Міністерство освіти і науки України  
Національний технічний університет України

«Київський політехнічний інститут ім. І. Сікорського»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

Лабораторна робота №1

з дисципліни «Інформаційне забезпечення безпеки комп’ютерних систем»

Тема: «Розробка простого TCP-сервера на Python»

Виконав:

студент 4-го курсу, НН ІАТЕ

групи ТР-23

Ровний Г. О.

Перевірив:

Тітов В.М.

КИЇВ 2025

**Мета роботи:**

Відпрацювати навички створення мережевих додатків на основі протоколу TCP

— засвоїти принципи роботи сокетів;

— реалізувати простий сервер, який приймає запити від клієнта;

— навчитися обробляти кілька підключень;

— протестувати роботу сервера та клієнта.

**Поставлене завдання:**

1. Підготувати середовище та імпортувати модуль socket.
2. Реалізувати TCP-сервер, що:

— здійснює прослуховування на заданому порті;

— приймає з’єднання клієнта;

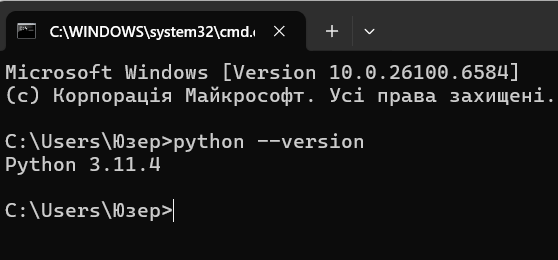
— отримує повідомлення, перетворює його у верхній регістр та повертає назад.

1. Реалізувати простого клієнта для тестування сервера.
2. Розширити сервер так, щоб він обслуговував кілька клієнтів послідовно (без багатопоточності).
3. При бажанні: додати обробку помилок і таймаутів.

**Результат виконання роботи**

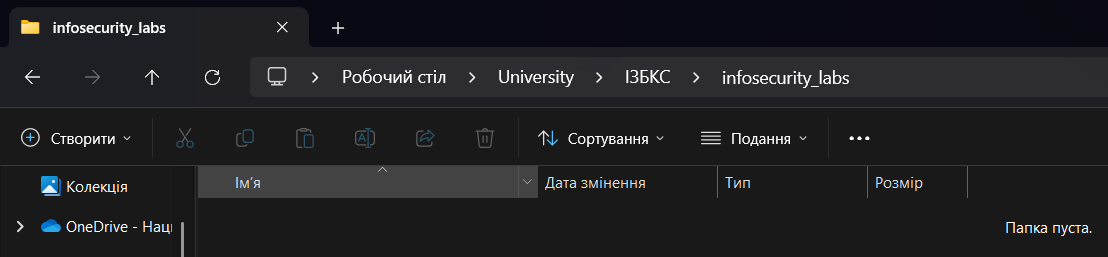
* 1. Підготовка середовища

Переконайся, що Python 3 встановлений:

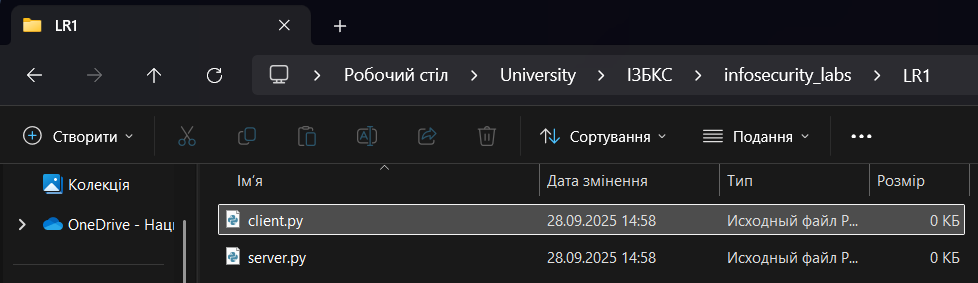


Створи папку для лабораторної, наприклад:

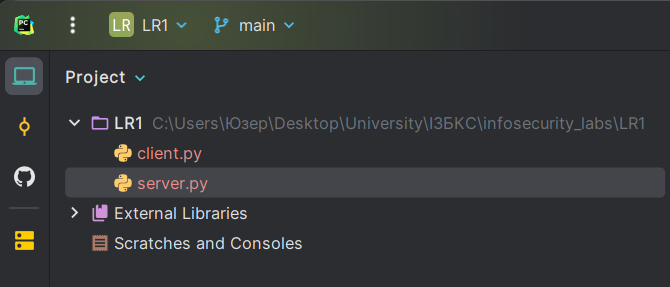
C:\Users\Юзер\Desktop\University\ІЗБКС\infosecurity\_labs



Усередині папки створи два файли:

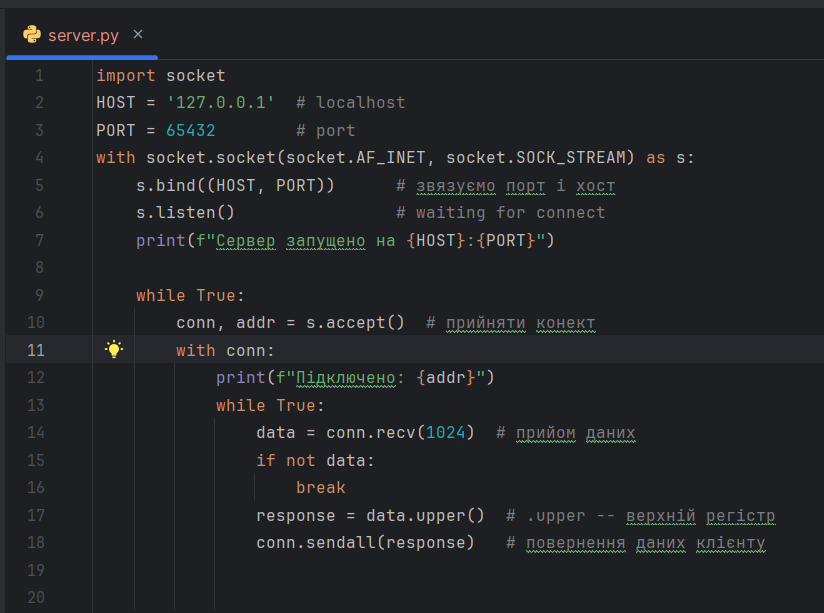


Перейдемо до PyCharm

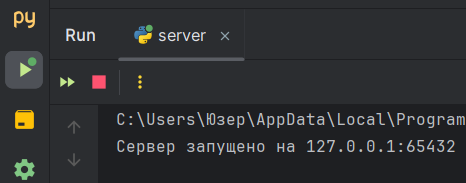


* 1. Реалізація ТСР сервера

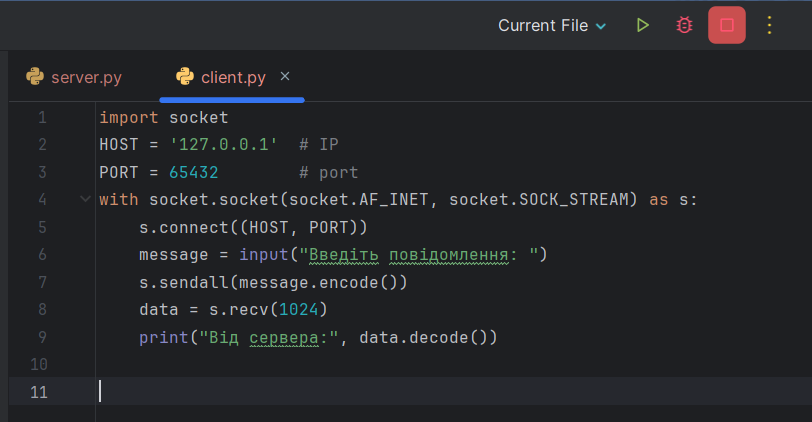
Імпортуйте модуль і створіть сокет:



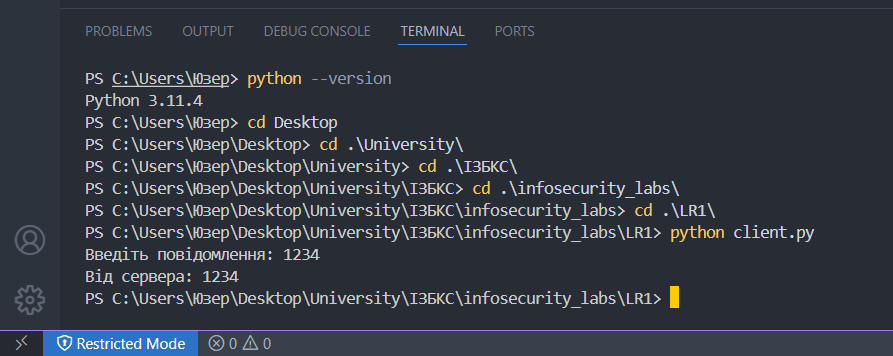
Запустимо server.py:



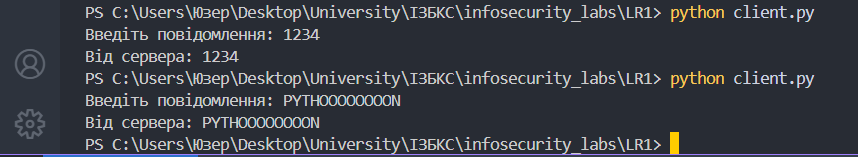
* 1. **Реалізація ТСР клієнта**

****

У новому вікні запустимо client.py і введено наприклад 1234:



* 1. **Обслуговування кількох клієнтів**



**Відповіді на запитання**

* 1. Яка роль bind() і чому важливо перевіряти помилки?

Bind () прив’язує сокет до певної IP-адреси та порту. Завдяки цьому сервер «знає», на якій адресі він має слухати клієнтів, і клієнти можуть підключатися саме до цього порту. Якщо, наприклад, порт уже зайнятий іншою програмою, або в нас немає прав для його використання, виклик bind() завершиться з помилкою. Саме тому важливо перевіряти помилки під час виконання цього кроку, адже без цього можна навіть не помітити, що сервер фактично не працює.

* 1. Як TCP забезпечує надійність доставки?

TCP був спеціально розроблений для того, щоб забезпечити надійність передачі даних. Він ділить дані на сегменти, кожен із яких отримує свій порядковий номер. Якщо пакети приходять не по черзі, TCP відновлює правильний порядок. Кожен отриманий сегмент підтверджується спеціальним повідомленням ACK, а якщо підтвердження не приходить, відправник надсилає дані повторно. Крім того, TCP перевіряє цілісність даних за допомогою контрольних сум, щоб упевнитися, що повідомлення не пошкодилось у дорозі. Ще однією важливою особливістю є керування потоком і перевантаженням: протокол стежить, щоб отримувач не був перевантажений, і водночас намагається не забивати мережу зайвими пакетами. Завдяки всім цим механізмам TCP гарантує, що дані або будуть доставлені правильно й у повному обсязі, або з’єднання буде розірвано.

* 1. Які обмеження послідовної обробки клієнтів?

Якщо ми будуємо сервер із **послідовною обробкою клієнтів**, це означає, що він може одночасно обслуговувати лише одного користувача. Тобто сервер приймає перше підключення, обробляє запити цього клієнта, і лише після того, як клієнт відключиться, він може перейти до наступного. З одного боку, це простий і зрозумілий підхід, який добре підходить для навчання чи невеликих тестових застосунків. Але з іншого боку, у нього є суттєві обмеження: якщо один клієнт зависне або триматиме з’єднання занадто довго, усі інші клієнти будуть змушені чекати. Такий підхід погано масштабується й не підходить для систем, які повинні обслуговувати десятки або сотні клієнтів одночасно. Саме тому в реальних умовах застосовують багатопоточність, асинхронні підходи або навіть окремі процеси для паралельної роботи з великою кількістю користувачів.

**Висновок:**

У результаті виконання лабораторної роботи було розроблено простий клієнт-серверний застосунок на основі протоколу TCP. Під час роботи було вивчено принципи створення сокетів, механізм прив’язки сокета до адреси та порту за допомогою функції bind(), а також особливості обробки вхідних з’єднань