**ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ**  
**«ОПТИКО-МЕХАНІЧНИЙ ФАХОВИЙ КОЛЕДЖ КИЇВСЬКОГО**  
**НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ІМЕНІ ТАРАСА ШЕВЧЕНКА»**  
Циклова комісія програмування та інформаційних технологій

**ЗВІТ**

**З НАВЧАЛЬНОЇ ПРАКТИКИ**

спеціальність *121 Інженерія програмного забезпечення*  
освітньо-кваліфікаційний рівень *«Фаховий молодший бакалавр»*

|  |  |
| --- | --- |
| ОЦІНКА  (оцінка цифрою та прописом)  Керівник практики від коледжу:  (підпис)  « \_» 20 року | Виконав:  студент 3-го курсу групи ІПЗ-32  *Ігонов Артем Олександрович*  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис виконавця)  « \_» 20 року |

КИЇВ - 2025

ЗАТВЕРДЖУЮ

Директор ВСП «ОМФК КНУ

ім. Тараса Шевченка»

\_\_\_\_\_\_\_\_\_\_\_Борис ГАПРИНДАШВІЛІ

«\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025р

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

на період *Навчальної* практики

(навчальної, технологічної, виробничої)

Ігонов Артем Олександрович

(П І.Б студента )

Спеціальність *121 Інженерія програмного забезпечення* Група *ІПЗ-32*

|  |  |  |
| --- | --- | --- |
|  | **Питання для вивчення** | **Відмітки про виконання** |
| **1** | Вступне заняття. Вивчення питань охорони праці. Ознайомлення з електронною бібліотекою коледжу. Вивчення правил оформлення звітної документації. |  |
| **2** | Знайомство з мовою програмування Python. Історія мови, особливості мови, сфери застосування мови |  |
| 3 | Основи мови Python. Синтаксис мови, середовище виконання, запуск програм |  |
| 4 | Змінні та літерали, типи даних. Вбудовані типи даних, ініціалізація змінних, змінювані і незмінювані типи |
| 5 | Програмування лінійних та розгалужених обчислювальних процесів |  |
| 6 | Програмування циклічних обчислювальних процесів |  |
| 7 | Типи даних визначені користувачем, магічні методи, виключення (Exceptions) та їх обробка |  |
| 8 | Вбудовані структури даних, складні типи |  |
| 9 | Функції |  |
| 10 | Потокове введення, виведення та обробка даних |  |
| 11 | Побудова проекту, модулі, пакети |  |
| 12 | Робота з файловою системою, оператор with, контекстний менеджер |  |
| 13 | Робота з протоколом HTTP, бібліотека requests |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Голова циклової комісії |  |  |  |
|  | (підпис) | (ініціали, прізвище) | (дата) |
| Керівник практики від коледжу |  |  |  |
|  | (підпис) | (ініціали, прізвище) | (дата) |

**Зміст**

[ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 4](#_Toc565466204)

[Вступ 5](#_Toc1137144064)

[1 Теоретичні відомості 7](#_Toc928310420)

[1.1 Вступне заняття 7](#_Toc1749277208)

[1.2 Знайомство з мовою програмування Python. Історія мови, особливості мови, сфери застосування мови 7](#_Toc1313556716)

[Історія мови Python 7](#_Toc1402805331)

[Особливості мови Python 8](#_Toc1124402173)

[Сфери застосування Python 8](#_Toc141203036)

[1.3 Основи мови Python. Синтаксис мови, середовище виконання, запуск програм 9](#_Toc975059889)

[Синтаксис мови Python 9](#_Toc1551059058)

[Середовище виконання Python 9](#_Toc1045124425)

[Запуск програм на Python 9](#_Toc886069885)

[1.4 Змінні та літерали, типи даних. Вбудовані типи даних, ініціалізація змінних, змінювані і незмінювані типи 10](#_Toc457620007)

[Змінні та літерали 10](#_Toc1001588155)

[Типи даних 10](#_Toc327235305)

[Ініціалізація змінних 10](#_Toc306893266)

[Змінювані та незмінювані типи 11](#_Toc2083310759)

[2 Програмування лінійних та розгалужених обчислювальних процесів 11](#_Toc1918252953)

[2.1 Програмування циклічних обчислювальних процесів 12](#_Toc151871907)

[2.2 Типи даних визначені користувачем, магічні методи, виключення (Exceptions) та їх обробка 12](#_Toc1332761692)

[Типи даних, визначені користувачем 12](#_Toc60247023)

[2.3 Вбудовані структури даних, складні типи 12](#_Toc141563928)

[Списки (Lists) 13](#_Toc327833781)

[Словники (Dictionaries) 13](#_Toc1190511440)

[Множини (Sets) 13](#_Toc1319207991)

[Складні типи даних 13](#_Toc491495181)

[3 Функції 14](#_Toc952659577)

[3.1 Потокове введення, виведення та обробка даних 14](#_Toc565199148)

[Потокове введення 14](#_Toc436532189)

[Потокове виведення 14](#_Toc1342440961)

[Обробка даних 14](#_Toc1003759454)

[Робота з файлами CSV 14](#_Toc1380895627)

[3.2 Побудова проекту, модулі, пакети 14](#_Toc1104850220)

[Модулі 14](#_Toc2043141505)

[Пакети 14](#_Toc622146731)

[Структура проекту 15](#_Toc1399309708)

[Віртуальне середовище 15](#_Toc945261717)

[Файл requirements.txt 15](#_Toc753743243)

[4 Робота з файловою системою, оператор with, контекстний менеджер 15](#_Toc669335905)

[4.1 Робота з протоколом HTTP, бібліотека requests 15](#_Toc1383604342)

[Основи HTTP 15](#_Toc1473466461)

[Бібліотека requests 15](#_Toc1749593572)

[5 Виконання завдань 16](#_Toc1141070261)

[5.1 Завдання 1 16](#_Toc1730155898)

[5.2 Завдання 2 17](#_Toc952682175)

[5.3 Завдання 3 18](#_Toc1979880078)

[6 Висновки 20](#_Toc937489042)

# **ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ**

* **API** (Application Programming Interface) - інтерфейс програмування додатків

# **Вступ**

Метою практики є поглиблення знань та набуття практичних навичок з використанням мови Python. Під час практики ми ознайомлювались з основами мови програмування: типи даних, функції, взаємодія з API та інше. Також виконували корисні для початкових інженерів програмного забезпечення практичні заняття.

У звіті описано основні етапи проходження практики, виконані завдання, використані технології та інструменти, а також отримані результати.

# **Теоретичні відомості**

## **1.1 Вступне заняття**

Метою вступного заняття було ознайомлення з програмою та основними напрямками курсу. Також розглянули основні питання охорони праці та організація практики, наприклад як саме проходитимуть заняття та терміни виконання робіт, дії під час повітряної тривоги та інше. Також нас ознайомили з електронною бібліотекою коледжу та правилами користування. Окрім цього нас ознайомили з оформленням звітної документації, що важливо для успішної здачі практики.

## **1.2 Знайомство з мовою програмування Python. Історія мови, особливості мови, сфери застосування мови**

### **Історія мови Python**

Python – інтепретована об’єктивно-орієнтована мова програмування високого рівня із суворою динамічною типізацією. Мову Python було розроблено в 1990 році Гвідо ван Россумом. Назву було запозичено у британського шоу «Монті Пайтон».

Перша версія Python була опублікована у 1991 році. З того часу Python постійно розвивався та вдосконалювався. У 2000 році вийшла версія Python 2.0, а у 2008 році - Python 3.0. Python 3.0 був значним кроком вперед, оскільки він містив багато змін, які покращили мову та її можливості.

### **Особливості мови Python**

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об’єктивно-орієнтованого програмування. Також є інші переваги які роблять його привабливим для розробників.

Наприклад:

* Чистий синтаксис.
* Переносніть програм.
* Великий вибір бібліотек.
* Можливість використання в діалоговому вікні.
* Відкритий код для можливості редагувати його іншим користувачем.
* Велика спільнота.

### **Сфери застосування Python**

Python використовується в різних сферах, зокрема:

* **Веб-розробка:** Python є популярним вибором для веб-розробки завдяки таким фреймворкам, як Django та Flask.
* **Аналіз даних та машинне навчання:** Python має потужні бібліотеки для аналізу даних та машинного навчання, такі як NumPy, Pandas та Scikit-learn.
* **Автоматизація:** Python використовується для автоматизації різних завдань, таких як обробка файлів, веб-скрепінг та тестування програмного забезпечення.
* **Розробка ігор:** Python може використовуватися для розробки ігор за допомогою бібліотеки Pygame.
* **Освіта:** Python є популярною мовою для навчання програмуванню завдяки своїй простоті та зрозумілості.

## **1.3 Основи мови Python. Синтаксис мови, середовище виконання, запуск програм**

### **Синтаксис мови Python**

Синтаксис Python відрізняється своєю лаконічністю та інтуїтивністю від основних мов програмування. Ось деякі ключові особливості:

* **Відсутність крапок з комою:** Python не використовує крапки з комою для позначення кінця оператора. Замість цього використовується перехід на новий рядок що доволі зручно, але може заплутати нових користувачів.
* **Відступи:** Python використовує відступи для позначення блоків коду. Це робить код більш читабельним та структурованим.
* **Динамічна типізація:** Python автоматично визначає тип змінної під час виконання програми.
* **Великий вибір вбудованих функцій та типів даних:** Python має велику кількість вбудованих функцій та типів даних, що полегшує розробку програм, також для зручності можна використовувати бібліотеки за допомогою яких виконання задач стане ще простіше.

### **Середовище виконання Python**

Для виконання програм на Python необхідно встановити інтерпретатор Python. Існує кілька способів встановлення Python:

* **Завантаження з офіційного сайту:** Ви можете завантажити останню версію Python з офіційного сайту python.org.
* **Використання дистрибутивів:** Існують різні дистрибутиви Python, такі як Anaconda та ActivePython, які включають в себе інтерпретатор Python та інші корисні інструменти.

Після встановлення Python ви можете використовувати різні середовища розробки для написання та виконання програм:

* **IDLE:** IDLE - це інтегроване середовище розробки, яке постачається разом з Python.
* **Visual Studio Code:** Visual Studio Code - це популярний редактор коду з підтримкою Python.
* **PyCharm:** PyCharm - це професійне середовище розробки для Python.

### **Запуск програм на Python**

Існує кілька способів запуску програм на Python:

* **Запуск з командного рядка:** Ви можете запустити програму Python з командного рядка, вказавши шлях до файлу з розширенням .py.
* **Запуск з інтегрованого середовища розробки:** Ви можете запустити програму Python безпосередньо з інтегрованого середовища розробки, натиснувши кнопку запуску.

## **1.4 Змінні та літерали, типи даних. Вбудовані типи даних, ініціалізація змінних, змінювані і незмінювані типи**

У Python змінні використовуються для зберігання даних які згодом можна змінювати. Літерали - це безпосередньо задані значення, наприклад, числа, рядки або логічні значення. Кожна змінна має тип даних, який визначає, які значення вона може зберігати. Python має кілька вбудованих типів даних, такі як.

### **Змінні та літерали**

Змінні в Python можуть зберігати значення різних типів. Ім'я змінної може складатися з літер, цифр та символу підкреслення, але воно повинно починатися з літери або символу підкреслення. Python є динамічно типізованою мовою, тому тип змінної визначається під час присвоєння їй значення.

### **Типи даних**

Python має кілька вбудованих типів даних, які використовуються для представлення різних видів інформації.

**Основні типи даних:**

* **int:** цілі числа (наприклад, 10, -5, 0)
* **float:** числа з плаваючою крапкою (наприклад, 3.14, -2.5, 0.0)
* **str:** рядки (наприклад, "Hello", "Python")
* **bool:** логічні значення (True або False)
* **list:** списки (впорядковані колекції елементів)
* **tuple:** кортежі (незмінювані списки)
* **dict:** словники (асоціативні масиви)
* **set:** множини (невпорядковані колекції унікальних елементів)

### **Ініціалізація змінних**

Ініціалізація змінної - це присвоєння їй першого значення. У Python змінні ініціалізуються за допомогою оператора присвоєння =.

### **Змінювані та незмінювані типи**

У Python типи даних поділяються на змінювані (mutable) та незмінювані (immutable).

* **Незмінювані типи:** значення незмінюваних типів не можуть бути змінені після створення. До них відносяться int, float, str, bool та tuple.
* **Змінювані типи:** значення змінюваних типів можуть бути змінені після створення. До них відносяться list, dict та set.

# **2 Програмування лінійних та розгалужених обчислювальних процесів**

## **2.1 Програмування циклічних обчислювальних процесів**

Цикли є однією з фундаментальних концепцій програмування, що дозволяє виконувати блок коду кілька разів. Python надає два основних типи циклів: for та while.

## **2.2 Типи даних визначені користувачем, магічні методи, виключення (Exceptions) та їх обробка**

### **Користувацькі типи даних**

У Python, крім стандартних типів даних, таких як цілі числа (int), числа з плаваючою комою (float), рядки (str) тощо, програмісти можуть створювати власні типи даних за допомогою класів. Класи дозволяють об’єднувати дані та методи в єдиний об’єкт, що робить програмний код більш структурованим і зручним для використання.

### **Спеціальні методи**

Спеціальні методи — це методи з особливими іменами, які починаються і закінчуються двома підкресленнями (\_\_). Python викликає ці методи автоматично в певних ситуаціях, дозволяючи змінювати поведінку об’єктів.

Найбільш поширені спеціальні методи:

* \_\_init\_\_(self, ...) — конструктор класу, викликається під час створення об’єкта.
* \_\_str\_\_(self) — повертає текстове представлення об’єкта для користувача.
* \_\_repr\_\_(self) — повертає офіційне представлення об’єкта, корисне для розробників.
* \_\_len\_\_(self) — визначає, яку довжину має об’єкт.
* \_\_getitem\_\_(self, key) — дозволяє звертатися до елементів об’єкта за ключем, як у словниках чи списках.
* \_\_setitem\_\_(self, key, value) — дає змогу встановлювати значення елементів за ключем.

## **2.3 Вбудовані структури даних, складні типи**

**Python має кілька базових структур даних, які дозволяють зберігати та обробляти множину значень. Найпоширенішими з них є:**

* **Списки (Lists)** — впорядковані, змінювані колекції елементів, які можуть містити дані різних типів.
* **Словники (Dictionaries)** — колекції пар «ключ-значення», де ключі є унікальними й незмінюваними. Порядок елементів не гарантовано до версії Python 3.7 включно, але у версіях 3.7+ словники зберігають порядок додавання.
* **Множини (Sets)** — невпорядковані колекції унікальних елементів, що дозволяють ефективно виконувати операції над множинами, такі як об'єднання та перетин.

### **Складні типи даних**

Python надає можливість створювати власні складні типи даних за допомогою класів. Це дозволяє об’єднувати дані та методи для їхньої обробки в одному об’єкті, забезпечуючи гнучкість і повторне використання коду.

Такі користувацькі типи особливо корисні для створення спеціалізованих структур даних або програм з об'єктно-орієнтованим підходом.

# **3 Функції**

### **3.1 Потокове введення, виведення та обробка даних**

**Потокове введення**  
Для отримання даних від користувача або з файлу застосовуються функції input() та open().

* input(): зчитує текстовий рядок, введений користувачем у консолі.
* open(): відкриває файл для читання або запису.

**Потокове виведення**  
Для передачі даних на консоль або у файл використовуються функції print() та write().

* print(): виводить дані на екран.
* write(): записує інформацію у файл.

**Обробка даних**  
Python пропонує широкий спектр засобів для обробки даних:

* **Оператори**: виконують арифметичні, логічні, а також порівняльні операції.
* **Функції**: можуть бути як вбудованими, так і користувацькими.
* **Модулі**: це бібліотеки, що містять додаткові функції та інструменти.
* **Структури даних**: включають списки, кортежі, словники та множини.

**Робота з файлами CSV**  
CSV (Comma Separated Values) — популярний формат для збереження табличних даних. Для роботи з файлами цього формату в Python використовується вбудований модуль csv.

### **3.2 Побудова проекту, модулі, пакети**

**Модулі**  
Модуль — це окремий файл із розширенням .py, що містить функції, класи та інші об’єкти. Використання модулів дозволяє розділити код на логічні частини, які можна повторно використовувати в різних місцях проекту.

**Пакети**  
Пакет — це каталог, що містить файл \_\_init\_\_.py (може бути порожнім) і модулі або підпакети. Пакети дозволяють створювати ієрархічну структуру для організації коду.

**Структура проекту**  
Зазвичай Python-проекти організовуються таким чином:

* **Кореневий каталог**: містить файли конфігурації та документацію.
* **Каталог з вихідним кодом**: може називатися src або відповідно до назви проекту.
* **Каталог тестів**: зазвичай називається tests.
* **Файл setup.py**: використовується для збірки та встановлення проекту.

**Віртуальне середовище**  
Щоб ізолювати залежності одного проекту від іншого, рекомендується використовувати віртуальне середовище. Інструмент venv дозволяє легко створити таке середовище.

**Файл requirements.txt**  
Цей файл містить перелік усіх залежностей проекту. Його можна згенерувати командою pip freeze > requirements.txt.

### **4 Робота з файловою системою, оператор with, контекстний менеджер**

**Контекстний менеджер** спрощує роботу з ресурсами, такими як файли, автоматично закриваючи їх після завершення роботи. Оператор with допомагає безпечно відкривати файли, забезпечуючи їхнє коректне закриття, навіть якщо виникла помилка під час виконання.

### **4.1 Робота з протоколом HTTP, бібліотека requests**

**Основи HTTP**  
HTTP (Hypertext Transfer Protocol) — це протокол, який забезпечує обмін даними між клієнтом (наприклад, браузером) і сервером. Він працює на основі запитів і відповідей: клієнт надсилає запит, а сервер повертає відповідь. Запит може містити метод (наприклад, GET, POST, PUT, DELETE), URL-адресу та заголовки. Відповідь містить код стану (200 OK, 404 Not Found) і тіло.

**Бібліотека requests**  
Бібліотека requests спрощує роботу з HTTP-запитами в Python. Вона дозволяє легко надсилати запити різних типів і обробляти відповіді.

Основні можливості requests:

* Передача параметрів через URL: requests.get(url, params={"param1": "value1"})
* Додавання заголовків: requests.get(url, headers={"User-Agent": "My-User-Agent"})
* Надсилання файлів: requests.post(url, files={"file": open("my\_file.txt", "rb")})
* Робота з cookies: response.cookies
* Таймаути для запитів: requests.get(url, timeout=5)

# **5 Виконання завдань**

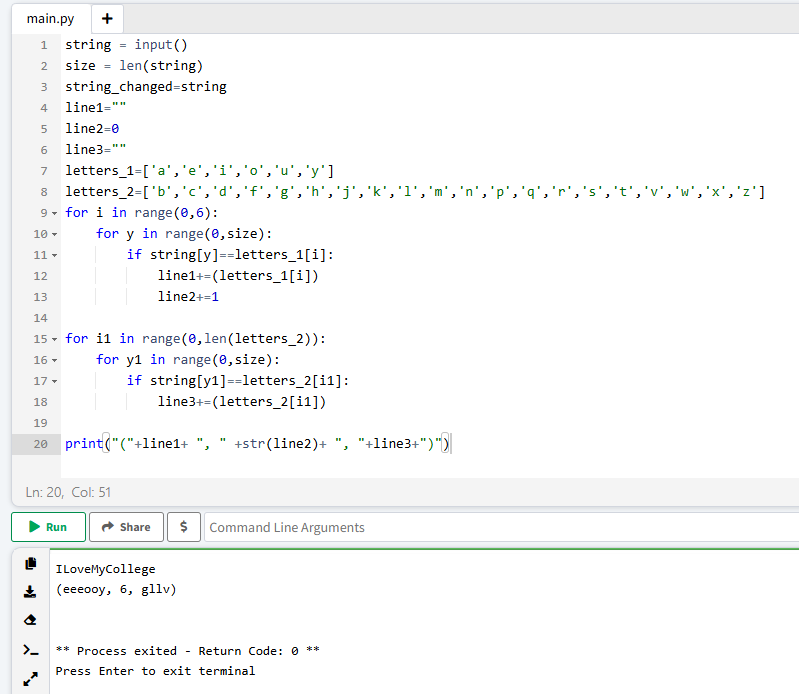
## **5.1 Завдання 1**

**Посилання на гіт хаб:**

https://github.com/tyomkich/-/blob/main/4\_Artem

**Пояснення:**

приймає на вхід строку і повертає кортеж з трьома елементами, в яких перший елемент має тип str і включає всі голосні літери з вхідної строки  (алфавітний порядок). Другий елемент - int  - кількість голосних літер, а останній - всі приголосні літери з вхідної строки (алфавітний порядок).



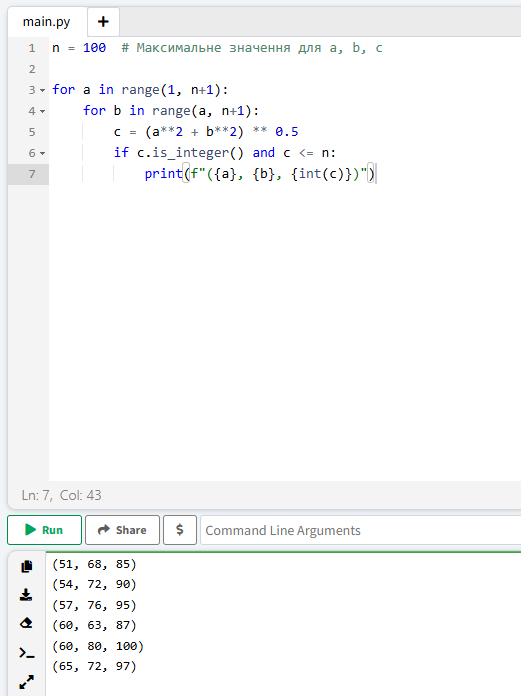
## **5.2 Завдання 2**

**Посилання на гіт хаб:**

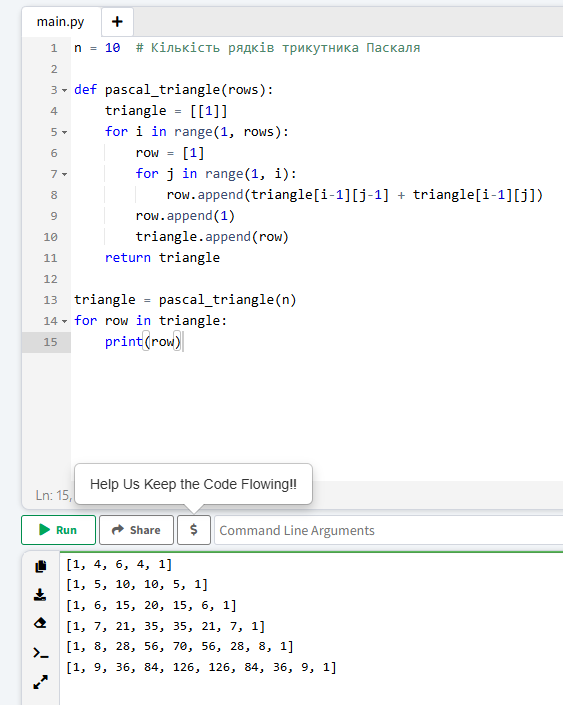
[https://github.com/tyomkich/-/blob/main/4\_Artem%20Task2](https://github.com/Rost589/Prc/blob/main/Практика/IPZ-32/13_Rostyslav/pr2.py)

**Пояснення:**

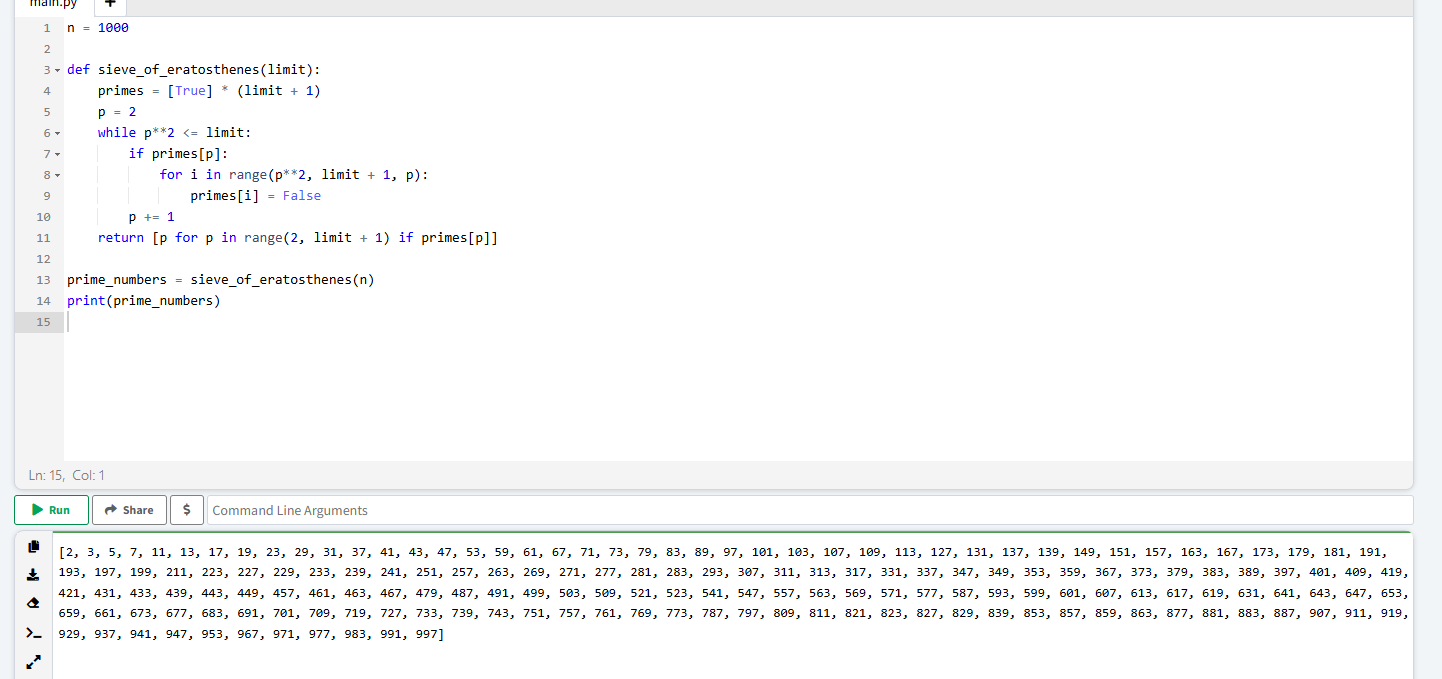
1)****Знаходження пифагорових трійок:**  
Код для цього завдання шукає всі трійки цілих чисел aaa, bbb, ccc, які не перевищують число nnn і задовольняють рівняння a2+b2=c2a^2 + b^2 = c^2a2+b2=c2. Це так звані пифагорові трійки.**



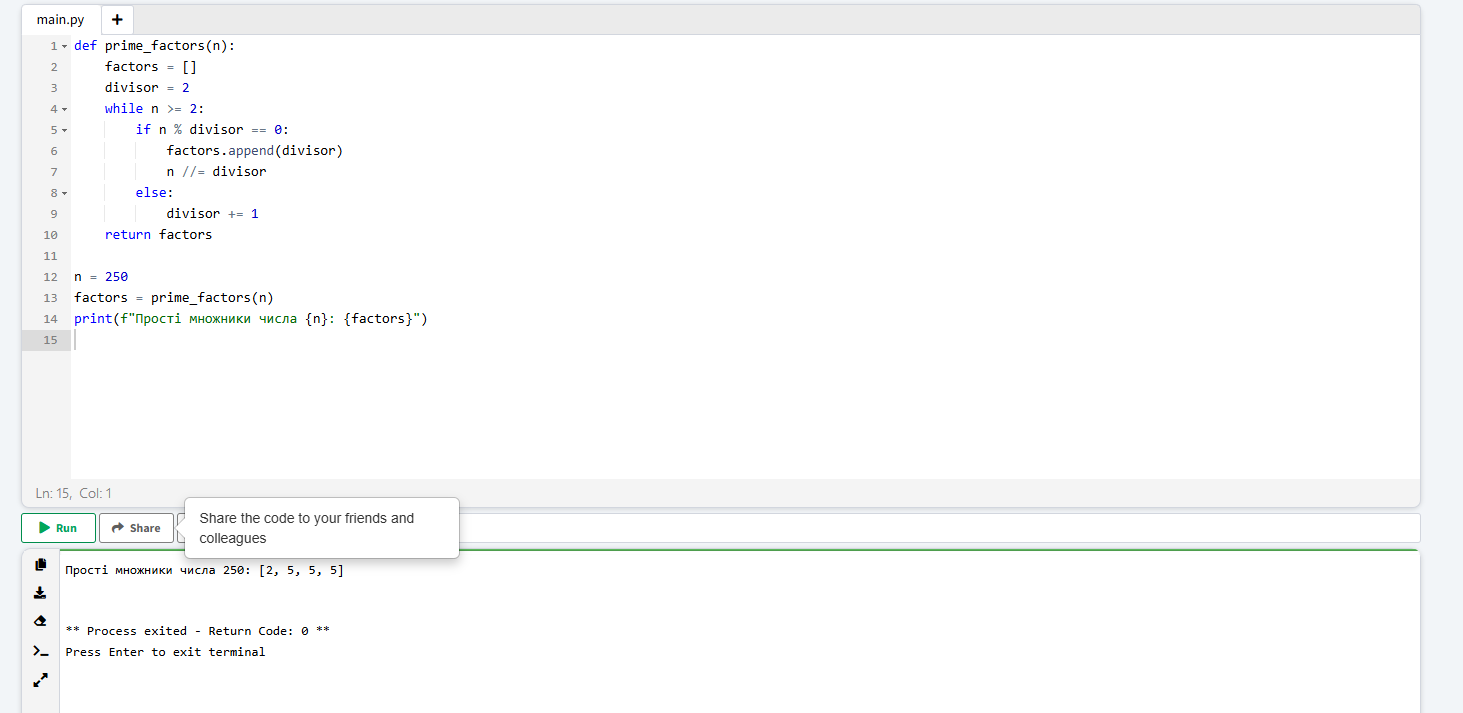
**2)Побудова трикутника Паскаля:  
Код генерує перші nnn рядків трикутника Паскаля. Кожен елемент у трикутнику — це сума двох елементів, розташованих безпосередньо над ним зліва і справа.**



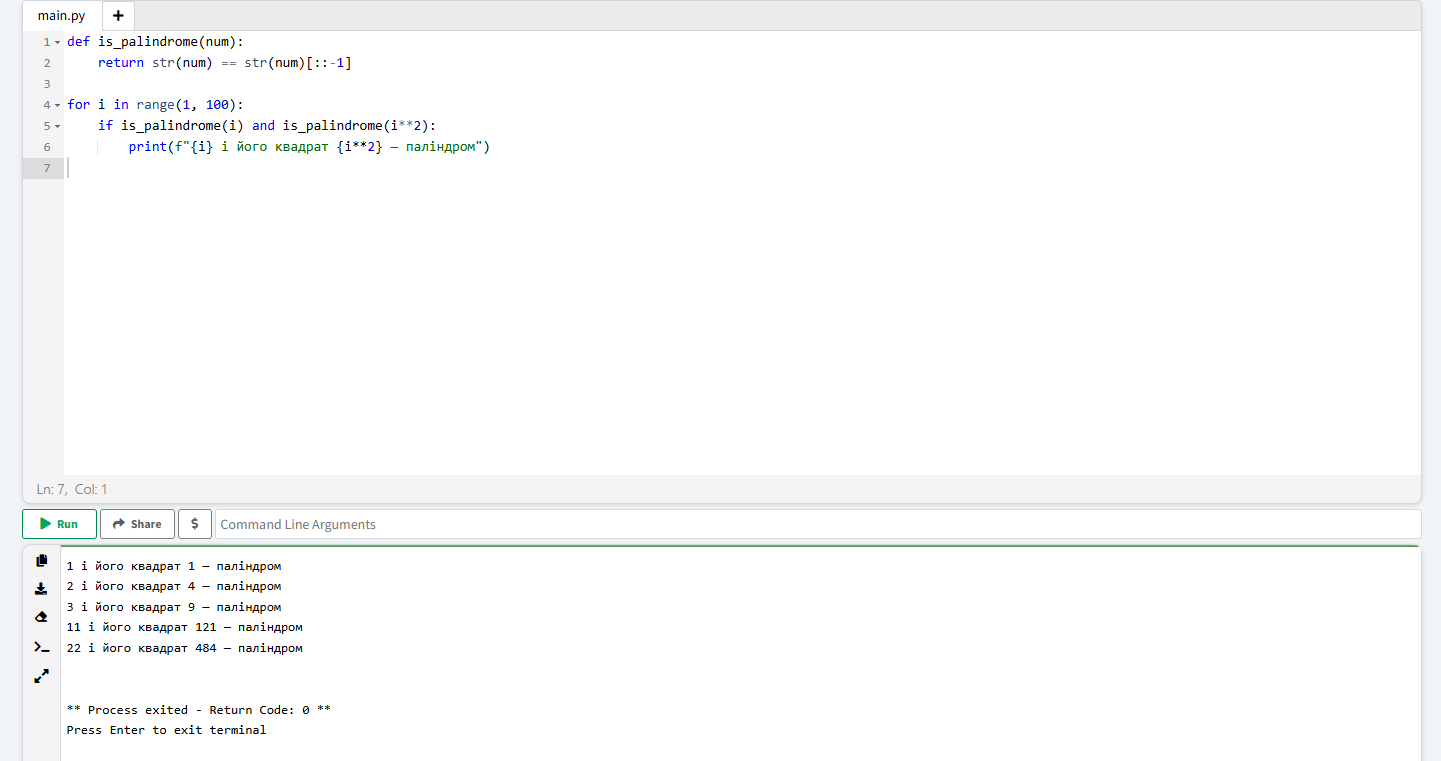
**3)Решето Ератосфена:  
Код реалізує алгоритм для пошуку всіх простих чисел від 2 до 1000. Він поступово викреслює всі складені числа, залишаючи лише прості**



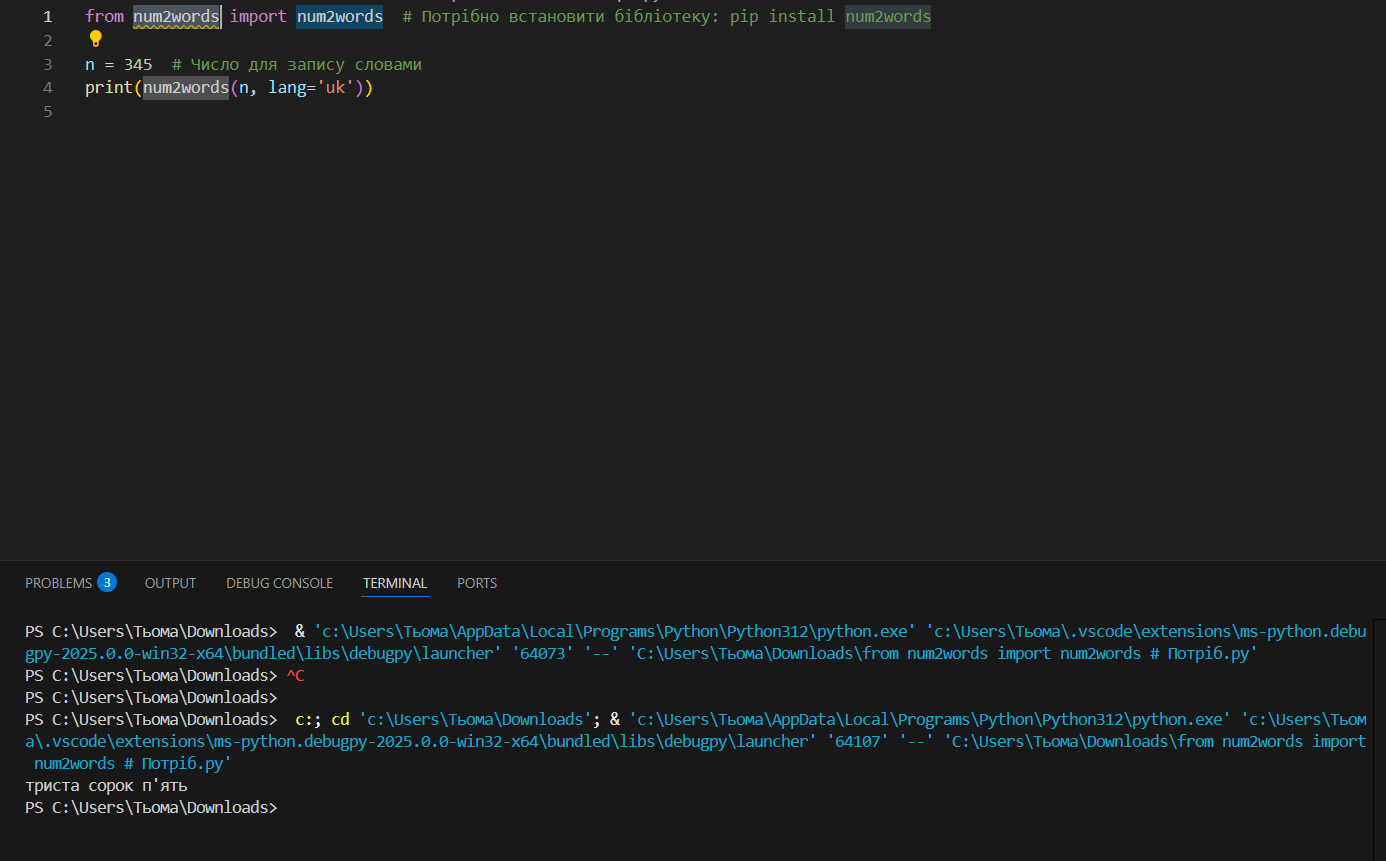
**4)Розкладання числа на прості множники:  
Код розкладає задане число на прості множники і повертає їх перелік, наприклад, для числа 250 результатом буде 2, 5, 5, 5.**



**5)Пошук паліндромів:  
Код знаходить усі числа-паліндроми в діапазоні від 100 до 999. Паліндром — це число, яке читається однаково зліва направо і справа наліво, наприклад, 484 або 121.**

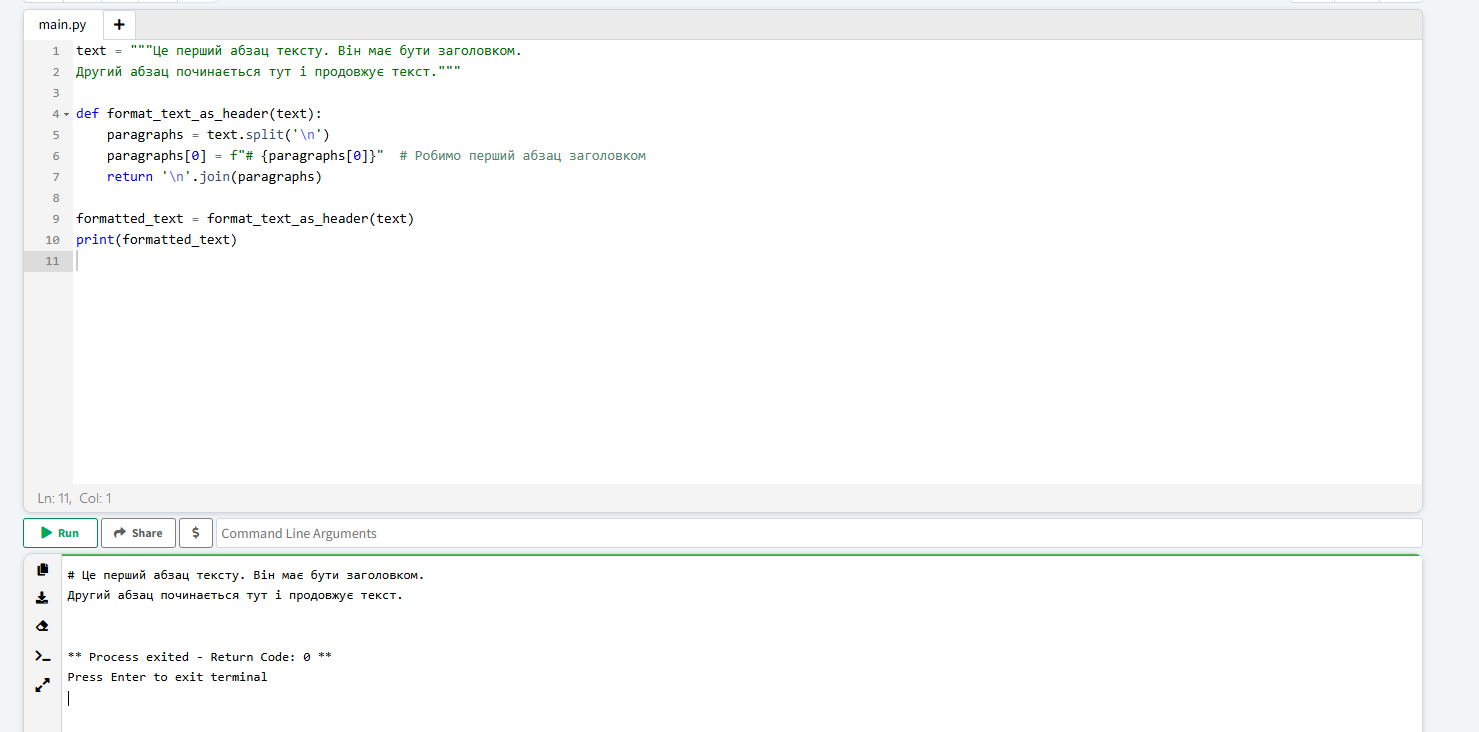


**6)Запис числа словами:  
Код приймає ціле число n≤1000n \leq 1000n≤1000 і виводить його словесне представлення, наприклад, «п'ятсот сорок три».**



**7)**Пошук пар простих чисел:**  
Код шукає всі пари простих чисел, різниця між якими дорівнює 2. Такі числа називаються «простими близнюками» (наприклад, 3 і 5 або 11 і 13).**

**8)**Форматування тексту:**  
Код приймає текст і натуральне число w>50w > 50w>50 і відформатовує його так, щоб кожен рядок мав ширину www символів, зберігаючи розриви слів і додаючи два символи нового рядка в кінці кожного рядка.**

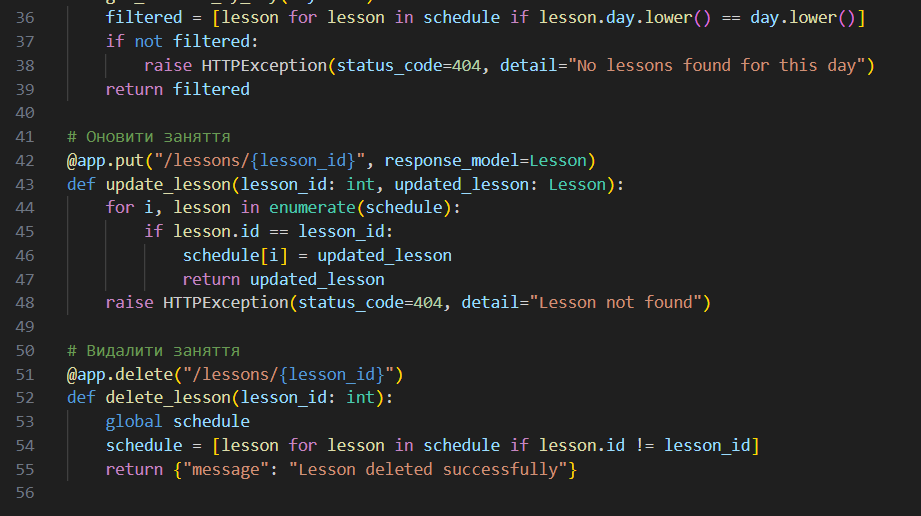
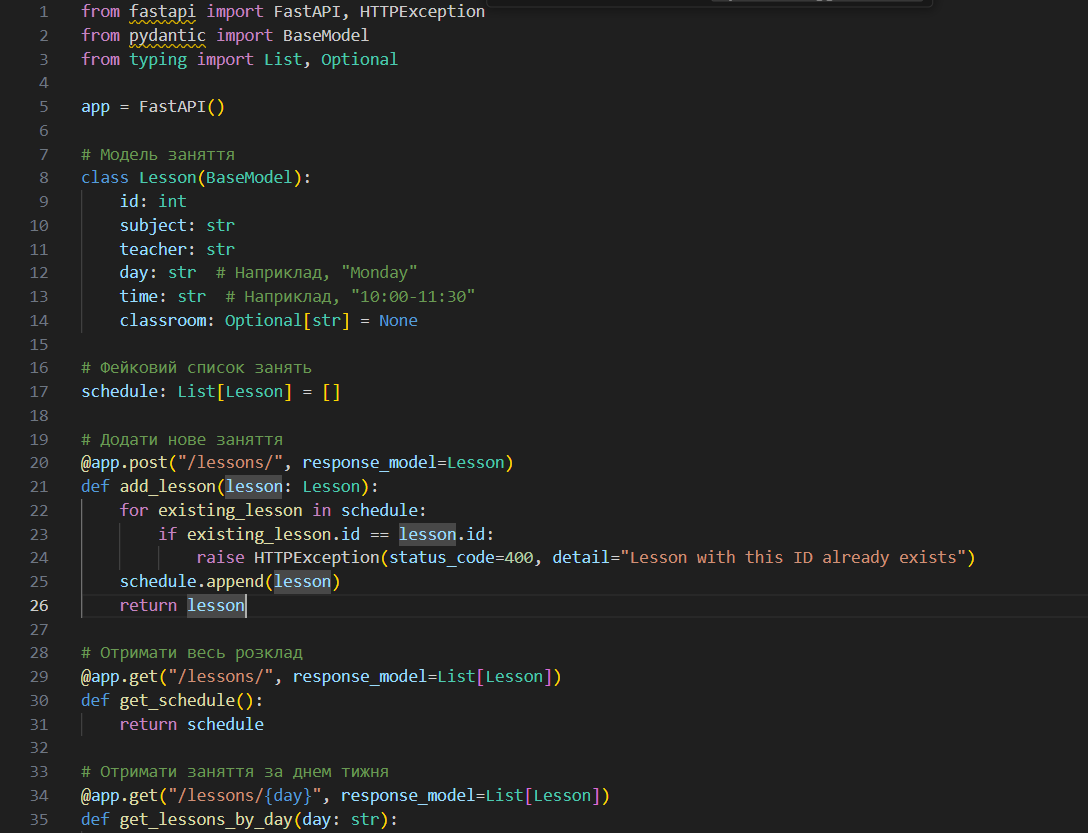


## **5.3**

## **Завдання 3**

**Посилання на гіт хаб:**

https://github.com/tyomkich/-/blob/main/task3/task3



**Інструкція та пояснення як він працює:**

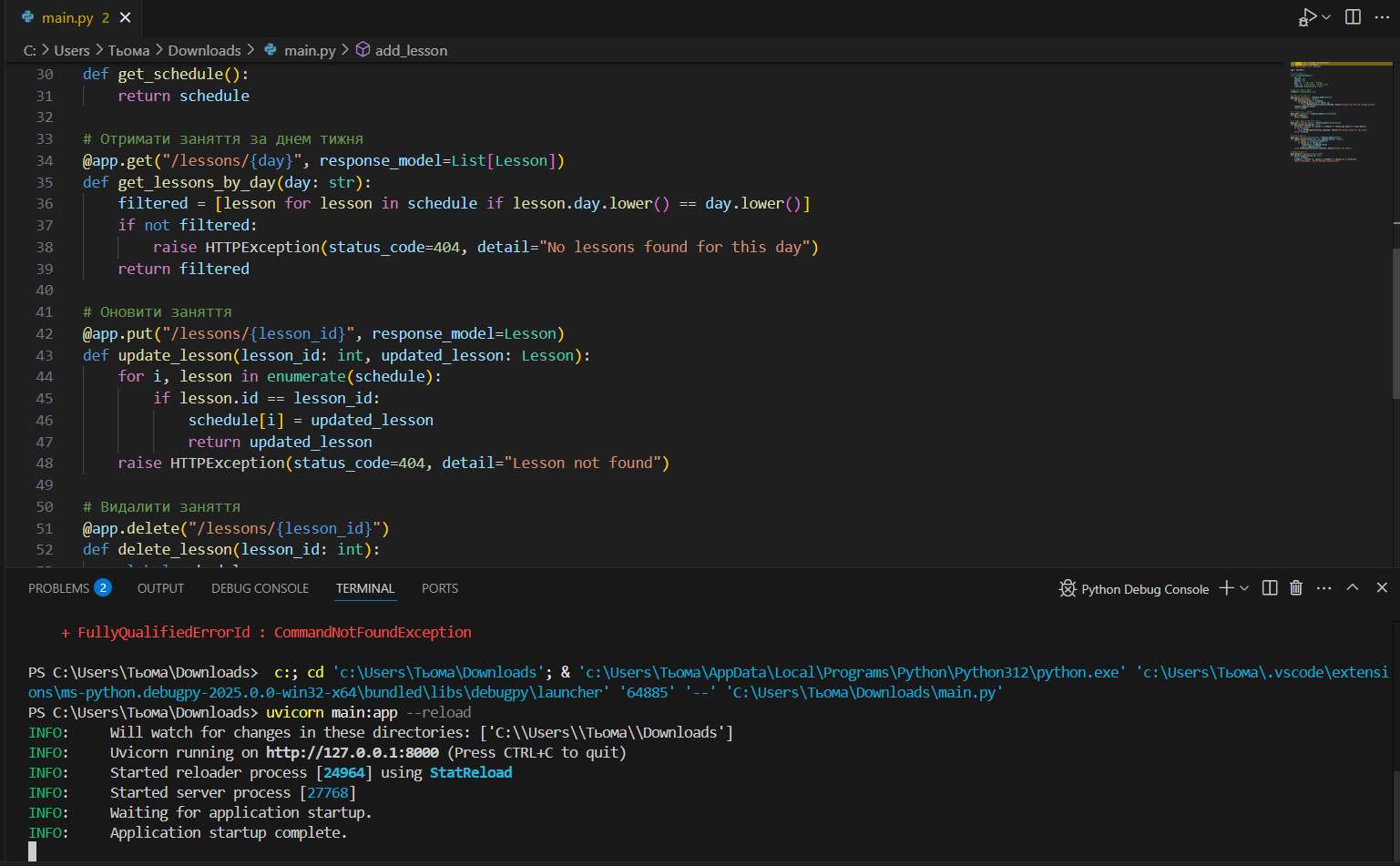
**Перед запуском коду потрібно встановити бібліотеки за допомогою команд:**

pip install fastapi та pip install uvicorn

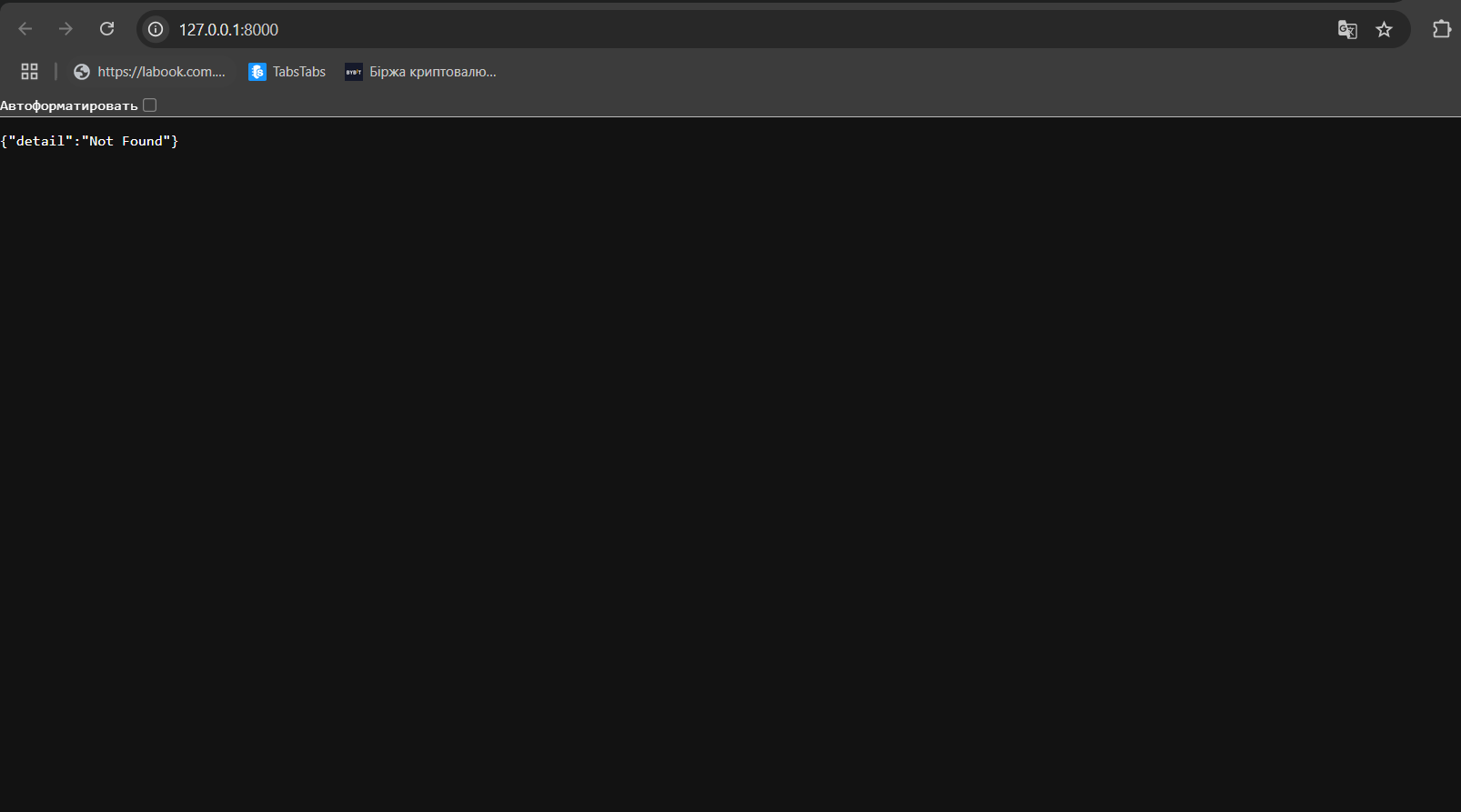
**Щоб запустити код потрібно ввести команду:**

uvicorn main:app --reload

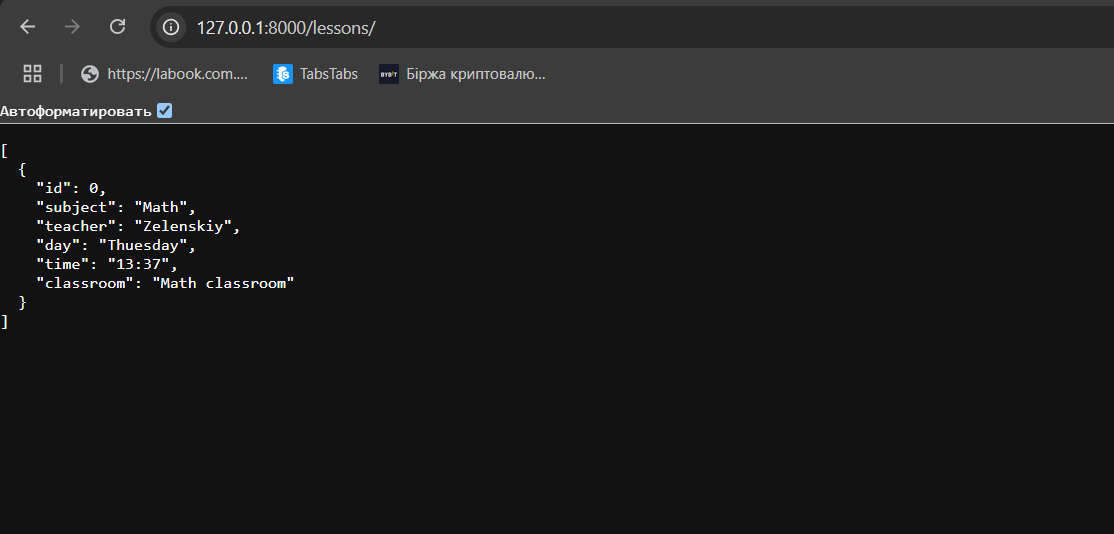
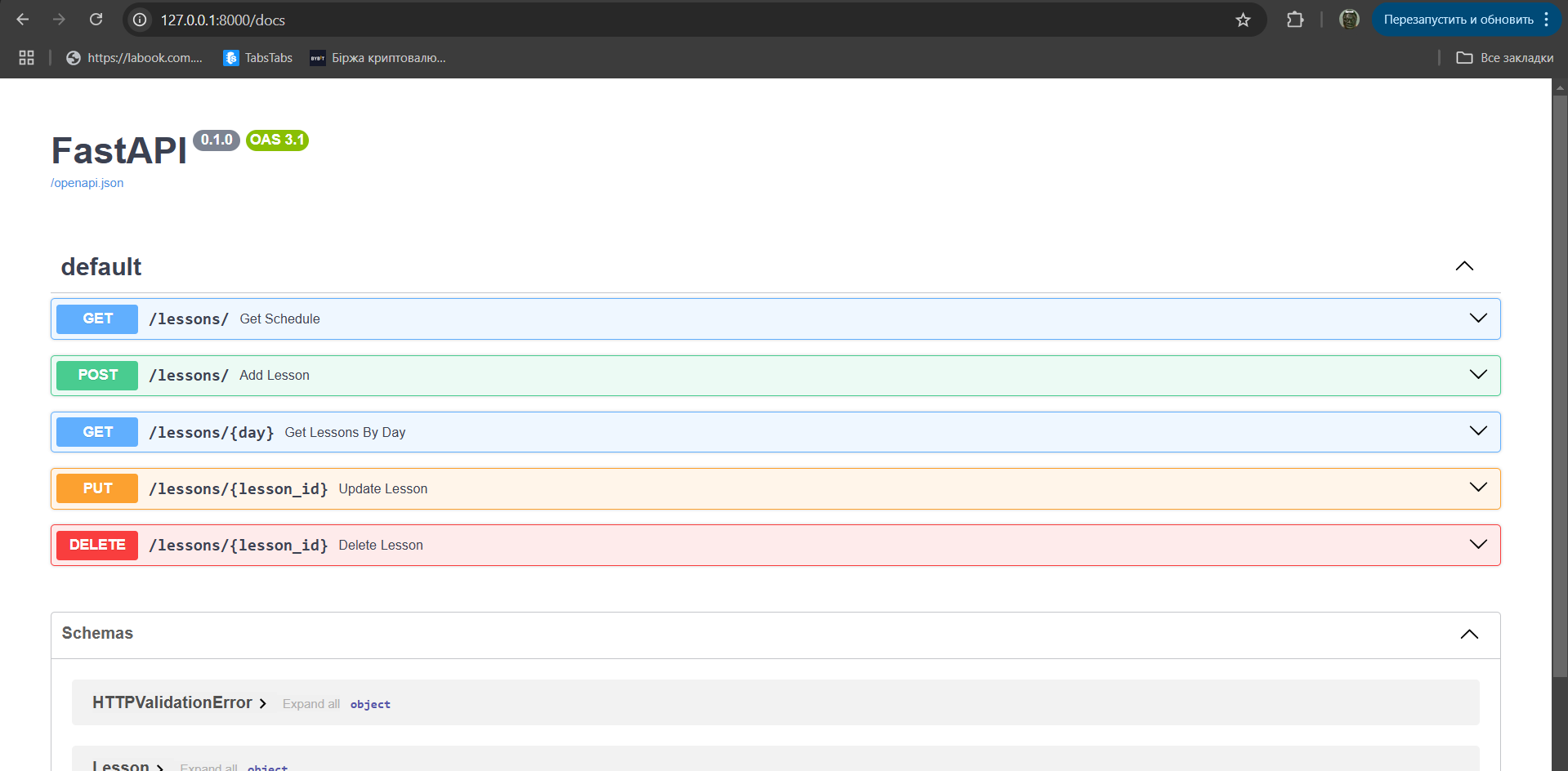
**Бот активований:**



**Переходимо по посиланні:** <http://127.0.0.1:8000/>і бачимо що код працює.



За посиланням [**http://127.0.0.1:8000/docs**](http://127.0.0.1:8000/docs) **ми можемо добавляти дані до бібліотеки нашого Api**



# **6 Висновки**

**Протягом цього періоду ми детально ознайомилися з основами мови програмування Python, поступово переходячи від базового синтаксису, структур даних і основних операцій до більш складних тем, таких як робота з базами даних, об’єктно-орієнтоване програмування, обробка файлів, виконання мережевих запитів і створення веб-додатків.**

**Здобуті знання та практичні навички не лише заклали міцний фундамент для подальшого розвитку в цій галузі, а й відкрили можливості для реалізації складних проєктів, автоматизації рутинних процесів та ефективної роботи з великими обсягами даних.**

# **ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ**

1. Посібник з мови Python<https://docs.python.org/3/reference/>
2. Офіційний посібник для початківців <https://docs.python.org/3/tutorial>
3. Початковий посібник для новачків <https://www.python.org/about/gettingstarted>
4. Документація стандартної бібліотеки Python <https://docs.python.org/3/library/>
5. Платформа для практики програмування <https://www.codewars.com/>