**\*\*Тестовое задание для стажера-программиста Python на разработку кода парсера на фреймворке Scrapy\*\***

Цель задания: Разработать парсер для сбора общедоступных данных с открытых веб-страниц из списка сайтов.

Задание:

1. Необходимо разработать парсер, который импортирует список url-адресов из txt файла domains.txt (<https://disk.yandex.ru/i/zogXqN9rE3jIMw>) и проверяет по списку запрещённых url-адресов block\_domains.txt [(https://disk.yandex.ru/i/UiSzb6IV\_iyY7A](https://disk.yandex.ru/i/UiSzb6IV_iyY7A) ) . Если url-адрес разрешён, то парсер переходит по данному url-адресу и выполняет поиск, и извлечение следующих данных с помощью селекторов XPATH, CSS, REGEX:

- Заголовок страницы (title)

- Мета-описание (description)

- Электронная почта (шаблон)

- Контактный телефон (шаблон)

- Почтовый адрес (шаблон)

- ИНН (шаблон)

- ОГРН (шаблон)

Примечание:

*Под шаблоном в контексте извлечения данных при парсинге понимается регулярное выражение, описывающее нужный формат данных.*

*Например, для поиска электронных адресов можно использовать такой шаблон:*

*“\b[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}\b”*

*Это регулярное выражение описывает общий формат email вида:*

*имя\_пользователя@имя\_домена.домен\_верхнего\_уровня*

*Основные принципы использования шаблонов:*

*- Шаблон задаёт последовательность символов, которая идентифицирует искомый формат данных.*

*- В шаблоне используются специальные символы регулярных выражений для описания вариантов и повторений.*

*- Шаблон более гибок по сравнению с точным указанием формата.*

*- С помощью шаблонов можно найти данные в тексте, даже если формат слегка варьируется.*

*- Шаблоны удобно использовать для поиска email, телефонов, дат и других структурированных данных.*

1. Глубина парсинга - не более 2 уровней от стартовой страницы.
2. Результаты парсинга сохранять в CSV файл. Формат: названия колонок в первой строке, кодировка UTF-8.
   1. Если данные не найдены, добавить в строку - н/д;
   2. Исключить дублирование данных;
3. Обрабатывать ошибки при парсинге, чтобы скрипт не падал.
4. Включить логирование ошибки. (Такую функцию можно активировать в Scrapy)
5. Реализовать многопоточность для ускорения работы парсера. (опционально)
6. Добавить таймаут для запросов (например, 3 секунд).
7. Оформить код с комментариями и в соответствии со стандартами PEP8.

Оценка результата:

- Корректность и полнота собранных данных

- Производительность и скорость работы парсера

- Структурированность и читаемость кода

- Обработка ошибок и исключений

Срок выполнения: 3-5 дней.

Примечания:

Для выполнения задания рекомендуется использовать Python библиотеку Scrapy, Selenium, в крайнем случае  BeatifulSoup4.

При сдаче задания, предоставьте исходный код вашего паука и собранные данные отправьте в телеграмм на @OptimumPrice\_ru на электронную почту [info@optimumprice.ru](mailto:info@optimumprice.ru) .

Удачи в выполнении задания! Если у вас возникнут вопросы или требуется

дополнительная информация - не стесняйтесь обращаться.

Справочная информация:

https://digitology.tech/docs/scrapy/index.html

<https://docs.scrapy.org/en/latest/intro/tutorial.html>

https://pythonru.com/biblioteki/sozdanie-parserov-s-pomoshhju-scrapy-i-python

<https://newtechaudit.ru/scrapy-dlya-obhoda-beskonechnogo-skrolla-pri-parsinge-vc-ru/>

<https://pycoder.ru/make-simple-spider-scrapy/>

<https://scrapfly.io/blog/web-scraping-with-scrapy/>