Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»
_____Інститут електроенергетики _____
(інститут)
_____факультет інформаційних технологій _____
(факультет)
Кафедра _____інформаційних технологій та комп'ютерної інженерії
_____
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня ____бакалавра___
(бакалавра, спеціаліста, магістра)

студента _____ Гриненко Руфіни Олегівни _____

(ПІБ)
академічної групи _____126-20з-1_____
(шифр)
спеціальності _____126 «Інформаційні системи та технології _____
(код і назва спеціальності)
за освітньо-професійною програмою _____
(за наявності)
_____«Інформаційні системи та технології»_____
(офіційна назва)

на тему __Інструмент аналізу та обробки csv-даних мовою Python_____
(назва за наказом ректора)

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи	проф. Нікулін С.Л.			
розділів:				

Рецензент				

Нормоконтролер	проф. Коротенко Г.М.			

Дніпро
2024

ЗАТВЕРДЖЕНО:
завідувач кафедри
_____інформаційних технологій _____
_____та комп'ютерної інженерії_____
(повна назва)
_____ _Гнатушенко В.В._
(підпис) (прізвище, ініціали)

«_____»_____2024 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня ____**бакалавра**_____
(бакалавра, спеціаліста, магістра)

студенту_____Гриненко Р.О._____ **академічної групи** _____126-20з-1____
(прізвище та ініціали) (шифр)
спеціальності _____126 «Інформаційні системи та технології»_____
за освітньою-професійною програмою _____
(за наявності)
_____«Інформаційні системи та технології»_____
на тему ___Інструмент аналізу та обробки csv-даних мовою Python_____

затверджену наказом ректора НТУ «Дніпровська політехніка» від 23.05.2024 р. .№ 470-с

Розділ	Зміст	Термін виконання
Розділ 1	Аналіз стану області рішення задачі	23.05.2024 – 02.05.2024
Розділ 2	Геопросторові інструменти та інтеграція csv-даних	03.06.2024 – 13.06.2024
Розділ 3	Розробка інструменту аналізу та обробки csv-даних мовою python	14.06.2024 – 24.06.2024

Завдання видано _____ _____С.Л. Нікулін_____
(підпис керівника) (прізвище, ініціали)

Дата видачі __23.05.2024 р._____

Дата подання до екзаменаційної комісії _____

Прийнято до виконання _____ _____Гриненко Р.О._____
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 70 стор., 34 рис., 2 додатки, 14 джерело.

Об'єкт розробки: інструмент аналізу та обробки csv-даних мовою Python.

Мета кваліфікаційної роботи: розробка інструменту аналізу та обробки csv-даних мовою Python з акцентом на геопросторовий аналіз та візуалізацію геоданих.

У вступі наведено стан проблеми та обґрунтована її актуальність.

Перший розділ включає в себе постановку завдання і характеристику предметної області.

У другому розділі був проведений аналіз і вибір геопросторових інструментів для реалізації поставленого завдання. Проведена інтеграція csv-даних.

У третьому розділі розроблений інструмент аналізу та обробки csv-даних мовою Python, який дозволяє створювати інтерактивні карти. Ці карти дозволяють візуально представити географічне розташування та характеристики даних, надаючи користувачам зрозумілий та інтуїтивно зрозумілий інструмент для аналізу.

Практичне значення роботи полягає у створенні інструменту аналізу та обробки csv-даних, який може бути використаний в різних сферах, таких як освітній процес, геоаналітика, маркетингові аналітики та інші, де потрібен аналіз геоданих.

Список ключових слів: ГЕОДАНІ, CSV-ДАНІ, НАВЧАЮЧИЙ ТРЕНАЖЕР, HTML, ІНТЕРАКТИВНІ КАРТИ, PHP, UML

ABSTRACT

Explanatory note: 70 pages, 34 figures, 2 appendices, 14 sources.

The object of development: csv data analysis and processing tool in Python.

The purpose of the thesis: to development of a tool for analyzing and processing csv data in Python with an emphasis on geospatial analysis and visualization of geodata.

The introduction describes the state of the problem and substantiates its relevance.

The first chapter includes the task statement and the characterization of the subject area.

In the second section, the analysis and selection of geospatial tools for the implementation of the task was carried out. Integration of csv data was carried out.

In the third chapter, a csv data analysis and processing tool is developed in Python, which allows you to create interactive maps. These maps allow you to visually represent the geographic location and characteristics of data, providing users with a clear and intuitive tool for analysis.

The practical significance of the work is to create a csv-data analysis and processing tool that can be used in various areas, such as the educational process, geo-analytics, marketing analytics and others where geo-data analysis is required.

Keywords: GEO DATA, CSV DATA, TRAINING TRAINER, HTML, INTERACTIVE MAPS, PHP, UML

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

OGC - Open Geospatial Consortium;

WMS – Web Mapping Service;

SLD – Styled Layer Descriptor;

TJS - Table Joining Service;

CSV - Comma-Separated Values;

ДГІ – добровільної географічної інформації;

ГІС – географічні інформаційні системи;

IDE – Integrated Drive Electronics.

.

ВСТУП

Актуальність роботи. У сучасному періоді стрімкого розвитку інформаційних технологій користувачі стають залежними від використання онлайн-ресурсів, веб-тренажерів та інших подібних інструментів для вдосконалення своєї професійної діяльності. Звичайні підручники втрачають актуальність через відсутність інтерактивності, яка наразі є надзвичайно популярною серед користувачів.

Онлайн-тренажери пропонують систему в Інтернеті, яка дозволяє користувачам навчатися самостійно або під керівництвом вчителя. Цей спосіб навчання користується попитом як серед звичайних користувачів, так і серед тих, хто бажає підвищити свою кваліфікацію, оскільки у сучасному світі технологій постійно розвиваються та змінюються їх можливості.

Тому користувачам необхідно мати актуальну базу знань для повноцінного та систематичного вивчення змін у цікавій їм сфері.

Об'єктом дослідження є інструмент аналізу та обробки csv-даних мовою Python.

Метою роботи є розробка інструменту аналізу та обробки csv-даних мовою Python.

1 АНАЛІЗ СТАНУ ОБЛАСТІ РІШЕННЯ ЗАДАЧІ

1.1 Актуальність задачі

Сьогодні, питання інформаційних процесів і систем, а також засобів інформатизації й автоматизації, стали надзвичайно актуальними. Практично в усіх сферах нашого життя впроваджуються сучасні технології з метою його покращення. З появою інтернету людство отримало вільний доступ до безмежної бібліотеки знань. Швидкий розвиток інформаційних технологій та зміни у сучасному світі породжують проблему з передачею великої кількості нової інформації для школярів і студентів. Також зростає вимоги до якості та обсягу знань навіть для початкових спеціалістів.

Щоб відповісти на нові вимоги до знань, потрібні нові підходи до подання інформації. Тому використання комп'ютерів з навчальними системами в школах та університетах може зробити заняття більш інтерактивними і сучасними. Навчально-розвивальні системи дозволяють подавати інформацію в цікавій формі, допомагаючи виявляти помилки або проблеми з розумінням певних тем. Це дозволяє вчителям ефективно реагувати на потреби кожного учня та допомагає усунути виникаючі проблеми. Крім того, використання комп'ютерних ігор у навчанні має великий потенціал. Вони створюють мотивацію та зацікавленість учнів, що стимулює їх активну участь у навчальному процесі. Використання ігрових прийомів допомагає залучити увагу студентів і сприяє кращому засвоєнню навчального матеріалу. Таким чином, інформаційні технології можуть допомогти покращити навчальний процес шляхом створення інтерактивних навчальних середовищ та використання ігрових методів, що сприяють мотивації та зацікавленості учнів.

Гейміфікація є ефективним засобом привернення уваги сучасних дітей, які активно використовують комп'ютерні ігри та соціальні мережі. Цей метод

вже успішно використовується в маркетингу, управлінні персоналом та особливо цінний у сфері освіти, [1].

Гейміфікація використовує елементи гри для стимулювання неігрової діяльності. Її особливість полягає в тому, що учасники фокусуються на меті реальної діяльності, а не просто на самій грі. Елементи гри вплетені в повсякденні ситуації та мотивують до конкретних дій у конкретних умовах, [2].

Процес залучення ігрових елементів, як одного з найефективніших методів підвищення результативності навчання, у розвинених країнах отримав назву «гейміфікація» або «навчання на основі цифрових ігор», [3].

Головна ідея гейміфікації полягає у використанні ігрових функцій і механізмів, таких як системи досягнень, рівні, конкуренція, співпраця, винагороди та значки, для заохочення до участі в певних заходах або конкретних діях, [4].

Гра завжди відігравала свою роль в освіті, але з підвищеним інтересом до комп'ютерних ігор останнім часом говорять про гейміфікацію як про важливий освітній тренд, [4].

Гейміфіковані освітні середовища поступово стають справжніми конкурентами традиційним освітнім методам завдяки використанню найкращих програм лояльності, найкращого ігрового дизайну та найкращої поведінкової економіки. Завдяки різноманітним компонентам, гейміфікація включає різноманітні прийоми організації навчального процесу.

Гейміфікація може мати кілька корисних функцій:

1. Залучення до навчання: використання гейміфікації створює захопливу навчальну атмосферу, що спонукає учнів активно долучатися до вивчення предмету.

2. Мотивація до досягнень: гейміфікація надає можливість створювати систему винагород та досягнень, що стимулює учнів до досягнення нових цілей у вивченні предмету.
3. Розвиток критичного та проблемного мислення: Через гейміфікацію можна створити ситуації, що сприяють розвитку критичного та проблемного мислення учнів.
4. Співпраця та командна робота: Гейміфікація підтримує співпрацю та командну роботу між учнями шляхом створення викликів та завдань, що потребують взаємодії.
5. Запам'ятовування та повторення матеріалу: Гейміфікація допомагає у повторенні та закріпленні вивченого матеріалу шляхом використання ігрових елементів, таких як квізи та міні-ігри, [4].

Загальна ідея полягає у тому, щоб навчальний процес був цікавим, мотивуючим та ефективним, а гейміфікація надає інструменти для досягнення цих цілей у вивченні предмету.

1.2 Аналіз програмних продуктів – аналогів

Розробка навчальної гри потребує індивідуального підходу та досвіду через її вузьку спеціалізацію та різноплановість інформації, яку гра чи тренажер візуалізує. Кожен крок гри відрізняється один від одного своїм змістом та даними, які вона проектує, тому кожен крок повинен бути продуманий та створений вручну.

Під час розробки навчальної гри важливо враховувати не лише освітній матеріал, але й особливості мети та цільової аудиторії. Кожна гра має відповідати конкретним навчальним цілям та сприйняттю учнів, щоб забезпечити максимальний ефект від навчання.

Створення кожного етапу гри вимагає аналізу та розробки відповідного набору завдань, викликів або питань, що відповідають конкретному матеріалу. Ці завдання повинні бути інтерактивними та цікавими для учнів, а також відповідати їхньому рівню знань та навичок.

Такий індивідуальний підхід та ретельне проектування кожного кроку гри є ключовими для створення ефективного навчального інструменту, який буде сприяти успішному засвоєнню матеріалу та мотивації учнів.

Прикладом доволі відомого тренажеру для вивчення англійської мови є EnglishDom (рис. 1.1), який містить велику кількість завдань різного типу: від аудіювання та щоденного закріплення лексичного набору слів до диктантів та логічного доповнення речень. Тренажер відомий своєю ефективністю, однак його недоліком є обмежений функціонал у разі безоплатного режиму доступу.

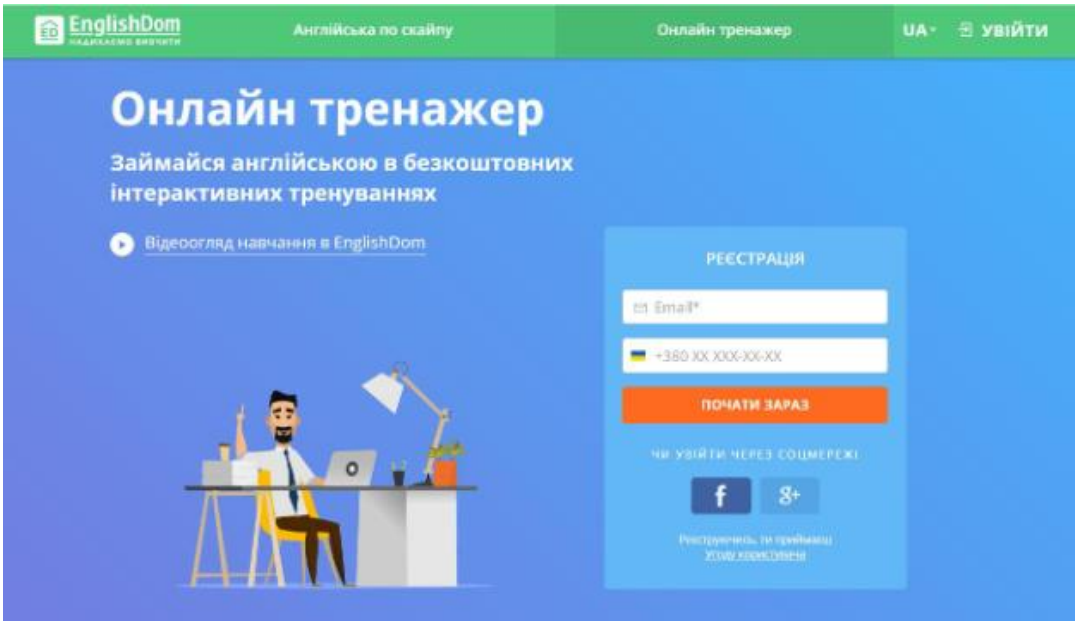


Рисунок 1.1 – Веб-тренажер для вивчення англійської EnglishDom

Ресурс Codecademy є інтерактивною онлайн-платформою, що присвячена вивченню різних мов програмування, таких як Python, PHP,

jQuery, JavaScript, Ruby, C++, Java, а також мов розмітки веб-сторінок HTML і CSS. Основною перевагою цього ресурсу є інтерактивність та певна міра гейміфікації, які роблять процес навчання більш захоплюючим і ефективним. На рис. 1.2 представлена початкова сторінка Codecademy, де користувачам пропонується пройти авторизацію або зареєструватися на сервісі.

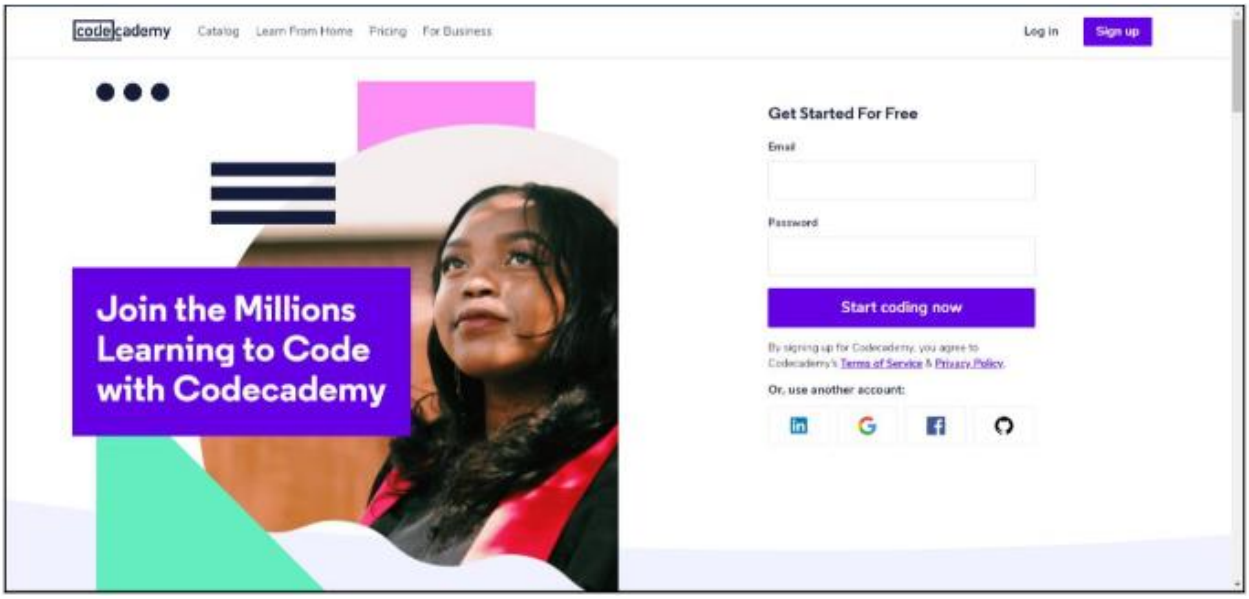


Рисунок 1.2 – Початкова сторінка Codecademy

Scratch – спеціальна віртуальна мова програмування, розроблена для навчання дітей програмуванню на основі мови Squeak. Вивчення Scratch відбувається в ігровій формі, що сприяє розвитку логічного та критичного мислення, уяви, а також навчає дітей вирішувати різнопланові задачі, (рис. 1.3).

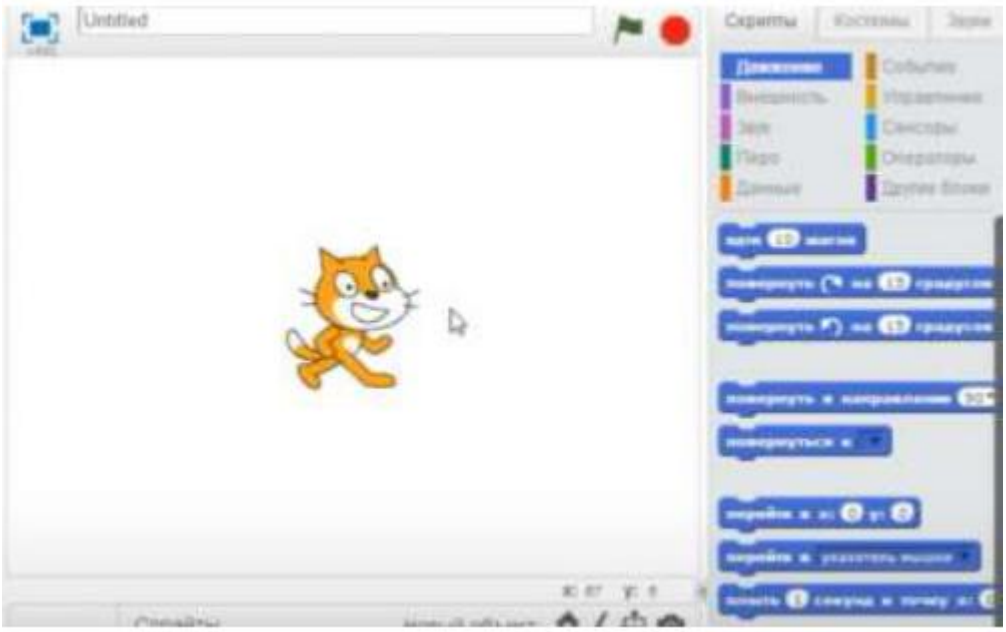


Рисунок 1.3 – Вікно навчальної гри Scratch

Один з головних мінусів Scratch полягає в тому, що він більше підходить для виконання завдань вдома, ніж у школах, оскільки орієнтований на творчу роботу, а не на стандартне навчання.

Mission Possible World Geography спрямована на поглиблення географічної обізнаності, включаючи знання про столиці, міста, річки, пустелі, озера, океани, гори та відомі місця. Використовуючи тему шпигунської місії для підвищення інтересу до вивчення географічних фактів, гра дозволяє дітям проникнути в дослідницький центр для пошуку важливих даних про глобальне потепління. Успіх місії безпосередньо залежить від того, наскільки добре діти можуть відповісти на низку питань з географії.

Запитання у грі мають різні формати, включаючи питання з кількома варіантами відповідей, заповнення пробілів, розшифровку слів і визначення місць на карті. Діти можуть потренуватися у вивченні фактів у навчальному режимі, перш ніж переходити до більш складного режиму місії. Приклад інтерфейсу гри наведено на рисунку 1.4.

Рисунок 1.4 – Інтерфейс гри Mission Possible World Geography

Таким чином, вивчивши дану методологію викладання та дослідивши ринок на наявність ігор та тренажерів для навчання, можна зробити висновок, що майбутня розробка є актуальною.

1.3 Вибір засобів реалізації

Оскільки метою кваліфікаційної роботи є розробка інструменту аналізу та обробки CSV-даних мовою Python, то природним вибором середовища розробки стала саме ця мова.

Python – популярна мова програмування, створена Гвідо ван Россумом у 1991 році, яка використовується в багатьох сферах:

- веб-розробка (на стороні сервера);
- розробка програмного забезпечення;
- математика;
- системні скрипти;

Особливості мови Python включають:

- створення веб-додатків на сервері;
- створення робочих процесів;
- підключення до систем баз даних;
- читання та змінення файлів;
- обробка великих даних та виконання складних математичних обчислень;
- швидка розробка програмного забезпечення, готового до виробництва;

Python працює на різних платформах, таких як Windows, Mac, Linux, Raspberry Pi тощо.

Python має простий синтаксис, схожий на англійську мову, що дозволяє розробникам писати програми з меншою кількістю рядків коду порівняно з іншими мовами програмування.

Python працює з системою інтерпретації, тобто код можна виконувати безпосередньо після написання. Це означає, що створення прототипу може бути дуже швидким.

Найновішою основною версією Python є Python 3, яка використовується для написання програм. Хоча Python 2 більше не оновлюється, за винятком оновлень безпеки, він все ще залишається досить популярним.

Python може бути записаний у текстовому редакторі. Можна писати Python в інтегрованому середовищі розробки (IDE), таких як Thonny, PyCharm, NetBeans або Eclipse, які особливо корисні при керуванні більшими колекціями файлів Python.

Синтаксис Python трохи схожий на інші мови програмування, але все ж має свої переваги:

- Python був розроблений для читабельності і має деяку схожість з англійською мовою, з впливом математики.
- Python використовує нові рядки для завершення команди, на відміну від інших мов програмування, які часто використовують крапки з комою або круглі дужки.
- для визначення області застосування Python покладається на відступ. Використовуючи пробіли, визначаються області циклів, функції та класи, тоді як інші мови програмування зазвичай використовують для цього фігурні дужки.

Python простий у вивченні завдяки своєму унікальному синтаксису, що полегшує читання коду. Розробники можуть легше читати та розуміти код Python порівняно з іншими мовами програмування. Це зменшує витрати на обслуговування та розробку програм, дозволяючи командам співпрацювати без значних мовних і досвідчених бар'єрів.

Крім того, Python підтримує використання модулів і пакетів, що означає, що програми можуть бути розроблені в модульному стилі, а код можна повторно використовувати в різних проектах. Розробивши потрібний модуль або пакет, його можна масштабувати для використання в інших проектах, легко імпортувати або експортувати ці модулі.

Однією з найбільш значущих переваг Python є те, що стандартна бібліотека та інтерпретатор доступні безкоштовно, як у двійковій, так і у

вихідній формі. Немає жодної ексклюзивності, оскільки Python та всі необхідні інструменти доступні на всіх основних платформах. Це робить Python привабливим варіантом для розробників, які не хочуть турбуватися про високі витрати на розробку.

Окремою перевагою Python є кількість різноманітних засобів візуалізації даних. Вже реалізований ряд бібліотек для різних цілей:

1. Matplotlib— базова бібліотека для побудови простих графіків;

2. Seaborn— бібліотека на базі matplotlib, інтегрована зі стеком PyData

(NumPy + SciPy + Pandas);

3. Plotly для побудови складних структур таких як тривимірні графіки.

Python має значні переваги завдяки розширеній екосистемі сторонніх бібліотек, таких як NumPy, Pandas, SciPy і багато інших. Ці бібліотеки дозволяють автоматизувати різноманітні завдання, такі як збір даних з Інтернету, обробка даних і наукові обчислення.

Наприклад, бібліотека NumPy надає широкі можливості для роботи з числовими даними і векторними обчисленнями, що робить Python потужним інструментом для наукових досліджень і обчислювальних завдань. Pandas, з іншого боку, забезпечує зручні засоби для обробки та аналізу даних у табличному форматі, що робить його ідеальним інструментом для роботи з великими обсягами даних.

А за допомогою бібліотеки Pandas можно працювати з табличними даними на мові програмування Python. З її допомогою можна легко і швидко читати, записувати і обробляти дані в різних форматах, включаючи CSV, Excel, SQL, JSON, HTML і т.д.

Завдяки цим бібліотекам та багатьом іншим, Python стає потужним і гнучким інструментом для автоматизації різноманітних завдань, що включають збір, обробку і аналіз даних.

1.4 Висновки

Ефективність навчання у навчальних комп'ютерних іграх визначається правильним поєднанням навчального матеріалу, ігрової механіки і дизайну гри. Гра повинна бути не лише цікавою і захоплюючою, але й ефективною з точки зору навчання.

Щоб досягти цієї мети, розробники повинні зосередитися на таких аспектах:

1. Навчальний матеріал: він повинен бути структурованим, легким для розуміння та відповідати освітнім стандартам або цілям навчання.
2. Ігрова механіка: гра повинна використовувати механіки, які сприяють навчанню, такі як виклики, нагороди, рівні складності, взаємодія з гравцем тощо.
3. Дизайн гри: він повинен бути привабливим та естетичним, а також підтримувати навчальний контекст.

Ефективні навчальні ігри здатні залучити учнів і студентів до навчання, забезпечуючи водночас ефективне засвоєння матеріалу та розвиток навичок. Такий підхід може стати значущим інструментом для підвищення ефективності навчального процесу.

Був розглянутий ринок навчальних систем, деякі приклад ігор і онлайн тренажерів, детально досліджена мета і механіка кожної з них.

Коротко описано про початкові інструменти які потрібні для розробки.

2 ГЕОПРОСТОРОВІ ІНСТРУМЕНТИ ТА ІНТЕГРАЦІЯ CSV-ДАНИХ

2.1 Веб-інструменти картографії та механізм стилізації шарів

Тематичні карти вважаються одними з найбільш популярних та важливих картографічних продуктів для візуалізації просторових даних. Вони є більш зрозумілими та чіткими, представляючи менш складну презентацію тематичних даних і дозволяючи користувачам пов'язувати інформацію з просторовим розташуванням. Дані часто збираються за територіальними одиницями, такими як країна, район та муніципалітет. Якісні та/або кількісні тематичні дані можуть бути відображені для показу просторового розподілу тем на просторових одиницях. Тематичні карти використовуються експертами, такими як географи, дослідники та містобудівники, як джерела інформації або для представлення результатів. З розвитком веб-технологій були розроблені різні стандарти для геопросторових веб-сервісів, що дозволяють розробляти тематичні веб-карти.

Веб-сервіси є програмними додатками зі стандартизованими програмними інтерфейсами, які дозволяють веб-додаткам взаємодіяти один з одним через Інтернет. Функціонал веб-сервісу представлений як послуга з методами та опублікованим інтерфейсом, доступним для клієнта. Клієнт – це програмне забезпечення, яке отримує доступ до веб-сервісу з сервера, формуючи клієнт-серверну модель. У багатьох випадках сервер працює на іншому комп'ютері, тому клієнт отримує доступ до веб-сервісу через комп'ютерну мережу.

Оскільки веб-сервіси пропонують інтерфейс для спілкування комп'ютерних програм, важливо, щоб вони були слабко пов'язані між собою. Це означає, що програми розгортаються незалежно одна від одної і, таким чином, програма виконується лише за потребою. Такий підхід відомий як сервіс-орієнтована архітектура (SOA). У SOA комп'ютерні програми представлені у вигляді сервісів, кожна програма відокремлена в окремий

самодостатній мережевий компонент, призначений для вирішення конкретної проблеми. SOA є особливо важливою для створення тематичних карт у кластерному комп'ютерному середовищі, оскільки часто комп'ютери, на яких розміщені геопросторові веб-сервіси, та комп'ютери, на яких зберігаються статистичні дані, розташовані віддалено.

Відкритий геопросторовий консорціум (Open Geospatial Consortium, OGC) [5] є головним авторитетом у розробці стандартів для веб-сервісів геоінформаційних технологій (GTI). Веб-сервіси OGC (OWS) надають інтерфейси для геопросторового контенту, обробки ГІС та пошуку даних. Повний опис стандартів OGC доступний на веб-сайті OGC. У наступних розділах наведено огляд стандартів, які використовувалися раніше, а також тих, що зараз використовуються для розробки та розповсюдження тематичних веб-карт

Стандарт Web Mapping Service (WMS) є одним з найперших стандартів, виданих OGC у 2002 році. Специфікація реалізації WMS надає HTTP-інтерфейс для запиту географічно прив'язаних картографічних зображень з однієї або декількох розподілених баз даних. WMS створюються картографічним сервером на основі даних, наданих базою даних ГІС; таким чином, тематичні карти можуть бути створені і збережені у вигляді картографічних зображень. Клієнти можуть запросити бажане картографічне зображення будь-якого розміру, що охоплює довільну географічну область. WMS часто використовують у поєднанні зі специфікацією OCG, яка називається Styled Layer Descriptor (SLD). Стандарт SLD використовується для опису зовнішнього вигляду веб-карткових сервісів. Можна опублікувати кілька індивідуальних стилів, які можна використовувати для стилізації тематичної карти, що відображається на WMS.

WMS має багато недоліків, незважаючи на те, що вона все ще використовується; вона базується на растровій моделі. Растрові моделі не дають користувачам доступу до географічних об'єктів, що лежать в їх основі;

отже, їх можна використовувати лише для відображення. Дослідження, проведене Cammack у 2007 році, дає широкий огляд використання та обмежень WMS у тематичному картографуванні. У цьому дослідженні одним з основних обмежень WMS є те, що вона тісно пов'язана з основними даними; таким чином, існує необхідність регенерувати всю WMS кожного разу, коли змінюється геометрія даних.

Ще однією проблемою WMS є те, що вона не кешується; таким чином, при кожному новому запиті від клієнта картографічні зображення повинні динамічно генеруватися "на льоту". Іноді дані мають такий великий обсяг, що кожне відображення за кожним запитом займає багато часу. Крім того, рендеринг WMS виконується сервером, що вимагає значних обчислювальних витрат, особливо коли на сервері є багато запитів на один і той самий ресурс. Як наслідок, загальна продуктивність та ефективність WMS як на стороні клієнта, так і на стороні сервера є дуже низькою.

Впровадження схем обробітку ґрунту для WMS покращило її здатність кешувати карти. Схеми розбиття дозволяють розбивати зображення карти на дискретні плитки зображень, що дало початок стандарту OGC Web Map Tile Service Standard (WMTS). Ця специфікація значно підвищила продуктивність, оскільки замість того, щоб комп'ютерні програми працювали з усім растровим зображенням, вони відображають лише ту частину плиток зображення, яку запитує клієнт. Tile Map Service (TMS) слугує основою WMTS, де растрове зображення розбивається на невеликі растрові плитки.

TMS - це специфікація з відкритим вихідним кодом для плиткових веб-карт, розроблена Open Source Geospatial Foundation (OSGeo). TMS використовує схему розбиття на плитки, де кожна плитка на одному рівні масштабування є батьківською для 4-х плиток на наступних рівнях масштабування (див. Рисунок 2.1). У випадку WMTS растрові тайли генеруються шляхом розбиття зображення карти на тайлову сітку. Положення кожної плитки базується на координатах сітки, що

використовується для її створення. Для того, щоб отримати конкретну плитку, запит повинен бути спрямований на координати плитки на сітці. Формат вилучення має вигляд z/x/y.format, де z - це рівень масштабування, x - позиція стовпчика, а y - позиція рядка. Наприклад, для запиту плитки на рівні 1 у стовпчику 1 і рядку 2 формат витягу буде 1/1/2.png.

Тим не менш, WMTS все ще страждає від тих же обмежень, що і WMS, оскільки обидві вони базуються на растровій моделі, і взаємодія або маніпуляції з географічними об'єктами неможливі [6].

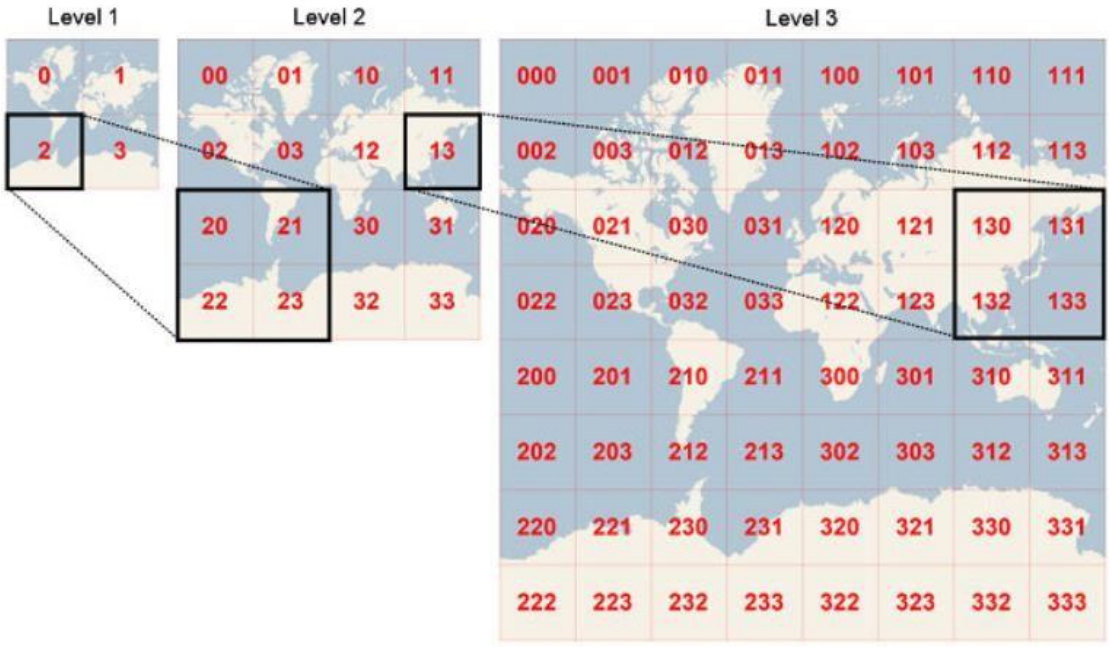


Рисунок 2.1 – Векторна схема тайлів на основі структури Quad tree, в якій кожен тайл є батьком 4-х тайлів на наступних рівнях масштабування в піраміді тайлів [6]

Мобільна карта - це найновіший спосіб відображення геопросторової інформації. До появи мобільних карт, основним недоліком традиційних веб-карт був брак мобільності порівняно з паперовими картами. Розвиток мобільних пристроїв робить екрани більшими та вдосконалює технології

позиціонування, які є базовою основою для реалізації мобільної карти. Найбільш помітною особливістю мобільної карти є можливість відображення поточної позиції користувача, що є характерною для сервісів, заснованих на визначенні місцезнаходження (LBS).

Парадигми взаємодії та питання юзабіліті в представленні геопросторових даних є предметом активних обговорень. Liqiu запропонував деякі дослідницькі питання, пов'язані з генерацією мобільних карт, такі як "Які типи завдань включають мобільні карти?", "Які методи генералізації є корисними для мобільних карт?" і "Як спрямувати увагу користувача на ключовий зміст мобільної карти?".

Згідно з дослідженнями мобільних карт, сучасні мобільні картографічні додатки можна класифікувати як підтримку мобільності, збір інформації та інформаційну комунікацію. Крім того, сучасні мобільні карти дозволяють два типи адаптації. Один з цих типів адаптації можна описати як "за допомогою сенсорів". Мобільний пристрій отримує поточне місцезнаходження та напрямок руху за допомогою GPS-приймача. Відображення карти автоматично оновлюється, що робить її унікальною для кожного окремого користувача. Інша адаптація залежить від введених користувачем даних - представлення мобільної карти базується на даних, введених користувачами .

З точки зору користувача, взаємодія між користувачем і мобільною картою є найбільш важливою частиною. Нижче наведено основні операції:

– панорамування: користувач може переміщати карту в будь-якому напрямку;

– масштабування: користувач може змінювати області відображення;

– перемикання: користувач може вибрати певний стиль представлення;

– запит: користувач може шукати будь-які об'єкти, вводячи ключові слова.

І останнє, але не менш важливе: перед тим, як випустити мобільний додаток карти на ринок, необхідно провести юзабіліті-тест. Це забезпечить високу якість створення карти. Вчені порівняли кілька методів перевірки зручності використання мобільної карти в природному середовищі. Крім того, зручність використання мобільної карти можна виміряти за показниками ефективності, задоволеності користувачів та результативності.

2.2 Інструменти для геоприв'язки та доступу до геоданих

Дослідження Hong та Lin було першим, де була розроблена система, що використовує веб-сервіси OGC для об'єднання атрибутивних даних з геопросторовими даними з метою створення тематичних веб-карт. Архітектура системи для їхнього дослідження базувалася на припущенні, що геопросторові дані та статистичні дані розміщуються на різних комп'ютерах у розподіленій комп'ютерній мережі. Для з'єднання цих двох комп'ютерів потрібен веб-сервіс, який об'єднує дані для створення тематичної веб-карти. Служба географічних зв'язків (Geolinking Service, GLS) визначає інтерфейс для сервісів, які дозволяють об'єднувати набори даних, що містять тематичні дані про географічні об'єкти, з географічними даними в іншому сховищі [7].

У цьому дослідженні GLS був розроблений для імітації об'єднання таблиць у реляційній базі даних. Припущення полягало в тому, що обидва набори даних мають географічні ідентифікатори для зв'язування. Результатом об'єднання стала тематична веб-карта, яка розповсюджується як WMS. Незважаючи на те, що основним продуктом був WMS, користувачі мали можливість надавати власні статистичні дані у форматі CSV для інтеграції з географічними даними. У 2010 році GLS було модернізовано до служби

об'єднання таблиць (TJS). TJS об'єднує GLS і GDAS, а не розглядає їх як окремі стандарти (рис.2.2).

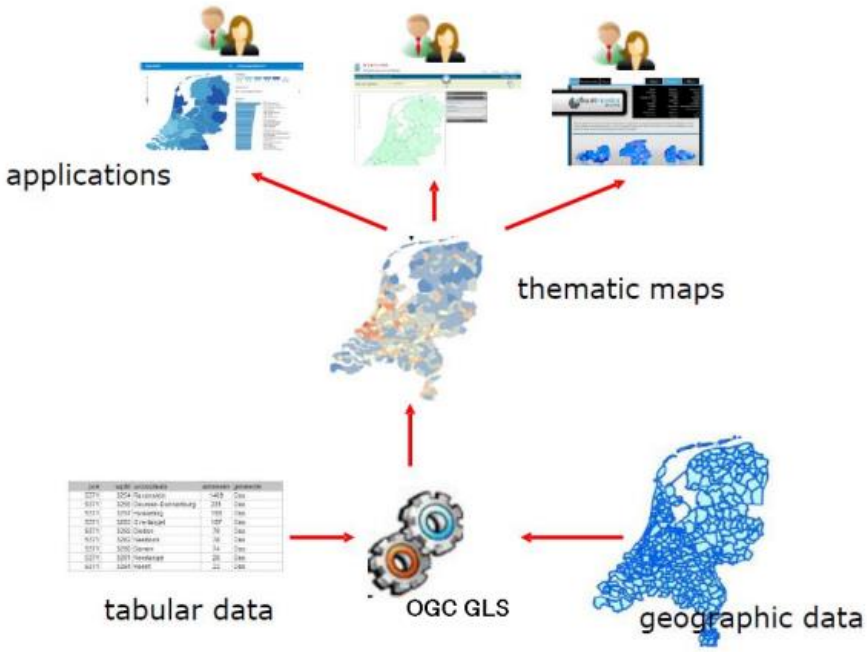


Рисунок 2.2 – Концепція GLS [7]

2.3 Інструменти для роботи з геотаблицями

TJS (Table Joining Service) існує вже трохи більше десяти років, проте досліджень, які б демонстрували його потенціал для об'єднання геопросторових даних та статистичних даних у розподіленому середовищі, було проведено недостатньо. TJS описує спосіб визначення та обміну даними, що містять інформацію про географічні об'єкти.

TJS приймає атрибутивні дані, які відносяться до географічних об'єктів, також відомі як геоприв'язані дані, і приєднує їх до геопросторової структури для можливості нанесення на тематичну веб-карту або використання в подальшій геообробці.

Атрибутивні дані в контексті TJS відносяться до даних про певний географічний простір, але ці дані безпосередньо не пов'язані з географічними координатами або геометрією. Атрибути кодуються у форматі, визначеному

OGC для TJS під назвою Geolinked Data Assess service (GDAS), який є форматом XML. Дані атрибутів містять поле з географічними ідентифікаторами для позначення географічного об'єкта, до якого вони відносяться.

Географічні ідентифікатори можуть бути адміністративними ідентифікаторами, такими як назва району, області, або постійними ідентифікаторами, такими як поштовий індекс та коди областей країни. Географічні ідентифікатори також використовуються як ключі для рамкових даних. За допомогою цих спільних географічних ідентифікаторів у двох наборах даних можна встановити зв'язок. Результатом операції зв'язування є нова геопросторова структура, заповнена атрибутивними даними.

Фреймворкові дані описують об'єкти, розташовані на поверхні Землі, такі як країни, екологічні регіони, річки або сітки перепису населення. Прикладами атрибутивних даних, які можуть бути пов'язані з географічними об'єктами, є дані про населення за країнами (рис.2.3). Фреймворкові дані можуть зберігатися локально в TJS або надаватися через веб-сервіс, такий як Web Feature Service (WFS), який надає геопросторові дані у векторному форматі. Відомі реалізації TJS використовують WFS, стандарт інтерфейсу OGC, що дозволяє запитувати необроблені географічні об'єкти через Інтернет. Однак, WFS не надає клієнтам адміністративних прав на дані, тому користувачі без адміністративного доступу можуть віртуально отримувати та змінювати об'єкти, але не можуть впливати на оригінальне сховище даних. Ця особливість забезпечує можливість одноразового публікування даних з їх подальшим багаторазовим використанням [8].

Також важливо відзначити, що TJS підтримує інтеграцію атрибутивних даних у форматі CSV (Comma-Separated Values). CSV є популярним форматом для зберігання табличних даних, де кожен рядок таблиці представляє собою один запис, а кожний стовпець відповідає конкретному

полю даних. Цей формат легко зрозуміти і обробляти як для людей, так і для машин.

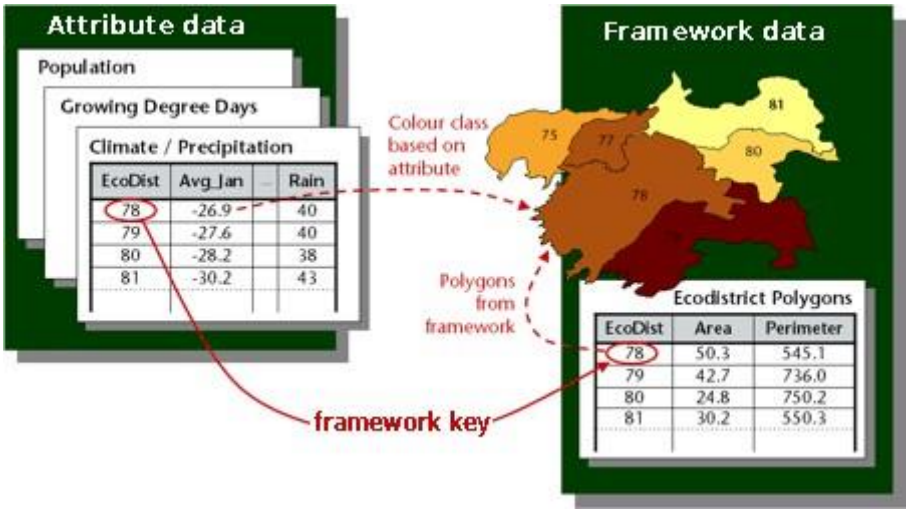


Рисунок 2.3 – Атрибутивні дані в геопросторовій структурі

Для інтеграції CSV даних в TJS користувач може завантажити свої статистичні дані у форматі CSV та вказати географічні ідентифікатори для зв'язування з географічними даними. Після цього TJS може автоматично зв'язати ці два набори даних, створюючи нову геопросторову структуру, яка містить атрибутивні дані з CSV таблиці та географічні дані. Ця можливість робить TJS гнучким і масштабованим рішенням для обробки різноманітних даних у геопросторовому контексті.

2.4 Створення карт на основі CSV-даних

CSV (Comma-Separated Values) - це формат для зберігання даних у вигляді таблиці, де кожен рядок представляє запис, а кожна кома (або інший розділювач) в рядку відокремлює колонки. Для створення географічних карт на основі CSV-даних, спершу необхідно визначити, які саме геодані ви хочете використовувати. Наприклад, це можуть бути географічні координати (широта та довгота) або адміністративні одиниці (назви країн, областей тощо).

Для візуалізації географічних даних на картах використовують різні інструменти та бібліотеки. У вашому випадку, ви можете використовувати Python з бібліотеками, такими як pandas для роботи з даними та matplotlib, geopandas, або folium для візуалізації.

Pandas - це потужна бібліотека для обробки та аналізу даних, включаючи зчитування CSV-файлів.

– Matplotlib - це основна бібліотека для візуалізації даних у Python. Вона може бути використана для створення базових графіків та карт;

– Geopandas - це розширення для pandas, яке дозволяє працювати з геоданими, такими як географічні формати даних та формати векторних даних;

– Folium - це бібліотека для створення інтерактивних карт, яка базується на Leaflet.js. Вона ідеально підходить для створення веб-карт з геоданими.

Після створення географічних карт на основі CSV-даних важливо правильно аналізувати та інтерпретувати отримані результати. Ось декілька кроків для цього:

– розгляд основних статистичних показників, таких як середнє, медіана, мода, дисперсія, тощо, для кращого розуміння розподілу населення за регіонами;

– аналіз просторових закономірностей та взаємозв'язків між географічними об'єктами та показниками населення. Наприклад, вивчення розподілу населення в різних адміністративних одиницях;

– оцінка кольорових схем, маркерів та інших візуальних елементів на картах для кращого розуміння даних та їхнього впливу на результати;

– засновані на аналізі результати можуть бути використані для розробки рекомендацій та стратегій для подальшого дослідження або планування ресурсів.

2.5 Використання геоданих

В останні роки спостерігається зростання популярності колективного інтелекту у зборі та поширенні різноманітних даних. Інтернет стає платформою для "користувацького контенту", що включає в себе зображення, відео та інформацію з соціальних мереж, таких як YouTube, Flickr, Facebook та Twitter. Цей контент генерується користувачами особистої ініціативи, незалежно від професійних обов'язків. Паралельно із цим розвиток географічних інформаційних систем (ГІС) сповільнюється, а збір даних вважається його слабким місцем. Відповідь на ці виклики надає поняття добровільної географічної інформації (ДГІ), яке досліджують Елвуд, Гудчайлд та Суї. Це поняття представляє новий підхід до створення та обміну географічною інформацією, розширюючи розуміння того, хто створює і ділиться географічними даними [9].

Сучасні дослідники журналістики шукають міждисциплінарні рішення для подолання обмежень традиційного підходу. Географічний аспект новин стає ключовим для розуміння та аналізу тематики та її впливу. Хоча новинні матеріали містять імпліцитні географічні дані, їх відсутність у більшості статей обмежує їх потенціал. Проте до 2008 року стандартів геотегування новин не існувало. Із розвитком геолокаційних технологій з'явилося багато інновацій, включаючи платформу "Bloom", яка допомагає видавцям визначити розташування новин. Ця платформа, разом із додатком "Blockfeed - новини Нью-Йорка", надає персоналізовані новини на основі місцезнаходження користувача [10].

2.6 Висновки

У даному розділі ми детально розглянули різноманітні інструменти та підходи, що використовуються для обробки, аналізу та візуалізації геопросторових даних на основі CSV-формату. Починаючи з веб-інструментів картографії та механізмів стилізації шарів, ми перейшли до

інструментів для геоприв'язки та доступу до геоданих, а також зупинились на інструментах для роботи з геотаблицями.

Особливий акцент було зроблено на створенні карт на основі CSV-даних, де ми досліджували різні підходи та методи візуалізації населення. Також ми розглянули можливості використання геоданих для забезпечення більш точного та зрозумілого представлення географічних даних.

Загальний аналіз показав, що комбінація різних інструментів та підходів може значно покращити якість та ефективність обробки геопросторових даних. Це відкриває широкі перспективи для подальших досліджень та розробки нових методів та інструментів у сфері геоінформатики та аналітики.

3 РОЗРОБКА ІНСТРУМЕНТУ АНАЛІЗУ ТА ОБРОБКИ CSV-ДАНИХ МОВОЮ PYTHON

3.1 Технологічний стек інструменту аналізу та обробки csv-даних

Для створення інструменту було використано декілька технологій. Інструмент аналізу та обробки csv-даних було створено на python разом з кількома допоміжними модулями, які описано нижче.

Python - це високорівнева інтерпретована мова програмування. Ми обрали її для реалізації проекту завдяки нашому досвіду роботи з нею, простоті вивчення та великому асортименту сторонніх пакетів. Використовувалася версія 3.10.

Pip - це інструмент для легкої установки пакетів Python. Всі необхідні пакети встановлювалися за допомогою pip.

Pandas - це пакет для ефективної роботи з даними. Він був ключовим для обробки даних про взаємодію з користувачем [11].

NumPy - це бібліотека для математичних операцій з масивами. Використовувалася для реалізації математичних формул [12].

PyInstaller модуль для пакування Python-скриптів в виконувані файли.

Tkinter бібліотека для створення графічного інтерфейсу в Python.

Os модуль для взаємодії з операційною системою.

PIL бібліотека для обробки зображень, включаючи Image та ImageTk.

Folium бібліотека для візуалізації географічних даних на основі OpenStreetMap.

CSV формат для збереження табличних даних.

Turtle модуль для графічного відображення.

Webbrowser модуль для відкриття веб-браузера з Python.

SPYDER IDE для Python, використовувалася для налагодження та компіляції коду [13].

3.2 Моделювання бізнес-процесу функціонування інструменту інтерактивної карти населення

Контекстна діаграма бізнес-процесу (рис.3.1) демонструє взаємодію програми з зовнішніми сутностями та основні функції, які вона виконує.

Рисунок 3.1 – Контекстна діаграма інструменту інтерактивної карти населення

Сутності:

– користувач;

– зовнішні скрипти (map.py, country_g.py, city_g.py, popul_g.py, world.py, analize.py, main.py);

– веб-браузер;

– csv-файл;

– файли та зображення.

Процеси:

– анімація об'єктів;

– пошук країни за назвою;

– пошук міста за назвою;

– пошук населення за кількістю;

– аналіз даних (карта або графік);

– гра "в штати".

Інтерфейсні дуги:

– користувач взаємодіє з програмою, надаючи дані або вибираючи опції;

– програма виконує зовнішні скрипти для аналізу та відображення даних;

– програма відображає результати на вікні інтерфейсу користувача.

Опис процесів: програма анімує рух прапора та тексту на фоновому зображенні. Користувач вводить назву країни. Програма виконує скрипт country_g.py, який аналізує та відображає інформацію про країну. Користувач вводить назву міста. Програма виконує скрипт city_g.py, який аналізує та відображає інформацію про місто. Користувач вводить кількісні значення населення. Програма виконує скрипт popul_g.py, який аналізує та відображає інформацію про населення. Користувач вибирає тип аналізу (карта або графік). Програма виконує відповідний скрипт (World.py або Analize.py) для відображення аналітичних даних. Користувач може грати в гру "В штати", використовуючи скрипт main.py.

UML-діаграма Use Case Diagram (рис.3.2) є графічним інструментом, який дозволяє відобразити функціональність системи та взаємодію між користувачами та системою. Для програми "Інтерактивна карта населення" можна створити таку діаграму, яка допоможе зрозуміти, як користувачі взаємодіють з програмою та які основні функції вона надає.

Основні компоненти:

– актори – користувач. Основний користувач програми, який може виконувати різні дії з програмою, такі як пошук інформації про країни, міста, населення, а також аналіз даних та гра "В штати".

Use Cases (Сценарії використання):

– пошук країни – користувач може ввести назву країни для отримання інформації про неї;

– пошук міста – користувач може ввести назву міста для отримання інформації про нього;

– пошук населення – користувач може ввести кількісні дані населення для отримання інформації про регіони з вказаним населенням;

– аналіз даних – користувач може вибрати тип аналізу (карта або графік) для отримання візуального представлення аналітичних даних;

– гра "В штати" – користувач може запустити гру, де можна взаємодіяти з географічними елементами на карті.

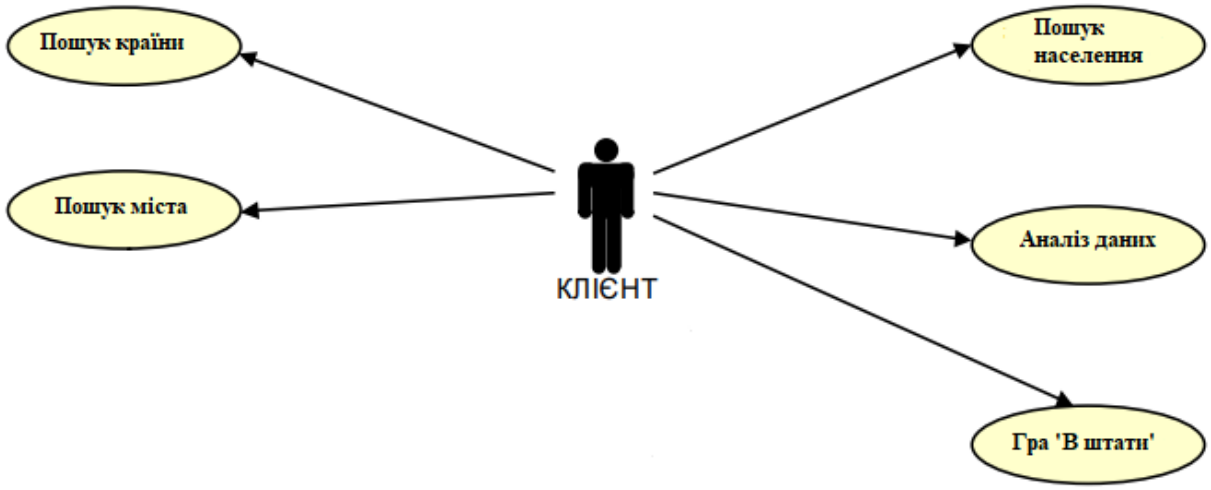


Рисунок 3.2 – UML-діаграма використання інструменту інтерактивної карти населення

На діаграмі рис.3.2 представлено актори та сценарії використання, які вони можуть виконувати. Актори представлені як людина, а сценарії використання — як еліпси. Взаємодія між ними буде показана за допомогою стрілок, які вказують на те, як користувач може взаємодіяти з програмою.

3.3 Опис структури проекту

Інструмент розроблено у зручному робочому середовищі Spyder. На рис. 3.3 показана архітектурна схема проекту.

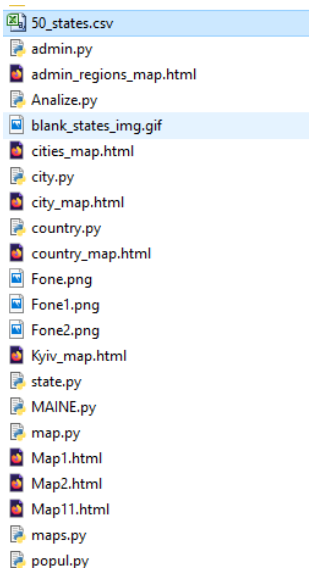


Рисунок 3.3 – Структура проекту

Розглянемо детально кожні файли:

– main.py. Головний файл, що ініціалізує графічний інтерфейс користувача. Відповідає за взаємодію з користувачем та запуск інших модулів на основі введених даних;

– admin.py. Цей файл, швидше за все, відповідає за адміністративні функції, такі як управління користувачами, налаштуваннями системи тощо. Він може включати функції для додавання, видалення та редагування даних.

– analize.py. Файл для аналізу даних. Можливо, цей модуль використовується для статистичного аналізу даних, графічного представлення даних або взаємодії з іншими модулями для аналітичних цілей;

– city.py. Модуль для роботи з даними міст. Включає функції для пошуку, відображення та обробки даних про конкретні міста;

– country.py. Модуль для роботи з даними країн. Відповідає за пошук, відображення та обробку даних про конкретні країни;

– map.py. Модуль для створення інтерактивних карт на основі геоданих. Використовує дані про міста та країни для створення візуальної репрезентації населення;

– popul.py. Модуль, який може відповідати за обробку та відображення загальних даних про населення, які можуть бути агреговані або оброблені з різних джерел.

Проект виглядає як класичний desktop-додаток з графічним інтерфейсом. Головний файл main.py відповідає за ініціалізацію графічного інтерфейсу та взаємодію з користувачем. Він використовує функціональні можливості інших модулів для виконання конкретних завдань, таких як відображення карт, пошук міст або країн, аналіз даних тощо.

Кожен модуль відповідає за конкретний аспект функціональності:

– city.py та country.py зосереджені на роботі з даними міст та країн відповідно;

– map.py створює інтерактивні карти на основі геоданих;

– analize.py може забезпечувати аналітичні функції для обробки та візуалізації даних;

– admin.py може включати управлінські функції для адміністраторів системи;

– popul.py може представляти функціональність, що стосується обробки загальних даних про населення.

Кожен модуль може взаємодіяти з базою даних, файлами або іншими ресурсами, щоб отримати, обробити та представити інформацію користувачу.

3.4 Вибір фреймворку графічного інтерфейсу для Python

Для розробки графічного інтерфейсу користувача на Python існує безліч варіантів. Нижче представлено невелику добірку поширених фреймворків.

PyQt5 - це набір прив'язок Python до п'ятої версії фреймворку Qt і складається з набору модулів Python [14]. Qt складається з набору C++ бібліотек та засобів розробки для створення як графічних інтерфейсів користувача, так і інтерфейсів прикладного програмування (API). Qt також

пропонує програмний інструмент під назвою Qt Designer, який дозволяє використовувати функції перетягування, що значно спрощує створення графічного інтерфейсу без особливих попередніх знань з програмування. Однак цей інструмент не було досліджено у цьому проекті.

TkInter вважається стандартним пакетом графічного інтерфейсу для Python і є тонким об'єктно-орієнтованим шаром поверх Tcl/Tk [5]. Основними перевагами TkInter є те, що він швидкий і входить до складу стандартної бібліотеки Python.

PySide2 є пізнішою прив'язкою до інструментарію Qt і розробляється компанією Qt Company [17]. wxPython - це крос-платформний інструментарій графічного інтерфейсу, який використовує бібліотеку C++ wxWidgets, але має модульні розширення для Python.

Графічний інтерфейс для цього проекту був розроблений за допомогою Tkinter, стандартної бібліотеки Python для створення графічних інтерфейсів користувача (GUI). Ця бібліотека дозволяє легко створювати вікна, кнопки, поля введення та інші інтерактивні елементи.

Функціонал графічного інтерфейсу:

– пошук країни: користувач може ввести назву країни у полі введення і побачити на карті всі міста цієї країни;

– пошук міста: користувач може ввести назву міста у полі введення і побачити на карті це місто з відображенням інформації про населення;

– пошук по кількості населення: користувач може ввести значення населення у полі введення і побачити на карті всі міста, населення яких перевищує введене значення;

– відображення карти: інтерактивна карта з різнокольоровими маркерами, які представляють міста з різними рівнями населення;

– анімація: анімований прапор україни, який рухається по канві;

– вибір виду аналізу: можливість вибору між представленням даних у вигляді карти або графіка.

На рис.3.4 наведено макет графічного інтерфейсу.

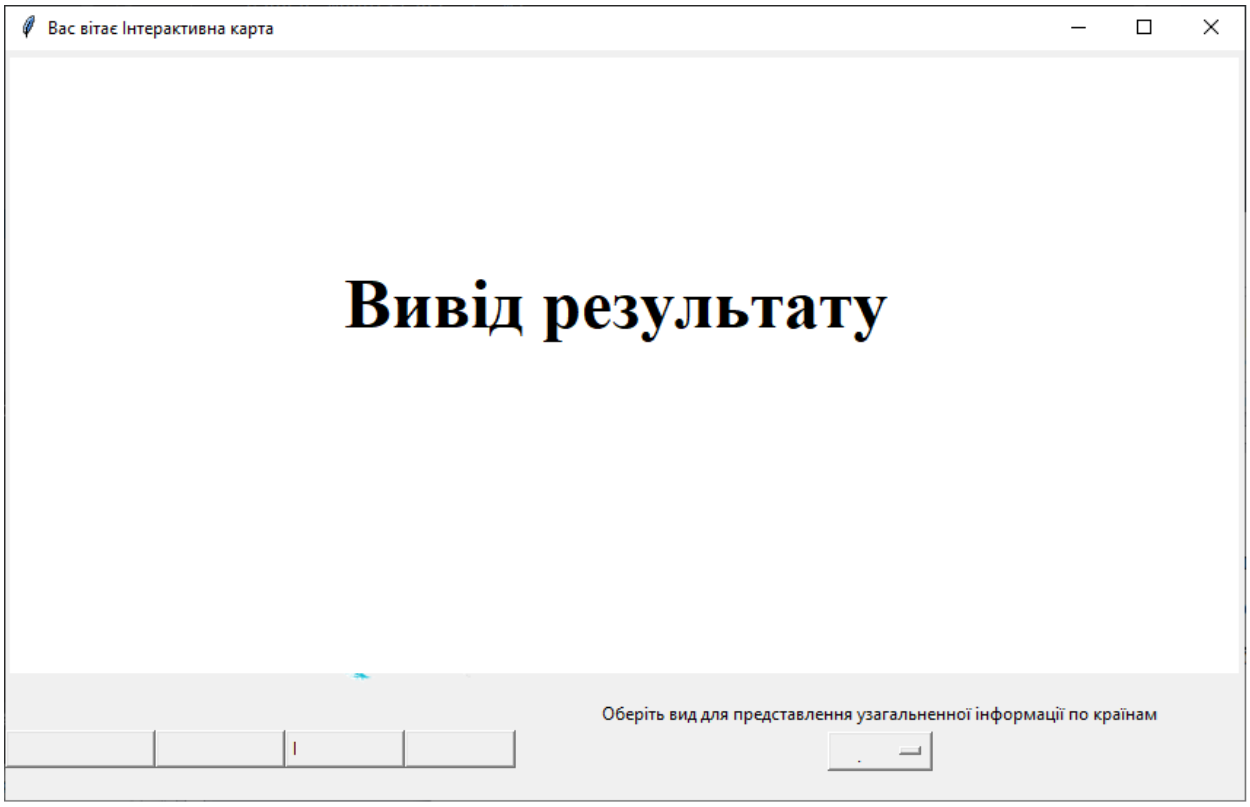


Рисунок 3.4 – Макет графічного інтерфейсу

Використані модулі та бібліотеки:

– Tkinter для створення графічного інтерфейсу;

– Pandas для обробки та аналізу даних з CSV-файлів;

– Folium для створення інтерактивних карт;

– Webbrowser для автоматичного відкриття створених HTML-файлів у веб-браузері;

– OS для роботи з операційною системою, такою як створення абсолютних шляхів до файлів;

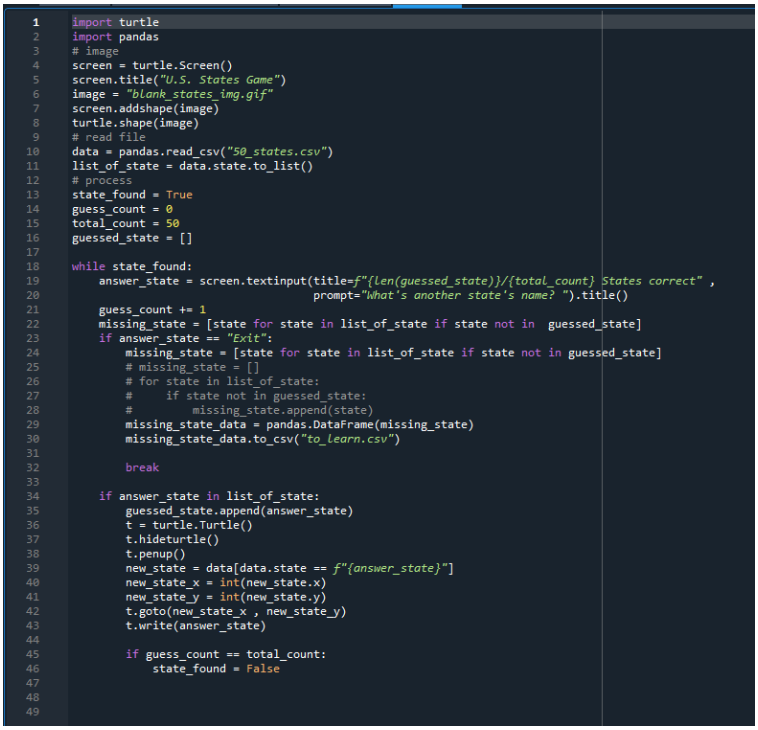
– PIL (Pillow) для роботи з зображеннями, такими як фонові зображення для графічного інтерфейсу.

3.5 Головне меню інструменту аналізу та обробки csv-даних

У роботі було розроблено головне меню графічного інтерфейсу для інструменту аналізу та обробки csv-даних "Інтерактивна карта". Додаток

розроблявся за допомогою мови програмування Python та фреймворку tkinter, який є стандартним пакетом графічного інтерфейсу для цієї мови. Це меню надає користувачеві зручний інтерфейс для вибору різноманітних функцій додатку (рис.3.5).

Меню включає в себе ряд функціональних можливостей, що охоплюють різні аспекти взаємодії користувача з додатком. Основні функції включають анімацію об'єктів, відображення інформації про населення країн, пошук інформації про країни та міста, а також можливість аналізу даних через візуалізацію на карті або графіку.



```
import turtle
import pandas
# image
screen = turtle.Screen()
screen.title("U.S. States Game")
image = "blank_states_img.gif"
screen.addshape(image)
turtle.shape(image)
# read file
data = pandas.read_csv("50_states.csv")
list_of_state = data.state.to_list()
# process
state_found = True
guess_count = 0
total_count = 50
guessed_state = []

while state_found:
    answer_state = screen.textinput(title=f"{len(guessed_state)}/{total_count} States correct" ,
                                    prompt="What's another state's name? ").title()
    guess_count += 1
    missing_state = [state for state in list_of_state if state not in  guessed_state]
    if answer_state == "Exit":
        missing_state = [state for state in list_of_state if state not in guessed_state]
        # missing_state = []
        # for state in list_of_state:
        #     if state not in guessed_state:
        #         missing_state.append(state)
        missing_state_data = pandas.DataFrame(missing_state)
        missing_state_data.to_csv("to_learn.csv")

        break

    if answer_state in list_of_state:
        guessed_state.append(answer_state)
        t = turtle.Turtle()
        t.hideturtle()
        t.penup()
        new_state = data[data.state == f"{answer_state}"]
        new_state_x = int(new_state.x)
        new_state_y = int(new_state.y)
        t.goto(new_state_x , new_state_y)
        t.write(answer_state)

        if guess_count == total_count:
            state_found = False
```

Рисунок 3.5 – Лістінг головного меню

Функція animate() відповідає за анімацію об'єктів на канві, таких як прапор України, який рухається по екрану. Ця анімація створює естетичний ефект і привертає увагу користувача.

Функція show_country_population() відкриває вікно з картою, яка відображає населення різних країн. Це дозволяє користувачеві отримати уявлення про густоту населення у різних регіонах світу.

Функції search_country() і search_city() дозволяють користувачеві ввести назву країни або міста та отримати інформацію про нього. Додаток відображає географічну інформацію та статистику населення для введених країн та міст.

Функція analyze_data(selection) надає можливість користувачу вибрати вид представлення даних: карта або графік. Це дозволяє зробити порівняльний аналіз даних у зручному форматі.

Функція game_map() інтегрує в додаток гру, яка допомагає користувачеві вивчити географічне розташування штатів.

Для реалізації графічного інтерфейсу було використано бібліотеку tkinter, яка є стандартом для створення графічних інтерфейсів на Python. Для роботи з зображеннями та анімації були використані бібліотеки PIL (Pillow) та os.

3.6 Розробка функціоналу для інструменту аналізу та обробки csv-даних

Функціонал "Карта", який відображає географічні дані різних міст світу на інтерактивній карті. Цей модуль розроблений з використанням бібліотеки folium для Python, що дозволяє легко створювати картографічні візуалізації.

Дані про міста та їх населення було зчитано з файлу "worldcities.csv" за допомогою бібліотеки pandas (рис.3.6). Після цього були видалені рядки з пропущеними значеннями для підготовки даних до візуалізації.

```
import pandas as pd

data = pd.read_csv("worldcities.csv")
```

Рисунок 3.6 – Робота з csv-файлами

Код га рис.3.6 використовує функцію read_csv з бібліотеки pandas для зчитування даних з CSV-файлу "worldcities.csv" із поточного каталогу. Цей файл містить інформацію про міста світу, включаючи їх географічні координати та населення.

Після зчитування даних вони були перевірені на наявність пропущених значень. Це є важливим кроком, оскільки пропущені дані можуть спричинити помилки під час візуалізації або аналізу даних.

```
data.dropna(axis=0, inplace=True)
```

Рисунок 3.7 – Візуалізація даних

На рис.3.7 dropna використовується для видалення рядків, де пропущені значення з'являються. Параметр axis=0 вказує, що видалення повинно проводитися вздовж рядків (а не стовпців), а inplace=True вказує, щоб зміни відбувалися безпосередньо в об'єкті data, не створюючи нового об'єкта.

Настпним етапом є створення карти. Використовуючи folium, було створено базову карту з початковими координатами та масштабом. В якості підкладки було використано OpenStreetMap.

Folium є потужною бібліотекою Python для візуалізації геоданих на інтерактивних картах. Вона використовується для створення інтерактивних карт, які можна переглядати у веб-браузері. Folium підтримує різноманітні типи карт, включаючи OpenStreetMap [14], Mapbox, Stamen Terrain, і багато інших.

Для створення базової карти з початковими координатами та масштабом, було використано функцію Map з бібліотеки folium. Ця функція дозволяє вказати початкові географічні координати та масштаб карти.

```
map = folium.Map(location=[38.58, -99.09], zoom_start=2, tiles="OpenStreetMap")
```

Рисунок 3.8 – Реаліщація функції Map

На рис.3.8 location=[38.58, -99.09] вказує початкові географічні координати для центру карти; zoom_start=2 визначає масштаб карти. Чим більше значення, тим ближче масштаб до земної поверхні.

В якості підкладки для створеної карти було обрано OpenStreetMap, яка є відкритою картографічною платформою, що базується на вільній ліцензії. OpenStreetMap надає високоякісні геодані, які можна використовувати безкоштовно для створення інтерактивних карт. У функції Map параметр tiles="OpenStreetMap" вказує, що для створення карти буде використана підкладка OpenStreetMap.

Третім етапом є Додавання маркерів. Для кожного міста, населення якого перевищує 2 мільйони осіб, був доданий маркер на карту. Кожен маркер має відповідний розмір та колір в залежності від кількості населення. Маркери дозволяють нам візуально позначити конкретні точки на карті, надаючи користувачеві зрозумілу інтерактивну інформацію. У контексті візуалізації геоданих, маркери можуть бути використані для позначення міст, місць або будь-яких інших географічних об'єктів.

Для додавання маркерів на карту за умовою, що населення міста перевищує 2 мільйони осіб, ми використовували функціонал бібліотеки folium. Кожен маркер створюється за допомогою CircleMarker, який дозволяє задати розмір, колір та інші параметри маркера.

```
for lt, ln, pop, ci, coun in zip(lat, lon, population, city, country):
    if pop >= 2000000.0:
        popname = ci + ", " + coun + "\n" + "Population: " + str(round(pop / 1000000.
        iframe = folium.IFrame(html=html % popname, width=180, height=115)
        fg.add_child(folium.CircleMarker(location=[lt, ln], radius=10, popup=folium.P
```

Рисунок 3.9 – Фрагмент лістингу CircleMarker

Цикл проходиться по кожному місту в нашому датасеті. Умова if pop >= 2000000.0: перевіряє, чи перевищує населення міста 2 мільйони осіб. CircleMarker створює маркер для кожного відповідного міста; radius=10

встановлює розмір маркера; fill_color=color_producer(pop) задає колір маркера в залежності від кількості населення за допомогою функції color_producer.

Маркери забарвлені в різні кольори в залежності від кількості населення міста. Функція color_producer приймає населення міста як вхідний параметр і повертає відповідний колір для маркера. Наприклад, міста з населенням від 2 до 4 мільйонів осіб позначаються оранжевим кольором, міста з населенням від 4 до 6 мільйонів осіб - червоним, а всі інші - темно-червоним.

```
def color_producer(population):
    if 2000000.0 <= population < 4000000.0:
        return 'orange'
    elif 4000000.0 <= population < 6000000.0:
        return 'red'
    else:
        return 'darkred'
```

Рисунок 3.10 – Функція color_producer

Завершальним етапом є інформаційні вікна. При натисканні на маркер користувачу відображається інформаційне вікно з назвою міста, країни та кількістю населення. Дана інформація форматується за допомогою HTML-шаблону для кращої читабельності.

Інформаційні вікна в картографії слугують ключовою ролью в наданні додаткової інформації про конкретні місця, позначені на мапі. Коли користувач клікає на маркер на карті, відображення цього місця розширюється детальною інформацією.

Для покращення читабельності і зручності відображення інформації в інформаційних вікнах, дані форматуються за допомогою HTML-шаблону. Це дозволяє нам структурувати інформацію у зрозумілий та привабливий спосіб.

```
html = """<h4>City information:</h4>
%s million
"""
```

Рисунок 3.11 – Фрагмент HTML-шаблон

На рис.3.11 *%s* використовується для підстановки значення місця, країни та кількості населення. Коли користувач клікає на маркер, ці дані автоматично підставляються в шаблон HTML, щоб надати детальний огляд інформації про вибране місце.

Після створення та налаштування карти, вона зберігається у файл з назвою "Map1.html". Цей файл містить всю необхідну інформацію для відображення інтерактивної карти. Після збереження, програма автоматично відкриває цей файл у веб-браузері для користувача.

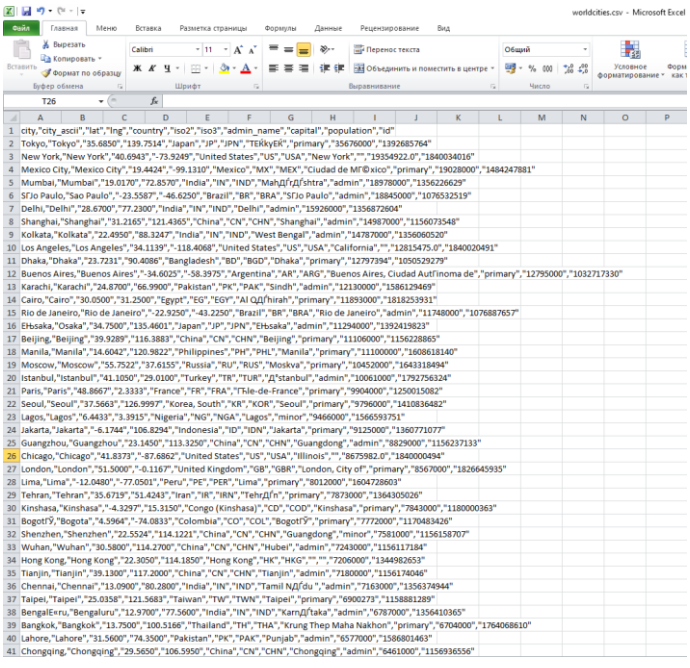


Рисунок 3.12 – Структура csv-файлу

CSV-файл "worldcities.csv" містить інформацію про різні міста світу та їхнє населення. Файл має наступну структуру стовпців:

– city: Назва міста (наприклад, Tokyo, New York, Mexico City);

– city_ascii: ASCII-формат назви міста (наприклад, Tokyo, New York, Mexico City);

– lat: Широта місця розташування міста в десятковому форматі (наприклад, 35.6850);

– lng: Довгота місця розташування міста в десятковому форматі (наприклад, 139.7514);

– country: Назва країни (наприклад, Japan, United States, Mexico);

– iso2: Код країни в ISO 3166-1 alpha-2 форматі (наприклад, JP, US, MX);

– iso3: Код країни в ISO 3166-1 alpha-3 форматі (наприклад, JPN, USA, MEX);

– admin_name: Назва адміністративного регіону (наприклад, TEЌkyEЌ, New York, Ciudad de MГ©xico);

– capital: Тип столиці або назва столиці (наприклад, primary, admin);

– population: Кількість населення міста (наприклад, 35676000, 19354922.0, 19028000);

– id: Ідентифікаційний номер міста.

3.7 Розробка функціоналу пошуку та візуалізації для інструменту аналізу та обробки csv-даних

Розглянемо розробку функціоналу для пошуку та візуалізації географічних даних, а також обговоримо ключові аспекти та інструменти, які були використані в цьому проекті.

3.7.1 Пошук міста

Функціональними можливостями кнопки «Пошук міста» є:

– користувач вводить назву міста у вікно консолі;

– програма шукає введене місто в csv-файлі;

– якщо місто знайдено, його координати та населення використовуються для створення маркера на карті;

– карта зберігається у файлі city_map.html та автоматично відкривається у веб-браузері.

Користуємося бібліотекою pandas для зчитування даних з файлу "worldcities.csv" та видалення рядків з пропущеними значеннями (рис.3.13).

Функція для введення назви міста get_city_input() відповідає за отримання назви міста від користувача через консоль. Вона перевіряє, чи введена назва не є порожньою, та повертає її.

Після отримання назви міста, ми здійснюємо пошук в CSV-даних. Якщо місто знайдено, отримуємо його географічні координати та населення. Створюємо інтерактивну карту за допомогою folium, додаємо маркер з інформацією про місто та зберігаємо карту у файл "city_map.html". Відразу відкриваємо створений файл у веб-браузері.

```
import folium
import pandas as pd
import webbrowser
import os
import tkinter as tk

# Функція для відображення карти міста
def show_city_map():
    # Отримуємо введення користувача з поля введення
    city_name = entry_city.get().strip()

    # Читаємо дані з файлу
    data = pd.read_csv("worldcities.csv")

    # Видаляємо рядки з пропущеними значеннями
    data.dropna(axis=0, inplace=True)

    # Шукаємо місто за введеною назвою
    matching_city = data[data['city_ascii'].str.lower() == city_name.lower()]

    # Перевіряємо, чи знайдено місто
    if len(matching_city) == 0:
        print(f"Місто з назвою '{city_name}' не знайдено.")
    else:
        # Отримуємо дані про місто
        city_data = matching_city.iloc[0]
        city_lat = city_data['lat']
        city_lon = city_data['lng']
        city_population = city_data['population']

        # Створюємо карту
        map = folium.Map(location=[city_lat, city_lon], zoom_start=10, tiles="OpenStreetMap")

        # Додаємо маркер на карту для знайденого міста
        popup_text = f"{city_data['city']}, {city_data['country']}<br>Population: {city_population}"
        folium.Marker(location=[city_lat, city_lon], popup=popup_text).add_to(map)

        # Зберігаємо карту у файл
        map_file = f"{city_name}_map.html"
        map.save(map_file)

        print(f"Карта з містом '{city_name}' та його населенням збережена у файлі {map_file}.")

        # Відразу відкриваємо файл у браузері
        webbrowser.open(map_file)

# Створюємо графічний інтерфейс
root = tk.Tk()
root.title("Пошук міста")

# Поле для введення назви міста
label_city = tk.Label(root, text="Введіть назву міста:")
label_city.pack()

entry_city = tk.Entry(root)
entry_city.pack()
```

Рисунок 3.13 – Лістінг кнопки «Пошук міста»

3.7.2 Пошук країни

Функціональними можливостями кнопки «Пошук країни» є:

– користувач вводить назву країни у вікно Tkinter;

– програма шукає всі міста введеної країни в CSV-файлі;

– за середніми координатами міст створюється карта з маркерами для кожного міста;

– карта зберігається у файлі з назвою країни та автоматично відкривається у веб-браузері.

Аналогічно до попереднього випадку, ми зчитуємо дані та видаляємо рядки з пропущеними значеннями.

Функція для відображення карти країни show_country_map():викликається при натисканні на кнопку "Пошук". Вона отримує назву країни, введену користувачем у поле введення. Після отримання назви країни, ми здійснюємо фільтрацію даних, щоб знайти всі міста цієї країни.

На основі знайдених міст обчислюємо середнє значення координат, яке використовується як центр мапи. Додаємо маркери для кожного міста на карту, з форматованим населенням. Зберігаємо карту у файл з назвою країни і відкриваємо його у веб-браузері.

3.7.3 Гра в штати

Гра "Гра в штати" є освітнім та розважальним проектом, який допомагає користувачам вивчати географічне розташування штатів США. Ця гра реалізована за допомогою Python та бібліотек turtle для графічного представлення та анімації та pandas для роботи з даними з файлу CSV.

Для реалізації виконуємо імпорт бібліотек:

– turtle: Використовується для створення графічного інтерфейсу та візуалізації гри;

– pandas: Використовується для обробки та управління даними.

Виконуємо ініціалізацію нового екрану та завантаження зображення штатів (рис.3.14).

```
screen = turtle.Screen()
screen.title("U.S. States Game")
image = "blank_states_img.gif"
screen.addshape(image)
turtle.shape(image)
```

Рисунок 3.14 – Налаштування екрану та зображення

Потім виконуємо зчитування списку штатів з CSV-файлу (рис.3.15).

```
data = pandas.read_csv("50_states.csv")
list_of_state = data.state.to_list()
```

Рисунок 3.15 - Зчитування даних CSV-файлу

Виконуємо перевірку введеного штату на правильність та відображення на карті (рис.3.16).

```
if answer_state in list_of_state:
    guessed_state.append(answer_state)
    t = turtle.Turtle()
    ...
    t.write(answer_state)
```

Рисунок 3.16 – Обробка вводу користувача

Після завершення гри користувачу пропонується зберегти список невідгаданих штатів у файлі CSV (рис.3.17).

```
if answer_state == "Exit":
    missing_state = [state for state in list_of_state if state not in guessed_state]
    missing_state_data = pandas.DataFrame(missing_state)
    missing_state_data.to_csv("to_learn.csv")
    break
```

Рисунок 3.17 – Збереження невідгаданих штатів

3.8 Тестування інструменту аналізу та обробки csv-даних

Інтерактивна карта - це інструмент, який надає користувачам можливість взаємодії з географічною інформацією за допомогою графічного інтерфейсу. Користувач може шукати країни, міста, а також аналізувати дані про населення країн (рис.3.18).

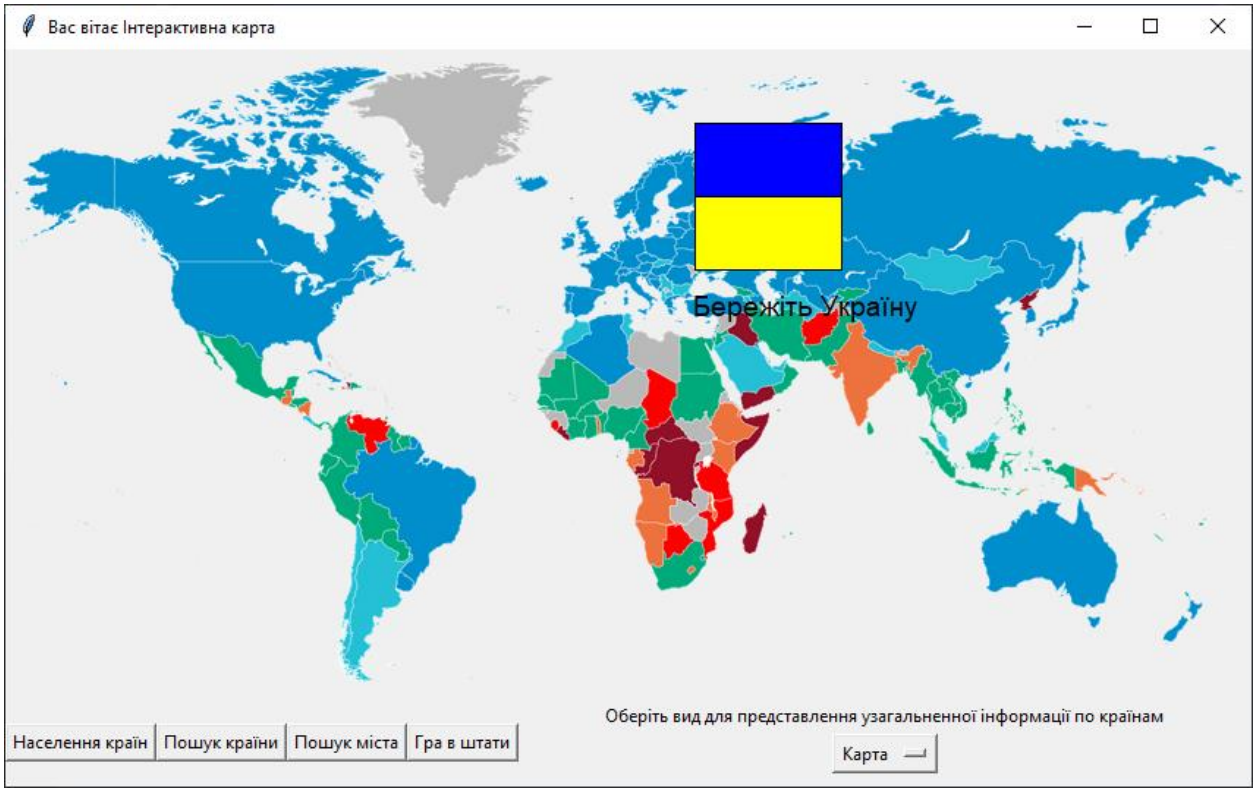


Рисунок 3.18 – Головний вид програмного інструменту

Основні функції:

– об'єкти на канві переміщуються анімовано зліва направо;

– при натисканні на кнопку "населення країн" відкривається карта, що відображає інформацію про населення країн;

– користувач може ввести назву країни у відповідному полі та натиснути кнопку "пошук країни" для отримання інформації про введену країну;

– користувач може ввести назву міста у відповідному полі та натиснути кнопку "пошук міста" для отримання інформації про введене місто;

– користувач може ввести кількість населення у відповідному полі та натиснути кнопку "пошук по кількості населення" для отримання інформації про країни з введеним населенням;

– користувач може запустити гру в штати, що допомагає вивчити географічне розташування штатів США;

– користувач може обрати одне з двох представлень - карта або графік - для отримання узагальненої інформації по країнам.

Кнопка "Населення країн" (рис.3.19) є частиною графічного інтерфейсу, який дозволяє користувачам взаємодіяти з географічною інформацією про міста за критерієм кількості населення. Ця кнопка дозволяє користувачам аналізувати та порівнювати різні міста згідно їхнього населення.

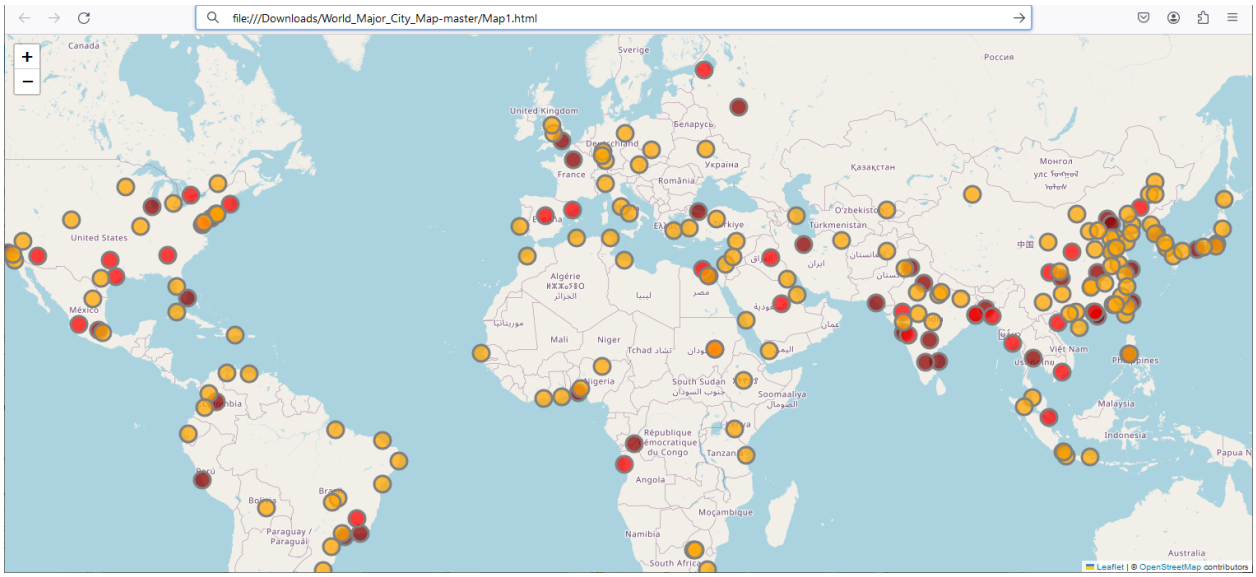


Рисунок 3.19 – Результат функціоналу "Населення країн"

Користувачу не потрібно вводити жодних додаткових даних, але він може змінити діапазон кількості населення безпосередньо в коді.

Після натискання кнопки "Населення країн", система автоматично створює та відображає карту, на якій позначені міста з відповідними кількостями населення. Система завантажує дані з файлу "worldcities.csv". Видаляє рядки з пропущеними значеннями.

Створює карту, на якій для кожного міста показана інформація про кількість населення та країну, до якої воно належить (рис.3.20). Міста позначені колірними маркерами залежно від кількості населення. Після генерації карти система зберігає її у файлі "Map1.html" та автоматично відкриває цей файл у веб-браузері користувача.

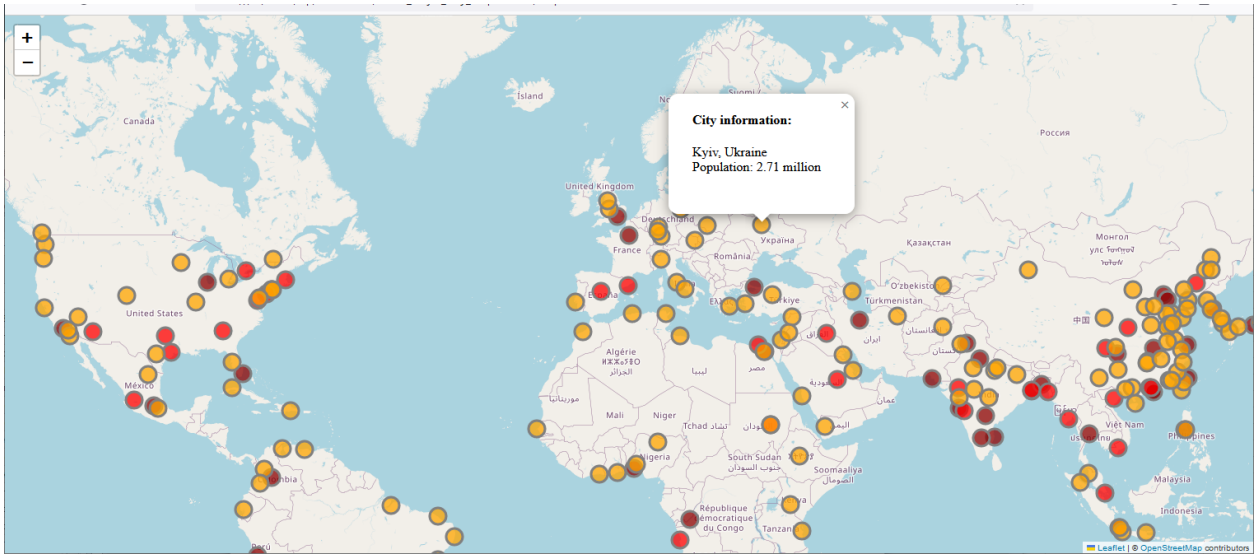


Рисунок 3.20 – Результат інструменту на прикладі м. Києва

В програмі можна виконувати пошук. Кнопка «Пошук» активує функцію, яка відображає на карті міста, де кількість населення відповідає введеному користувачем значенню. Користувач вводить кількість населення, для якої хоче знайти міста, у поле для введення. Після введення даних користувач натискає кнопку "Пошук". Система читає дані з файлу "worldcities.csv". Видаляє рядки з пропущеними значеннями. Знаходить міста з введеною кількістю населення. Визначає центральні координати для відображення на карті. Створює карту із маркерами для кожного знайденого міста, після чого зберігає цю карту у файл "coutries_map.html" (рис.3.21) або

"cities_map.html" (рис.3.23). Після обробки даних система автоматично відкриває файл "cities_map.html" у браузері, де користувач може побачити міста, які відповідають введеному критерію населення. На рисунку 3.22 наведено результат за критерієм країна «Україна».

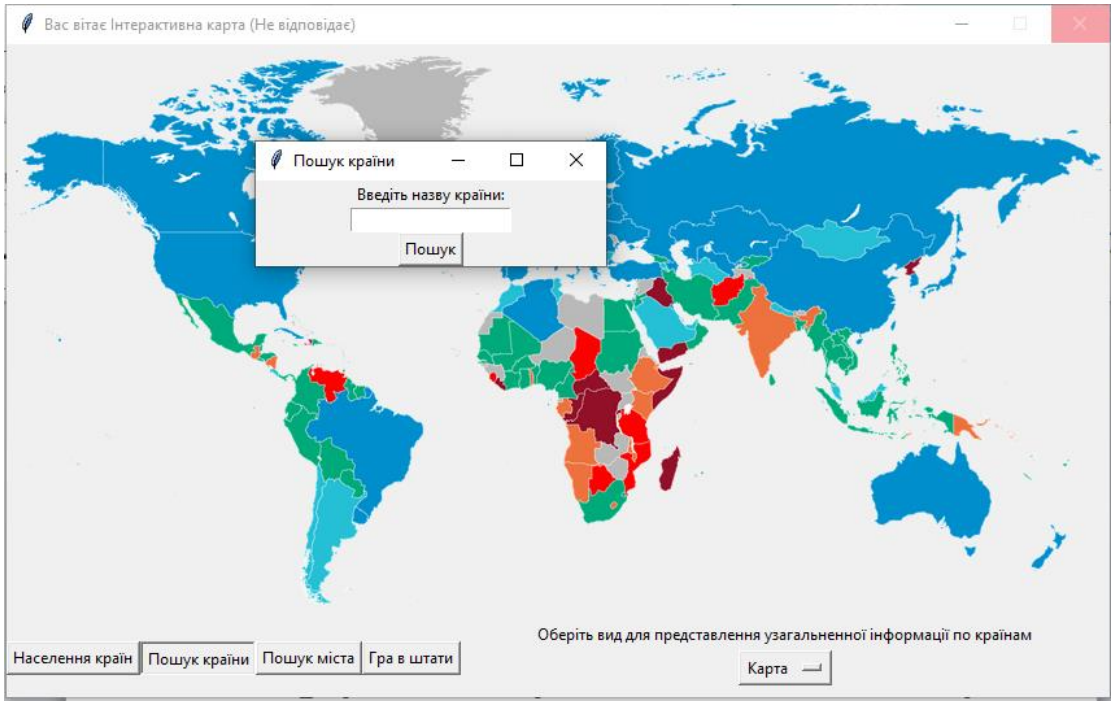
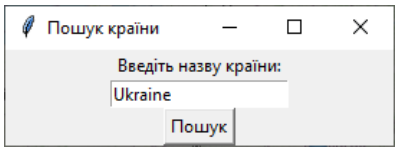


Рисунок 3.21 – Форма для пошуку країн



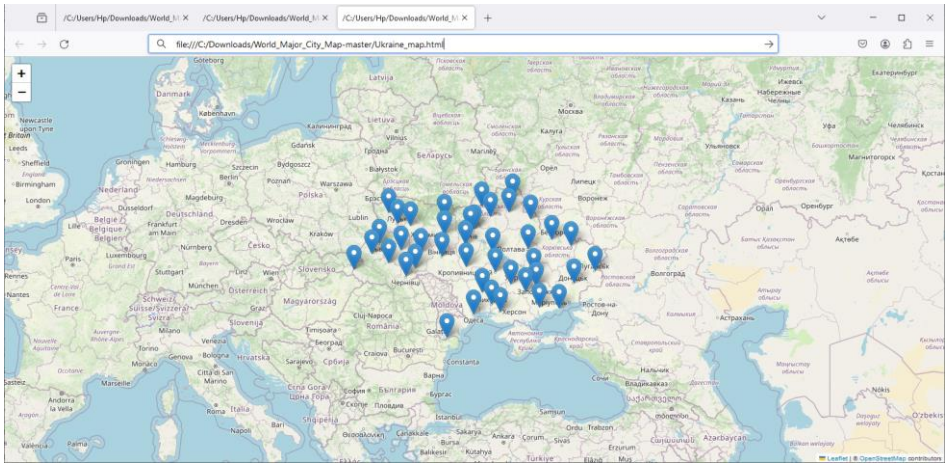


Рисунок 3.22 – Результат «Пошук країни» на прикладі України

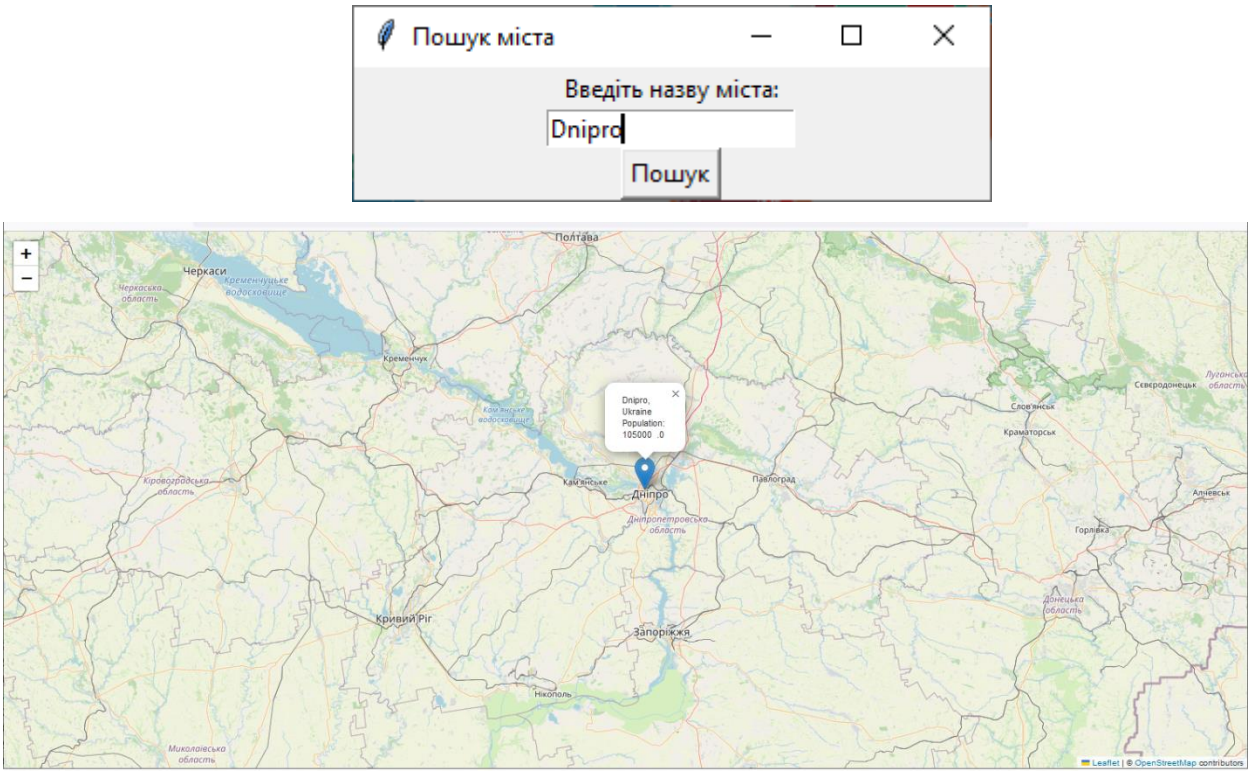


Рисунок 3.23 – Результат «Пошук країни» на прикладі м. Дніпро

Кнопка «Гра в штати» - користувач повинен вгадати назву одного з 50 американських штатів на основі графічного зображення (рис.3.24).

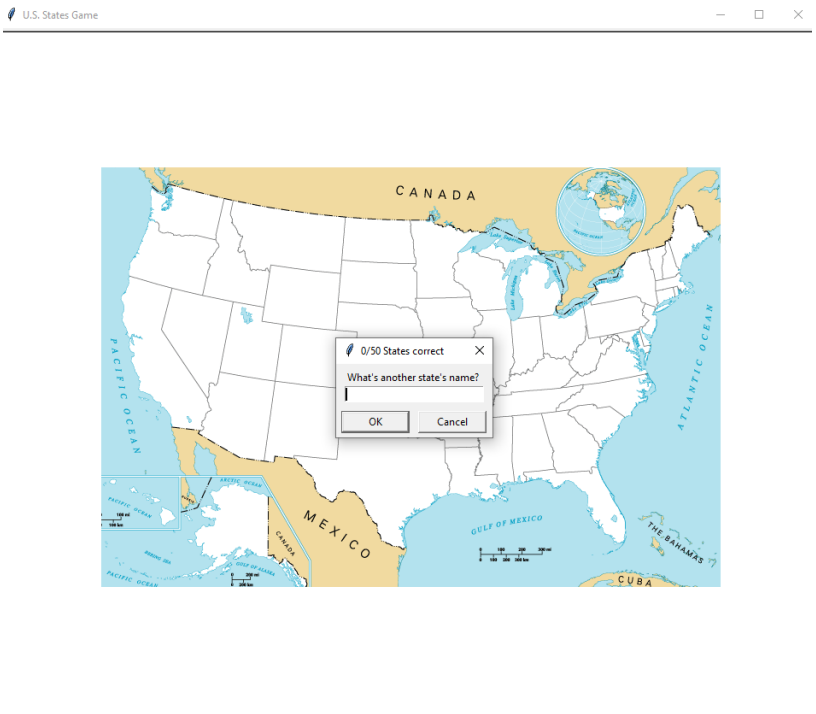


Рисунок 3.24 – Інтерфейс Гра в штати

Користувач запускає код, який відкриває вікно із зображенням картки США. Потім вводить назву штату англійською мовою, який він вважає правильним.Код перевіряє, чи є введена відповідь дійсно існуючим штатом. Якщо відповідь вірна, на картці з'являється напис із назвою відгаданого штату. Гра триває до тих пір, поки користувач не вгадає всі штати або не вибере вийти. Після завершення гри виводиться файл to_learn.csv, в якому містяться назви штатів, які користувач не вгадав. Гра завершується, коли користувач вирішує вийти, введення слова "Exit" або коли всі штати вгадані.

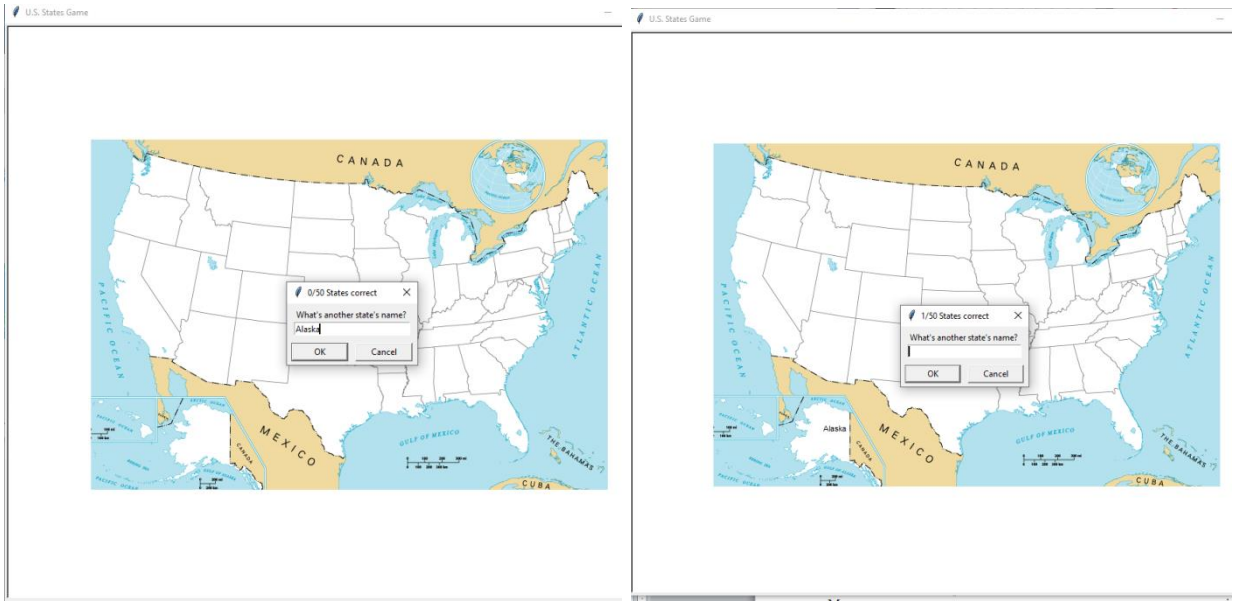


Рисунок 3.25 – Результат гри на прикладі штату Alaska

Користувач може обрати узагальнений вид інформації і обрати кнопку «Карта», генерує інтерактивну карту світу, яка показує міста в залежності від їхньої кількості населення.

Користувач запускає код, що відкриває веб-браузер з відображенням інтерактивної карти світу. На карті з'являються маркери для міст, які мають популяцію більше або рівну 2 мільйонам. Користувач може навести курсор миші на маркер, щоб побачити інформацію про місто, таку як назва міста, країна та кількість населення. Маркери можуть мати різний колір в залежності від кількості населення. Наприклад (рис.3.26):

– помаранчевий: 2 - 4 мільйони;
– червоний: 4 - 6 мільйони;
– темно-червоний: більше 6 мільйонів.

Збереження та відкриття карти: Після створення інтерактивної карти, вона зберігається в файлі Map1.html, який автоматично відкривається у веб-браузері для користувача.

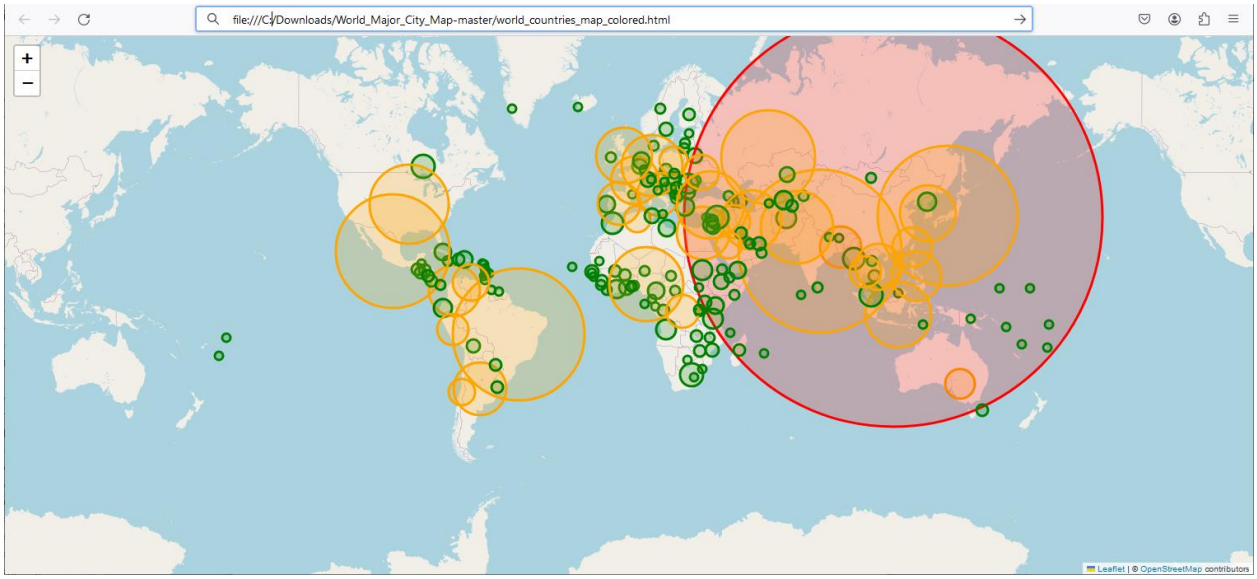


Рисунок 3.26 – Результа даних по кількості населення

Є функціональна кнопка, яка аналізує дані про населення міст у різних країнах та візуалізує результати на графіку. Користувач запускає код, який обробляє дані та створює графік. Код завантажує дані з файлу worldcities.csv та групує їх за країною. Далі обчислює загальну кількість населення міст для кожної країни. З аналізу отриманих даних обираються п'ять країн з найбільшою загальною кількістю населення міст. На основі вибраних країн будується стовпчикова діаграма, де по горизонталі відображаються назви країн, а по вертикалі - загальна кількість населення міст у мільйонах. Графік відображається у вікні користувача, де він може зрозуміти, які країни мають найбільшу кількість міст на основі загального населення (рис.3.27).

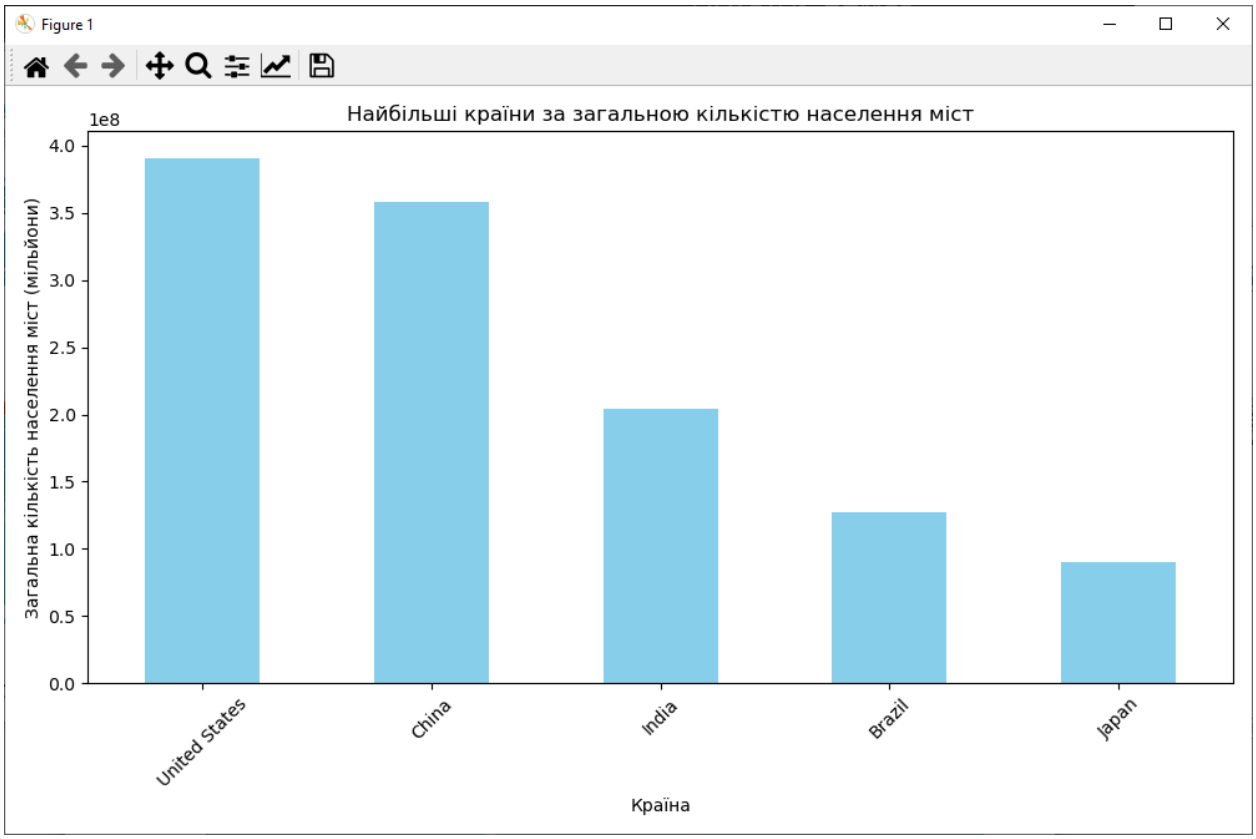


Рисунок 3.27 – Аналіз даних про населення

3.9 Висновки

Інтерактивна карта є зручним інструментом для вивчення географії, аналізу та взаємодії з географічними даними. Вона надає ряд функцій для пошуку та візуалізації інформації, що робить її корисною для освіти, дослідження та розваг.

Завдяки інформаційним вікнам користувач може отримати докладну інформацію про кожне місце на карті, просто клікаючи на маркер. Це робить взаємодію з додатком більш інформативною та зручною, а використання HTML-шаблонів покращує читабельність та привабливість відображеної інформації.

Додавання маркерів на карту забезпечує інтерактивний та візуальний спосіб представлення геоданих. Використовуючи бібліотеку folium, ми змогли ефективно створити маркери, які відображають міста з населенням

понад 2 мільйони осіб, і використовувати кольорову схему для підкреслення важливості цих міст.

Кнопка "Населення країн" надає користувачам простий інструмент для візуалізації географічного розташування міст на основі кількості населення. Вона дозволяє легко аналізувати і порівнювати різні регіони світу згідно розподілу населення.

Гра "Гра в штати" є простим та ефективним інструментом для навчання та розваги. Вона демонструє можливість використання Python для створення освітніх ігор, які можуть допомагати користувачам вивчати нову інформацію у цікавій та інтерактивній формі.

ВИСНОВКИ

У даній кваліфікаційній роботі було розроблено та досліджено інструмент аналізу та обробки CSV-даних з використанням мови програмування Python. Робота включає докладний аналіз геопросторових інструментів та інтеграцію з CSV-даними, а також розробку програмного забезпечення для візуалізації та аналізу даних.

У роботі проведено докладний аналіз стану геоданих, що включає їх походження, формати, структури та потенційні області використання. Було розглянуто різноманітні джерела геоданих та їх особливості, що слугують основою для подальшої обробки та аналізу.

Розроблено механізм завантаження, обробки та інтеграції CSV-даних. Цей процес включає в себе читання файлів, фільтрацію, сортування та перетворення даних для подальшого аналізу. Додатково було вивчено та використано методи валідації даних для підтвердження їх коректності та точності.

На основі оброблених даних розроблено інтерактивні карти з використанням Python та бібліотеки Folium. Ці карти дозволяють візуально представити географічне розташування та характеристики даних, надаючи користувачам зрозумілий та інтуїтивно зрозумілий інструмент для аналізу.

Було обрано оптимальний технологічний стек для реалізації інструменту, включаючи Python як основну мову програмування, бібліотеки для роботи з даними (pandas) та геоданими (Folium). Детально описано структуру проекту, особливості інтерфейсу користувача, а також реалізовані функціональні можливості для аналізу та візуалізації даних.

Отже, розроблений інструмент може бути використаний в різних сферах, таких як освітній процес, геоаналітика, маркетингові аналітики та інші, де потрібен аналіз геоданих.

ПЕРЕЛІК ПОСИЛАНЬ

1. Гейміфікація в освіті. [Електронний ресурс] – Режим доступу до ресурсу: http://blog.gioschool.com/gamification

2. Дядікова О. А. Гра як інструмент: що таке гейміфікація? [Електронний ресурс] – Режим доступу до ресурсу: https://mistosite.org.ua/uk/articles/hra-iak-instrument-shcho-take-heimifikatsiia/

3. Salen K., Zimmerman E. Rules of Play: Game Design Fundamentals. Cambridge: MIT Press, 2003. 688 pp.

4. Переяславська С. О., Смагіна О. О. Гейміфікація як сучасний напрям вітчизняної освіти. Відкрите освітнє Е-середовище сучасного університету: електронне наук. фахове видання. 2019. 250–260 с. [Електронний ресурс] – Режим доступу до ресурсу: https://doi.org/10.28925/2414-0325.2019s24

5. Bocher, E., Ertz, O. (2018). A redesign of OGC Symbology Encoding standard for sharing cartography. PeerJ Computer Science 4, e143.

6. Balog, D., Houtmeyers, R. (2017). Testbed-12 Vector Tiling Implementation Engineering Report. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.ogc.org/per/16-067r4.html.

7. Grothe, M., Brentjens, T., 2013. Joining tabular and geographic data – Merits and possibilities of the Table Joining Service 53.

8. Fujimura, H., Sanchez, O., Ferreiro, D., Kayama, Y., Hayashi, H., Iwasaki, N., Mugambi, F., Obukhov, T., Motojima, Y., Sato, T. (2019). DESIGN AND DEVELOPMENT OF THE UN VECTOR TILE TOOLKIT. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLII-4/W14, 57–62.

9. Kyriakidis, P., Hadjimitsis, D., Skarlatos, D., Mansourian, A. (2019). Geospatial Technologies for Local and Regional Development: Proceedings of the 22nd AGILE Conference on Geographic Information Science. Springer.

10. GÖBEL, F., KIEFER, P., & RAUBAL, M. (2017). FeaturEyeTrack: a vector tile-based eye tracking framework for interactive maps. In Societal Geo-

Innovation: Short Papers, Posters and Poster Abstracts of the 20th AGILE Conference on Geographic Information Science. Wageningen University and Research (pp. 9-12).

11. Pandas Documentation. [Електронний ресурс] – Режим доступу до ресурсу: https://pandas.pydata.org/docs/

12. NumPy v1.20 Manual. [Електронний ресурс] – Режим доступу до ресурсу: https://numpy.org/doc/stable/

13. Spyder. [Електронний ресурс] – Режим доступу до ресурсу: https://www.spyder- ide.org/

14. Map Projection. [Електронний ресурс] – Режим доступу до ресурсу: https://www.usna.edu/Users/oceano/pguth/md_help/html/mapb0iem.htm

Додаток А. Фагмент лістингу програми

```
import matplotlib.pyplot as plt
import pandas as pd
# Групуємо дані за країною та обчислюємо загальну кількість населення для кожної країни
data = pd.read_csv("worldcities.csv")
country_population = data.groupby('country')['population'].sum()

# Вибираємо кілька найбільших країн
top_countries = country_population.nlargest(5)

# Створюємо графік
plt.figure(figsize=(10, 6))
top_countries.plot(kind='bar', color='skyblue')
plt.title('Найбільші країни за загальною кількістю населення міст')
plt.xlabel('Країна')
plt.ylabel('Загальна кількість населення міст (мільйони)')
plt.xticks(rotation=45)
plt.tight_layout()

# Відображаємо графік
plt.show()

import folium
import pandas as pd
import webbrowser
import os
import tkinter as tk
```

```
# Функція для відображення карти з містами з введеною кількістю населення
def show_cities_map():
    # Отримуємо введення користувача з поля введення
    population_threshold = float(entry_population.get())

    # Читаємо дані з файлу
    data = pd.read_csv("worldcities.csv")

    # Видаляємо рядки з пропущеними значеннями
    data.dropna(axis=0, inplace=True)

    # Пошук міст з введеною кількістю населення
    matching_cities = data[data['population'] == population_threshold]

    # Визначаємо центр мапи
    center_lat = matching_cities['lat'].mean()
    center_lon = matching_cities['lng'].mean()

    # Створюємо карту
    map = folium.Map(location=[center_lat, center_lon], zoom_start=4,
tiles="OpenStreetMap")

    # Додаємо маркери на карту для кожного знайденого міста
    for index, city in matching_cities.iterrows():
        # Округлюємо кількість населення до мільйонів і форматуємо число як
"2.7 мільйони"
        population_formatted = '{:.1f} мільйони'.format(city['population'] /
1_000_000)
```

```
        popup_text = f"{city['city']}, {city['country']}<br>Population: {population_formatted}"
        folium.Marker(location=[city['lat'], city['lng']], popup=popup_text).add_to(map)

    # Зберігаємо карту у файл
    map_file = "cities_map.html"
    map.save(map_file)

    print("Карта з містами з введеною кількістю населення збережена у файлі cities_map.html.")

    # Відразу відкриваємо файл у браузері
    webbrowser.open(map_file)

# Створюємо графічний інтерфейс
root = tk.Tk()
root.title("Пошук по кількості населення")

# Поле для введення кількості населення
label_population = tk.Label(root, text="Введіть кількість населення для відображення міст:")
label_population.pack()

entry_population = tk.Entry(root)
entry_population.pack()

# Кнопка для пошуку міст з введеною кількістю населення
btn_search_population = tk.Button(root, text="Пошук", command=show_cities_map)
```

```
btn_search_population.pack()

# Запускаємо головний цикл вікна
root.mainloop()

import folium
import pandas as pd
import webbrowser
import os
import tkinter as tk

# Функція для відображення карти країни
def show_country_map():
    # Отримуємо введення користувача з поля введення
    country_name = entry_country.get().strip()

    # Читаємо дані з файлу
    data = pd.read_csv("worldcities.csv")

    # Видаляємо рядки з пропущеними значеннями
    data.dropna(axis=0, inplace=True)

    # Шукаємо всі міста введеної країни
    matching_cities = data[data['country'].str.lower() == country_name.lower()]

    # Перевіряємо, чи знайдено міста в країні
    if len(matching_cities) == 0:
        print(f"Міста в країні '{country_name}' не знайдено.")
```

```
    else:
        # Визначаємо центр мапи за середнім значенням координат міст
        center_lat = matching_cities['lat'].mean()
        center_lon = matching_cities['lng'].mean()

        # Створюємо карту
        map = folium.Map(location=[center_lat, center_lon], zoom_start=5, tiles="OpenStreetMap")

        # Додаємо маркери на карту для кожного знайденого міста
        for index, city in matching_cities.iterrows():
            # Форматуємо кількість населення з роздільником тисяч
            population_formatted = '{:,.0f}'.format(city['population'])
            popup_text = f"{city['city']}, {city['country']}<br>Population: {population_formatted}"
            folium.Marker(location=[city['lat'], city['lng']], popup=popup_text).add_to(map)

        # Зберігаємо карту у файл
        map_file = f"{country_name}_map.html"
        map.save(map_file)

        print(f"Карта з містами в країні '{country_name}' збережена у файлі {map_file}.")

        # Відразу відкриваємо файл у браузері
        webbrowser.open(map_file)

# Створюємо графічний інтерфейс
root = tk.Tk()
```

```
root.title("Пошук країни")

# Поле для введення назви країни
label_country = tk.Label(root, text="Введіть назву країни:")
label_country.pack()

entry_country = tk.Entry(root)
entry_country.pack()

# Кнопка для пошуку країни
btn_search_country = tk.Button(root, text="Пошук",
command=show_country_map)
btn_search_country.pack()

# Запускаємо головний цикл вікна
root.mainloop()
```

Додаток Б. Тестування програмного застосунку

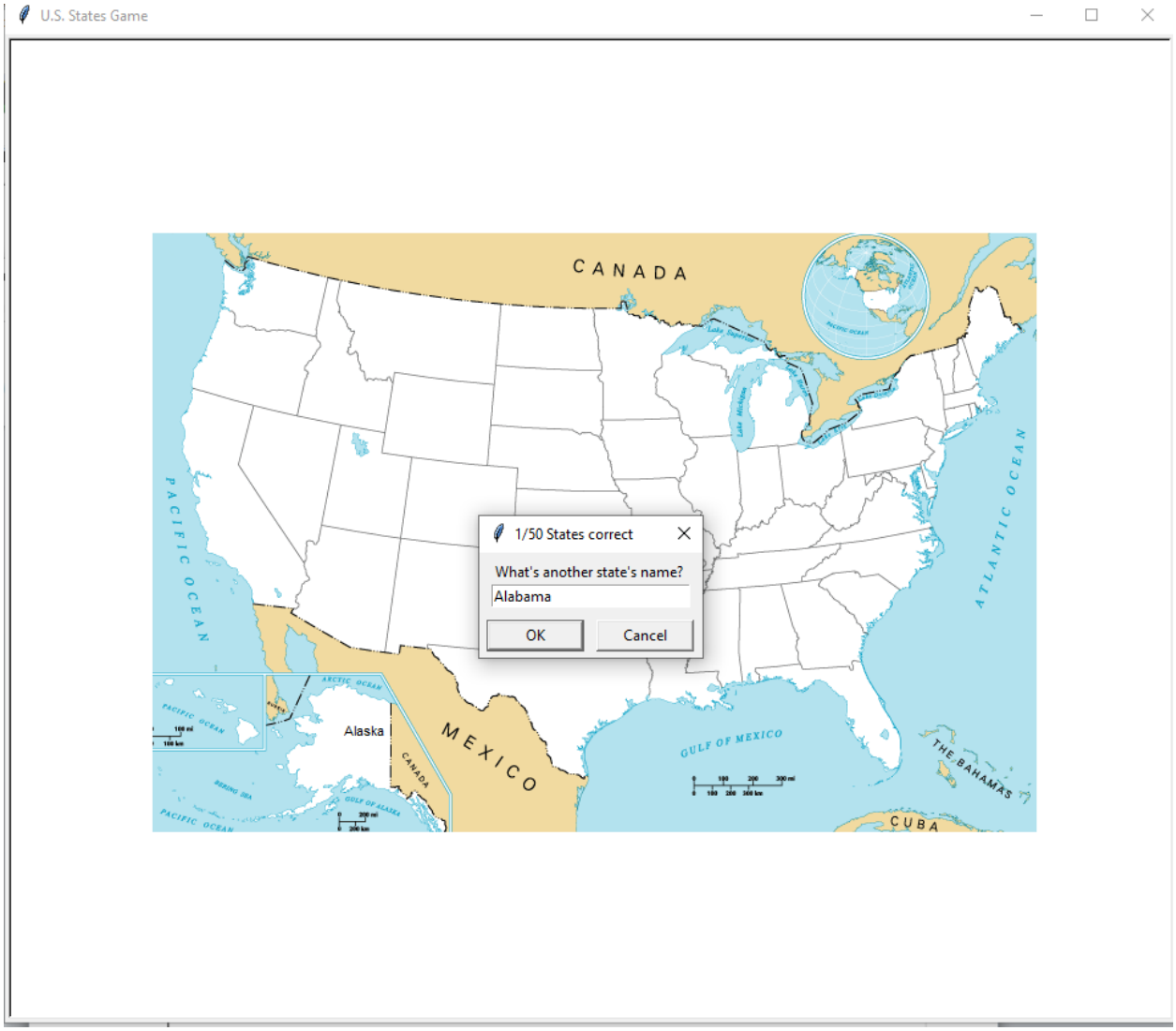


Рисунок Б.1 – Введення штату

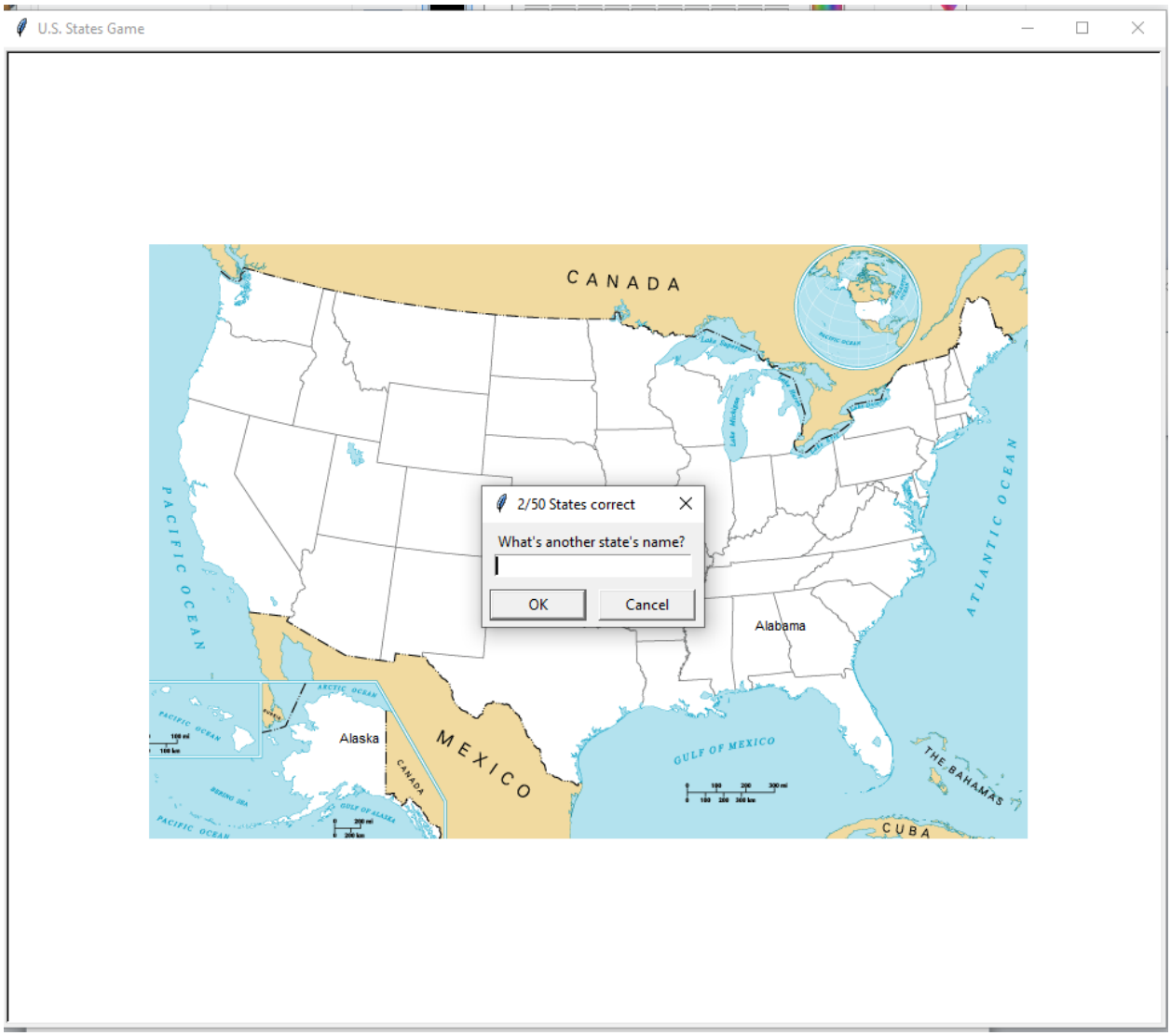


Рисунок Б.2 – Нанесено на карту штат Alabama

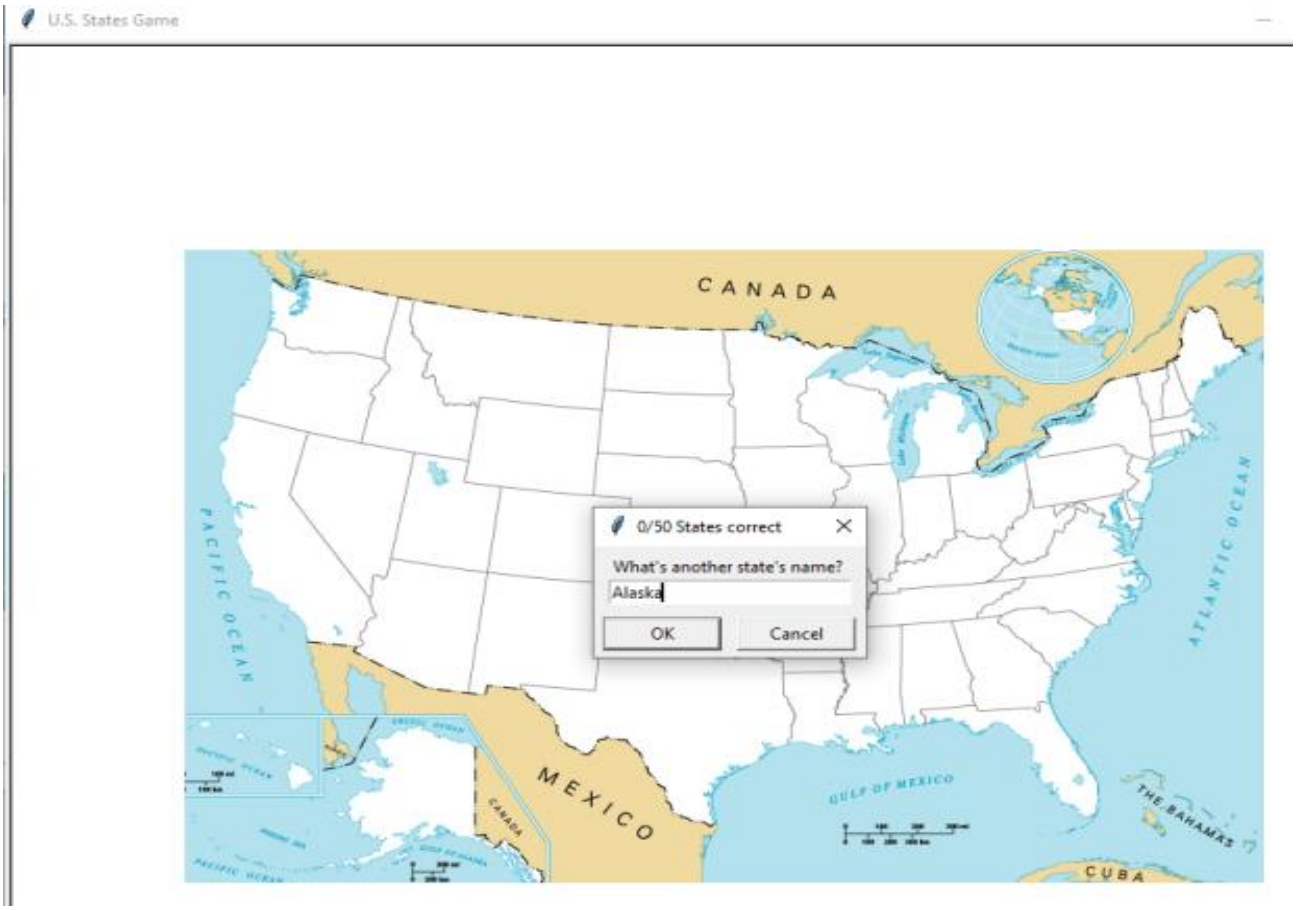


Рисунок Б.3 – Введення штату Alaska

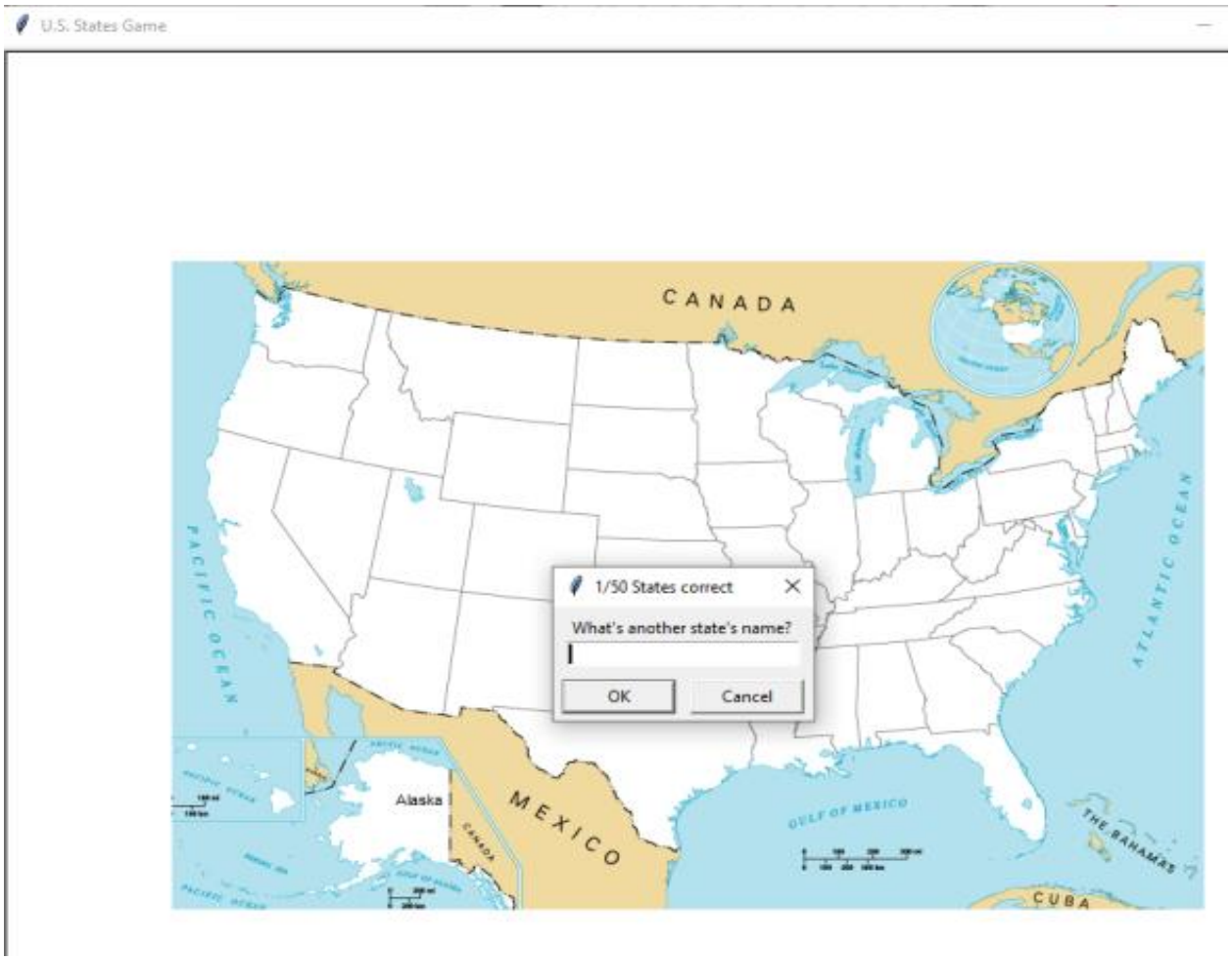


Рисунок Б.4 – Нанесено на карту штат Alaska