**МІЖНАРОДНИЙ ГУМАНІТАРНИЙ УНІВЕРСИТЕТ**

**Факультет кібербезпеки програмної інженерії та комп’ютерних наук**

**Кафедра комп’ютерних наук**

**ЗВІТ**

про проходження навчальної практики

здобувача освітнього ступеню «бакалавр»

на базі практики освітнього фонду “KeepSolid”

Виконав: студент III курсу

Спеціальності

123 «Комп’ютерна інженерія»

Андрушко Ярослав Васильович

Керівник Вакарчук А. О.

Одеса – 2023

**ЗМІСТ**

Стор.

[ВСТУП 3](#_Toc155030261)

[1 ОПИС БАЗИ ПРАКТИКИ 4](#_Toc155030262)

[1.1 Відомості про компанію 4](#_Toc155030263)

[1.2 Мета та завдання практик 6](#_Toc155030264)

[2 ОПИС ПРОЕКТУ 7](#_Toc155030265)

[2.1 Загальні відомості 7](#_Toc155030266)

[2.2 Мета проекту та цільова аудиторія 8](#_Toc155030267)

[3 РЕАЛІЗАЦІЯ ПРОЕКТНОЇ РОБОТИ 9](#_Toc155030268)

[3.1 Налаштування оточення 9](#_Toc155030269)

[3.2 Розробка програми 11](#_Toc155030270)

[ВИСНОВОК 19](#_Toc155030271)

[ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 20](#_Toc155030272)

[ДОДАТОК А – Лістинг файлу bot.py 21](#_Toc155030273)

# **ВСТУП**

У даному документі документі описаний звіт про проходження навчальної практики студента Андрушко Ярослава Васильовича, який проходив на базі практики освітнього фонду “KeepSolid”.

Згідно навчального плану виробнича практика є невід’ємною частиною підготовки студентів професійного спрямування «Комп’ютерна інженерія». Дана практика проводиться на 3 курсі.

Практика забезпечує узагальнення та поглиблення вивчених фахових дисциплін, формування їх взаємозв'язків і проявів у різних сферах програмування, розуміння студентами особливостей застосування теоретичних положень у практичній діяльності, їх адаптацію до умов ринку праці та вимог потенційних роботодавців.

Основним завданням практики було отримання практичних навичок по теоретичній базі, яку ми отримали протягом навчання. В рамках цього завдання я розробляв, тестував, влагоджував особистий програмний продукт для месенджеру Telegram.

# **1 ОПИС БАЗИ ПРАКТИКИ**

* 1. **Відомості про компанію**

Організація ТОВ "КІПСОЛІД УКРАЇНА" зареєстрована 04.07.2011 за юридичною адресою Україна, \*\*4, Одеська обл., місто Одеса, ПРОСПЕКТ ШЕВЧЕНКА, будинок \*\*А. Керівником організації є БОЧАРНІКОВ ОЛЕГ ПАВЛОВИЧ. Розмір статутного капіталу складає 80 000,00 грн. На момент останнього оновлення даних 22.12.2022 стан організації – Не перебуває в процесі припинення.

Види діяльності:

* 1. Комп'ютерне програмування.

77.33 Надання в оренду офісних машин і устаткування, у тому числі комп'ютерів.

58.21 Видання комп'ютерних ігор.

58.29 Видання іншого програмного забезпечення

* 1. Консультування з питань інформатизації.

62.09 Інша діяльність у сфері інформаційних технологій і комп'ютерних систем.

63.11 Оброблення даних, розміщення інформації на веб-вузлах і пов'язана з ними діяльність.

* 1. Веб-портали.
  2. Видання інших інформаційних послуг.

68.20 Надання в оренду й експлуатацію власного чи орендованого нерухомого майна.

Місцем проходження практики був освітній фонд “Кіпсолід” де я вже мав змогу проходити практику.

У цьому році через труднощі практика проходила у дистанційній формі, але це не як не заважало набувати нові знання та навички. Освітній фонд “кіпсолід” був офіційно зареєстрований у 2021 році. Але до цієї події було 15 років співпраці з навчальними закладами міста Одеса, досвід 8 років програми “Літня інтернатура”, 4 роки програми “Вступ до спеціальності” для першокурсників в ІТ спеціальностей ВНЗ Одеси, створення студентських лабораторії у ВНЗ Одеси, спроби співпрацювати з ІТ кластерами та Асоціаціями. Окрім досвіду створення програм для студентів, за ці роки компанія KeepSolid набула й розуміння обмеженості програм, спрямованих лише на відбір найкращих студентів.

Місія фонду – це поліпшити рівень життя в Україні шляхом підвищення рівня ІТ освіти та підготовки спеціалістів, здатних створювати передові ІТ-продуктів.

* 1. **Мета та завдання практик**

Метою практики є набуття та закріплення практичних навичок, закріплення теоретичних знань та можливість використати їх на практиці. Вивчення та аналіз ринку, дослідження доцільності використання тих чи інших технологій в тих чи інших випадках. Студенті узагальнюють свої дослідження, знаходять рішення на поставлені проблеми, та допомагають їх впровадити на підприємстві або організації.

Завданням практики є вивчення:

* Засобів пошуку, порівняння та опрацювання інформації при виборі найбільш прийнятних систем, технологій та програм за економічними та технічними характеристиками.
* Працювати зі спеціальною технічною та науковою літературою, та правилами оформлення технічної документації з програмного забезпечення.
* опанувати розробку концепцій комп‘ютерних систем, проектування баз знань та даних, прикладного програмного забезпечення інформаційних систем і технологій, розробку документації на програмне забезпечення.

# **ОПИС ПРОЕКТУ**

* 1. **Загальні відомості**

Чат-бот – це автоматизований багатофункціональний помічник, який може збирати та показувати інформацію [підписникам](https://sendpulse.ua/knowledge-base/chatbot/manage-user-data#telegram) за їхнім запитом згідно зі заздалегідь підготовленим сценарієм.

Найчастіше чат-боти використовуються для автоматизації обслуговування клієнтів. Боти можуть швидко та ефективно відповідати на запитання та запити клієнтів 24/7, автоматично збирати зворотний зв'язок від клієнтів після покупки чи обслуговування. Для цього вам необхідно створити сценарій бота з необхідними командами, які вибиратиме підписник для розв'язання своїх питань.

Боти можуть надсилати миттєві повідомлення через [розсилки](https://sendpulse.ua/knowledge-base/chatbot/telegram/send-telegram-campaign) про нові акції, події або важливі оновлення бізнесу, що допомагає тримати клієнтів у курсі подій та підвищувати впізнаваність бренду.

Навіщо використовувати чат ботів?

* На 92% збільшився обсяг ринку чат-ботів за останні кілька років. Чат-боти є найшвидшим каналом комунікації у брендів.
* 87,2% користувачів-клієнтів повідомляють про нейтральний або позитивний досвід роботи з чат-ботами.
* 58% B2B-компаній використовують чат-ботів, B2C-бізнеси – у 42% випадків.
* До 90% відгуків може згенерувати взаємодія чат бота в Телеграм з найбільш зацікавленими клієнтами. У середньому ж чат-боти генерують 35-40% відповідей.
* За словами 68% користувачів-клієнтів, їм подобаються чат-боти, тому що вони дають швидкі відповіді.

## **Мета проекту та цільова аудиторія**

Метою проекту у мене був телеграм бот для пошуку товарів в продовольчих магазинах для порівняння цін та знаходження найдешевшого варіанту.

Цільовою аудиторією цього боту може бути будь-яка людина, адже всі люблять економити та купляти щось дешевше. Тому цей бот має стати корисним помічником при рішенні питання того, де купити той чи інший товар.

# **РЕАЛІЗАЦІЯ ПРОЕКТНОЇ РОБОТИ**

* 1. **Налаштування оточення**

Перше, що потрібно було для себе зрозуміти – це стек технологій, які мали б використовуватись при розробці чат боту. Я вирішив використовувати мову програмування Python. Більшість ботів для телеграму пишуться на потужній мові з легким синтаксисом – Python. Завдяки мові Python і бібліотеці pyTelegramBotAPI можна легко розробляти ботів із різноманітним функціоналом, від простих луна-ботів до складніших із кнопками та обробкою користувацького введення.

Так як це сумісний проект мене та Артема Тараненко, то ми поділили обов’язки. Я маю написати основну логіку боту, а Артем інтегрувати це в телеграм бота.

Для написання коду я використав IDE PyCharm. PyCharm – це кроссплатформенна інтегрована середовище розробки мови програмування Python, розроблена компанією JetBrains на основі IntelliJ IDEA. Надає користувачеві комплекс засобів для написання коду та візуальний відладчик.

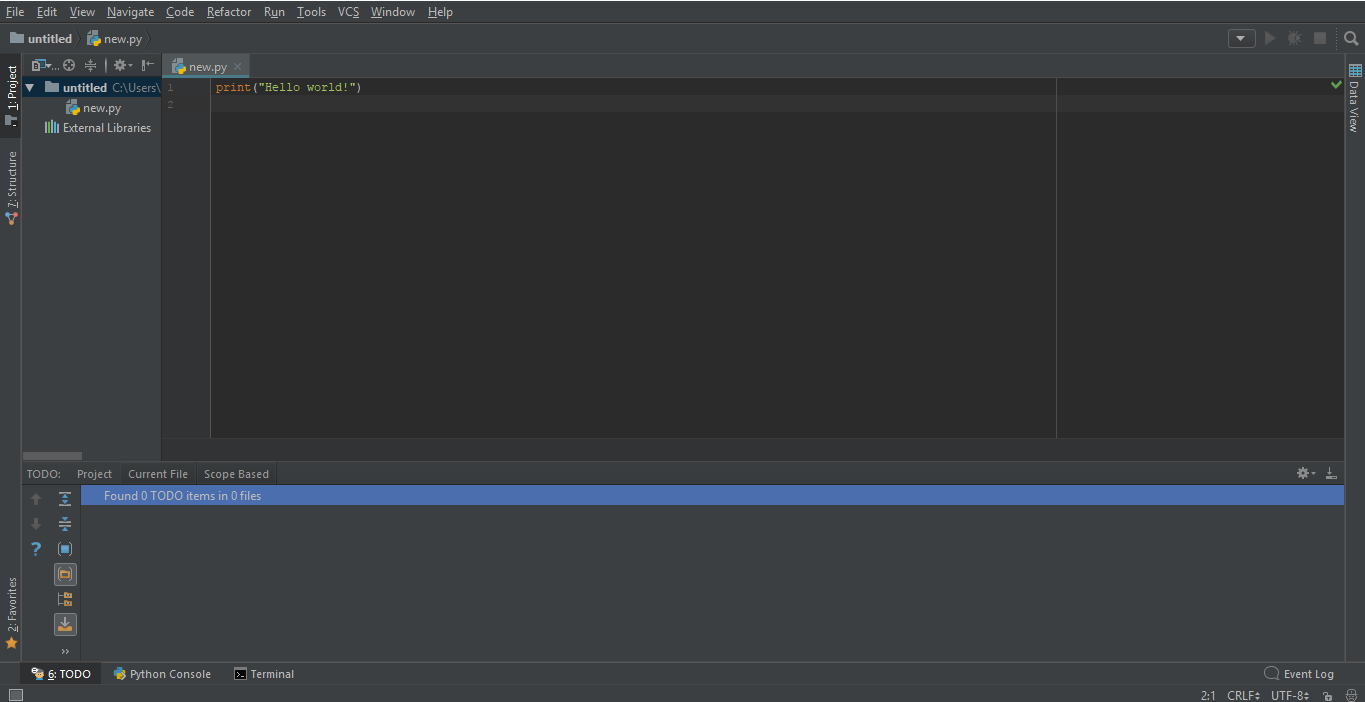


Рисунок 1 – Інтерфейс PyCharm

Перед початком написання коду потрібно було встановити бібліотеку pyTelegramBotAPI для подальшої праці. [pyTelegramBotAPI](https://pytba.readthedocs.io/en/latest/index.html) або [telebot](https://pytba.readthedocs.io/en/latest/index.html) – це простий та зручний фреймворк для створення Telegram ботів на Python. Він дозволяє швидко розробляти прості боти, але має обмежений функціонал порівняно з іншими фреймворками.

Для того щоб встановити бібліотеку pyTelegramBotAPI треба ввести команду: “pip install pyTelegramBotAPI“. Після встановлення бібліотеки потрібно її імпортувати разом з іншими бібліотеками.

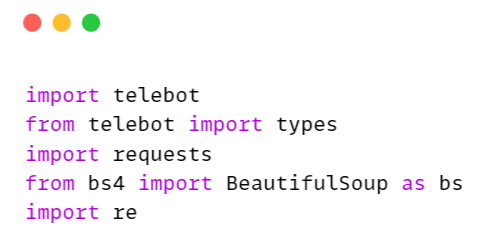


Рисунок 2 – Код імпортування бібліотек для проекту

На цьому рисунку є наступні команди:

* Import telebot – додавання бібліотеки pyTelegramBotAPI.
* From telebot import types – додавання класів та об’єктів для створення кнопок, клавіатур та медіа файлів.
* Import requests – додавання бібліотеки для роботи з HTTP запросами.
* From bs4 import BeautifulSoup as bs – бібліотека для парсингу HTML та XML бібліотек.
* Import re – додавання модуля регуляргних виразів.

## **3.2 Розробка програми**

Після імпортування бібліотек я почав розробляти код, який мав би підключатися до сайтів за допомогою запитів, які будуть вводити користувачі. На прикладі наступного коду я поясню як саме працює алгоритм.



Рисунок 3 – Код отримання запиту та входу на сайт магазину.

Цей код взаємодіє з веб-сайтом ATB Market, щоб здійснити пошук продуктів за певним запитом.

def search\_in\_atb(chat\_id): – ця стрічка визначає функцію search\_in\_atb з одним параметром chat\_id.

zapit = user\_inputs[chat\_id]['query'] – отримує значення запиту (пошукового запиту) для конкретного chat\_id зі збережених користувацьких вводів.

url\_atb=f"https://www.atbmarket.com/sch?page=1&lang=uk&location=1154&query={zapit}" – формує URL-адресу для пошуку на веб-сайті ATB Market з використанням отриманого запиту.

html\_content\_atb = get\_html\_content(url\_atb) – викликає функцію get\_html\_content, яка отримує вміст HTML сторінки за вказаним URL-адресом.

products\_atb=extract\_atb\_data(html\_content\_atb) – викликає функцію extract\_atb\_data, яка, ймовірно, обробляє HTML-вміст сторінки та витягає інформацію про продукти.

send\_results(chat\_id,"АТБ",products\_atb) – викликає функцію send\_results з параметрами chat\_id, назвою магазину ("АТБ") та отриманими даними про продукти products\_atb. Це використовується для відправки результатів пошуку користувачеві.

Після тестування коду я зрозумів, що є якась помилка, яка не дозволяє отримати інформацію зі сторінки. Для тесту я додав стрічку, яка виводить статус код і отримав число 403, яке вказує на те, що сервер отримав запит але не хоче його авторизувати.

Для вирішення цієї проблеми я вирішив емулювати вхід з комп’ютера щоб сервер перестав забороняти авторизацію.



Рисунок 4 – Код, який емітує вхід з браузера

Розглянемо цей код.

headers = {...} – створює словник headers, який містить інформацію про HTTP-заголовки. У цьому випадку, вказано User-Agent, що імітує веб-браузер Google Chrome.

response = requests.get(url, headers=headers) – викликає функцію get з бібліотеки requests, яка виконує HTTP GET-запит за вказаним URL-адресою. Заголовки передаються для імітації запиту від браузера.

return response.text – повертає текстовий вміст відповіді. response.text містить HTML-код сторінки або вміст, який можна подальше обробити.

Отже, ця функція використовує бібліотеку requests для виконання HTTP GET-запиту за вказаним URL-адресою, використовуючи певні HTTP-заголовки для імітації веб-браузера, і повертає текстовий вміст відповіді, який містить HTML-код сторінки.

Таким чином я вирішив проблему з авторизацією та міг приступити до наступної частини виконання практичного завдання.

Наступною задачею було отримання даних з сайтів магазині. Для цього я використовував бібліотеку BeautifulSoup, яка дозволяє парсити HTML та XML файли.

Після вивчення бібліотеки та розробки алгоритму я створив наступну функцію, яку видно на рисунку 5.

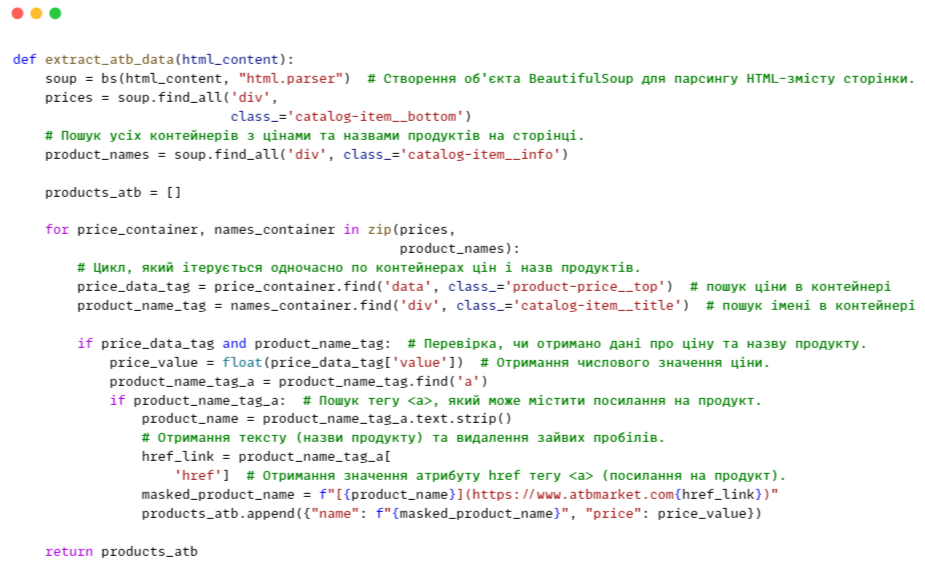


Рисунок 5 – Код, який знаходить потрібну інформацію на сайті АТБ

Цей код призначений для обробки HTML-коду сторінки з веб-сайту ATB Market та видобуття інформації про продукти, такі як ціни та назви.

soup = bs(html\_content, "html.parser"): Створюється об'єкт BeautifulSoup, який вкористовується для парсингу HTML-вмісту сторінки.

prices = soup.find\_all('div', class\_='catalog-item\_\_bottom'): Знаходяться всі контейнери із цінами на продукти, використовуючи метод find\_all з бібліотеки BeautifulSoup. Знаходяться всі елементи <div> з класом catalog-item\_\_bottom.

product\_names = soup.find\_all('div', class\_='catalog-item\_\_info'): Знаходяться всі контейнери із назвами продуктів. Аналогічно, використовуючи метод find\_all з бібліотеки BeautifulSoup, знаходяться всі елементи <div> з класом catalog-item\_\_info.

Створюється порожній список products\_atb, який буде містити інформацію про продукти.

for price\_container, names\_container in zip(prices, product\_names):: Використовується цикл for, який ітерується одночасно по контейнерах із цінами і назвами продуктів.

price\_data\_tag=price\_container.find('data',class\_='product-price\_\_top'): Знаходиться тег <data> в контейнері із ціною.

product\_name\_tag=names\_container.find('div', class\_='catalog-item\_\_title'): Знаходиться тег <div> в контейнері із назвою продукту.

if price\_data\_tag and product\_name\_tag:: Перевіряється, чи вдалося знайти дані про ціну та назву продукту.

price\_value = float(price\_data\_tag['value']): Отримується числове значення ціни з атрибуту 'value' тегу <data>. Ціна перетворюється в число з плаваючою комою.

product\_name\_tag\_a = product\_name\_tag.find('a'): Знаходиться тег <a> в контейнері із назвою продукту, який може містити посилання на продукт.

if product\_name\_tag\_a:: Перевіряється, чи знайдено тег <a>.

product\_name = product\_name\_tag\_a.text.strip(): Отримується текст (назва продукту) та видаляються зайві пробіли.

href\_link = product\_name\_tag\_a['href']: Отримується значення атрибуту 'href' тегу <a>, що містить посилання на продукт.

masked\_product\_name=f"[{product\_name}](https://www.atbmarket.com{href\_link})": Створюється відформатована назва продукту з посиланням, що приховане за допомогою Markdown-формату.

products\_atb.append({"name":f"{masked\_product\_name}","price":price\_value}): Додається словник з інформацією про продукт (назва та ціна) до списку products\_atb.

return products\_atb: Повертається список із словниками, що містять інформацію про продукти (назва та ціна).

Цей алгоритм не є універсальним, адже структура HTML сторінки в кожному магазині відрізняється і для кожного треба писати свій код і адаптувати його так, щоб ми отримували ті ж самі дані. Але загальна структура алгоритму схожа. Таким чином я навчився збирати дані з сайту магазину АТБ та отримувати їх у вигляді повідомлення.

Після того, як я налагодив доступ до сайту та отримання інформації з HTML сторінки я прийнявся за те, щоб цю інформацію зручно вивести користувачеві у вигляді повідомлення. Я вирішив виводити назву продукту, посилання, яке буде замасковане під назву продукту та ціну на цей самий продукт. Таким чином я створив функцію send\_results, яку можна побачити на рисунку 6.

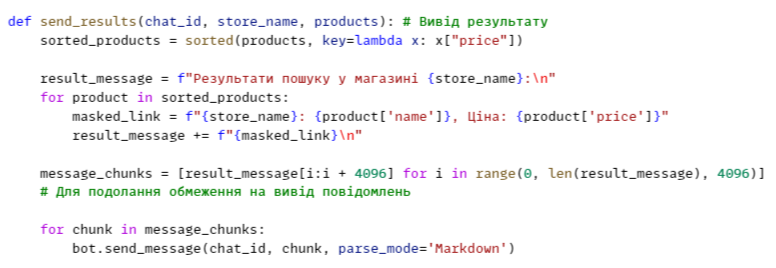


Рисунок 6 – Код який виводить дані отримані з магазинів

Ця функція send\_results призначена для відправки результатів пошуку користувачеві у вигляді повідомлення.

sorted\_products = sorted(products, key=lambda x: x["price"]): Сортує список продуктів за ціною у зростаючому порядку. Використовується функція sorted, при цьому ключ для сортування задається за допомогою лямбда-функції.

result\_message = f"Результати пошуку у магазині {store\_name}:\n": Створюється початковий рядок повідомлення, який містить ім'я магазину.

for product in sorted\_products:: Запускається цикл для ітерації по відсортованим продуктам.

masked\_link = f"{store\_name}: {product['name']}, Ціна: {product['price']}": Створюється рядок, який містить інформацію про продукт, його назву та ціну.

result\_message += f"{masked\_link}\n": Додається інформація про продукт до загального рядка повідомлення.

message\_chunks = [result\_message[i:i + 4096] for i in range(0, len(result\_message), 4096)]: Для подолання обмеження на вивід довгих повідомлень Telegram, рядок повідомлення розбивається на частини розміром не більше 4096 символів (максимальний розмір повідомлення в Telegram).

for chunk in message\_chunks:: Запускається цикл для ітерації по частинах повідомлення.

bot.send\_message(chat\_id, chunk, parse\_mode='Markdown'): Кожна частина повідомлення надсилається користувачеві за допомогою бота. При цьому використовується форматування Markdown для відображення посилань у тексті.

Після цього я почав тестування коду. Результатом виконання моєї частини стала можливість користувача ввести запит, вибрати магазин, в якому він хоче знайти товар та вивести результат.

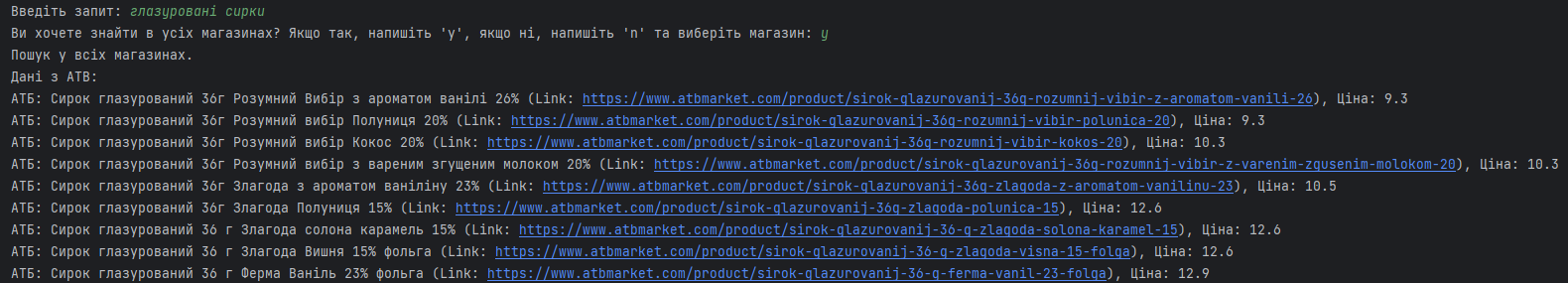


Рисунок 7 – Результат виконання програми

Далі потрібно було інтегрувати цей код в телеграм бота і провести тестування. Займався інтеграцією коду та налаштуванням боту Артем Тараненко і в результаті у нас вийшов чат-бот, який отримає запит, вводить його на обраний користувачем сайт та виводить назву продукту з замаскованим під назву посиланням та ціною цього продукту.

Для більш коректної роботи з ботом ми придумали правила, які полегшують пошук тих чи інших товарів, а також не забули про поради, які потрібно дати в разі відсутності результату виконання програми.

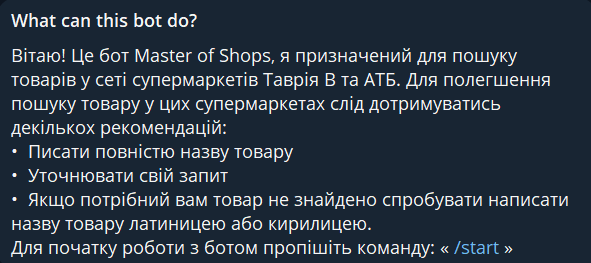


Рисунок – Правила та рекомендації для коректної роботи боту

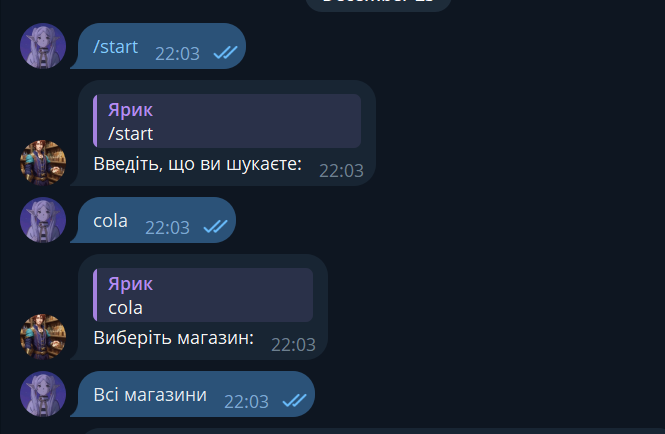


Рисунок 8 – Приклад роботи з вводом запиту в чат-боті

На рисунку 7 можна побачити сценарій роботи з чат ботом, який ми передбачували і реалізували. А на рисунку 8 можна побачити результат цього запиту.

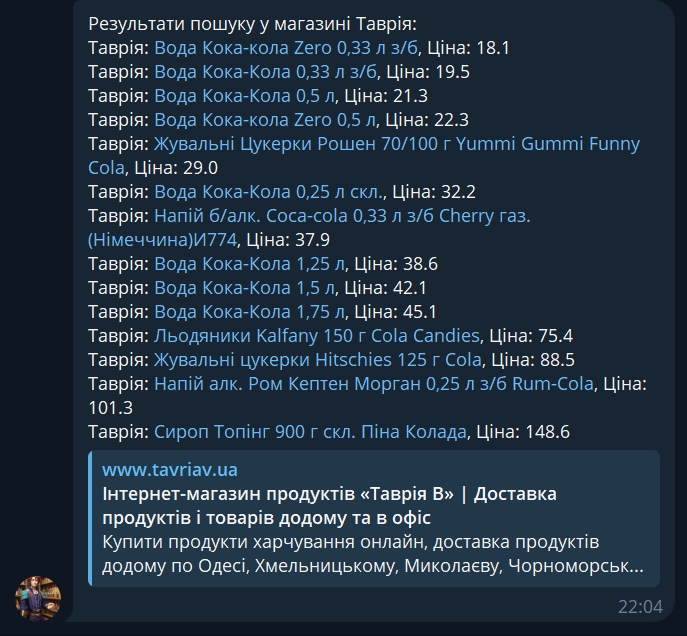
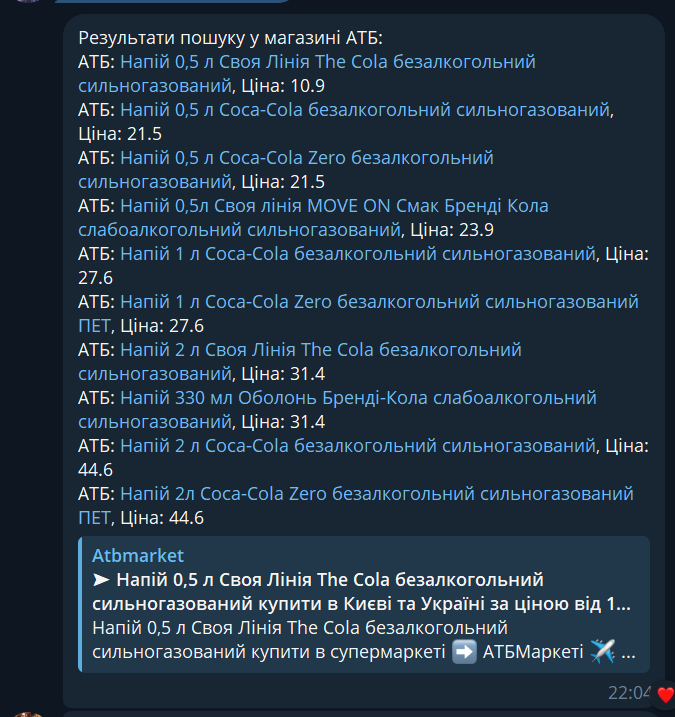


Рисунок 8 – Результат роботи програми

Таким чином ми реалізували телеграм бота, який реалізовує пошук того чи іншого продовольчого товару в магазинах АТБ та Таврія, сортує товари за ціною та виводить товар користувачеві для того, щоб той обрав, де йому вигідніше купити.

# 

# **ВИСНОВОК**

Під час проходження практики на базі освітнього фонду “Keepsolid” я здобув нові теоретичні, та підкріпив старі знання на практиці. Я ознайомився з принципами та методами праці з обробки інформації. Практика в освітньому фонді “KeepSolid” надала змогу розробити свій програмний продукт та працювати на кожному етапі розробки.

Під час проходження практики мені довелося зустрітися з усіма процесами розробки телеграм боту. Від розробки самої ідеї та функціоналу, до налаштування оточення та написання коду. Також навчився тестувати та виправляти помилки, які з’являлися під час тестування. Також я закріпив теоретичні знання, отримані в процесі навчання у Міжнародному гуманітарному університеті. Тобто, виконав усе, з чим зустрічаються програмісти у своїй роботі.

# **ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ**

1. Інформація з сайту URL: https://www.keepsolid.com/.
2. Інформація з сайту освітнього фонду KeepSolid URL: <https://keepsolideducationfund.org/>.
3. Інформація з сайту URL: <https://youcontrol.com.ua/ru/catalog/company_details/37761009/>.
4. Інформація з сайту Марченко Маркетинг URL :https://marchenko.marketing/scho-take-chat-bot-ta-yak-stvoriti-bota-v-telegram/
5. Інформація з сайту SendPuls про створення телеграм бота URL: [https://sendpulse.ua/knowledge-base/chatbot/telegram/create-telegram-chatbot#](https://sendpulse.ua/knowledge-base/chatbot/telegram/create-telegram-chatbot)
6. Інформація з сайту PyPi документація до pyTelegramBotAPI https://pypi.org/project/pyTelegramBotAPI/

# 

# **ДОДАТОК А – Лістинг файлу bot.py**

def extract\_atb\_data(html\_content):  
 soup = bs(html\_content, "html.parser") # Створення об'єкта BeautifulSoup для парсингу HTML-змісту сторінки.  
 prices = soup.find\_all('div',  
 class\_='catalog-item\_\_bottom')  
 # Пошук усіх контейнерів з цінами та назвами продуктів на сторінці.  
 product\_names = soup.find\_all('div', class\_='catalog-item\_\_info')  
  
 products\_atb = []  
  
 for price\_container, names\_container in zip(prices,  
 product\_names):  
 # Цикл, який ітерується одночасно по контейнерах цін і назв продуктів.  
 price\_data\_tag = price\_container.find('data', class\_='product-price\_\_top') # пошук ціни в контейнері  
 product\_name\_tag = names\_container.find('div', class\_='catalog-item\_\_title') # пошук імені в контейнері  
  
 if price\_data\_tag and product\_name\_tag: # Перевірка, чи отримано дані про ціну та назву продукту.  
 price\_value = float(price\_data\_tag['value']) # Отримання числового значення ціни.  
 product\_name\_tag\_a = product\_name\_tag.find('a')  
 if product\_name\_tag\_a: # Пошук тегу <a>, який може містити посилання на продукт.  
 product\_name = product\_name\_tag\_a.text.strip()  
 # Отримання тексту (назви продукту) та видалення зайвих пробілів.  
 href\_link = product\_name\_tag\_a[  
 'href'] # Отримання значення атрибуту href тегу <a> (посилання на продукт).  
 masked\_product\_name = f"[{product\_name}](https://www.atbmarket.com{href\_link})"  
 products\_atb.append({"name": f"{masked\_product\_name}", "price": price\_value})  
  
 return products\_atb  
  
  
def extract\_tavria\_data(html\_content):  
 soup = bs(html\_content, "html.parser")  
 products = soup.find\_all('div', class\_='products\_\_item')  
  
 products\_tavria = []  
  
 for product in products:  
 price\_data\_tag = product.find('p', class\_='product\_\_price') # пошук ціни в контейнері  
 product\_name\_tag = product.find('p', class\_='product\_\_title') # пошук імені в контейнері  
  
 if price\_data\_tag and product\_name\_tag:  
 price\_value\_match = re.search(r'(\d+\.\d+)', price\_data\_tag.text)  
 price\_value = float(price\_value\_match.group()) if price\_value\_match else float('inf')  
  
 product\_name\_tag\_a = product\_name\_tag.find('a')  
 if product\_name\_tag\_a:  
 product\_name = product\_name\_tag\_a.text.strip()  
 href\_link = product\_name\_tag\_a['href']  
 masked\_product\_name = f"[{product\_name}](https://www.tavriav.ua{href\_link})"  
 products\_tavria.append({"name": f"{masked\_product\_name}", "price": price\_value})  
  
 return products\_tavria

def search\_in\_atb(chat\_id):  
 zapit = user\_inputs[chat\_id]['query']  
 url\_atb = f"https://www.atbmarket.com/sch?page=1&lang=uk&location=1154&query={zapit}"  
 html\_content\_atb = get\_html\_content(url\_atb)  
 products\_atb = extract\_atb\_data(html\_content\_atb)  
 send\_results(chat\_id, "АТБ", products\_atb)  
  
  
def search\_in\_tavria(chat\_id):  
 zapit = user\_inputs[chat\_id]['query']  
 url\_tavria = f"https://www.tavriav.ua/catalog/search/?query={zapit}"  
 html\_content\_tavria = get\_html\_content(url\_tavria)  
 products\_tavria = extract\_tavria\_data(html\_content\_tavria)  
 send\_results(chat\_id, "Таврія", products\_tavria)

def send\_results(chat\_id, store\_name, products): # Вивід результату  
 sorted\_products = sorted(products, key=lambda x: x["price"])  
  
 result\_message = f"Результати пошуку у магазині {store\_name}:\n"  
 for product in sorted\_products:  
 masked\_link = f"{store\_name}: {product['name']}, Ціна: {product['price']}"  
 result\_message += f"{masked\_link}\n"  
  
 message\_chunks = [result\_message[i:i + 4096] for i in range(0, len(result\_message), 4096)]  
 # Для подолання обмеження на вивід повідомлень  
  
 for chunk in message\_chunks:  
 bot.send\_message(chat\_id, chunk, parse\_mode='Markdown')