapy,
необходимо установить ее. Установка библиотеки
производится с помощью следующей команды pip:
"pip install scrapy"
После установки мы импортируем
библиотеку Scrapy в свой код:
"import scrapy"
Для создания парсера с помощью
Scrapy необходимо создать новый проект Scrapy
с помощью командной строки:
"scrapy startproject myproject"
После создания проекта, необходимо создать
спайдера - класс, который определяет
логику парсинга веб-страниц. В методе parse спайдера вы
можете определить, какие данные нужно
собрать и какие
действия нужно совершить на странице.

```
<br>
<br>
```

После создания спайдера
парсер с помощью команды scrapy

Спаидеры Scrapy предоставляют
анализ собранных данных.


```
<br>
<br>
```

В примере на изображении №10 мы инициализируем экспортер CSV и открываем файл `data.csv`, в который будут сохраняться данные. Затем мы вызываем метод `export_item` для сохранения элементов на каждой странице. По завершению работы мы вызываем метод `finish exporting`.

Затем создается экземпляр класса веб-драйвера для выбранного браузера:


```
<span style="color: #ff6600;">"driver = webdriver.Chrome()  
    # Создание веб-драйвера Chrome"</span><br>
```

У объекта веб-драйвера есть множество методов для управления веб-страницами.

Например, метод get используется для перехода на определенную веб-страницу:


```
<span style="color: #ff6600;">"driver.get("https://example.com")  
    # Переход на страницу example.com"</span><br>
```

Методы класса webdriver также позволяют выполнять различные действия на веб-страницах, такие как заполнение полей формы, нажатие кнопок, скроллинг страницы и многое другое.

Основной метод получения информации из веб-страницы с использованием Selenium - это метод find_element_by, который находит элемент на странице на основе различных атрибутов (Class Name, CSS Selector, ID, Link Text и другие). Он используется для нахождения нужной информации на веб-странице.


```
<span style="color: #ff6600;">"element =  
    driver.find_element_by_css_selector('#my-element')  
    # Найти элемент по селектору CSS "</span><br>
```

```
<span style="color: #ff6600;">"print(element.text)  
    # Извлечь текст элемента"</span><br>
```

В представленном выше примере мы находим элемент с помощью метода

```
find_element_by_css_selector, передавая ему CSS-селектор  
элемента. Затем мы выводим текстовое  
содержимое найденного элемента.<br>
```

После завершения работы необходимо закрыть окно браузера, чтобы освободить ресурсы.

```
Такое действие реализуется с помощью метода close:<br>  
<span style="color: #ff6600;">"driver.close() # Закрытие окна
```


браузера"

Также возможно закрыть полностью веб-драйвер с помощью метода quit:

```

                <span style="color:
#ff6600;">"driver.quit() # Полное завершение работы
        веб-драйвера"</span><br>

```

Важно помнить, что закрывать окно
браузера нужно после завершения всех
нужных
операций с веб-страницей.

По умолчанию браузер Selenium работает в видимом режиме, и можно видеть,

как он выполняет действия на веб-странице. Однако библиотека Selenium также предоставляет

возможность
работы в фоновом режиме с помощью драйвера PhantomJS или
Headless Chrome.

PhantomJS: `
`Пример использования

```
from selenium.webdriver import  
PhantomJS
```

```

#ff6600;">driver = PhantomJS() # Создание экземпляра
PhantomJS
веб-драйвера"</span><br>

```

```

                <span style="color:
#ff6600;">"driver.get("https://example.com")
        # Переход на страницу example.com"</span><br>
        <br>

```

2.2. Средства хранения данных
при парсинге сайтов

2.2.1. Использование CSV при парсинге сайтов

Одним из способов организации и хранения данных, полученных при парсинге web-страниц, является использование SCV-формата (Structured Comma Separated Values).

SCV-формат представляет собой специальный тип текстового файла, который используется

для структурированного хранения данных в виде таблицы.
Файл SCV содержит данные, разделенные
запятыми

• Ограниченность структуры данных. SCV-формат предоставляет ограниченные возможности для структурирования данных. Например, сложные связи между таблицами или специфические типы данных могут быть сложно или невозможно представить в формате SCV.

• Ограничения на объем данных. SCV-формат является текстовым и не предназначен для хранения больших объемов данных. В случае работы с очень большими массивами информации, использование SCV может снизить производительность и эффективность обработки данных.

• Возможные проблемы с кодировкой. В некоторых случаях могут возникнуть проблемы с кодировкой данных при сохранении и чтении SCV-файлов. Это может привести к искажению данных или их неправильному представлению.

Отметим, что использование SCV-формата при парсинге сайтов с помощью Python предоставляет достаточно удобный и эффективный способ организации и сохранения полученных данных. Однако, необходимо учитывать некоторые ограничения данного формата, как на структуру данных, так и на возможное возникновение проблем с кодировкой.

2.2.2. Использование JSON при парсинге сайтов

JSON (JavaScript Object Notation) - это формат обмена данными, основанный на языке JavaScript. Он широко используется веб-разработчиками для передачи и обмена структурированными данными между клиентской и серверной сторонами приложения. При парсинге сайтов, JSON используется для извлечения и преобразования данных, полученных из HTML-разметки в более удобный и легкодоступный формат.

JSON предоставляет удобное и легко читаемое представление данных, которое можно использовать для анализа и дальнейшей обработки. Он позволяет сократить объем кода и сделать

процесс парсинга более эффективным и гибким.
Для начала, необходимо получить HTML-код страницы. Это можно сделать, используя библиотеку HTTP-запросов, такую как requests, которая была упомянута выше. После получения HTML-кода, можно использовать BeautifulSoup, lxml или другие парсеры, чтобы извлечь данные из HTML-разметки.

Затем, если данные, которые нужно извлечь, представлены в формате JSON на странице, мы можем использовать методы работы с JSON для их извлечения. JSON-данные обычно представляются в виде пар "ключ-значение" и могут быть представлены в виде объектов, массивов, чисел, строк и логических значений.

В простых случаях, когда данные находятся в корневом уровне страницы, можно просто обратиться к ним через общий для большинства языков программирования синтаксис обращения к элементам словаря.

Следует учитывать, что в некоторых случаях данные могут находиться во вложенных структурах и требуют достаточно сложных запросов. В таких случаях следует применить такие методы работы с JSON, как использование пути к элементу или циклов для обхода всех элементов массива или объекта.

Кроме того, необходимо учитывать, что при парсинге сайтов, некоторые веб-страницы могут быть динамически построены с помощью JavaScript, что может усложнить процесс извлечения данных из JSON. В таких случаях может потребоваться использование инструментов для автоматического выполнения JavaScript, таких как Selenium WebDriver.

2.2.3. Использование SQL при парсинге сайтов

Для эффективного хранения и организации данных, собранных в процессе парсинга веб-страниц часто используется язык структурированных запросов SQL (Structured Query Language) и реляционные базы данных.

Перед началом работы с SQL для хранения данных, нужно выбрать соответствующую базу данных, которая будет отвечать требованиям проекта. Существуют различные реляционные базы данных, такие как MySQL, PostgreSQL, SQLite и другие, которые предоставляют различные функциональности и возможности. Выбор конкретной системой управления базой данных будет зависеть от предпочтений команды разработчиков, требований к масштабируемости, производительности и некоторых других факторов.

Одним из основных преимуществ SQL является его способность обеспечивать структурированное хранение данных. SQL базы данных состоят из таблиц с заданными столбцами и типами данных для каждого столбца. Это позволяет легко организовать данные и обеспечить соответствие структуре данных, что облегчает работу с ними.

В то время как CSV и JSON являются неструктурированными форматами данных, которые сохраняют информацию в виде текста без явного определения структуры данных. В CSV данные хранятся в простом текстовом формате с разделителями, а в JSON данные хранятся в виде пар "ключ-значение".

Это может усложнить управление и обработку данных при использовании этих форматов.

SQL предоставляет средства для выполнения мощных и гибких запросов к данным.

Имеется возможность использовать язык SQL для извлечения нужных данных, применять фильтры, сортировать результаты и связывать данные из разных таблиц. Благодаря этому, SQL дает возможность быстро получить нужные данные и осуществить сложные операции с ними.

С другой стороны, в CSV и JSON форматах отсутствуют встроенные механизмы запросов и фильтрации данных. Для извлечения отдельных данных или выполнения сложных операций придется обрабатывать

данные вручную с использованием других инструментов или кода Python. В отдельных случаях может быть более трудоемким и менее эффективным в сравнении с использованием SQL запросов.

SQL базы данных обеспечивают поддержку целостности данных и механизмы контроля доступа. Имеется возможность определить ограничения на значения в таблицах для обеспечения целостности данных, а также настроить права доступа для пользователей и ролей. Это значительно упрощает управление и контроль данных, особенно в случаях, когда нужно обрабатывать чувствительные данные или предоставлять доступ к базе данных другим пользователям.

С другой стороны, в CSV и JSON форматах данных нет встроенной поддержки целостности данных или контроля доступа, что может привести к выполнению таких операций вручную, оказаться трудоемким и потенциально приводить к ошибкам.

SQL базы данных предоставляют хорошую масштабируемость и производительность для хранения и обработки больших объемов данных. Они обладают эффективными механизмами работы с данными, оптимизацией запросов и поддержкой индексов для ускорения выполнения запросов и улучшения производительности.

В то время как CSV и JSON форматы имеют свои преимущества, они не всегда подходят для хранения и обработки больших объемов данных. Они могут стать неэффективными при работе с большими файлами, а также создавать сложности при выполнении операций с данными.

2.2.4. Использование Hadoop при парсинге сайтов

Hadoop - это фреймворк для обработки больших объемов данных на

2.3.1. Общая информация по
возможностям языка Python

для анализа данных

Python является одним из наиболее
популярных и широко используемых языков
программирования в мире. Он обладает простым и интуитивно
понятным синтаксисом, что делает его
идеальным выбором для анализа данных. Python является
мощным и гибким языком программирования,
который предоставляет широкий спектр инструментов и
библиотек для анализа данных.

Он легко изучается и его использование для анализа данных
может значительно упростить

и ускорить процесс.

Важно отметить, что Python
является языком с открытым исходным кодом,
что означает, что он бесплатен и доступен для всех. Это
делает его использование актуальным

для широкого круга пользователей и организаций, и
позволяет использовать его для различных
целей, включая научные исследования, анализ данных,
разработку приложений и многое другое.

Одной из самых популярных
библиотек Python для работы с данными
является Pandas. Данная библиотека предоставляет удобный
и гибкий интерфейс для работы
с таблицами данных, анализа и манипулирования ими. Pandas
предлагает функциональность,
которая позволяет легко выполнять операции, такие как
фильтрация, сортировка, объединение
и агрегирование данных.

Еще одной мощной библиотекой
Python для анализа данных является

NumPy. Данная библиотека предоставляет поддержку для
многомерных массивов, а также
широкий набор функций для работы с ними. NumPy позволяет
выполнять математические
операции, такие как сложение, умножение и
тригонометрические функции, на массивах
данных. Это позволяет рассматривать NumPy как идеальный
инструмент для работы

с большими объемами числовых данных.

Для визуализации данных Python
предлагает библиотеку

Matplotlib. Данная библиотека позволяет создавать различные типы графиков, включая линейные, столбчатые, круговые диаграммы, диаграммы рассеяния и другие. Matplotlib также предоставляет возможность настройки внешнего вида графиков, включая метки осей, легенды и цвета. В сочетании с Pandas и NumPy, Matplotlib делает Python мощным инструментом для визуализации и анализа данных.

Также стоит отметить библиотеку Seaborn, построенную на Matplotlib и предназначенную для создания статистических визуализаций. Seaborn хорошо интегрирован с Pandas и позволяет делать выразительные построения наборов статистических данных, понятные как исследователю, так и конечному пользователю аналитического продукта.

Еще одной популярной библиотекой для анализа данных на Python является SciPy. SciPy предоставляет набор функций для решения сложных математических и научных задач. Он включает в себя функции для оптимизации, статистики, интерполяции, алгебры и многое другое. SciPy значительно расширяет возможности Python для анализа данных и делает его подходящим для широкого спектра приложений.

Для машинного обучения и искусственного интеллекта Python предлагает несколько популярных библиотек, таких как TensorFlow, Keras и Scikit-Learn. TensorFlow является мощной библиотекой для создания и обучения нейронных сетей. Keras предоставляет простой и интуитивно понятный интерфейс для создания моделей машинного обучения. Scikit-Learn является библиотекой для машинного обучения, предоставляющей реализацию широкого спектра алгоритмов, таких как классификация, регрессия, кластеризация и многое другое.

Python также имеет широкую поддержку для работы с базами данных. Он предоставляет библиотеки, такие как SQLAlchemy, для работы с различными типами баз данных, включая SQL и NoSQL. Python также имеет возможности для работы с API и получения данных из различных источников сети Интернет и различных веб-сервисов, что частично было описано в предыдущих разделах.

Также важно отметить, что одним из преимуществ использования Python для анализа данных является огромное сообщество разработчиков и наличие качественно составленной технической документации для большинства инструментов языка. При возникновении вопросов у разработчиков и аналитиков всегда есть возможность обратиться к сообществу и найти ответы.

2.3.2. Основы использования библиотеки Pandas для

анализа данных

Pandas предоставляет удобный и мощный инструментарий для работы с табличными данными и выполнения различных операций над ними.

В этой статье мы рассмотрим основные возможности и функции библиотеки Pandas, которые делают ее незаменимым инструментом для анализа данных.

Основным объектом в библиотеке Pandas является DataFrame

- двумерный массив данных, представляющий собой таблицу с метками строк и столбцов.

DataFrame позволяет хранить и манипулировать данными различных типов, включая числа, строки, даты и другие.

Одной из первых задач при работе с данными является их загрузка.

Pandas предоставляет множество функций для чтения данных из различных источников, включая

CSV-файлы, Excel, SQL-базы данных и даже веб-страницы. Например, функция `read_csv()`

позволяет загрузить данные из CSV-файла и создать

`DataFrame`:

Изображение № 11

После загрузки данных вы становится возможным выполнение различных

операций над ними. Например, можно узнать размерность данных с помощью атрибута `shape`:

Изображение № 12

Получить первые или последние строки данных можно с помощью методов

`head()` и `tail()`:

Изображение № 13

Pandas предоставляет множество функций для фильтрации, сортировки

и манипуляции данными. Например, можно отфильтровать данные, выбрав только те строки,

которые удовлетворяют определенным условиям:

Изображение № 14

Можно выполнять агрегирование данных, например, подсчитывать

среднее значение, максимальное или минимальное значение столбца с помощью метода `describe()`:

Изображение № 15

Pandas также предлагает удобные функции для группировки данных, выполнения

операций при соединении таблиц и многое другое. Например, функция `groupby()` позволяет сгруппировать

Matplotlib предоставляет широкий спектр возможностей для создания различных типов графиков, включая линейные, столбчатые, круговые диаграммы, диаграммы рассеяния и т.д., а также предлагает гибкую настройку внешнего вида графиков, включая настройку легенд, меток осей, цветов и т.д.

Для создания графиков с помощью Matplotlib, первым шагом является

импорт необходимых модулей:

Изображение № 17

Затем вы можете создать простой график, используя функции plot()

и show():

Изображение № 18

Matplotlib также предлагает множество возможностей для настройки

внешнего вида графиков. Существует возможность добавить метки для осей, заголовки,

легенды и многое другое с помощью соответствующих функций:

Изображение № 19

Seaborn, основанная на Matplotlib, позволяет легко создавать

привлекательные графики с помощью простых кодовых конструкций. Для использования

Seaborn сначала необходимо импортировать необходимые модули:

Изображение № 20

Затем можно использовать функции Seaborn для создания различных

типов графиков, таких как столбчатые диаграммы, гистограммы, ящики с усами и др. Например, создать столбчатую диаграмму с помощью функции `barplot()`:

`
`

Изображение № 21

``

`
`

Seaborn также предлагает функции для визуализации статистической

информации, например графики ящика с усами для отображения распределения данных и

показателей центра и размаха:

`
`

Изображение № 22

``

`
`

Более глубокие представления о библиотеке Matplotlib будут

реализованы в практической части работы и приложениях.

`
`

2.4. Использование SQL для анализа данных

`
`

SQL является одним из наиболее распространенных языков

программирования для работы с базами данных. Он позволяет эффективно извлекать д

анные из базы данных, выполнять сложные запросы, агрегирующие функции и другие операции.

SQL предоставляет возможность извлечения, обновления, добавления

и удаления данных из базы данных. Он основан на наборе команд и операторов, которые позволяют

работать с различными типами данных, такими как строки, числа, даты и другие. SQL поддерживает

операции выборки данных (SELECT), вставки данных (INSERT), изменения данных (UPDATE) и удаления данных (DELETE).

Оператор SELECT является основным для анализа данных в SQL. Он позволяет

выбрать данные из одной или нескольких таблиц базы данных, исходя из определенных условий.

Ключевое слово SELECT заставляет SQL вернуть определенный набор данных, определенный в запросе.

Вот простой пример оператора SELECT:

Изображение № 23

В примере, приведенном на изображении №22 мы выбираем все данные

из таблицы employees. Звездочка (*) указывает, что мы выбираем все столбцы из таблицы. Можно

также указать конкретные столбцы, которые требуется выбрать в запросе.

Оператор WHERE позволяет фильтровать данные в запросе на основе

определенных условий. Например:

Изображение № 24

В запросе, приведенном на изображении №23, мы выбираем только те

строки таблицы employees, в которых значение столбца age больше 30. Мы можем комбинировать

различные условия, используя логические операторы, такие как AND, OR, NOT.

SQL также предлагает возможности агрегации данных, такие

как сумма, среднее и максимальное значение столбца, или подсчет количества строк,

удовлетворяющих определенным условиям. Например:

Изображение № 25

Запрос, приведенный на изображении №24 возвращает количество сотрудников,

у которых возраст больше 30.

Более сложные операции агрегации могут быть выполнены с помощью операторов

GROUP BY и HAVING. GROUP BY позволяет группировать данные по определенным столбцам, а HAVING

позволяет фильтровать данные, на основе агрегированной информации. Например:

BI инструменты являются мощным средством для анализа данных и вывода результатов, что позволяет отдельным исследователям и организациям использовать свои данные на максимально возможном уровне. Правильное использование BI инструментов помогает принимать обоснованные и информированные решения, а также повышать эффективность операционной деятельности.

В современном мире использование BI инструментов становится все более необходимым для тех, кто хочет

быть конкурентоспособным и успешным.

Одной из основных целей BI - это сделать данные доступными и понятными

для широкого круга пользователей. С помощью BI инструментов пользователи получают возможность

легко анализировать большие объемы данных и создавать аналитические отчеты и дашборды. BI системы

обладают мощными функциями, которые позволяют пользователю запрашивать данные, объединять их,

агрегировать и визуализировать. Это позволяет пользователям просматривать данные в реальном

времени, отслеживать KPI, выявлять тренды и прогнозировать будущие результаты.

BI позволяет пользователям создавать информативные и наглядные графики,

диаграммы и дашборды, которые помогают визуализировать данные и быстро идентифицировать тренды

и паттерны. Это особенно полезно для менеджеров и руководителей, которым нужно принимать оперативные

решения на основе данных.

Еще одним важным аспектом BI инструментов является построение прогнозов и

моделирование данных. С помощью алгоритмов и методов машинного обучения BI инструменты позволяют

пользователям строить прогнозы и моделировать будущие результаты на основе исторических данных.

Это позволяет принимать решения на основе точных прогнозов и планировать свою деятельность

более эффективно.

2.6. Использование HTML/CSS для вывода результата работы

HTML и CSS являются основными базовыми технологическими инструментами для создания веб-страниц. Команда проекта в случае наличия данных технологий в своем портфеле всегда может представить результат своей работы в вебе. Это может послужить альтернативой использованию Git, так как результат будет представлен не в виде репозитория, а в виде аналога книги с навигацией.

С использованием языка разметки HTML можно легко структурировать и организовать содержимое, создавая разделы и подразделы, а также добавляя заголовки и различные элементы форматирования. Кроме того, HTML позволяет добавлять гиперссылки для облегчения навигации между разделами страницы дипломной работы.

CSS в свою очередь предоставляет возможность стилизовать элементы страницы. С помощью CSS можно изменять цвета, шрифты, размеры, добавлять иллюстрации и фоновые изображения, что поможет сделать страницу более привлекательной и профессиональной.

Еще одним важным аспектом создание таблиц. С помощью HTML и CSS можно создавать различные таблицы, которые помогут представить результаты исследования и обобщить информацию. CSS позволяет настраивать внешний вид таблиц, добавлять стили для заголовков и ячеек, а также добавлять разделители и анимацию.

HTML и CSS также позволяют добавлять интерактивные элементы. Если расширить технологический стек с помощью JavaScript, который можно интегрировать в HTML-код, можно создать раскрывающиеся списки, вкладки и кнопки, которые позволят пользователю взаимодействовать с содержимым страницы, например, просматривать дополнительную информацию или скрывать часть содержимого по выбору.

Важным аспектом является также адаптивный дизайн. HTML и CSS позволяют создавать страницы, которые будут автоматически адаптироваться под различные устройства и экраны.

Это значит, что страница отчета будет выглядеть хорошо на компьютере, планшете или мобильном устройстве, положительно влияя на опыт пользователей.

</p>

</div>

</div>

</div>

</section>

</body>

</html>

<head>

<meta charset="UTF-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Глава 2</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<header>

<div class="center">

<div class="logo">

На главную

<!-- GoogleDrive -->

</div>

<div class="menu">

Содержание

Введение

Глава I

<!-- Глава II -->

Заключение

Приложения

Список литературы

<!-- Репозиторий Github

Ссылка -->

</div>

</div>

</header>

```
<section class="main">
```

```
<div class="center">
```

```
<div class="main cta">
```

Глава II. Практическая часть.

<p align="justify">

3.1. Парсер на языке Python с загрузкой данных в SQL формат

Первичной практической задачей при создании поисково-аналитического механизма было принято создание на языке Python обходчика сайтов по формируемой в процессе

исполнения основного файла базе адресов. В качестве первичной базы адресов в сети интернет

был обнаружен и скачан текстовый файл с 5 млн. адресами сайтов рунета (5019692) для дальнейшей

выемки случайным образом необходимого количества строк адресов для проведения эксперимента.

В качестве поэтапного увеличения характеристик эксперимента разработанный обходчик решено

тестировать на линейке из 10, 100, 1000 случайных адресов сайтов с использованием библиотеки random.

Для простоты первичной фазы эксперимента были выбраны встроенные библиотеки `requests` и `sqlite3`

языка Python для работы с веб-страницами и SQL-форматом соответственно. Для подсчета времени

работы программы использовалась библиотека time.

В ходе разработки программа будет создавать базу данных SiteBase

с таблицей SiteTable, в которой будут содержаться текстовые столбцы «www» и «html».

Программа открывает существующий в текущей папке файл base.txt и записывает во вновь создаваемый

файл `addresses.txt` необходимое количество строк, которые представляют из себя адреса веб-страниц.

Далее создается база данных и с каждой строки файла `adresses.txt` мы забираем адрес страницы

в сети интернет, записываем это значение в ячейку столбца «www», соединяемся со страницей

по адресу и записываем ее содержимое в соответствующую ячейку столбца «html» таблицы SiteTable

базы данных. В случае ошибки соединения в столбец «html» таблицы SiteTable записывается значение

NULL. В процессе исполнения программы ведется подсчет количества скачанных сайтов, количества неудавшихся соединений, а также времени исполнения кода.

Когда закончатся строки файла addresses.txt

программа печатает строку "адреса закончились", выводит количество скачанных сайтов, количество неудачных соединений и время, затраченное на скачивание страниц.

Изображение № 27. Первичное состояние перед исполнением программы

Изображение № 28. Код программы

Запускаем код для 10 случайных сайтов, получаем

результат:

Изображение № 29. Результат обработки 10 случайных адресов

Фиксируем появление в папке проекта файлов

addresses.txt и SiteBase.db:

Изображение № 30. Изменения в папке проекта после исполнения программы

Запускаем программу для выборки из 100 адресов, получаем

следующий результат:

Изображение № 31. Результат обработки 100 случайных адресов

Запускаем программу для выборки из 1000 адресов, получаем

следующий результат:

Изображение № 32. Результат обработки 1000 случайных адресов

В процессе разработки для верификации возможности дальнейшей

работы с полученными данными были разработаны простейшие механизмы проверки. Для

начала был написан код на Python, выводящий на печать строки «www» из таблицы

SiteTable базы данных:

Изображение № 33. Код программы для печати содержимого таблицы БД

После запуска кода в терминале происходит печать содержимого

ячеек «www» в виде первичного html кода, с сохранением пробелов и отступов, как это

было реализовано разработчиками сайтов с соблюдением Document Object Model.

Изображение № 34. Печать содержимого БД в терминале

Далее для подтверждения возможности работы с базой данных

в разрезе их анализа была реализована первичная аналитическая программа, позволяющая

выявлять частотность слов на всем массиве содержимого столбцов «html» таблицы SiteTable.

При запуске кода (изображение № 36) мы получаем CSV файл:

Изображение № 35. Содержимое сформированного CSV файла

requirements.txt

pip freeze >

requirements.txt

Все актуальные библиотеки можно посмотреть в файле requirements.txt

- Установить зависимости при клонировании репозитория локально:

```
python -m pip install -r requirements.txt
```

4. Загрузить файл с доменом ru

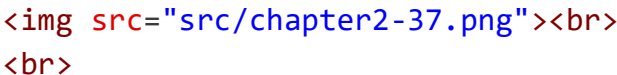
Ссылка на сайт с базой зарегистрированных доменов в приложении.

Скачиваем zip папку и распаковываем ее в нашу папку с проектом.

Изображение № 36. Папка с проектом, содержащая txt файл со списком url - адресов



Изображение № 37. Фрагмент списка с url адресами



В последующем нам придется дописать https://

ко всем строчкам в этом списке, чтобы мы могли зайти на все эти сайты.

Список состоит из 5 млн адресов. Наша задача будет состоять в том, чтобы

спросить как можно больше сайтов. Пройтись по всем не получится, так как

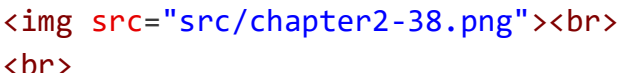
сайты в данном списке могут быть хорошо защищены от парсинга, а могут

и не быть запущены на сервере.

5. Создание скрипта в файле spider_parser.py

Импортируем библиотеки

Изображение № 38. Импорт библиотек



• threading - это стандартная библиотека Python, которая предоставляет инструменты для многозадачности. threading позволяет управлять потоками. Потоки позволяют выполнять несколько задач одновременно, что полезно для выполнения параллельных операций. Мы воспользуемся этой библиотекой, чтобы распараллелить потоки обработки парсинга сайтов.

• requests - это библиотека Python для выполнения HTTP-запросов. Она облегчает отправку HTTP-запросов к веб-серверам и получение ответов. requests позволяет нам взаимодействовать с веб-ресурсами, получать данные, отправлять данные и многое другое.

• csv - это модуль Python для работы с файлами в формате CSV (Comma-Separated Values). Формат CSV используется для хранения табличных данных, где значения разделены запятыми (или другими разделителями). Модуль csv предоставляет функциональность для чтения CSV-файлов, записи данных в CSV-файлы и обработки табличных данных.

• re - это модуль Python для работы с регулярными выражениями. Регулярные выражения (или регулярные выражения) используются для поиска и манипуляции текстовой информацией на основе шаблонов. Модуль re позволит нам отфильтровать полученный текст по русским символам. В итоге мы получим чистый текст с русскими символами.

Создать файл modif_file.py, прописать функцию, которая будет к каждой строке url адресов добавлять https:

Изображение № 39. Функция, модифицирующая файл

Импортируем функцию url_mod в основной файл spider_parser.py

Изображение № 40. Импорт функции

 3.3 Анализ полученных данных
средствами SQL

 Возьмем нашу базу, полученную при
помощи формирования данных
языком Python в CSV – файл и преобразуем для чтения с
помощью SQL. При помощи программы

EXCEL подготовим данные для таблицы.

Изображение № 47. Подготовка данных

 Данные необходимо сохранить в
формате :65001: Юникод (UTF-8),

указать на формат данных: с разделителем. Вы можете
начать импорт с любой строки.

Указать заголовки – выбрав галочкой «Мои данные содержат
заголовки»

(См. Изображение № 48).

Изображение № 48.

 Нажав кнопку далее, определим
разделители. В нашем случае,

это «знак табуляции» и «пробел» (См. Изображение № 49). И
кнопку считать

последовательные разделители одним. Выставляем
ограничение строк до 100.

Изображение № 49.

При количестве строк величиной 100 получается 15100 –
ячеек.

Изображение № 50.

Сохраняем файл в формате CSV (разделитель- запятая).

Изображение № 51.

В файле закаченной базы данных 12924 строк. Разбираем данные по файлам поскольку с большим объемом в 500 строк и в 75500 ячеек MySQL не справился.

Изображение № 52.

Изображение № 53.

12924 строки делим по 150 строк в файл и получаем 85 документов для анализа.

Заходим в MYSQL в базе данных необходимо выбрать кодировки согласно рисунку 8.

Изображение № 54.

Нажимаем правой кнопкой мыши на вкладке справа.

Изображение № 55.

«Table Data Import Wizard» Появляется табло скачивания файла.

Изображение № 56.

Нажимаем кнопку «NEXT». Появляется окно с символами из нашего файла.

Изображение № 57.

Нажимаем кнопку «NEXT».

Изображение № 58.

Таблица появилась в окне «OUTput». Меняем Имя второй колонки, поскольку

наличие синтаксических символов мешают программированию.

Изображение № 63.

Поскольку нам необходимо найти совпадение в тексте

с использованием

примеров.»

Изображение № 64.

Вставляем в заголовок столбец URL, нажимаем на «+»,

во второй столбец пишем «content»

выставляем на

основе 200 строк в окне, разделитель – точка с запятой. –
извлекаем.

Изображение № 65.

Появляется таблица с колонками, выделяем столбец, содержащий строковые

значения, в которых нужно найти повторы.

Изображение № 66.

Нажимаем в верхнем меню во вкладке «Преобразование» кнопку «Извлечение»,

«Текст после разделителя».

Изображение № 67.

Вводим текст в нашем случае это «адрес», «товар», «услуги». С формированием

одноименных столбцов.

Изображение № 68.

[illegible]

[illegible]

и практический опыт, полученные в результате обучения аналитическим специализациям на платформе GeekBrains. Накопленные знания и опыт были применены для решения задач поиска информации в сети Интернет с последующим хранением и анализом данной информации.

Основными компонентами технологического стека, представленного GeekBrains в рамках курсов аналитических специальностей, по мнению членов команды, являются языки Python и SQL, позволяющие решать широкий спектр аналитических задач и разрабатывать как небольшие самостоятельные инструменты, так и целые аналитические комплексы. Также отличным инструментарием для анализа и отображения данных является BI, кратко разобранный в теоретической части. Наибольшее внимание в главе, посвященной теории, было уделено различным библиотекам языка Python, которые позволяют собирать информацию в сети Интернет и анализировать полученные данные. Для библиотек requests, BeautifulSoup, Scrapy, Selenium, Pandas, Matplotlib были даны характеристики и описаны особенности использования данных инструментов. Также в главе представлено описание использования различных форматов хранения полученных данных, таких как CSV и JSON, проведено сравнение использования данных инструментов, в том числе относительно использования SQL-форматов.

В практической части команда попыталась реализовать различные варианты создания парсинговых механизмов и применить несколько подходов для анализа полученных данных. В связи с тем, что проект изначально реализовывался на отдельных локальных машинах, было принято решение ограничиться работой только с текстовой информацией, на заходя в область больших данных. Изначальный файл с адресами для программы-обходчика

В перспективе проект будет реализован командой уже не на локальных машинах. Планируется разворачивание поисково-аналитического механизма на внешних серверах, что придаст проекту совершенно иные возможности. Также в перспективе будут задействованы механизмы обработки, хранения и анализа больших данных, что потребует от команды вложения в проект денежных средств, что должно быть обусловлено практической обоснованностью проекта в коммерческом плане.

</p>

</div>

</div>

</div>

</section>

</body>

</html>

<head>

<meta charset="UTF-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Список литературы</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<header>

<div class="center">

<div class="logo">

На главную

<!-- GoogleDrive -->

</div>

<div class="menu">

Содержание

Введение

Глава I

Глава II


```

        <a href="conclusion.html">Заключение</a>
        <a href="appendix.html">Приложения</a>
        <!-- <a href="literature.html">Список литературы</a> -->
        <!-- <a
href="https://github.com/BoginskiyVV/webpageDiploma">Репозиторий Github</a>
        <a class="btn-menu" href="#">Ссылка</a> -->
    </div>
</div>
<!--Center-->
</header>
</header>
<section class="main">
    <div class="center">
        <div class="main_cta">
            <h2>Список использованной литературы</h2>
            <p align="justify">
                <br>
                <!-- &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<b>Заключение</b><br><br>
-->

                <br>
                1. Беляева И. В. Архитектура информационных
                систем: учебное пособие / И. В. Беляева. – Ульяновск: УлГТУ,
                2019. 192 с.<br>
                2. Митчелл Райан. Современный скрапинг
                веб-сайтов с помощью Python. Санкт-Петербург: Издательство
                «Питер», 2021. 497 с.<br>
                3.Равив Г. Power Query в Excel и Power
                BI / Г. Равив. – Санкт-Петербург: БХВ, 2021. 480 с.<br>
                4. База зарегистрированных доменов разбитая
                по зонам [Электронный ресурс].
                URL: https://purecrawl.com/download/domains
                (дата обращения: 12.10.2023).<br>
                5. Базы данных. Практическое применение
                СУБД SQL и NoSQL-типа для проектирования информационных
систем:
                учеб. пособие / С.А. Мартишин, В.Л. Симонов, М.В. Храпченко.
                – Москва: «ФОРУМ», 2018. 368 с. [Электронный ресурс].
                URL: http://znanium.com/ catalog/product/926871/(дата
обращения:
                23.09.2023).<br>
                6. Библиотека Pandas // ru.wikipedia.org
                [Электронный ресурс]. URL: ru.wikipedia.org/wiki/Pandas
                (дата обращения: 12.10.2023).<br>
                7. Библиотека Requests // python.ru

```

[Электронный ресурс]. URL: python.ru/post/97/ (дата обращения: 12.10.2023).

8. Библиотека Python для извлечения данных из файлов HTML и XML [Электронный ресурс]. URL: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/> (дата обращения: 25.09.2023)

9. Гуриков, С. Р. Основы алгоритмизации и программирования на Python : учеб. пособие / С.Р. Гуриков. – Москва: «ФОРУМ», 2018. 343 с. [Электронный ресурс]. URL: <https://znanium.com/catalog/product/924699> (дата обращения: 20.01.2020).

10. Документация по BeautifulSoup // wiki.python.org/su [Электронный ресурс]. URL: wiki.python.org/su (дата обращения: 12.10.2023).

11. Краткое руководство по веб-парсингу [Электронный ресурс]. URL: <https://habr.com/ru/companies/selectel/articles/754674/> (дата обращения: 18.10.2023)

12. Метод загрузки больших файлов GitHub [Электронный ресурс]. URL: <https://all-python.ru/osnovy/threading.html?ysclid=lmxayom1wc333926930> (дата обращения: 10.10.2023).

13. Миронов М. «Парсинг»: Лучшие программы для парсинга / М. Миронов // idatica.com: [Электронный ресурс]. URL: [https:// idatica.com/ blog/programmy-dlya-parsinga-dannykh-v-2020-godu](https://idatica.com/blog/programmy-dlya-parsinga-dannykh-v-2020-godu) (дата обращения: 25.09.2023)

14. Питинов А. «Парсер простыми словами»: как его настроить и пользоваться / А. Питинов // semantica.in: [Электронный ресурс]. URL: [https:// semantica.in/blog/chto-takoe-parser.html](https://semantica.in/blog/chto-takoe-parser.html) (дата обращения: 02.10.2023)

15. Парсинг Json в Python 3 // all-python.ru [Электронный ресурс]. URL: all-python.ru/osnovy/json.html (дата обращения: 12.10.2023).

16. Руководство разработчика MySQL Connector/NET [Электронный ресурс]. URL: <https://dev.mysql.com/doc/connector-net/en/> (дата обращения 16.10.2023)

ресурс].

17. Северянин С. «Парсинг»: Что такое парсер и как он работает / С. Северянин // timeweb.ru: [Электронный ресурс].
URL: [https:// timeweb.com/ru/community/articles/chto-takoe-parser](https://timeweb.com/ru/community/articles/chto-takoe-parser)
(дата обращения: 04.10.2023)

18. Pandas – библиотека Python [Электронный ресурс].
URL: [https:// blog.skillfactory.ru/glossary/pandas/](https://blog.skillfactory.ru/glossary/pandas/)
(дата обращения: 05.10.2023)

19. Pandas [Электронный ресурс].
URL: [https://pandas.pydata.org/ getting_started.html/](https://pandas.pydata.org/getting_started.html/)
(дата обращения: 05.10.2023)

20. Parser [Электронный ресурс].
URL: [https://github.com /FiryuzaLapteva/Parser](https://github.com/FiryuzaLapteva/Parser)
(дата обращения: 15.10.2023)

21. SQL или NoSQL [Электронный ресурс].
URL: [https://habr.com/ru /company/ruvds/blog/324936/](https://habr.com/ru/company/ruvds/blog/324936/) (дата обращения: 07.10.2023).

Threading в Python 3: работа с потоками [Электронный ресурс].
URL: <https://all-python.ru/osnovy/threading.html?ysclid=lmxayom1wc333926930>
(дата обращения: 11.10.2023).

22. Threading в Python 3: работа с потоками [Электронный ресурс].
URL: <https://all-python.ru/osnovy/threading.html?ysclid=lmxayom1wc333926930>
(дата обращения: 11.10.2023).

</p>

</div>

</div>

</div>

</section>

</body>

</html>