**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**БУДІНИЦТВА І АРХІТЕКТУРИ**

**КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Звіт з лабораторної роботи №4**

**Дисципліни «Програмування мовою Python»**

**Варіант 23**

**Студента 2-го курсу**

**групи ІСТ-23**

**Шепель Р.С.**

**Перевірив:**

**Асистент кафедри**

**Долгополов С.Ю.**

**Київ-2024**

**Лабораторна робота № 4. Обробка рядків та регулярні вирази**

Мета: поглибити розуміння роботи з рядками в Python, освоїти методи форматування рядків та навчитися ефективно використовувати регулярні вирази для обробки текстової інформації.

**Хід Роботи:**

**Тема:** Аналіз юридичних документів. Витяг дат, імен, номерів справ та телефонних номерів.

1. **Створення файлу програми**

* Відкрив середовище розробки (VS Code)
* Створив новий файл з назвою «Lab4\_1.py»

****

Рисунок 1.1 Створення нового .ру файлу у IDE VS Code

1. **Імпорт необхідних бібліотек**



Рисунок 1.2 Імпорт бібліотек

Даний блок коду імпортує модуль re для роботи з регулярними виразами та клас Counter з модуля collections для підрахунку частоти елементів.

1. **Реалізація функцій для обробки рядків**

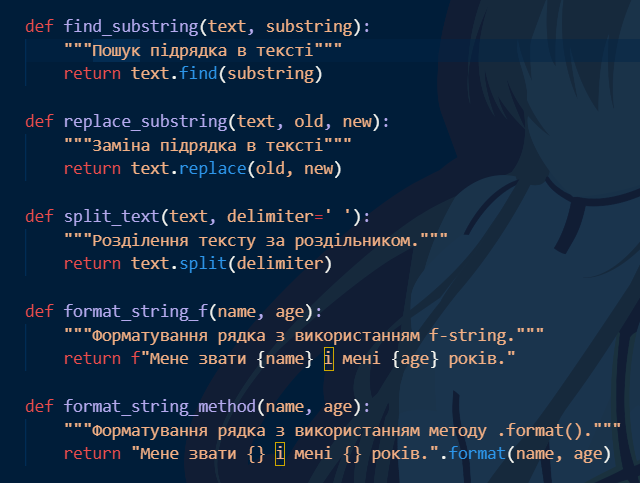
****

Рисунок 1.3 Реалізація функцій для обробки рядків

Даний блок коду містить функції для базових операцій з рядками, включаючи пошук підрядка, заміну підрядка, розділення тексту та форматування рядків з використанням f-strings та методу .format().

1. Реалізація функцій для роботи з регулярними виразами

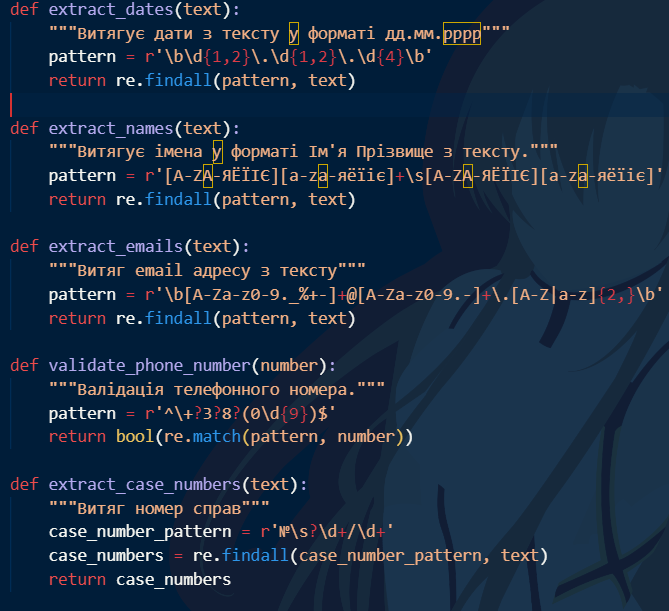


Рисунок 1.4 Реалізація функцій для роботи з регулярними виразами

Даний блок коду реалізує функції для роботи з регулярними виразами, включаючи витяг email адрес, витяг дати, валідацію телефонних номерів, витяг імен та витяг номерів справ.

1. Реалізація функцій для аналізу тексту

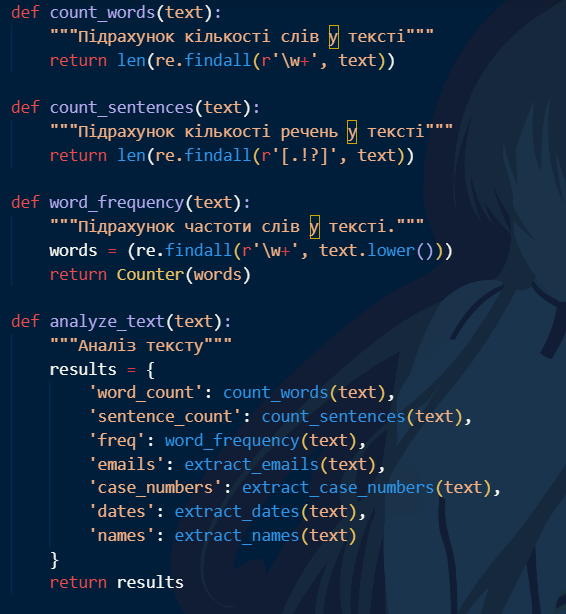


Рисунок 1.5 Реалізація функцій для аналізу тексту.

Даний блок коду містить функції для аналізу тексту, включаючи підрахунок слів, речень, визначення частоти слів, а також функцію analyze\_text(), яка об'єднує всі ці аналізи в один результат

1. Реалізація функції для форматування результатів

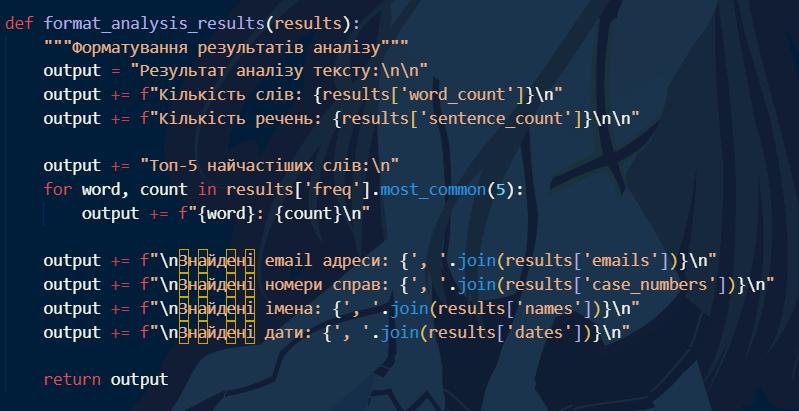


Рисунок 1.6 Реалізація функції для форматування результатів

Даний блок коду реалізує функцію для форматування результатів аналізу тексту у зручний для читання формат.

1. Реалізація головної функції програми

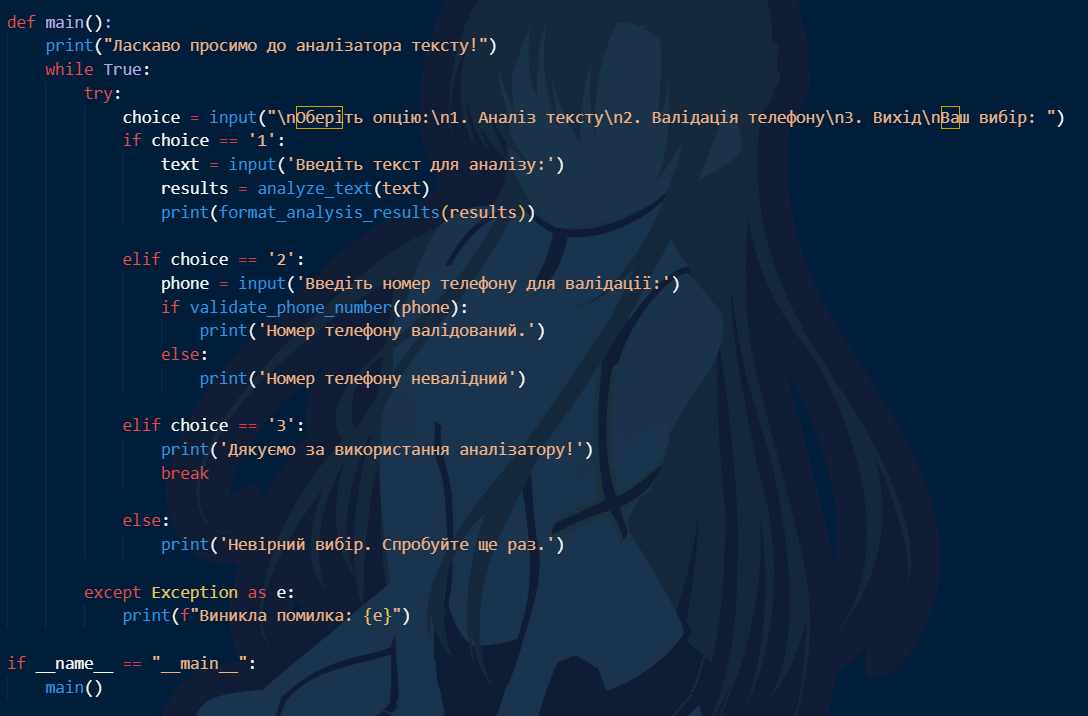


Рисунок 1.7 Реалізація головної функції програми

Даний блок коду реалізує головну функцію програми, яка забезпечує інтерактивний інтерфейс для користувача, дозволяючи вибирати між аналізом тексту та валідацією телефонного номера

1. Створення наборів тестів



Рисунок 1.8 Створення нового .ру файлу у IDE VS Code

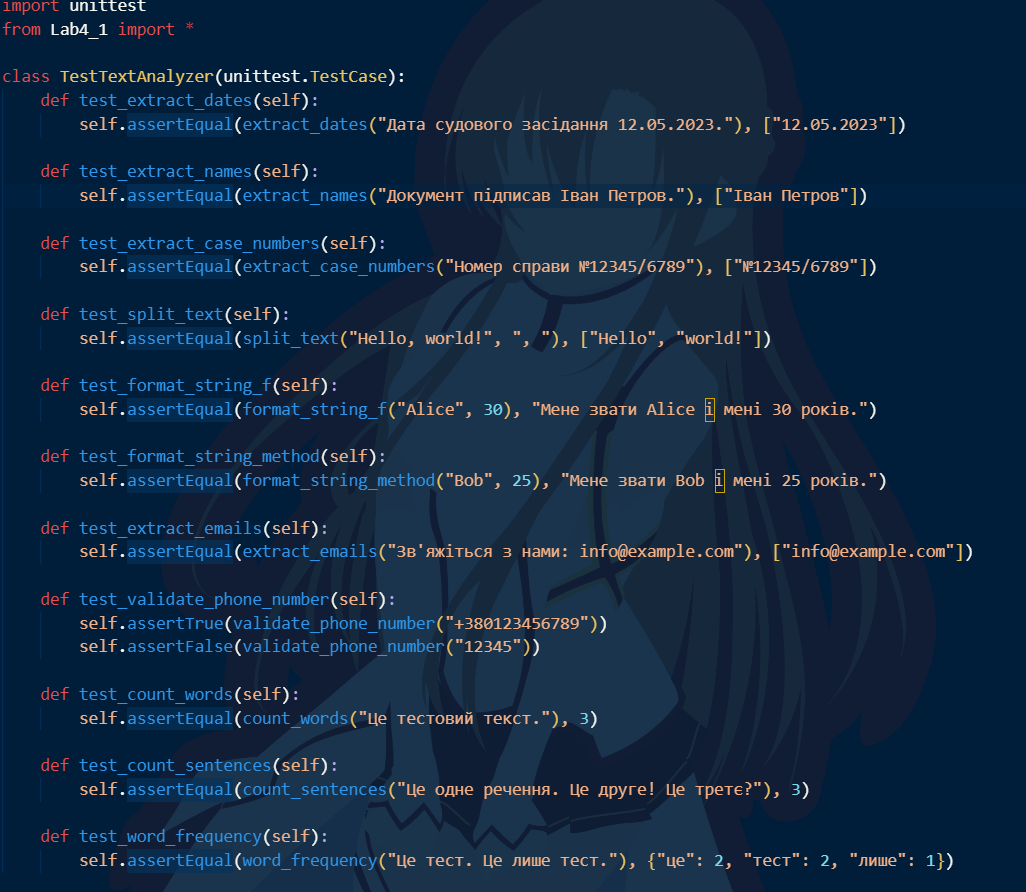


Рисунок 1.9 Реалізація тестів №1

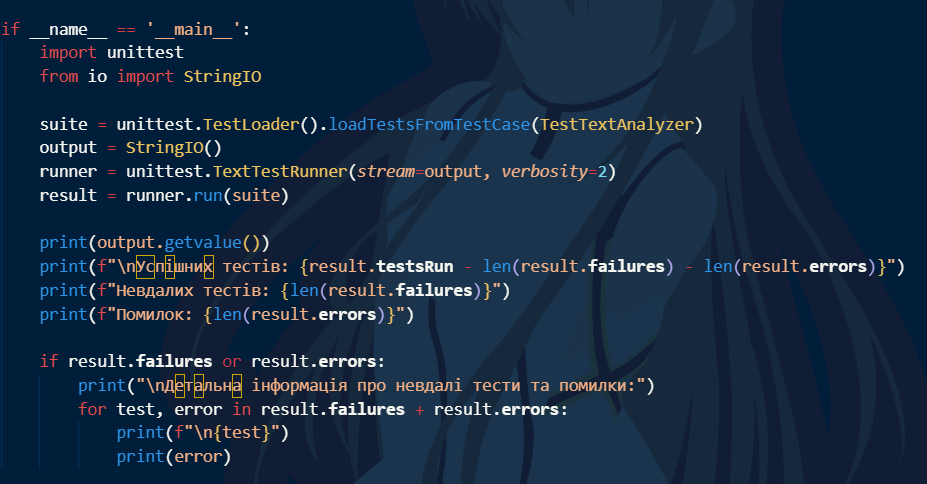


Рисунок 1.10 Реалізація функції тестування програми.

Даний блок коду створює набір тестів для перевірки коректності роботи всіх функцій програми. Він також включає код для запуску тестів та виведення детальної інформації про результати тестування.

**test\_extract\_dates**: перевіряє функцію extract\_dates. Тест витягує дату «12.05.2023» з речення «Дата судового засідання 12.05.2023.» і очікує, що дата буде знайдена.

**test\_extract\_names**: перевіряє функцію extract\_names. Тест перевіряє, чи коректно витягується ім'я «Іван Петров» з речення «Документ підписав Іван Петров.»

**test\_extract\_case\_numbers**: перевіряє функцію extract\_case\_numbers. Тест шукає номер справи «№12345/6789» у рядку «Номер справи №12345/6789» і очікує, що він буде знайдений.

**test\_split\_text**: перевіряє функцію split\_text. Розділяє рядок «Hello, world!» за роздільником «, » і очікує отримати список ["Hello", "world!"].

**test\_format\_string\_f**: перевіряє функцію format\_string\_f, яка використовує f-strings. Тест перевіряє, чи правильно форматується рядок з ім'ям «Alice» та віком 30.

**test\_format\_string\_method**: аналогічний попередньому, але перевіряє функцію format\_string\_method, яка використовує метод .format(). Тест перевіряє форматування з ім'ям «Bob» та віком 25.

**test\_extract\_emails**: перевіряє функцію extract\_emails. Тест перевіряє, чи правильно витягуються email адреси з тексту «Зв'яжіться з нами: info@example.com».

**test\_validate\_phone\_number**: перевіряє функцію validate\_phone\_number. Тест перевіряє, чи коректно валідується правильний номер телефону «+380123456789» і чи відхиляється некоректний номер «12345».

**test\_count\_words**: перевіряє функцію count\_words. Тест підраховує кількість слів у реченні «Це тестовий текст.» і очікує отримати результат 3.

**test\_count\_sentences**: перевіряє функцію count\_sentences. Тест перевіряє, чи правильно підраховується кількість речень у тексті «Це одне речення. Це друге! Це третє?» (очікується 3 речення).

**test\_word\_frequency**: перевіряє функцію word\_frequency. Тест підраховує частоту слів у реченні «Це тест. Це лише тест.» і очікує, що слово «тест» з'являється 2 рази.

Кожен з цих тестів перевіряє окрему функціональність програми, забезпечуючи, що всі основні операції з обробки тексту працюють коректно. Використання різних методів assert (як assertEqual, assertTrue, assertFalse) дозволяє перевірити різні аспекти роботи функцій.

Додатковий код в блоці if \_\_name\_\_ == '\_\_main\_\_': налаштовує виконання тестів, забезпечуючи детальний вивід результатів, включаючи кількість успішних та невдалих тестів, а також детальну інформацію про помилки, якщо такі виникнуть.

**Примітка**: Для коректного запуску тестів виконайте python test\_text\_analyzer.py у терміналі.

1. Приклад використання програми

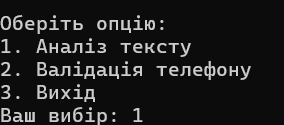


Рисунок 1.11 Інтерфейс користувача

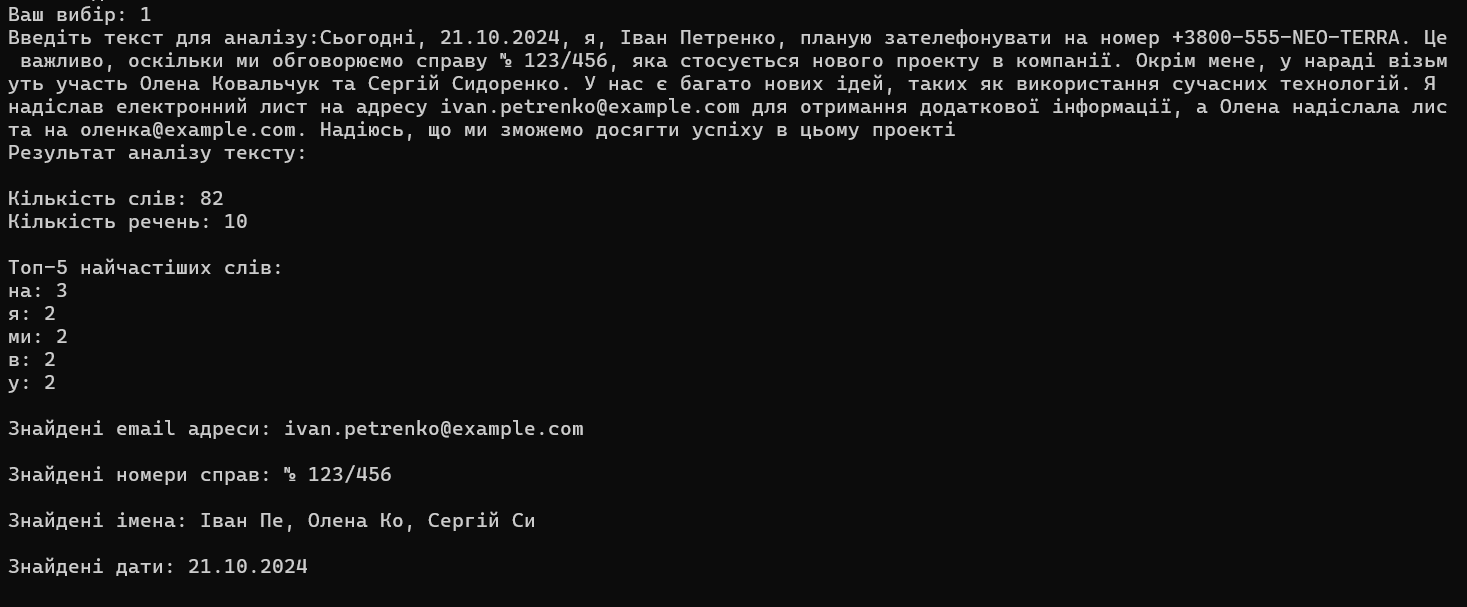


Рисунок 1.12 результат виконання опції №1

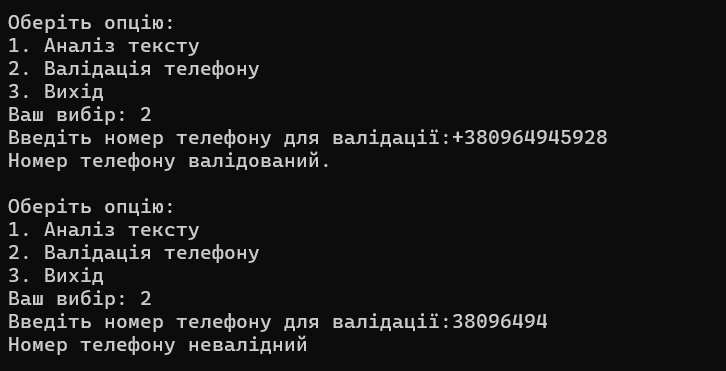


Рисунок 1.13 Результат виконання опції №2

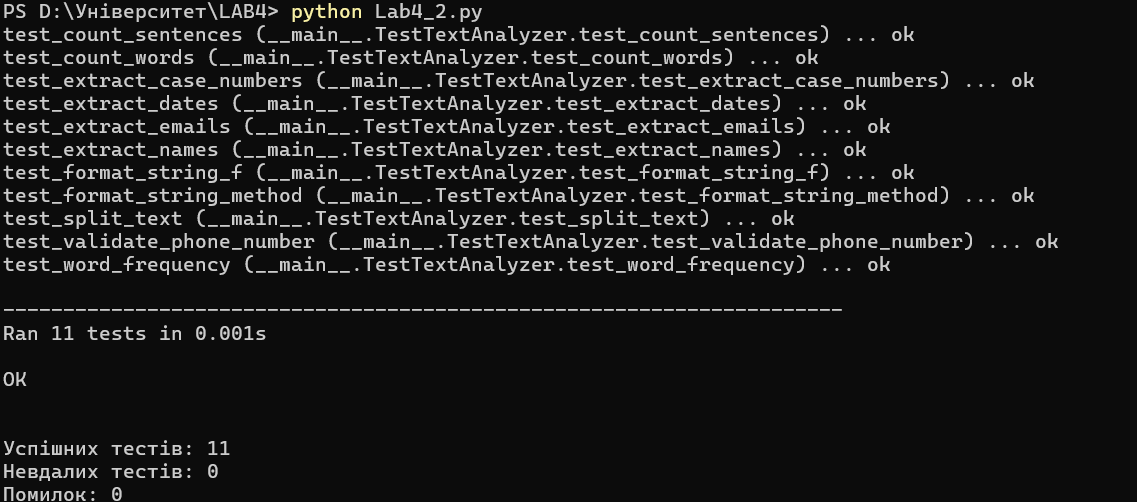


Рисунок 1.14 Результат тестування функцій

**Повний код програми**

**Lab4\_1.py**

*# 1. Обробка юридичних документів: витяг дат, імен, номерів справ.*

*# 2. Телефонні номери (наприклад, +3000-555-NEO-TERRA).*

import re

from collections import Counter

def find\_substring(text, substring):

    """Пошук підрядка в тексті"""

    return text.find(substring)

def replace\_substring(text, old, new):

    """Заміна підрядка в тексті"""

    return text.replace(old, new)

def split\_text(text, delimiter=' '):

    """Розділення тексту за роздільником."""

    return text.split(delimiter)

def format\_string\_f(name, age):

    """Форматування рядка з використанням f-string."""

    return f"Мене звати {name} і мені {age} років."

def format\_string\_method(name, age):

    """Форматування рядка з використанням методу .format()."""

    return "Мене звати {} і мені {} років.".format(name, age)

def extract\_dates(text):

    """Витягує дати з тексту у форматі дд.мм.рррр"""

    pattern = r'\b\d{1,2}\.\d{1,2}\.\d{4}\b'

    return re.findall(pattern, text)

def extract\_names(text):

    """Витягує імена у форматі Ім'я Прізвище з тексту."""

    pattern = r'\b[A-ZА-ЯЁЇІЄ][a-zа-яёїіє]+(?:\s[A-ZА-ЯЁЇІЄ][a-zа-яёїіє]+)+\b'

    return re.findall(pattern, text)

def extract\_emails(text):

    """Витяг email адресу з тексту"""

    pattern = r'\b[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'

    return re.findall(pattern, text)

def validate\_phone\_number(number):

    """Валідація телефонного номера."""

    pattern = r'^\+?3?8?(0\d{9})$'

    return bool(re.match(pattern, number))

def extract\_case\_numbers(text):

    """Витяг номер справ"""

    case\_number\_pattern = r'№\s?\d+/\d+'

    case\_numbers = re.findall(case\_number\_pattern, text)

    return case\_numbers

def count\_words(text):

    """Підрахунок кількості слів у тексті"""

    return len(re.findall(r'\w+', text))

def count\_sentences(text):

    """Підрахунок кількості речень у тексті"""

    return len(re.findall(r'[.!?]', text))

def word\_frequency(text):

    """Підрахунок частоти слів у тексті."""

    words = (re.findall(r'\w+', text.lower()))

    return Counter(words)

def analyze\_text(text):

    """Аналіз тексту"""

    results = {

        'word\_count': count\_words(text),

        'sentence\_count': count\_sentences(text),

        'freq': word\_frequency(text),

        'emails': extract\_emails(text),

        'case\_numbers': extract\_case\_numbers(text),

        'dates': extract\_dates(text),

        'names': extract\_names(text)

    }

    return results

def format\_analysis\_results(results):

    """Форматування результатів аналізу"""

    output = "Результат аналізу тексту:\n\n"

    output += f"Кількість слів: {results['word\_count']}\n"

    output += f"Кількість речень: {results['sentence\_count']}\n\n"

    output += "Топ-5 найчастіших слів:\n"

    for word, count in results['freq'].most\_common(5):

        output += f"{word}: {count}\n"

    output += f"\nЗнайдені email адреси: {', '.join(results['emails'])}\n"

    output += f"\nЗнайдені номери справ: {', '.join(results['case\_numbers'])}\n"

    output += f"\nЗнайдені імена: {', '.join(results['names'])}\n"

    output += f"\nЗнайдені дати: {', '.join(results['dates'])}\n"

    return output

def main():

    print("Ласкаво просимо до аналізатора тексту!")

    while True:

        try:

            choice = input("\nОберіть опцію:\n1. Аналіз тексту\n2. Валідація телефону\n3. Вихід\nВаш вибір: ")

            if choice == '1':

                text = input('Введіть текст для аналізу:')

                results = analyze\_text(text)

                print(format\_analysis\_results(results))

            elif choice == '2':

                phone = input('Введіть номер телефону для валідації:')

                if validate\_phone\_number(phone):

                    print('Номер телефону валідований.')

                else:

                    print('Номер телефону невалідний')

            elif choice == '3':

                print('Дякуємо за використання аналізатору!')

                break

            else:

                print('Невірний вибір. Спробуйте ще раз.')

        except Exception as e:

            print(f"Виникла помилка: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**Lab4\_2.py**

import unittest

from Lab4\_1 import \*

class TestTextAnalyzer(unittest.TestCase):

    def test\_extract\_dates(self):

        self.assertEqual(extract\_dates("Дата судового засідання 12.05.2023."), ["12.05.2023"])

    def test\_extract\_names(self):

        self.assertEqual(extract\_names("Документ підписав Іван Петров."), ["Іван Петров"])

    def test\_extract\_case\_numbers(self):

        self.assertEqual(extract\_case\_numbers("Номер справи №12345/6789"), ["№12345/6789"])

    def test\_split\_text(self):

        self.assertEqual(split\_text("Hello, world!", ", "), ["Hello", "world!"])

    def test\_format\_string\_f(self):

        self.assertEqual(format\_string\_f("Alice", 30), "Мене звати Alice і мені 30 років.")

    def test\_format\_string\_method(self):

        self.assertEqual(format\_string\_method("Bob", 25), "Мене звати Bob і мені 25 років.")

    def test\_extract\_emails(self):

        self.assertEqual(extract\_emails("Зв'яжіться з нами: info@example.com"), ["info@example.com"])

    def test\_validate\_phone\_number(self):

        self.assertTrue(validate\_phone\_number("+380123456789"))

        self.assertFalse(validate\_phone\_number("12345"))

    def test\_count\_words(self):

        self.assertEqual(count\_words("Це тестовий текст."), 3)

    def test\_count\_sentences(self):

        self.assertEqual(count\_sentences("Це одне речення. Це друге! Це третє?"), 3)

    def test\_word\_frequency(self):

        self.assertEqual(word\_frequency("Це тест. Це лише тест."), {"це": 2, "тест": 2, "лише": 1})

if \_\_name\_\_ == '\_\_main\_\_':

    import unittest

    from io import StringIO

    suite = unittest.TestLoader().loadTestsFromTestCase(TestTextAnalyzer)

    output = StringIO()

    runner = unittest.TextTestRunner(*stream*=output, *verbosity*=2)

    result = runner.run(suite)

    print(output.getvalue())

    print(f"\nУспішних тестів: {result.**testsRun** - len(result.**failures**) - len(result.**errors**)}")

    print(f"Невдалих тестів: {len(result.**failures**)}")

    print(f"Помилок: {len(result.**errors**)}")

    if result.**failures** or result.**errors**:

        print("\nДетальна інформація про невдалі тести та помилки:")

        for test, error in result.**failures** + result.**errors**:

            print(f"\n{test}")

            print(error)

**Пояснення до коду**

1. **Модуль re**: Використовується для роботи з регулярними виразами, що дозволяє ефективно виконувати складні операції пошуку та обробки тексту, такі як витягування дат, імен, email адрес та телефонних номерів.
2. **Клас Counter з модуля collections**: Застосовується для підрахунку частоти слів у тексті. Це спрощує аналіз тексту та забезпечує ефективне зберігання результатів у вигляді словника, де ключами є слова, а значеннями — їх частота.
3. **Функції find\_substring, replace\_substring та split\_text**: Демонструють базові операції з рядками, використовуючи вбудовані методи Python. Це забезпечує простоту та читабельність коду.

* find\_substring виконує пошук підрядка в тексті,
* replace\_substring дозволяє замінювати частини тексту,
* split\_text ділить рядок на частини за вказаним роздільником.

1. **Форматування рядків**: Реалізовано двома способами — за допомогою f-strings (в format\_string\_f) та методу .format() (в format\_string\_method). Це демонструє різні підходи до форматування рядків в Python.
2. **Регулярні вирази**: Використовуються в функціях extract\_emails, validate\_phone\_number, extract\_hashtags, extract\_mentions, extract\_dates, extract\_names та extract\_case\_numbers для точного пошуку специфічних патернів у тексті. Наприклад, extract\_emails витягує email адреси, а validate\_phone\_number перевіряє, чи відповідає номер телефону певному формату.
3. **Функція analyze\_text**: Об'єднує різні методи аналізу тексту в одну операцію, що дозволяє отримати комплексний результат аналізу тексту за один виклик функції. Вона повертає словник з результатами аналізу.
4. **Функція format\_analysis\_results**: Відповідає за форматування результатів аналізу, генеруючи зручний для читання вихідний текст. Це покращує читабельність коду, дотримуючись принципу розділення відповідальності.
5. **Головний цикл програми**: Реалізований у функції main(), що забезпечує інтерактивну взаємодію з користувачем через консольне меню. Користувач може обирати різні функції програми.
6. **Обробка винятків**: Виконується в головному циклі програми, що підвищує стійкість програми до помилок введення та інших непередбачених ситуацій. Це забезпечує коректність роботи програми навіть при неправильних даних.
7. **Тести**: Реалізовані з використанням модуля unittest, що забезпечує структурований підхід до тестування та можливість автоматизації процесу перевірки коду. Кожен тест відповідає за перевірку конкретної функції, що дозволяє легко ідентифікувати проблеми при внесенні змін до коду.
8. **Використання StringIO в тестах**: Дозволяє перехоплювати вивід тестів для подальшого аналізу, що корисно для автоматизованого тестування. Це дає можливість отримати детальний звіт про результати тестування.
9. **Виведення статистики тестування**: Реалізовано після виконання всіх тестів, що дозволяє швидко оцінити загальний стан програми та виявити проблеми.

**Повний код для вилучення інформації зі спільного тексту (ID-кодів жителів)**

**Lab4\_3.py**

import re

def extract\_id\_codes(text):

    """

    Вилучає ID-коди жителів з тексту.

    ID-код має формат: дві літери, цифра, літера, дефіс, п'ять цифр,

    дефіс, дві літери.

    Наприклад: NT3K-78901-XZ

    """

    pattern = r'\b[A-Z]{2}\d[A-Z]-\d{5}-[A-Z]{2}\b'

    return re.findall(pattern, text)

*# Приклад використання*

futuristic\_text = """

Вітаємо вас у NeoTerra 3000, де технології та природа зливаються воєдино!

Наше місто, засноване в 2150 році, є домом для 5 мільйонів жителів та

незліченної кількості ШІ-помічників.

Кожен житель NeoTerra має свій унікальний ID-код, наприклад NT3K-78901-XZ,

який використовується для доступу до всіх міських сервісів. Наші квантові

комп'ютери, такі як QC-9000, обробляють петабайти даних щосекунди,

забезпечуючи безперебійну роботу міста.

…

"""

*# Вилучення ID-кодів*

id\_codes = extract\_id\_codes(futuristic\_text)

*# Виведення результатів*

print("Знайдені ID-коди жителів:")

for code in id\_codes:

    print(code)

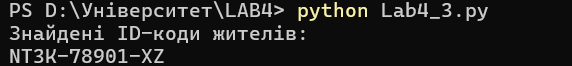
****

Рисунок 1.15 Результат вилучення ID-кодів з тексту

**Пояснення до коду**

* 1. **Імпорт модуля re**:
* Модуль re використовується для роботи з регулярними виразами в Python. Це дозволяє виконувати складні пошукові та обробні операції на рядках.
  1. **Функція extract\_id\_codes(text)**:
* Ця функція призначена для вилучення ID-кодів жителів з наданого тексту.
* ID-код має специфічний формат, що складається з двох літер, однієї цифри, однієї літери, дефіса, п'яти цифр, ще одного дефіса та двох літер.
  1. **Регулярний вираз**:
* pattern = r'\b[A-Z]{2}\d[A-Z]-\d{5}-[A-Z]{2}\b': Цей вираз описує структуру ID-коду:
* \b: Позначає межу слова, що гарантує, що ID-код не є частиною іншого слова.
* [A-Z]{2}: Вимагає дві великі літери на початку.
* \d: Вимагає одну цифру.
* [A-Z]: Вимагає одну велику літеру.
* -: Вимога наявності дефісу.
* \d{5}: Вимагає п’яти цифр.
* -: Ще один дефіс.
* [A-Z]{2}: Вимога наявності двох великих літер в кінці.
* \b: Ще одна межа слова, що завершує пошук.
  1. **Метод re.findall(pattern, text)**:
* Цей метод виконує пошук усіх входжень вказаного патерну в тексті та повертає їх у вигляді списку. Якщо жодного входження не знайдено, повертається порожній список.
* Виконання функції дозволяє отримати ID-коди, які є частиною тексту, і зберегти їх у змінній id\_codes.

**Контрольні запитання №1**

Які основні методи доступні для роботи з рядками в Python?

1. **Основні методи рядків**:

* .lower(): Перетворює всі літери рядка на малі.
* .upper(): Перетворює всі літери рядка на великі.
* .title(): Перетворює перші літери слів на великі.
* .strip(): Видаляє пробіли з початку і кінця рядка.
* .split(): Розділяє рядок на список підрядків за заданим роздільником.
* .join(): Об’єднує список рядків в один рядок за допомогою заданого роздільника.

1. **Методи пошуку**:

* .find(): Повертає індекс першого входження підрядка, або -1, якщо не знайдено.
* .rfind(): Повертає індекс останнього входження підрядка.
* .count(): Повертає кількість входжень підрядка в рядку.

1. **Методи заміни**:

* .replace(): Замінює всі входження підрядка на інший підрядок.

1. **Форматування рядків**:

* f-рядки: Використовуються для форматування рядків (наприклад, f"Hello, {name}!").
* .format(): Метод для форматування рядків.

**Контрольні запитання №2**

Як виконати заміну тексту з використанням функції замість простого рядка в re.sub()?

* Для того щоб виконати заміну в тексті з використанням функції re.sub(), передаємо функцію заміни як другий аргумент.