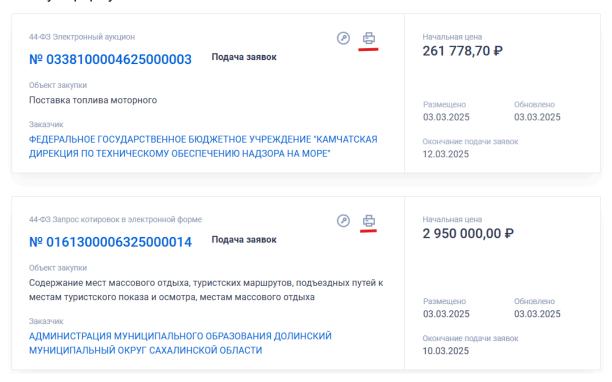
Сайт государственных закупок (ЕИС), главная страница поиска тендеров: https://zakupki.gov.ru/epz/order/extendedsearch/results.html, с постраничной разбивкой по 10 тендеров.

Нужно реализовать программу на Python, которая обходит первые две страницы по 44Ф3:

- https://zakupki.gov.ru/epz/order/extendedsearch/results.html?fz44=on&pageNumber
 =1
- https://zakupki.gov.ru/epz/order/extendedsearch/results.html?fz44=on&pageNumber
 =2

При обходе, у каждого элемента списка (тендера), нужно собирать ссылку на его печатную форму:



Эти ссылки имеют вид

https://zakupki.gov.ru/epz/order/notice/printForm/view.html?regNumber=0338100004625000 003.

Заменив view.html на viewXml.html, получим ссылку на печатную XML-форму (https://zakupki.gov.ru/epz/order/notice/printForm/viewXml.html?regNumber=033810000462 5000003). Распарсив этот XML, для каждого тендера нужно получить значение XML-поля publishDTInEIS, или None в случае его отсутствия.

Для простоты, результат вывести прямо в консоль (например, обычным print()), в виде пары "ссылка на печатную форму"-"дата публикации".

Рекомендуемый стек:

- Python3
- пакет Requests https://docs.python-requests.org/en/latest/ для сетевых запросов
- пакет BeautifulSoup https://beautiful-soup-4.readthedocs.io/en/latest/ для парсинга HTML и получения ссылок на печатные формы
- пакет XmlToDict https://github.com/martinblech/xmltodict для парсинга XML

Пакеты из списка выше - лишь рекомендации, если удобнее, то можно, например, парсить XML через методы стандартной библиотеки <u>xml.etree.ElementTree</u>, или парсить HTML через <u>xml/XPath</u>.

Основной трудный момент этого тестового задания - нужно распараллелить задачи обхода списка и парсинга XML-форм, разбить написанный выше скрипт на отдельные таски. Для этого нужно использовать пакет \mathtt{Celery} -

https://docs.celeryproject.org/en/latest/ . Рекомендуемая разбивка на (асинхронные) подзадачи:

- сбор ссылок с каждой страницы отдельный таск (для каждой страницы)
- парсинг печатных XML-форм отдельный таск (для каждой формы)

Для тестирования рекомендуется использовать т.н. eager-режим (синхронное выполнение) - https://docs.celeryproject.org/en/stable/userguide/testing.html

Плюсом при оценке программы будет использование классов для тасков Celery:

```
import celery
class Task(celery.Task):
    def run(self):
        # ваш код здесь
```

•••

Результат выложить на Г	Гитхаб или прислать	на почту ir@lmx.su в виде а	архива.
-------------------------	---------------------	-----------------------------	---------

По всем вопросам можно писать мне на почту ir@lmx.su (Иван) я буду оперативно отвечать.