МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота №1 з дисципліни
“Бази даних. Частина 2”
тема “Вивчення базових операцій обробки XML-документів”

Варіант 1

Виконав
студент III курсу
групи КП-82
Анікєєв Ігор Анатолійович
Посилання на репозиторій:https://github.com/flain1/bdlabs2

Мета

Метою роботи є здобуття практичних навичок створення програм, орієнтованих на обробку XML-документів засобами мови Python.

Постановка завдання

1. На основі базової адреси Web-сайту виконати обхід наявних сторінок сайту, відокремлюючи текстову та графічну інформацію від тегів HTML. Пошук вузлів виконувати засобами XPath. Наступну сторінку для аналізу **цього ж сайту** обрати як одне із гіперпосилань на даній сторінці (тег <a href=”url”/>). Обмежитись аналізом 20 сторінок сайту. Зберегти XML у вигляді файлу. Формат XML-документу:
2. Виконати аналіз отриманих даних засобами XML згідно варіанту та вивести результати у консольне вікно. Відбір вузлів та розрахунки за варіантом **виконувати засобами XPath**.
3. Проаналізувати вміст Web-сторінок інтернет-магазину (див. варіант). Отримати ціну, опис та зображення для 20 товарів з нього за допомогою DOM-парсеру та мови XPath для пошуку відповідних вузлів. Результат записати в XML-файл.
4. Перетворити отриманий XML-файл у XHTML-сторінку за допомогою мови XSLT. Дані подати у вигляді XHTML-таблиці та записати його у файл.

Сайт для пунктів 1-2: https://kpi.ua

Завдання для пункту 2: Вивести максимальну кількість текстових елементів.

Сайт для пунктів 3-4: https://rozetka.com.ua/

Приклади коду

part_1.py

```
from typing import List from
urllib.request import urlopen
from lxml import etree
import xml.etree.cElementTree as ET

INITIAL_URL = "https://kpi.ua"


def parse_kpi_website() -> int:
    """Scrape images and text from pages of the KPI
website. Save to an XML file.
    Return the total number of text tags.
    """ xml_root =
    ET.Element("data") htmlparser
    = etree.HTMLParser()

    # Parse INITIAL_URL and get a list of the
subsequent URLs to be parsed.
    urls_to_parse: List[str] =
parse_initial_page(xml_root, htmlparser)

    # Go through the list of URLs and append them to
the XML root.
    for url in urls_to_parse:
        url_to_parse = INITIAL_URL + url
        response = urlopen(url_to_parse) tree =
        etree.parse(response, htmlparser)
        parse_url(xml_root, tree, INITIAL_URL)

    # Write all the parsed pages to an XML file
    xml_tree = ET.ElementTree(xml_root)
    xml_tree.write("kpi_website.xml",
```

```
encoding="UTF-8")

   text_tags_count =
len(xml_tree.findall(".//fragment[@type='text']"))

   return text_tags_count


def parse_initial_page(xml_root, htmlparser) ->
List[str]:
   """ Scrape data from `INITIAL_URL` and
determine which other urls should be parsed. """
response = urlopen(INITIAL_URL) tree =
etree.parse(response, htmlparser) urls =
tree.xpath("//a/@href") urls_to_parse = urls[1:20]
parse_url(xml_root, tree, INITIAL_URL) return
urls_to_parse


def parse_url(xml_root, tree, url) -> int:
   """ Extract all the text and image elements from
a webpage and appends them to an XML file.

       Return the number of text elements found on
the page.
    """ text_pieces:
   List[str] = [ text_piece
   for text_piece in
(tree.xpath("//body//text()")) if
       any(char.isalpha() for char in text_piece)
   ]
```

```
   image_urls: List[str] = [image.attrib["src"] for
image in tree.xpath("//body//img")]

   # Create a new "page" entry for the output XML
doc page = ET.SubElement(xml_root, "page",
url=url) for text in text_pieces:
       ET.SubElement(page,
"fragment", type="text").text = text
for image_url in image_urls:
       ET.SubElement(page,
"fragment", type="image").text =
image_url return len(text_pieces)


if __name__ == "__main__":
   number_of_text_tags = parse_kpi_website()

   print(f"Total number of text tags accumulated:
{number_of_text_tags}")
```

part2.py

```
import json import os import
webbrowser from typing import
List, Tuple from urllib.request
import urlopen from lxml import
etree import
xml.etree.cElementTree as ET
```

```
INITIAL_URL =
"https://rozetka.com.ua/chateau_de_montifaud_3550142
637970/p74680656/"


def parse_rozetka_website() -> None:
    """Scrape images and text from pages of the KPI
website. Save to an XML file.
    Return the total number of text tags.
    """ xml_root =
    ET.Element("shop") htmlparser
    = etree.HTMLParser()

    # Parse INITIAL_URL and get a list of the
subsequent URLs to be parsed.
    urls_to_parse: List[str] =
parse_initial_page(xml_root, htmlparser)

    # Go through the list of URLs and append them to
the XML root.
    for url in urls_to_parse:
        response = urlopen(url) tree =

        etree.parse(response, htmlparser)

        parse_url(xml_root, tree, INITIAL_URL)

    # Write all the parsed pages to an XML file
    xml_tree = ET.ElementTree(xml_root)
    xml_tree.write("rozetka_website.xml",
encoding="UTF-8")

    transform =
etree.XSLT(etree.parse("./transform.xsl"))
    result =
```

```
transform(etree.parse("./rozetka_website.xml"))
   result.write("./parsed_rozetka.xhtml",
pretty_print=True, encoding="UTF-8")
   webbrowser.open('file://' +
os.path.realpath("./parsed_rozetka.xhtml"))


def parse_initial_page(xml_root, htmlparser) ->
List[str]:
   """ Scrape data from `INITIAL_URL` and
determine which other urls should be parsed. """
response = urlopen(INITIAL_URL) tree =
etree.parse(response, htmlparser) urls =
tree.xpath("//a[@class='lite-

   tile__title']/@href") urls_to_parse =

   urls[1:20] parse_url(xml_root, tree,

   INITIAL_URL) return urls_to_parse


def parse_url(xml_root, tree, url) -> None:
   """ Extract from Rozetka product page name,
image, description and price of the product.
       Create an entry in the XML tree.
    """ product_name: str =
   parse_product_name(tree) product_image_url:
   str =
parse_product_image_url(tree)
   product_description, product_price =
parse_product_description_and_price(tree)

   # Create page entry in the XML tree page
   = ET.SubElement(xml_root, "product",
```

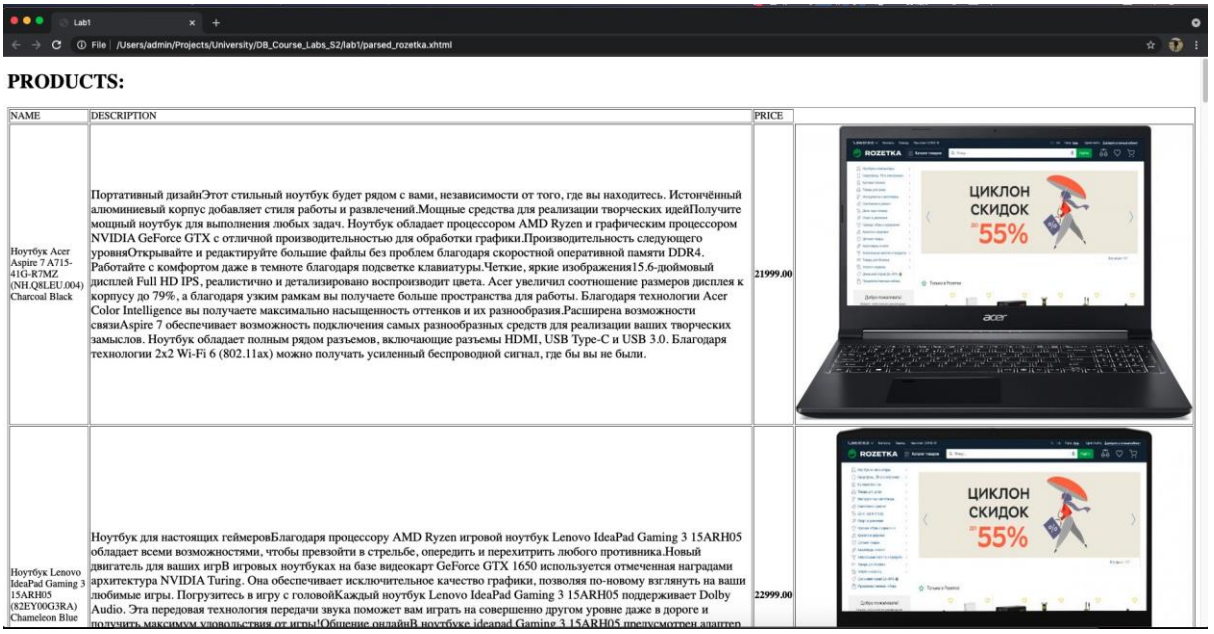
```
url=url)
    ET.SubElement(page, "name").text = product_name
    ET.SubElement(page, "description").text =
product_description
    ET.SubElement(page, "image").text =
product_image_url
    ET.SubElement(page, "price").text = product_price


def parse_product_description_and_price(tree) ->
Tuple[str, str]:
    """ Fetch and decode the product description from
Rozetka """
    product: str =
tree.xpath("//script[@data-seo='Product']")[0].text
    product: dict = json.loads(product)
    product_description: str = product['description']
    product_description: str =
product_description.encode('latin1').decode('utf8')

    offers: dict = product['offers']

    price: str = offers['price']

    return product_description, price


def parse_product_name(tree):
    """ Get product's name from its Rozetka page """
    product_name: str =
tree.xpath("//h1[@class='product__title']/text()")[0
] return
product_name.encode('latin1').decode('utf8')
```

```
def parse_product_image_url(tree):
    """ Get product's image URL from Rozetka """
    return
tree.xpath("//img[@class='product-photo__picture']/@
src")[0]


def cleanup():
    try:
        os.remove("parsed_rozetka.xhtml")
        os.remove("rozetka_website.xml")
    except OSError:
        pass


if __name__ == "__main__":
    cleanup()
    parse_rozetka_website()
```

Скріншоти роботи програми

```
/Users/admin/Projects/University/DB_Course_Labs_S2/lab1/venv/bin/python /Users/admin/Projects/University/DB_Course_Labs_S2/lab1/part_1.py
Total number of text tags accumulated: 4432
Page with most text elements: https://kpi.ua/almamater; Number of elements: 302
```

Завдання 1-2



PRODUCTS:

NAME	DESCRIPTION	PRICE	
Ноутбук Acer Aspire 7 A715-41G-R7MZ (NH.Q8LEU.004) Charcoal Black	Портативный дизайнЭтот стильный ноутбук будет рядом с вами, независимости от того, где вы находитесь. Источённый алюминиевый корпус добавляет стиля работы и развлечений.Мощные средства для реализации творческих идейПолучите мощный ноутбук для выполнения любых задач. Ноутбук обладает процессором AMD Ryzen и графическим процессором NVIDIA GeForce GTX с отличной производительностью для обработки графики.Производительность следующего уровняОткрывайте и редактируйте большие файлы без проблем благодаря скоростной оперативной памяти DDR4. Работайте с комфортом даже в темноте благодаря подсветке клавиатуры.Четкие, яркие изображения15.6-дюймовый дисплей Full HD IPS, реалистично и детализировано воспроизводит цвета. Acer увеличил соотношение размеров дисплея к корпусу до 79%, а благодаря узким рамкам вы получаете больше пространства для работы. Благодаря технологии Acer Color Intelligence вы получаете максимально насыщенность оттенков и их разнообразия.Расширена возможности связиAspire 7 обеспечивает возможность подключения самых разнообразных средств для реализации ваших творческих замыслов. Ноутбук обладает полным рядом разъемов, включающие разъемы HDMI, USB Type-C и USB 3.0. Благодаря технологии 2x2 Wi-Fi 6 (802.11ax) можно получать усиленный беспроводной сигнал, где бы вы не были.	21999.00	
Ноутбук Lenovo IdeaPad Gaming 3 15ARH05 (82EY00G3RA) Chameleon Blue	Ноутбук для настоящих геймеровБлагодаря процессору AMD Ryzen игровой ноутбук Lenovo IdeaPad Gaming 3 15ARH05 обладает всеми возможностями, чтобы превзойти в стрельбе, опередить и перехитрить любого противника.Новый двигатель для ваших игрВ игровых ноутбуках на базе видеокарт GeForce GTX 1650 используется отмеченная наградами архитектура NVIDIA Turing. Она обеспечивает исключительное качество графики, позволяя по-новому взглянуть на ваши любимые игры. Погрузитесь в игру с головойКаждый ноутбук Lenovo IdeaPad Gaming 3 15ARH05 поддерживает Dolby Audio. Эта передовая технология передачи звука поможет вам играть на совершенно другом уровне даже в дороге и получить максимум удовольствия от игры!Общение онлайнВ ноутбуке ideapad Gaming 3 15ARH05 предусмотрен адаптер	22999.00	

Завдання 3-4

Висновок

У результаті роботи було відпрацьовано навички створення програм, орієнтованих на обробку XML-документів засобами мови Python.