Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

СИСТЕМА ПРОГНОЗУВАННЯ РОЗВИТКУ ЗАБРУДНЕНЬ ВОДНИХ ТА ЗЕМНИХ ПОВЕРХОНЬ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-18-1

Ткачов В.А.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник ст.викл. Кіношенко Д.К.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«____»_____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Ткачову Володимиру Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Система прогнозування розвитку забруднень водних та земних поверхонь
затверджена наказом університету від 16 травня 2022 року № 541Ст

2. Термін подання студентом роботи до екзаменаційної комісії 28 травня 2022 р.

3. Вихідні дані до роботи персональний комп'ютер на операційній системі Windows 10, мова програмування Python. сережовище розробки PyCharm.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз програмного забезпечення, аналогічного до розроблюваного, виявлення його переваг та недоліків.
2.Розгляд методу полікоординатних перетворень
3. Розроблення власного програмного продукту та користувацького інтерфейсу

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) користувацькі інтерфейси аналогічних програм, формули, приклади роботи програми, приклади застосування інструментів.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	доцент Бєлова Н.В.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	12.04.2022	
2	Аналіз завдання, підбір літератури	13.04.22-15.04.22	
3	Аналіз літератури з досліджуваної проблеми	16.04.22-17.04.22	
4	Аналіз підходів до розпізнавання та класифікації образів на зображенні	18.04.22-28.04.22	
5	Розробка методу розпізнавання та класифікації образів кулінарних страв	29.04.22-14.05.22	
6	Програмна реалізація	15.05.22-23.05.22	
7	Оформлення пояснювальної записки	24.05.22-26.05.22	
8	Перевірка на плагіат	27.05.22	
9	Рецензування	28.05.22	
10	Підготовка презентації та доповіді	29.05.22-30.05.22	
11	Занесення роботи в електронний архів	31.05.22	
12	Попередній захист кваліфікаційної роботи	02.06.22	

Дата видачі завдання 18 квітня 2022 р.

Студент _____
(підпис)

Керівник роботи _____ ст. викл. Кіношенко Д.К.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Робота містить 55 сторінок, 37 малюнків, 4 формули, 1 додаток та 28 джерел.

PYTHON, PYCHARM, ПОЛІКООРДИНАТНІ ПЕРЕТВОРЕННЯ, ФУНКЦІЇ, ГРАФІКИ, MATPLOTLIB, MATPLOTLIB.PYPLOT, MATPLOTLIB.LINES, MATPLOTLIB.PATH

Об’єктом роботи є процес розробки програми для надання графічної інформації про забруднення землі та водної поверхні на основі даних, введених користувачем..

Метою роботи є створення програмного забезпечення для прогнозування розвитку забруднень водних та земних поверхонь. Аналіз подібної програми, виявивлення її сильних та слабких сторон, розгляд полікоординатних перетвореннь і бібліотек Python.

У ході роботи було розглянуто PyCharm як середовище розробки. Використана мова програмування Python. Розроблено програмний продукт для прогнозування розвитку забруднення водних та земних поверхонь.

PYTHON, PYCHARM, POLYCOORDINATE TRANSFORMATIONS, FUNCTIONS, GRAPH, MATPLOTLIB, MATPLOTLIB.PYPLOT, MATPLOTLIB.LINES, MATPLOTLIB.PATH

The object of the work is the process of developing a program to provide graphical information on land and water pollution based on data entered by the user.

The aim of the work is to create software for forecasting the development of pollution of water and land surfaces. Analysis of such a program, identification of its strengths and weaknesses, consideration of multi-coordinate transformations and Python libraries.

In the course of the work, PyCharm was considered as a development environment. Python programming language used. A software product for forecasting the development of pollution of water and land surfaces has been developed.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Python – мова програмування

PyCharm – середовище розробки

AutoCad – система для проектування

Django - фреймворк

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

IDE – Integrated Development Environment

NumPy – Numerical Python

Tk – Tkinter

UI – User Interface

GUI – Graphical User Interface

LISP – LIst Processing Language

ВСТУП

Сучасна людська цивілізація, окрім покращення рівня сімейного комфорту, призвела також до швидкого погіршення екологічних умов світу. Згодом зруйноване цивілізацією середовище може мати катастрофічні наслідки.

Слово «екологія» походить з грецької і означає «вивчати місця, де ми живемо». Отже, немає людини на землі, яка б не впливала на екологічні проблеми. Наприклад, забруднене заводами повітря рухається разом із вітром, і важкі частинки потрапляють у річки та океани.

Забруднення моря є найнебезпечнішим. Райони, які використовуються для морського транспорту, мають найвищий рівень забруднення. Наприклад, Балтійське море є одним з найбільш забруднених регіонів Землі. Радіоактивний розлив у Фукусімі, Японія, забрав життя багатьох риб та інших водних мешканців.

Запобігання екологічним надзвичайним ситуаціям і катастрофам є однією з найгостріших проблем, які стоять перед науковцями та інженерами. Надзвичайні ситуації на морі зазвичай супроводжуються забрудненням водного середовища нафтопродуктами. У цьому випадку завданням прогнозування розвитку надзвичайної ситуації є прогнозування поширення забруднення.

В даний час існує досить велика кількість математичних моделей, що описують переміщення антропогенних домішок у водних середовищах. Для прогнозування розвитку забруднення використовуються комп’ютерні програми та системи. Одна з таких систем реалізована в моєму дипломному проекті.

1 ЗАДАЧА РОЗРОБКИ СИСТЕМИ ПРОГНОЗУВАННЯ

Сьогодні вчені намагаються розробити програмне забезпечення для запобігання розвитку забруднення на нашій планеті. Метою мого дипломного проекту є впровадження системи розвитку забруднення земних та водних поверхонь.

1.1 Призначення

Призначенням системи є надання графічної інформації про забруднення землі та водної поверхні на основі введених користувачів. Система включає функції введення даних для побудови графіків і використання спеціальних методів для отримання кінцевих результатів. Необхідні функції, які повинна забезпечувати система:

- зручний інтерфейс для введення даних користувача;
- надання інформацію у вигляді діаграми з осями;
- можливість збереження графіки в PNG, PDF, SVG тощо;
- можливість задавати колір графіки;
- можливість задати розмір фігури;
- можливість встановлення кількості кроків;
- можливість встановлення векторів.

1.2 Підсистеми

1.2.1 Функції

Для обробки даних використовуються функції, тобто кожен блок має функцію. Це полегшує розуміння та читання коду.

1.2.2 Користувацький інтерфейс

Система реалізована на мові програмування Python, тому для розробки інтерфейсу використовуються спеціальні бібліотеки, призначені для цієї мети. Щоб вирішити цю проблему, ми вирішили розробити програму, а не веб-додаток. Вам не потрібно закривати або знову відкривати програму, щоб перезапустити її. Інтерфейс повинен дозволяти користувачеві вводити дані, перезапускати програму та виходити з програми.

1.3 Постановка задачі

На сьогоднішній день найбільш актуальним є забруднення водойм (річок, озер, океанів, підземних вод тощо), тому всім відома приказка «вода – це життя». Без води людина не може прожити три дні, але навіть усвідомлюючи важливість води в її житті, вона продовжує забруднювати водойми, незворотно змінюючи їх природний стан через скидання сміття і відходів.

Об’єктом роботи є процес розробки програми для надання графічної інформації про забруднення землі та водної поверхні на основі даних, введених користувачем.

Метою мого дипломного проекту є впровадження системи розвитку забруднення води та поверхні.

Враховуючи мету роботи, необхідно вирішити наступні завдання:

– аналіз сучасного стану розвитку програм виявлення забруднення в Україні та за кордоном;

– вибрати інструменти для створення програми;

– реалізувати метод полікоординатних перетворень;

– зробити детальний опис різних етапів розробки програми;

– протестувати розроблену програму та проаналізувати результати;

– зробити висновки про виконану роботу.

2 ОПИС ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

2.1 Структура програми

Як зазначалося вище, програми пишуться за допомогою функцій, тому структура системи точно відповідає функціям у коді. Функції пов’язані між собою так само, як одна функція викликає інші функції. У головній функції, де визначена глобальна змінна, відбувається виклик функції, що відповідає за інтерфейс. Ця функція викликає функцію, відповідальну за обробку введених користувачами даних. Також є виклик функції, що відповідає за правильність введених у поле даних, іншими словами, перевірку поля.

Мабуть, однією з найважливіших функцій є здатність обробляти дані для отримання результатів. Тобто після отримання даних користувача функція обробляє їх, виконує необхідні обчислення, потім будує графік і користувач отримує результат.

2.2 Огляд існуючих систем

Дана система має аналог, розроблений на мові програмування LISP. Одним із істотних недоліків такого моделювання є те, що для запуску та використання програми потрібна будь-яка версія сторонньої платформи AutoCad. AutoCAD використовується для створення автоматизованих проектів або програмних додатків. За допомогою AutoCad ви можете розробляти 2D та 3D-додатки, надаючи інструменти для проектування програмного забезпечення для будівництва та управління проектами.

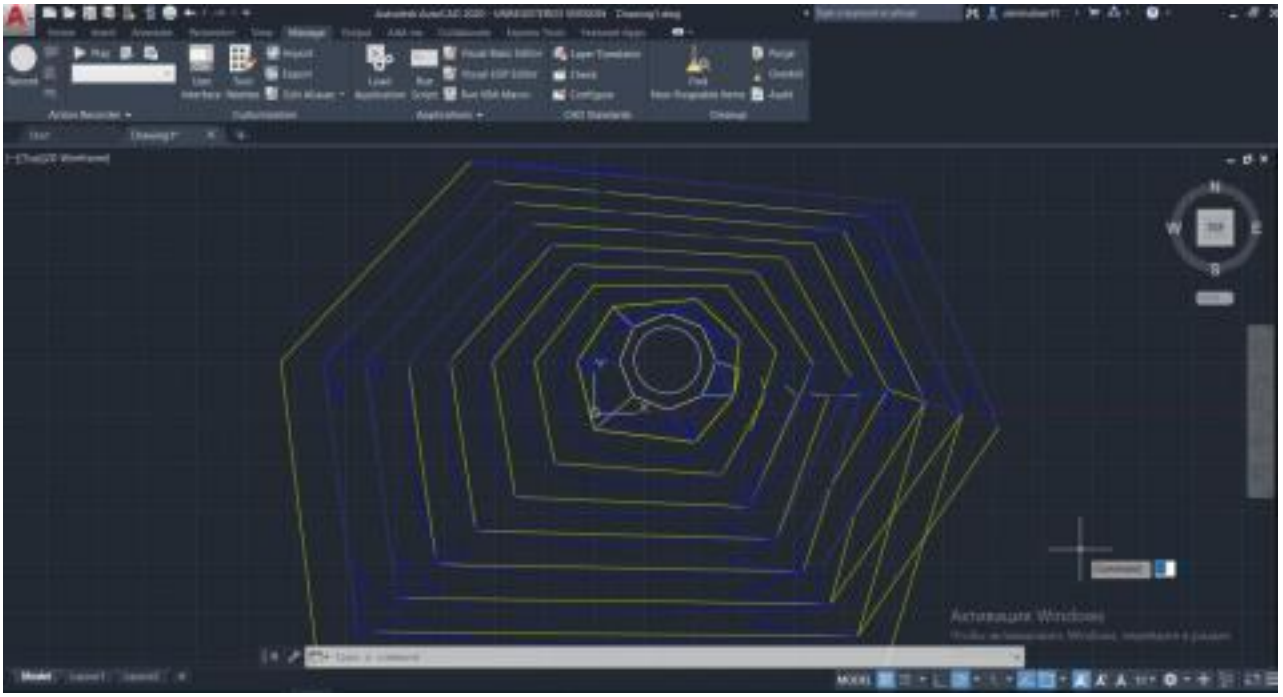

Рисунок 2.1 – Результат запуску аналогу

Щоб завантажити і продовжувати використовувати AutoCad, вам потрібна платна підписка, що робить цю симуляцію незручною у використанні. Хоча використання AutoCad є перевагою, оскільки він має більш зручний і чіткий вихід, але для його завантаження потрібен час. Моїй системі не потрібно завантажувати сторонню платформу, тобто користувач може просто натиснути на файл, щоб запустити програму. Крім того, аналогію неможливо нормально використовувати без спеціальних інструкцій, оскільки користувач повинен мати певні навички для використання AutoCad.

2.3 Огляд існуючих методів

Відомі методи прогнозування розвитку забруднення в основному базуються на методах регресії для кожної точки точок забруднення. Наприклад, одним з таких

підходів є інформаційні технології. Однією з переваг є відносно невеликий обсяг даних, переважно регіональних гідрометеорологічних факторів. Недоліком цього підходу є досить громіздкий математичний аппарат, і це також вимагає великої роботи над впровадженням програмного забезпечення. Тому мета цієї роботи – знайти простий шлях до вирішення глобального рівня забруднення. Комбінуючи різні методи, встановлено, що найбільш підходящим є метод полікоординатних перетворень.

2.4 Засоби розробки

Основним середовищем розробки є PyCharm, яке надає інструменти аналізу коду, графічний налагоджувач, інструменти для виконання модульних тестів і підтримку веброзробки на Django. PyCharm розроблено JetBrains на основі IntelliJ IDEA. Також було використано середовище розробки IDLE, створене за допомогою бібліотеки Tk. Ця бібліотека дозволяє використовувати IDLE у багатьох операційних системах, таких як Windows, Mac OS і Unix-подібні операційні системи.

Безпосередньо використано мову програмування Python версії 3.8.2 для розробки програми. Python — це мова програмування високого рівня загального призначення, яка також використовується для розробки вебзастосунків. Мова орієнтована на підвищення продуктивності розробників і читання коду. Python підтримує кілька парадигм програмування: структуроване, об’єктно-орієнтоване, функціональне, імперативне та аспектно-орієнтоване. Ця мова має динамічний введення, автоматичне керування пам’яттю, повний самоаналіз, механізм обробки винятків, підтримку багатопоточних обчислень і зручні високорівневі структури даних. Код Python організований у функції та класи, які можна об’єднати в модулі, які, у свою чергу, можна об’єднати в пакети.

2.4.1 Мова програмування Python

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

– веб-розробка (на стороні сервера);

– розробка програмного забезпечення;

– математика;

– системні скрипти.

Python також має такі особливості:

– створити веб-додаток на сервері;

– створювати робочі процеси;

– підключення до системи бази даних.

– можна читати та змінювати файли.

– обробка великих даних та виконання складної математики.

–швидка розробка програмного забезпечення, готового до виробництва. Python працює на різних платформах (Windows, Mac, Linux, Raspberry Pi тощо).

Python має простий синтаксис, схожий на англійську. Python має синтаксис, який дозволяє розробникам писати програми з меншою кількістю рядків, ніж інші мови програмування.

Python працює з системою перекладу, тобто код можна виконувати як написаний. Це означає, що створення прототипу може бути дуже швидким.

Найновішою основною версією Python є Python 3, яка використовується при написанні програми. Однак Python 2, хоча не оновлюється нічим іншим, окрім оновлень безпеки, все ще залишається досить популярним.

Python може буде записаний у текстовому редакторі. Можна писати Python в інтегрованому середовищі розробки, 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 одно має свої переваги порівняно з іншими мовами:

– Python був розроблений для читабельності і має деякі 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 круглими дужками;

– для визначення області застосування Python покладається на відступ.

Використовуючи пробіли, наприклад, область циклів, функції та класи. Інші мови програмування зазвичай використовують для цього фігурні дужки.

Python простий, тому його легко вивчити, оскільки він вимагає унікального синтаксису лише для читання. Розробники можуть читати та перекладати код Python легше, ніж інші мови. З іншого боку, це зменшує витрати на обслуговування та розробку програми, 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 вам модуль або пакет, його можна масштабувати для використання в інших проектах, і легко імпортувати або експортувати ці модулі.

Однією з найбільш перспективних переваг Python є те, що і стандартна бібліотека, і інтерпретатор доступні безкоштовно, як у двійковій, так і у вихідній формі. Ексклюзивності також немає, оскільки Python та всі необхідні інструменти доступні на всіх основних платформах. Тому це привабливий варіант для розробників, які не хочуть турбуватися про оплату високих витрат на розробку.

2.4.2 Середовище розробки PyCharm

PyCharm - це гібридна платформа, розроблена JetBrains як IDE для Python, що забезпечує широкий спектр необхідних інструментів для розробників Python, щоб

створити зручне середовище для продуктивної розробки програм на мові програмування Python. Він підтримує дві версії: v2.x та v3.x.

Ми можемо запускати PyCharm в Windows, Linux або Mac OS. Крім того, він містить модулі та пакети, які допомагають програмістам розробити програмне забезпечення, використовуючи Python, за менший час та з мінімальними зусиллями. Деякі основні функції, надані PyCharm:

Розумний редактор коду:

– завдяки цьому програмування стає більш зручнішим та швидшим;

– складається з кольорових виділень для ключових слів, класів та функцій;

– допомагає збільшити читабельність та розуміння коду;

– допомагає легко визначити помилки;

– надає функцію автозаповнення та інструкції щодо заповнення коду.

Навігація по коду:

– допомагає розробникам в редагуванні та вдосконаленні коду з меншими зусиллями та часом;

– за допомогою навігації з кодом розробник може легко перейти до функції, класу чи файлу;

– програміст може знайти елемент, символ або змінну у вихідному коді протягом найкоротшого часу;

– крім того, використовуючи режим об'єктива, розробник може ретельно перевірити і налагодити весь вихідний код.

Рефакторинг:

– має перевагу в тому, щоб зробити ефективні та швидкі зміни як локальних, так і глобальних змінних;

– рефакторинг в PyCharm дозволяє розробникам вдосконалити внутрішню структуру, не змінюючи зовнішню продуктивність коду;

– також допомагає розділити більш розширені класи та функції за допомогою методу вилучення.

Допомога для багатьох інших вебтехнологій:

– допомагає розробникам створювати вебзастосунки в Python;

– підтримує популярні вебтехнології, такі як HTML, CSS та JavaScript;

– розробники мають вибір редагування в реальному часі за допомогою цього IDE.

– Одночасно вони можуть переглядати створену / оновлену вебсторінку;

– розробники можуть стежити за змінами безпосередньо у веббраузері;

– PyCharm також підтримує AnglularJS та NodeJS для розробки вебзастосунків.

Підтримка популярних фреймворків Python:

– PyCharm підтримує фреймворки, такі як Django;

– надає функцію автозаповнення щодо параметрів Django;

– допомагає в налагодженні кодів Django;

– також підтримує web2py та Pyramid, інші популярні фреймворки.

Підтримка бібліотек Python:

–PyCharm підтримує бібліотеки Python, такі як Matplotlib, NumPy та Anaconda;

– бібліотеки допомагають будувати проекти Data Science та Machine Learning;

– складається з інтерактивних графіків, які допомагають розробникам розуміти дані;

– здатний інтегруватися з різними інструментами, такими як IPython, Django та Pytest. Ця інтеграція допомагає впроваджувати унікальні рішення.

Під час розробки системи була використана версія PyCharm 2019.3.3.

Для розробки системи було використано декілька бібліотек Python, а саме Matplotlib, Matplotlib.pyplot, Matplotlib.lines, Matplotlib.path – для візуалізації результату та Math – для обчислення формул.

Matplotlib – це бібліотека для створення статичних, анімованих та інтерактивних візуалізацій на Python, яка була розроблена Джоном Хантером, який разом із численними учасниками вкладав безмежну кількість часу та зусиль, щоб тисячі вчених та звичайних людей використовували її у своїх цілях.

Matplotlib створює фігури у різних форматах та інтерактивних середовищах на різних платформах. Matplotlib може використовуватися в скриптах Python, оболонці

Python та IPython, серверах вебзастосунків та різних інструментах графічного інтерфейсу користувача.

Matplotlib має багато переваг над іншими схожими бібліотеками для створення візуалізацій. Ось деякі функції та переваги даної бібліотеки:

– можливість створювати діаграми, графіки, фігури за допомогою лише кількох рядків коду;

– можливість використовувати інтерактивні фігури, які можна збільшувати, зменшувати та змінювати ;

– повний контроль над стилями рядків, властивостями шрифту, властивостями осей;

– можливість експортувати та зберігати в деяких форматах файлів ;

– можливість дослідити спеціалізовані функції, що надаються сторонніми пакетами;

– можливість дізнатися більше про Matplotlib за допомогою багатьох зовнішніх навчальних ресурсів;

– можливість створювати 3D-фігури.

Matplotlib поставляється з декількома додатковими наборами інструментів, включаючи 3D-графіки з *mplot3d*, помічниками осей у *axes_grid1* та помічниками осей у *axartist*.

Велика кількість сторонніх пакетів розширюється та ґрунтується на функціональності Matplotlib, включаючи кілька графічних інтерфейсів вищого рівня (seaborn, HoloViews, ggplot, ...) та два набори інструментів для проектування та картографування (Basemap та Cartopy).

Пакет підтримує багато видів графіків і діаграм:

– графіки (line plot)

– діаграми розкиду (scatter plot);

– стовпчасті діаграми (bar chart) і гістограми (histogram);

– кругові діаграми (pie chart);

– стовбур-лист діаграми (stem plot);

– контурні графіки (contour plot);

– поля градієнтів (quiver);

– спектральні діаграми (spectrogram).

Matplotlib складається з безлічі модулів. Модулі наповнені різними класами та функціями, які ієрархічно пов’язані. Однією з найважливіших особливостей matplotlib, яку я хочу підкреслити, і яку, на мою думку, matplotlib полегшує створення чисел для наукових публікацій, це те, що всі аспекти чисел будуть

управління програмним забезпеченням. Це важливо для відтворюваності та простоти, коли вам потрібно змінити форму за допомогою оновлених даних або змінити зовнішній вигляд.

У дипломному проекті також використовується один із модулів бібліотеки Matplolib під назвою Pyplot. Pyplot — це модуль Matplotlib, який надає інтерфейс, подібний до MATLAB. Matplotlib розроблений так само, як MATLAB, з можливістю використовувати Python і скористатися перевагами вільності та відкритості. Кожна функція труби вносить певні зміни форми: наприклад, вона створює фігуру, створює графічну область фігури, малює деякі лінії в області графіка, прикрашає область мітками тощо. . У Pyplot ви можете створити графік, діаграму, гістограму, 3D графік, зображення, контур і полярність. Pyplot дозволяє набагато зручніше розробляти систему і вимагає менших рядків для створення графіків:

```
1  import matplotlib.pyplot as plt
2
3  plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
4  plt.axis([0, 6, 0, 20])
5  plt.show()
6
```

Рисунок 2.2 – Код для побудови графіка

Результат програми:

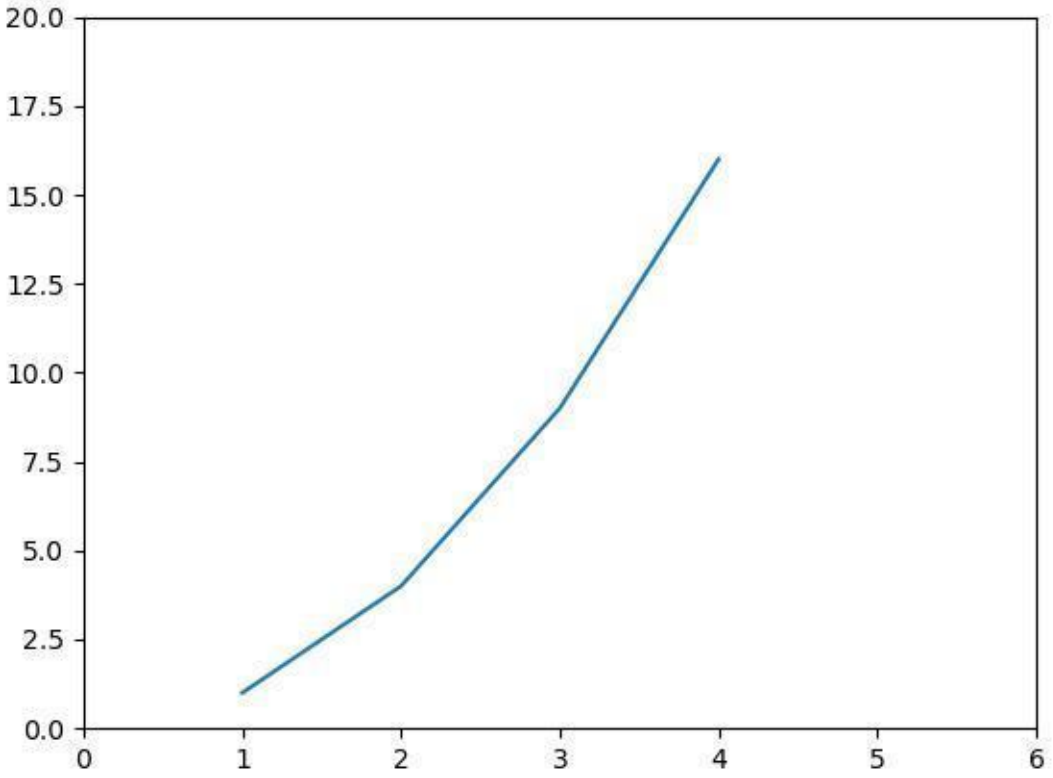

Рисунок 2.3 – Результат програми

Функція графіку позначає *x*-координати (1, 2, 3, 4) та *y*-координати (1, 4, 9, 16) у лінійному графіку із заданими масштабами. Ця функція приймає параметри, які дозволяють нам встановлювати масштаби осей і форматувати графіки. Ці параметри згадані нижче:

– *plot(x, y)*: графік *x* і *y*, використовуючи стиль та колір лінії за замовчуванням;

– *plot.axis([xmin, xmax, ymin, ymax])*: масштабує вісь *x* і вісь *y* від мінімальних до максимальних значень;

– *plot.(x, y, color='green', marker='o', linestyle='dashed', linewidth=2, markersize=12)*: координати *x* і *y* позначаються круговими маркерами розміром 12 і зеленою лінією;

– *plot.xlabel('X-axis')*: називає вісь *x*;

– *plot.ylabel('Y-axis')*: називає вісь *y*.

Для наступного прикладу ми будемо використовувати набори даних про споживання електроенергії Індії та Бангладеш. Тут ми використовуємо Загальнодоступні дані Google як джерело даних. Код прикладу:

```
import matplotlib.pyplot as plt

year = [1972, 1982, 1992, 2002, 2012]
e_india = [100.6, 158.61, 305.54, 394.96, 724.79]
e_bangladesh = [10.5, 25.21, 58.65, 119.27, 274.87]

plt.plot(year, e_india, color='orange',
         label='India')
plt.plot(year, e_bangladesh, color='g',
         label='Bangladesh')

plt.xlabel('Years')
plt.ylabel('Power consumption in kWh')

plt.title('Electricity consumption per capita\
of India and Bangladesh')
plt.legend()
plt.show()
```

Рисунок 2.4 – Код для побудови графіку споживання електроенергії

Результат програми:

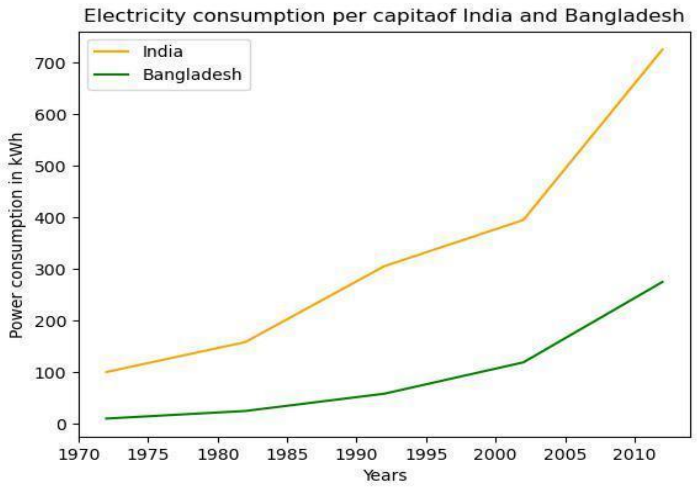

Рисунок 2.5 – Результат виконаного коду

Тому створити зображення matplolib досить просто і легко. Написати зразок коду зі знанням процедур у цій бібліотеці набагато швидше, ніж за допомогою сторонніх служб.

Бібліотека Math також використовується для обчислення геометричних формул. Математичний модуль Python є важливим компонентом, призначеним для роботи з математичними операціями. Постачається зі стандартною версією Python. Більшість функцій математичного модуля - це тонкі обгортки навколо математичних функцій платформи C. Оскільки основні його функції записані в CPython, математичний модуль ефективний і відповідає стандарту C. Модуль математики Python дає можливість виконувати загальні та корисні математичні обчислення у вашій програмі. Ось кілька практичних застосувань для математичного модуля:

– обчислення комбінацій та перестановок за допомогою факторіалів;

– обчислення висоти полюса за допомогою тригонометричних функцій;

– обчислення радіоактивного розпаду за допомогою експоненціальної функції;

– обчислення кривої підвісного моста за допомогою гіперболічних функцій;

– розв’язування квадратичних рівнянь;

– моделювання періодичних функцій, таких як звукові та світлові хвилі, використовуючи тригонометричні функції.

Для обчислення системи рівнянь був використан пакет NumPy. NumPy - бібліотека, яка використовується для роботи з масивами. Він також має функції для роботи в області лінійної алгебри, перетворення Фур'є та матриць. NumPy був створений у 2005 році Тревісом Оліфантом. Це проект з відкритим кодом і безкоштовний для використання. NumPy — це абревіатура від Numerical Python. Навіщо використовувати NumPy? У Python у нас є списки, які обробляють цілі масиви, але їх обробка повільна. NumPy прагне створити об’єкт масиву в 50 разів швидше, ніж традиційний Python. Об’єкт масиву в NumPy називається ndarray, він надає багато допоміжних функцій, завдяки чому дуже легко працювати з ndarray. Поля частіше використовуються в науці даних, де

Швидкість і ресурси дуже важливі. Наука про дані: це область інформатики, де ми досліджуємо, як ви зберігаєте, використовуєте та аналізуєте дані для отримання інформації. Чому NumPy швидший за списки? NumP зберігаються в енергонезалежній пам’яті, на відміну від списків, що дозволяє процесам отримувати доступ до них і ефективно керувати ними. Цю характеристику інформатики називають точкою відліку. Це основна причина, чому NumPy швидше, ніж списки. Він також оптимізований для роботи з останньою архітектурою процесора. NumPy — це бібліотека Python і має функції, написані на Python, але більшість функцій, які вимагають швидких обчислень, написані на C або C ++.

Спосіб зміни координат в точках представлено в роботі, яка полягає в наступному.

2.4.4 Полікоординатні перетворення

Визначення. Лінійною політканиною розміру $\boldsymbol{p}$ у просторі R^2 будемо називати сукупність $\boldsymbol{p}$ *сімей* прямих виду (див. рис. 3.5-а при p=5):

$$\beta_i = a_i x + b_i y + c_i,\ i = 1, 2, \ldots, p \geq 3\ . \qquad (3.1)$$

Нехай на площині в декартовій системі координат *xOy* задана політканина (***p**-тканина*) за допомогою $\boldsymbol{p}$ лінійних функцій-координат (3.1). Перетворимо цю ***p**-тканину*, тобто змінимо деяким чином положення координатних функцій. Нова ***p**-тканина* буде визначена новою системою рівнянь (3.1):

$$\varphi_i = a_i x + b_i y + c_i, i = 1, 2, \ldots, p \geq 3. \qquad (3.2)$$

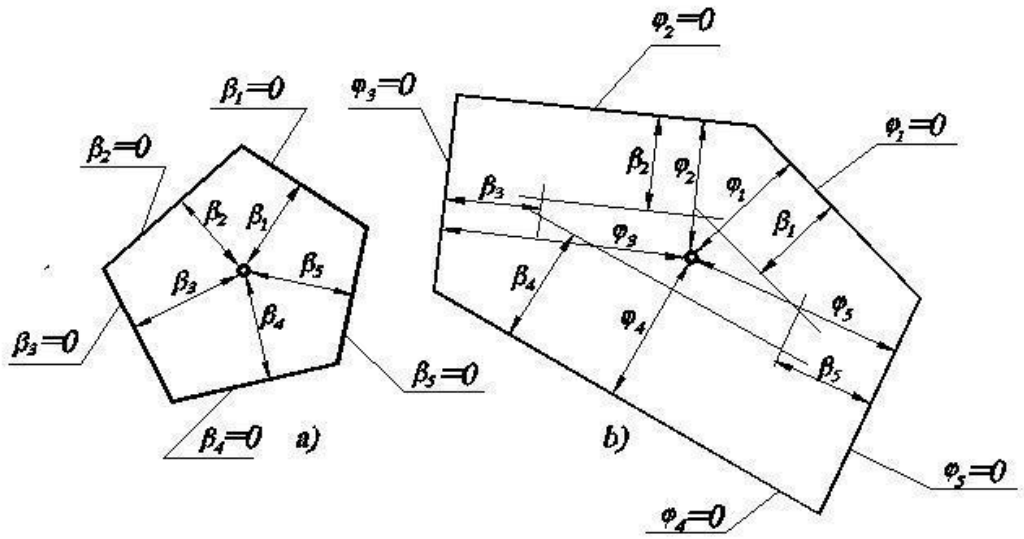

Рис. 3.5. Політканинні перетворення точки

Положення будь-якої точки площини в попередній ***p****-тканині визначається* координатами *(x,y)* – в системі координат *xOy* – або також сукупністю координат і, *i=1,2,...,p≥3.* Аналогічно, після перетворення політканини – *(x',y')* і φ*i, i=1,2,...,p≥3*. Якщо розглядати конкретну точку, то її політканинні координати до перетворення і після не співпадуть тобто $\beta_i \neq \varphi_i$, i=1,2,...,p≥3. Якщо ж прийняти протилежне, тобто намагатись побудувати точку в перетвореній *p*-тканині при умові $\beta_i = \varphi_i$, i=1,2,...,p≥ 3, то замість точки отримаємо багатокутник, конфігурація якого однозначно визначається орієнтацією координатних функцій політканини (3.2). Таким чином, виникає необхідність в тому, щоб отримати однозначний розв'язок задачі політканинного перетворення площини, тобто встановлення функціональної взаємозв'язку між координатами β_i і ω_i , ω_i=1,2,...,p≥3 обох політканин.

Приймаючи до уваги те, що координата точки в політканині є аналогом її віддаленості від відповідної координатної лінії, можна записати:

$$\varphi_i = \omega_i \beta_i, i = 1, 2, \ldots, p \geq 3. \qquad (3.3)$$

Таким чином, (3.2) буде виглядати:

$$\omega_i \beta_i = a_i x + b_i y + c_i, i = 1, 2, \ldots, p \geq 3. \quad (3.4)$$

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

3.1 Користувацький інтерфейс

Пакет appjar використовується для покращення інтерфейсу. Appjar – це найпростіший спосіб створення графічних інтерфейсів на Python. Майбутні віджети використовуються для фіксації взаємодії з користувачем шляхом натискання, введення або перетягування. Зазвичай вони виконують три функції:

– ADD - створює віджет;

– GET - отримує результат / стан віджета;

– SET - змінює те, що знаходиться у віджеті.

Записи використовуються для фіксації набраного вводу від користувача. Існує п'ять видів записів:

– NumericEntry - це дозволяє вводити лише числа;

– SecretEntry – з’являться зірки замість введених букв - корисно для збору паролів;

– AutoEntry - має список слів для автоматичного заповнення;

– ValidationEntry - може бути встановлено як дійсне / недійсне / очікування -

зафарбує межу зеленим / червоним / чорним та покаже ✔ / ✖ / ★;

– OpenEntry / SaveEntry / DirectoryEntry - забезпечує кнопку для вибору файлу каталогу та автоматично заповнює запис.

Щоб створити застосунок appJar потрібно імпортувати gui з бібліотеки appJar та створіть змінну gui(рис. 3.6). Для цього треба додати наступний рядок на початку файлу вихідного коду:

```
1  # import the library
2  from appJar import gui
3
4  # let app be name of gui variable
5  app = gui()
```

Рисунок 3.1 – Код для створення додатку

За допомогою змінної програми треба налаштувати зовнішній вигляд програми та логіку кожного віджета. Наприклад, тут створюється вікно, яке відображає «Hello World» за допомогою змінної програми.

```
7  app.addLabel("title", " Hello World! ")
8
9  app.setLabelBg("title", "blue")
```

Рисунок 3.2 – Код для створення вікна

Нарешті, потрібно запустити застосунок, додавши у свій код наступну команду:

```
11 app.go()
12
```

Рисунок 3.3 – Команда app.go()

Скомпілюємо даний код та отримаємо результат:

Рисунок 3.4 – Результат програми

Кнопка, яку можна натиснути, викликає функцію. Це ключ до запуску інтерактивної програми. GUI циклічно чекає, коли щось станеться. Натискання кнопок є класичним способом взаємодії з графічним інтерфейсом користувача.

```
1   from appJar import gui
2
3   def press(btn):
4       print(btn)
5
6   app=gui()
7   app.addButton("One", press)
8   app.addButton("Two", press)
9   app.addButton("Three", press)
10  app.go()
```

Рисунок 3.5 – Код для створення кнопки

Коли ви запускаєте код, ви отримуєте меню з трьома кнопками: одна, дві, три. Після натискання однієї з кнопок активується функція натискання, тобто відображається назва натиснутої користувачем кнопки.

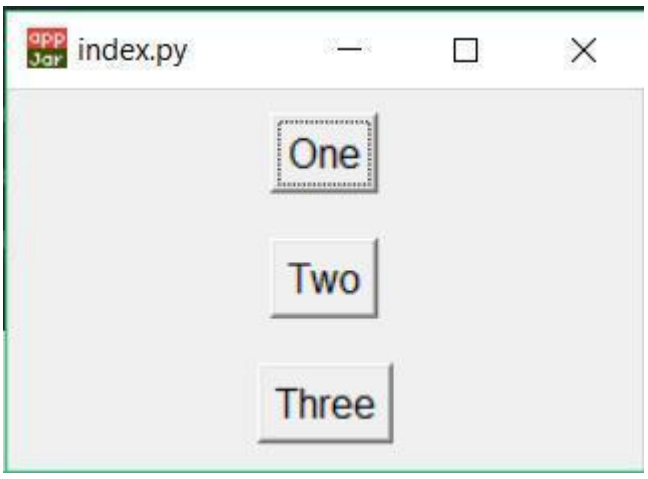

Рисунок 3.6 – Результат програми

Щоразу, коли функція викликається в графічному інтерфейсі, заголовок віджета, який вона викликає, передається як параметр. Тому кілька віджетів можуть

використовувати одну функцію, але ви можете виконувати різні дії залежно від імені, переданого як параметр.

Мітки використовуються для відображення тексту в графічному інтерфейсі. Це добре для заголовків у верхній частині графічного інтерфейсу, який зазвичай складається з кількох стовпців. Вони дуже корисні для записів і атак, щоб пояснити їх призначення. А в GUI дуже корисно бачити результати дії.

```
1  from appJar import gui
2
3  app = gui()
4
5  app.addLabel("l1", "Label 1")
6  app.addLabel("l2", "Label 2")
7  app.addLabel("l3", "Label 3")
8  app.addLabel("l4", "Label 4")
9  # common set functions
10 app.setLabelBg("l1", "red")
11 app.setLabelBg("l2", "yellow")
12 app.setLabelBg("l3", "purple")
13 app.setLabelBg("l4", "orange")
14
15 app.go()
```

Рисунок 3.7 – Команда app.go()

Результат програми:

Рисунок 3.8 – Результат програми

3.2 Програмна реалізація

Розроблена система має мати такий функціонал:

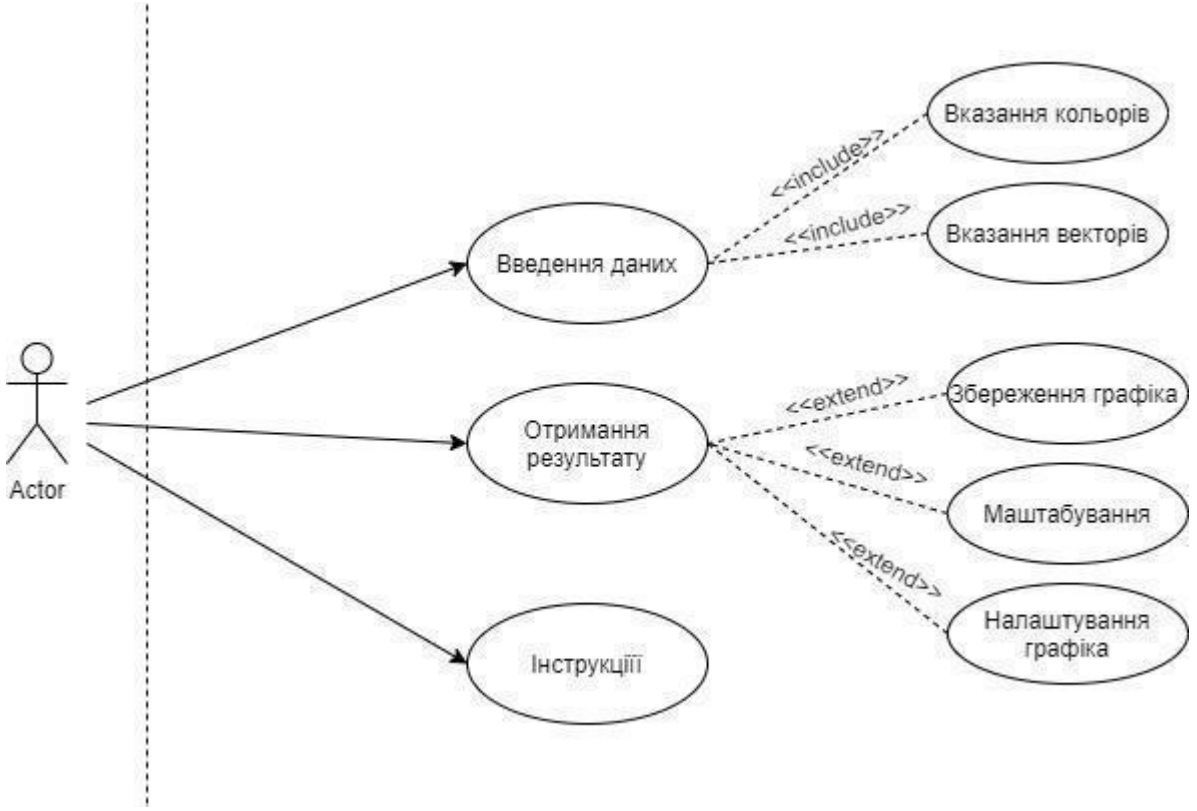

Рисунок 3.9 – Функціонал програмного модуля

Після розробки системи спочатку додаються бібліотеки, описані в попередньому розділі. Імпорт дозволяє коду Python в одному модулі отримати доступ до коду в іншому модулі. Це означає, що коли модуль імпортується, Python виконує весь код у файлі.

Крім того, такі функції, як *importlib.import_module()* і вбудований *import()*, можна використовувати для завантаження та монтування пакетів або бібліотек.

Імпортна декларація поєднує обидві операції; шукає названий модуль, а потім прив’язує результат цього пошуку до імені локальної області. Пошук оператора

імпорту визначається як виклик функції *import()* із відповідними аргументами. Повернене імпорту значення використовується для виконання операції зв’язування від імені оператора імпорту.

Прямий виклик *import()* просто знаходить модуль і, якщо він його знаходить, створює модуль. Хоча деякі побічні ефекти можуть виникати, наприклад, імпортування старих пакетів та оновлення різних кешів (включаючи *sys.modules*), лише оператор імпорту створює ім’я, яке пов’язує операцію.

Коли виконується команда імпорту, викликається вбудована функція імпорту за замовчуванням. Інші механізми, що викликають систему імпорту (наприклад, *importlib.import_module()*), можуть обійти *import ()* і використовувати власні рішення для виконання семантики імпорту.

Коли вперше імпортується модуль Python, він шукає модуль, а коли його знаходить, створює модульний об’єкт, який ініціалізує його. Якщо названий модуль не знайдено, виникає помилка *ModuleNotFoundError*. Python реалізує різні стратегії пошуку для іменованого модуля під час виклику імпорту. Ці стратегії можна модифікувати та розширювати різними способами, тому що конструкції іноді використовуються для полегшення використання бібліотек. Це означає, що замість імені бібліотеки, коли використовується, пишеться не повне ім'я, а ім'я, яке має дати розробник цієї бібліотеки. У моєму випадку я змінив назву математичного пакета на літеру *m* і *matplotlib.pyplot на plt* (рисунок 3.10).

Рисунок 3.10 – Використання *import … as …*

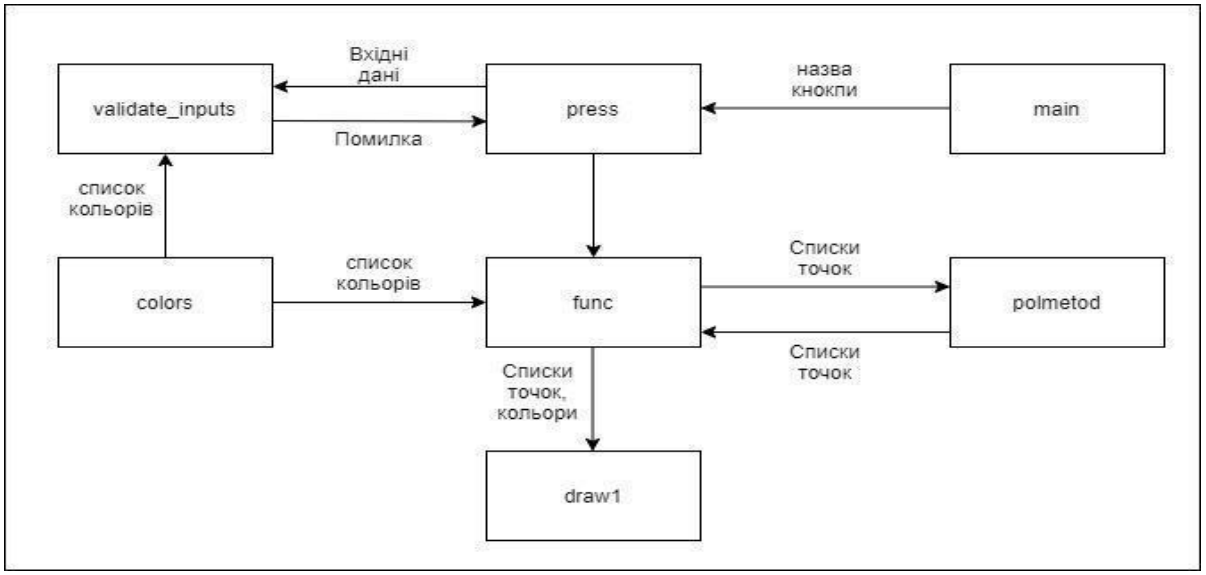

Рисунок 3.11 – Діаграма

Після підключення пакетів йдуть функції, в яких створюється інтерфейс, будується графік та відбуваються обчислення. Функція – це блок коду, який запускається лише тоді, коли він викликається. Можна передавати дані, тобто параметри, у функції. Також функція може повертати дані.

В Python функція визначається за допомогою ключового слова def (рисунок 3.12).

```
1    def my_function():
2      print("Hello from a function")
```

Рисунок 3.12 – Використання *def*

Щоб викликати функцію потрібно використовувати ім'я функції з круглими дужками:

```
1    def my_function():
2      print("Hello from a function")
3
4    my_function()
```

Рисунок 3.13 – Виклик функції

Інформація може передаватися у функції як аргументи. Аргументи задаються після назви функції, всередині дужок. Можна додати скільки завгодно аргументів, просто розділивши їх комою.

Останніми рядками коду є виклик головної функції main. Основна функція - точка входу будь-якої програми. Але інтерпретатор python виконує код вихідного файлу послідовно і не викликає жодного методу, якщо він не входить до коду. Але якщо це безпосередньо частина коду, він буде виконуватися, коли файл імпортується як модуль.

Існує спеціальна методика визначення основного методу в програмі python, щоб він виконувався лише тоді, коли програма запускається безпосередньо, а не виконується при імпорті в якості модуля. Приклад визначання головної функції python у простій програмі.

Рисунок 3.14 – Визначення головної функції

Коли програма Python виконується, інтерпретатор починає виконувати код всередині неї. Він також встановлює кілька неявних змінних значень, одне з них - *__name__*, значення якого встановлено як *__main__*. Для основної функції python, потрібно визначити функцію, а потім використовувати, якщо *__name__ == '__main__'*, умову для виконання цієї функції. Якщо вихідний файл Python імпортується як модуль, інтерпретатор встановлює значення *__name__* на ім'я

модуля, тому умова if поверне помилку і основний метод не буде виконаний. Python надає нам можливість зберігати будь-яке ім’я для основного методу, однак найкращою практикою називати його як метод main ().

Для перевірки введення даних користувачем була створена функція, за допомогою якої відбувається валідація полей і виводиться помилка, якщо дані не відповідають дійсності.

```
def validate_inputs(r,r1,color1,color2, num_st):
```

Рисунок 3.15 – Функція validate_inputs

Дана функція має назву *validate_inputs*. Вона приймає п’ять параметрів *r,r1,color1,color2, num_st*, тобто радіус кола, радіус многокутника, колір кола, колір многокутника та кількість кроків відповідно. Користувач може ввести лише числове значення в параметри *r , r1 ma num_st*. Тобто якщо користувач буде вводити у дані поля все, окрім цифр, то ці дані вводитися не будуть. Також є перевірка на введення тільки додатних чисел у параметри *r, r1 ma num_st*, тобто користувач зможе ввести від’ємні числа у строку, але при запуску програми виникне помилка, яка повідомить про те, що потрібно ввести інше число.

За умовою радіус многокутника повинен бути більшим, ніж радіус кола, тому існує перевірка, яка порівнює дані числа та виводить помилку, якщо радіус многокутника менший, ніж радіус кола.

```
178     if r1 <= r:
179         errors = True
180         error_msgs.append("Please enter a valid radius of polygon")
```

Рисунок 3.16 – Перевірка даних

Для параметрів *color* та *color1* також існують перевірки. Для коректного вводу даних параметрів була створена функція colors, яка повертає список усіх кольорів.

```
def colors():
    cnames = ['aliceblue', 'antiquewhite', 'aqua', 'aquamarine', 'azure', 'beige', 'bisque',
              'black', 'blanchedalmond', 'blue', 'blueviolet', 'brown', 'burlywood', 'cadetblue',
              'chartreuse', 'chocolate', 'coral', 'cornflowerblue', 'cornsilk', 'crimson', 'cyan', 'darkblue',
              'darkcyan', 'darkgoldenrod', 'darkgray', 'darkgreen', 'darkkhaki', 'darkmagenta', 'darkolivegreen',
              'darkorange', 'darkorchid', 'darkred', 'darksalmon', 'darkseagreen', 'darkslateblue', 'darkslategray',
              'darkturquoise', 'darkviolet', 'deeppink', 'deepskyblue', 'dimgray', 'dodgerblue', 'firebrick',
              'floralwhite', 'forestgreen', 'fuchsia', 'gainsboro', 'ghostwhite', 'gold', 'goldenrod', 'gray',
              'green', 'greenyellow', 'honeydew', 'hotpink', 'indianred', 'indigo', 'ivory', 'khaki', 'lavender',
              'lavenderblush', 'lawngreen', 'lemonchiffon', 'lightblue', 'lightcoral', 'lightcyan', 'lightgoldenrodyellow',
              'lightgreen', 'lightgray', 'lightpink', 'lightsalmon', 'lightseagreen', 'lightskyblue', 'lightslategray',
              'lightsteelblue', 'lightyellow', 'lime', 'limegreen', 'linen', 'magenta', 'maroon', 'mediumaquamarine', 'mediumblue',
              'mediumorchid', 'mediumpurple', 'mediumseagreen', 'mediumslateblue', 'mediumspringgreen', 'mediumturquoise',
              'mediumvioletred', 'midnightblue', 'mintcream', 'mistyrose', 'moccasin', 'navajowhite', 'navy', 'oldlace', 'olive',
              'olivedrab', 'orange', 'orangered', 'orchid', 'palegoldenrod', 'palegreen', 'paleturquoise', 'palevioletred',
              'papayawhip', 'peachpuff', 'peru', 'pink', 'plum', 'powderblue', 'purple', 'red', 'rosybrown', 'royalblue',
              'saddlebrown', 'salmon', 'sandybrown', 'seagreen', 'seashell', 'sienna', 'silver', 'skyblue', 'slateblue', 'slategray',
              'snow', 'springgreen', 'steelblue', 'tan', 'teal', 'thistle', 'tomato', 'turquoise', 'violet', 'wheat', 'white',
              'whitesmoke', 'yellow', 'yellowgreen']
    return cnames
```

Рисунок 3.17 – Функція, яка повертає список кольорів

За допомогою конструкції *if ... not in ...* я шукаю колір зі списку, який співпадає із введеними даними у параметрах *color* та *color1*. Оцінюється як true, якщо він не знаходить змінної у зазначеній послідовності, а false в іншому випадку. Якщо введеного кольора не існує, то користувач отримує помилку.

Задля швидшого заповнення полей, де потрібно ввести кольори фігур, існує функція автозаповнення, для якої використовується та ж сама функція *colors*, яка повертає список кольорів.

```
words = colors()
app.addAutoEntry("Color of circle: ", words)
app.setAutoEntryNumRows("Color of circle: ", 4)
```

Рисунок 3.18 – Функція автозаповнення

Спочатку повертається список кольорів і записується змінними словами, потім функція addAutoEntry викликається з пакетом Appjar, щоб уточнити список кольорів. Наступна функція визначає кількість стовпців, які відображаються користувачеві під час введення даних у поля. Це дозволяє користувачеві швидко вводити дані без будь-яких помилок.

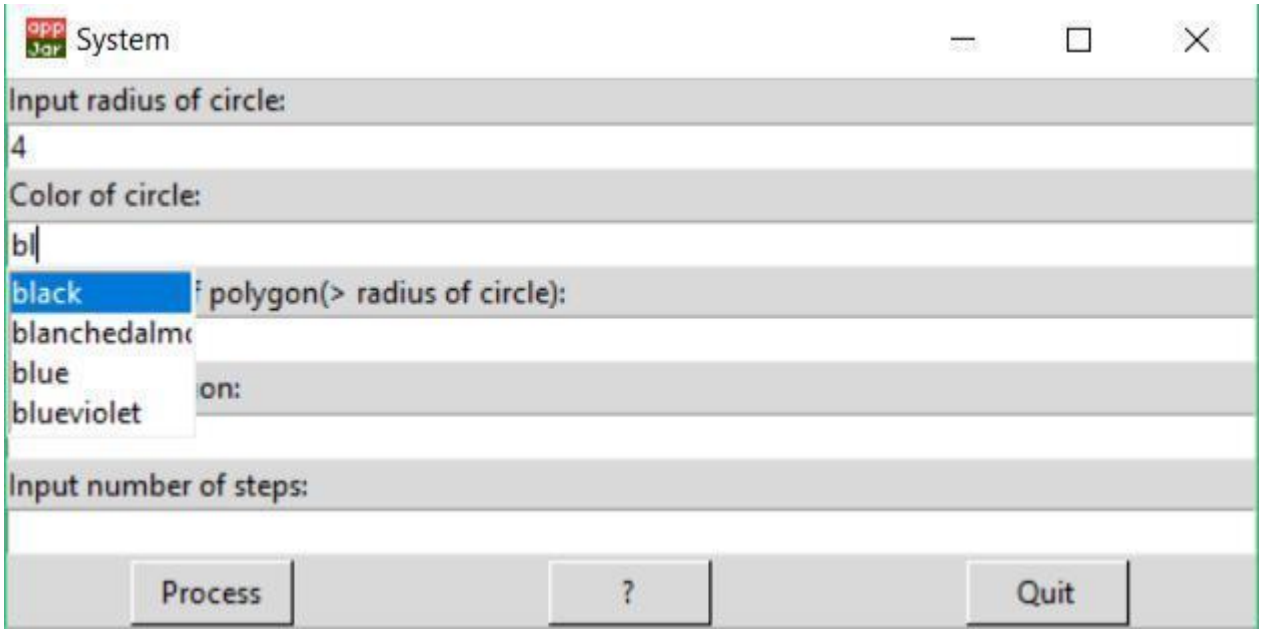

Рисунок 3.19 – Автозаповнення

Якщо користувач не може правильно ввести дані, доступна кнопка «?», яка містить інструкції для успішного запуску програми. Для обробки кнопок виконується функція натискання, яка отримує параметр кнопки. Це означає, що при натисканні однієї з кнопок назва цієї функції змінюється.

Далі оператор if використовується для обробки даного імені та дії, яка виконується залежно від значень змінних під час перевірки статусу.

Натиснувши кнопку Редагувати, програма приймає введені дані та викликає функцію перевірки даних. Якщо виникають якісь помилки, повідомте про них. Якщо помилок не виникає, програма викликає функцію функції, під час якої будуть виконуватися всі обчислення для отримання кінцевого результату. Коли користувач натискає кнопку? , програма викликає .infoBox і повідомляє користувачеві інструкції. В іншому випадку, якщо натиснути кнопку Вийти, програма закриється.

```
def press(button):

    if button == "Process":
        r = app.getEntry('Input radius of circle: ')
        r1 = app.getEntry('Input radius of polygon(> radius of circle): ')
        color1 = app.getEntry("Color of circle: ")

        color2 = app.getEntry("Color of polygon: ")
        num_st = app.getEntry('Input number of steps: ')

        errors, error_msg = validate_inputs(r,r1, color1, color2, num_st)
        if errors:
            app.errorBox("Error", "\n".join(error_msg), parent=None)
        else:
            func(r,r1, color1, color2,num_st)
    elif button == "?":
        app.infoBox("Instructions", "1. Enter radius of circle >0 \n2. Enter color from dropbox \n3. Enter radius of rectangle > radius of circle \
                                    "4. Enter number of ssteps > 0 \n5. Click Process \n6. Click Quit to exit from the program", parent=None)
    else:
        app.stop()
```

Рисунок 3.20 – Визначення функції press

В найголовнішій функції func відбуваються обробка даних, обчислення за формулами та вивід кінцевого результату. Були використані списки,

Список є найбільш універсальним типом даних, доступним у Python, який може бути записаний у вигляді списку розділених комами значень (елементів) між квадратними дужками. Важливим у списку є те, що елементи в списку не повинні бути одного типу.

Створення списку так само просто, як і розміщення різних розділених комами значень між квадратними дужками. Наприклад:

```
# empty list
my_list = []

# list of integers
my_list = [1, 2, 3]

# list with mixed data types
my_list = [1, "Hello", 3.4]
```

Рисунок 3.21 – Код для створення списків

3.3 Робота користувача з програмою

3.3.1 Запуск системи

Для запуску розробленої програми у форматі .py необхідно:

–скачати IDLE з офіційного сайту;

–запустити програму, натиснувши на неї або через командний рядок, вказавши назву файлу;

Для запуску програми у форматі .exe потрібно всього лише два рази клікнути на неї.

3.3.2 Інтерфейс користувача

При запуску системи користувачу випадає меню, в якому потрібно вказати деякі параметри (рисунок 3.22).

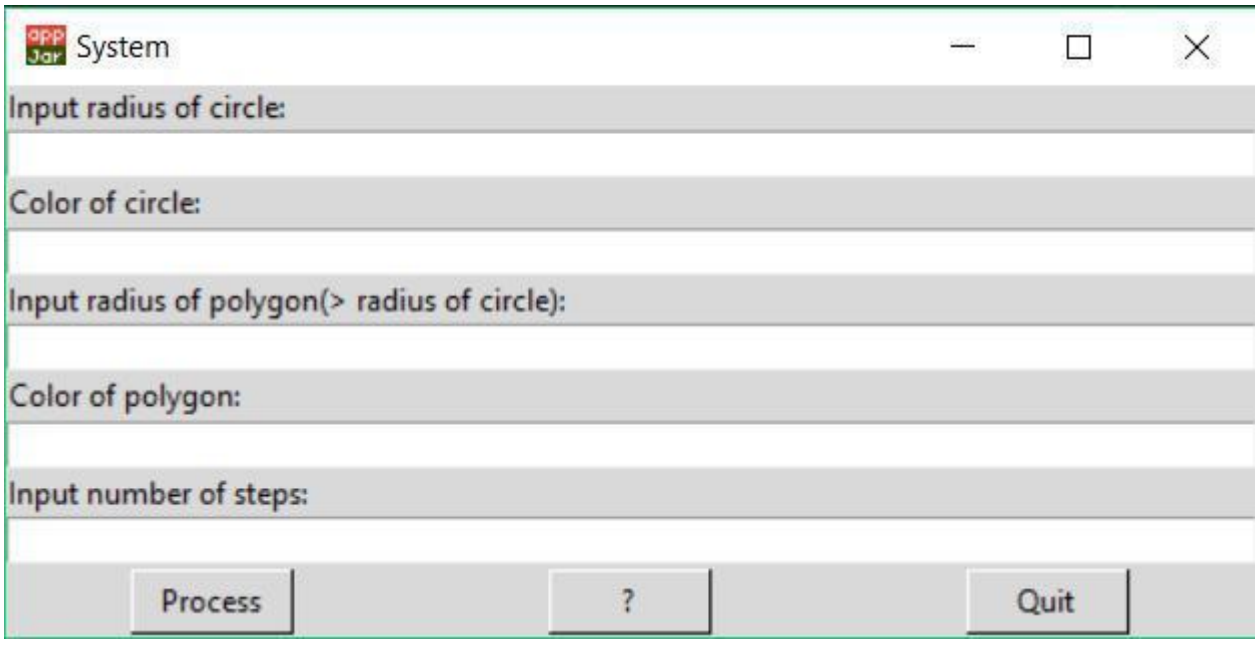

Рисунок 3.22 – Меню

Після вводу параметрів в полях, вікно має буде таким (рисунок 3.23).

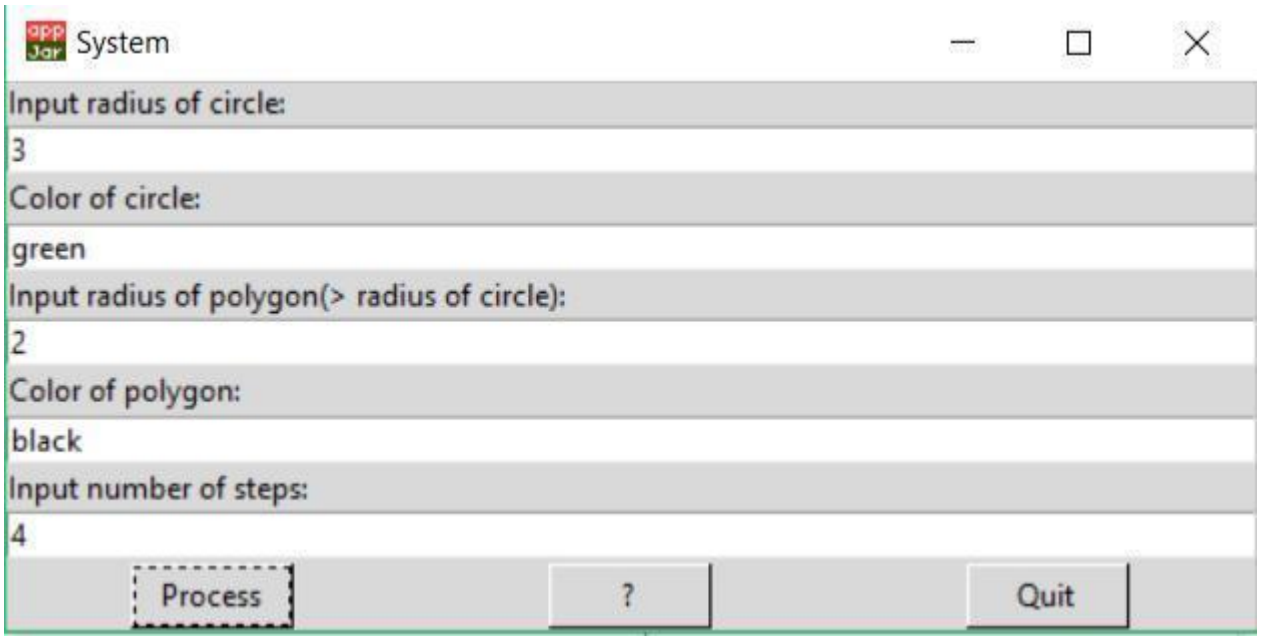

Рисунок 3.23 – вікно з даними у полях

Після того як користувач натисне кнопку Process під формами вводу, у разі успішного вводу даних, відкриється нове вікно з результатом програми (рисунок 3.24).

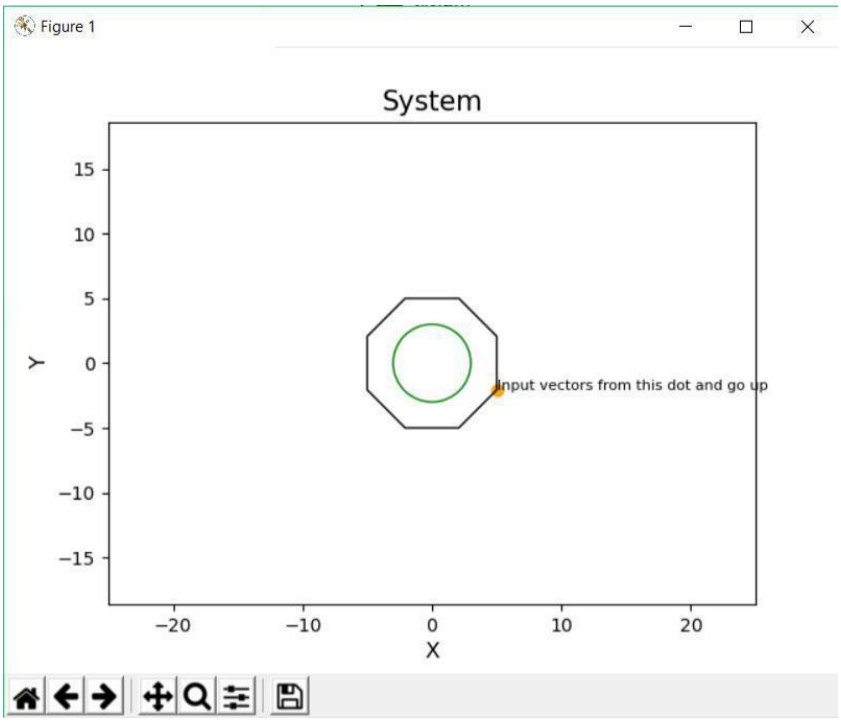

Рисунок 3.24 – Результат програми

У першому полі необхідно ввести радіус кола. Якщо користувач хоче ввести текст у це поле, він не вводиться, тому що в першому полі та в третьому, де потрібно ввести радіус кола та багатокутника, є поле перевірки, тому є лише одне поле. номер. Якщо ви введете дані успішно, програма запуститься.

Треба ввести радіус багатокутника в третє поле. Зазвичай він повинен бути більшим за радіус кола, тому, якщо користувач введе невеликий радіус, виявляється помилка (рисунок 3.25). Рисунок 3.26 – Помилка введення кольорів кола та багатокутників

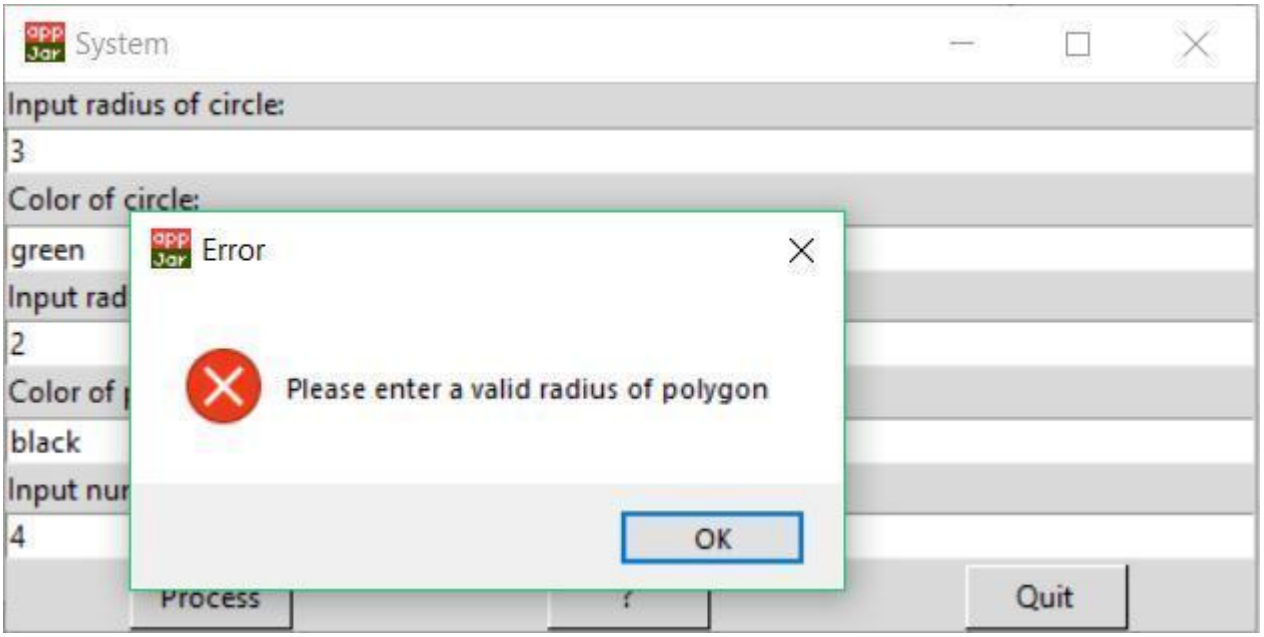

Рисунок 3.25 – Помилка введення радіусу багатокутника

У другому полі та у четвертому, де потрібно ввести колір кола та багатокутника, є перевірка на введення даних. Якщо буде введено будь-який текст або числа, то програма при натиску кнопки Process, повідомить про помилку(рисунок 3.26). В дані поля потрібно вводити назву кольора на англійській мові. Якщо введені дані будуть правильними, то програма буде працювати.

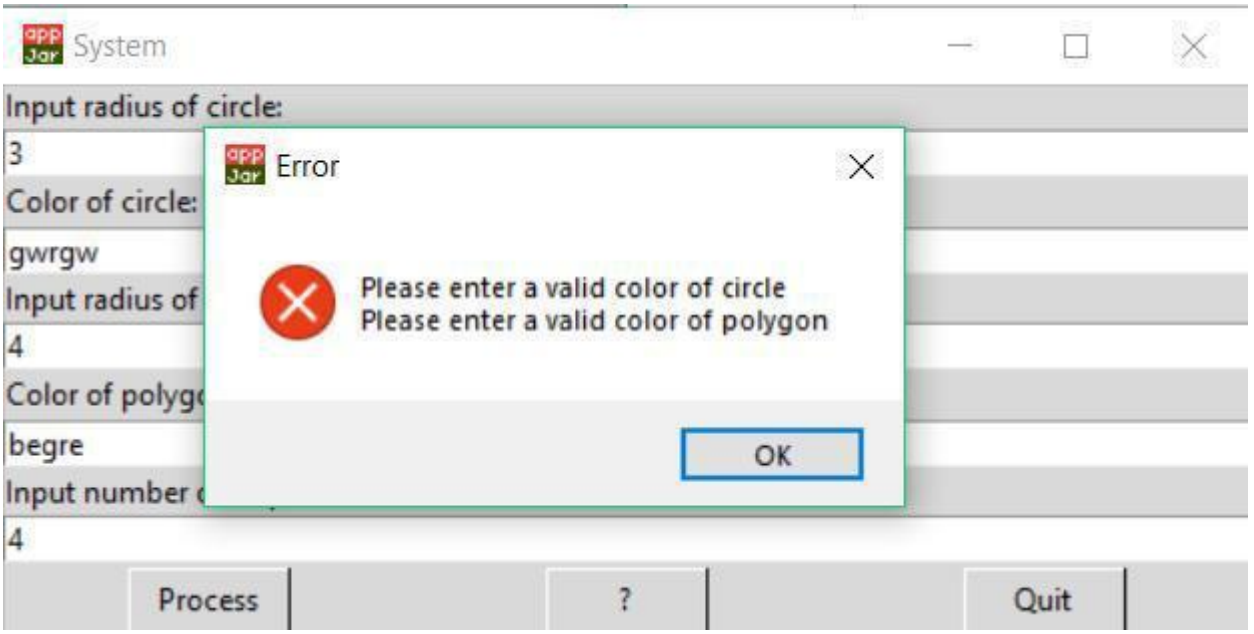

Рисунок 3.26 – Помилка введення кольорів кола та багатокутника

Після того, як користувач успішно введе дані та натисне кнопку Обробка, з’явиться нове вікно з побудованим колом і багатокутником (рис. 3.27). У цьому вікні вам потрібно буде ввести багатокутні вектори, починаючи з вибраної точки і збільшуючи числа. Користувач повинен натиснути на графік біля введених точок. Потім на графіку розміщуються червоні точки (рисунок 3.28). Від точок многокутника до зазначених точок проводять вектори (рисунок 3.29).

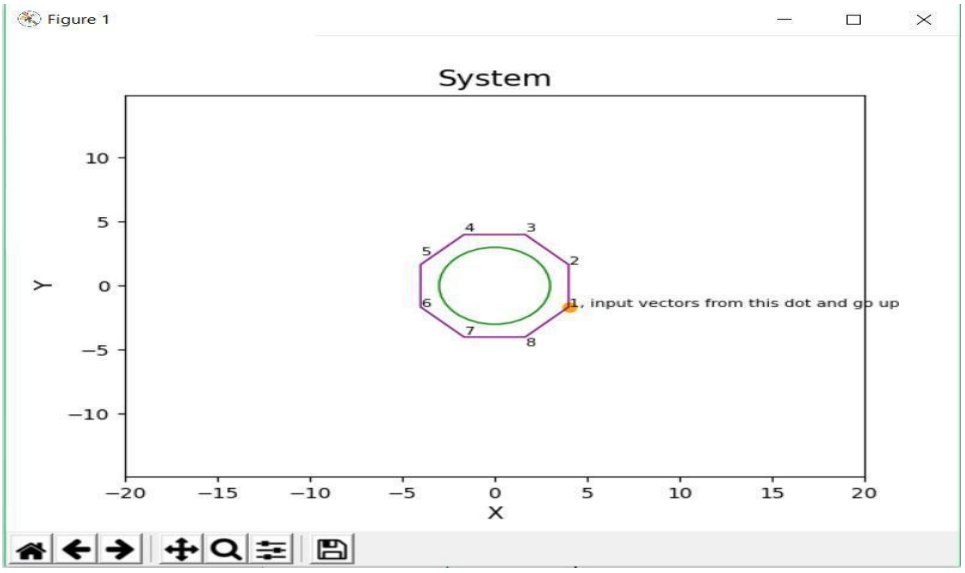

Рисунок 3.27 – Вікно з побудованим колом та багатокутником

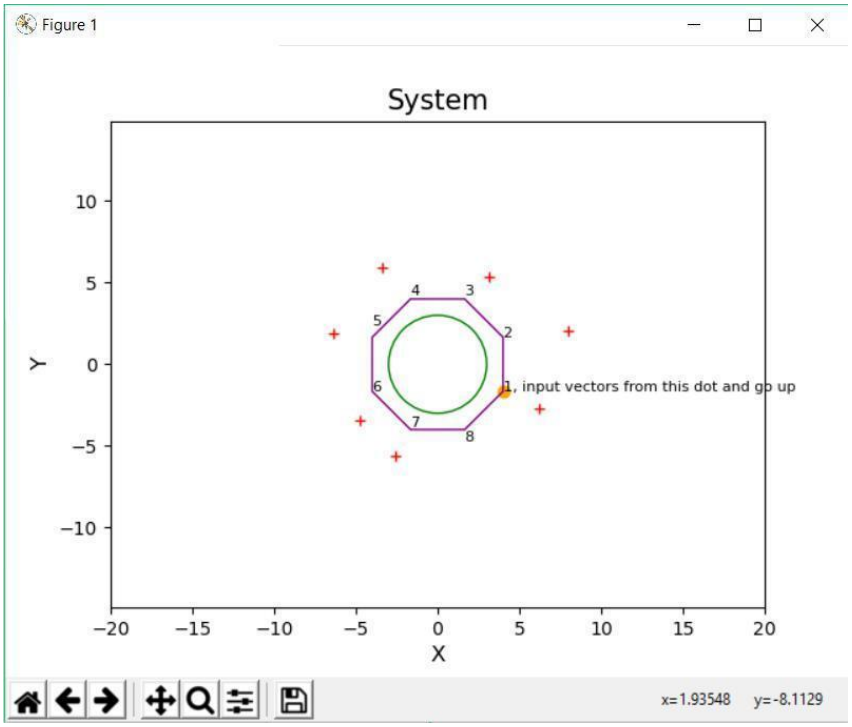

Рисунок 3.28 – Введення точок для побудови векторів

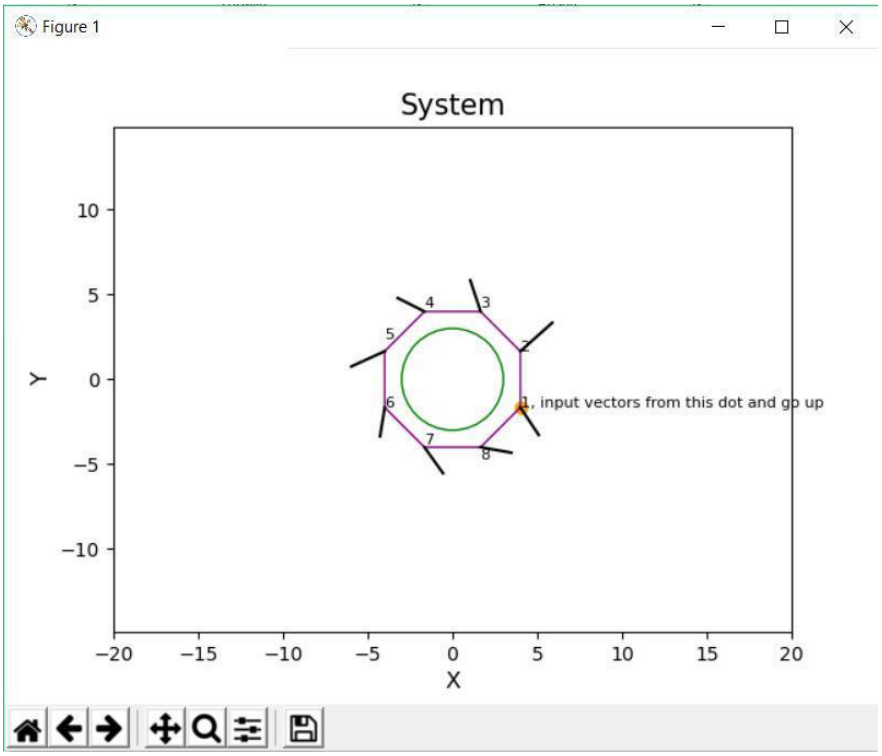

Рисунок 3.29 – побудовані вектори

Після успішного введення векторів, на даному графіку з’явиться кінцевий результат(рис.3.30).

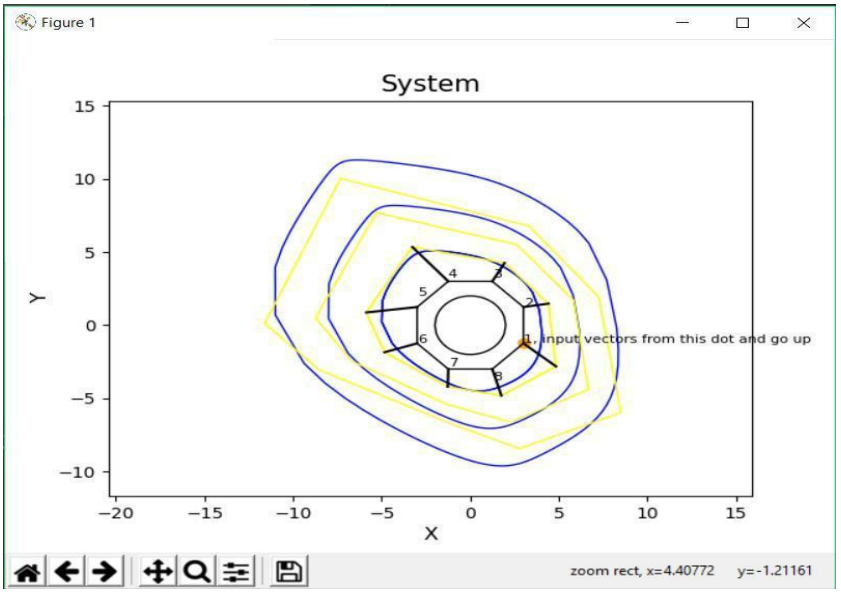

Рисунок 3.30 – Результат програми

Якщо користувач не зможе правильно запустити програму, то є кнопка зі знаком питання(рис.3.31). Натиснувши на неї, відкриється вікно з усіми інструкціями для успішного запуску програми(рис. 3.32).

Рисунок 3.32 – Кнопка з інструкціями

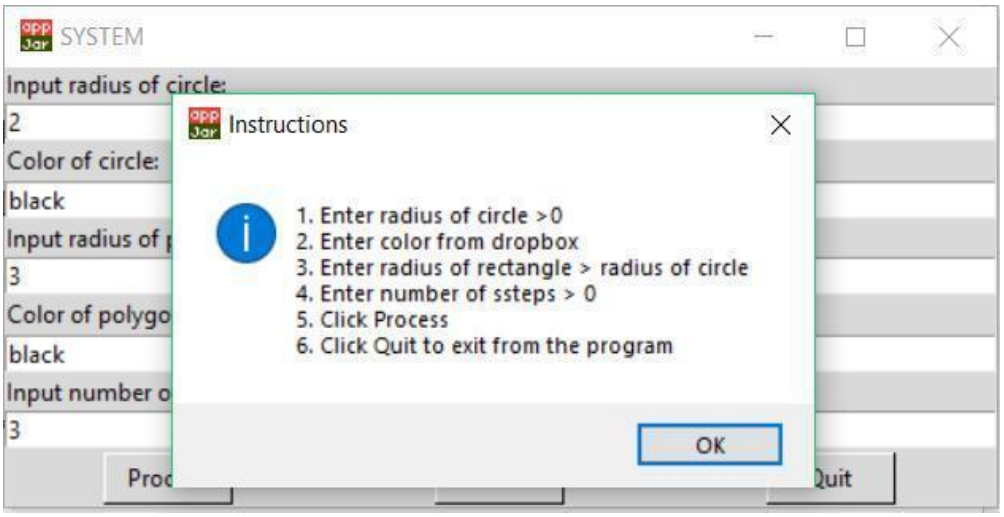

Рисунок 3.33 – Вікно з інструкціями

ВИСНОВКИ

У рамках кваліфікаційної роботи було покращено навички розробки програмного забезпечення за допомогою мови програмування Python та середовища розробки програмного забезпечення PyCharm.

Отримав досвід використання бібліотек Python. Вони допомагають легко і швидко створювати складні графіки. Також бібліотеки значно полегшують створення зручного для користувача інтерфейсу.

Результатом став програмний продукт, який завдяки гнучкості мови програмування Python та зрозумілому інтерфейсу легко досягає цілей кваліфікаційної роботи.

У процесі розробки я поглибив знання бібліотек Mathplotlib, Math та NymPy. Вони є дуже актуальними та затребуваними в розробці сучасних застосунків. Також проведено аналіз та інтегрування в програму методу полікоординатних перетворень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Python, R. (2019). Python. *Python Releases for Windows*, *24*.
2. vanRossum, G. (1995). Python reference manual. *Department of Computer Science [CS]*, (R 9525).
3. Islam, Q. N. (2015). *Mastering PyCharm*. Packt Publishing Ltd.
4. Ari, N., & Ustazhanov, M. (2014, September). Matplotlib in python. In *2014 11th International Conference on Electronics, Computer and Computation (ICECCO)* (pp. 1-6). IEEE.
5. Tosi, S. (2009). *Matplotlib for Python developers*. Packt Publishing Ltd.
6. Saha, A. (2015). *Doing Math with Python: Use Programming to Explore Algebra, Statistics, Calculus, and More!*. No Starch Press.
7. Daszkiewicz, B., & Nading, J. Internet of Things Information Display.
8. Oliphant, T. E. (2006). *A guide to NumPy* (Vol. 1, p. 85). USA: Trelgol Publishing.
9. Бадаєв, Ю. І., & Блиндарук, А. О. (2014). Компютерна реалізація проектування криволінійних обводів методом NURBS-технологій вищих порядків. *Сучасні проблеми моделювання*, (2), 3-8.
10. Мокин, В. Б., & Варчук, И. В. (2013). Моделирование распространения загрязняющих веществ в воздухе города с использованием геоинформационных технологий. *Вісник Вінницького політехнічного інституту*, (5), 13-18.
11. Бадаєв Ю.І. Геометричне моделювання криволінійних обводів складних об’єктів.-К.:ТЕХТ,2015,-172с.
12. Бадаєв, Ю. І., & Лагодіна, Л. П. (2014). Дослідження щодо застосування вагових коефіцієнтів в полікоординатних відображеннях. *Водний транспорт*, (2), 140-144.
13. Родионова, Ю. В. (2021). Применение геоинформационных технологий в создании виртуальных локаций. *Творчество и современность*, (2 (15)), 77-80.
14. Бадаєв, Ю. І., & Лагодіна, Л. П. (2013). Властивості полікоординатних відображень за однією координатою. *Водний транспорт*, (1), 162-168.

15. Маккинни, У. (2022). *Python и анализ данных*. Litres.

16. Лагодіна, Л. П., Сілантьєва, Ю. О., Поляков, В. В., & Бадаєв, Ю. І. Розроблення програмного забезпечення моделювання розповсюдження екологічних забруднень. In *Наукові праці Третьої міжнар. наук.-практ. конф.«Сучасні тенденції розвитку інформаційних систем і телекомунікаційних технологій», 25–26 січня 2021 р.(Київ, Україна).–К.: НУХТ, 2021.–181 с. У працях конференції наведено доповіді за напрямами:• світові тенденції в розробленні інформаційних систем і* (p. 120).

17. Творошенко, І. С., & Подласенко, Є. П. (2019). Дослідження методу розпізнавання геоінформаційних ситуацій в системах моніторингу територій.

18. Tvoroshenko, I. S., & Kramarenko, O. O. (2019). Software determination of the optimal route by geoinformation technologies. *Radio Electronics, Computer Science, Control*, (3), 131-142.

19. Мокін, В. Б., & Варчук, І. В. (2013). Моделювання поширення забруднювальних речовин у повітрі міста з використанням геоінформаційних технологій. *Вісник Вінницького політехнічного інституту*, 13-18.

20. Бігвава, Н. М. (2018). Предмет забруднення або псування земель в Україні. *Молодий вчений*, (4 (2)), 655-658.

21. Кисельов, Ю., & Шутак, К. (2019). ГОСПОДАРСЬКА ДІЯЛЬНІСТЬ ЯК ЧИННИК ЗАБРУДНЕННЯ ВОДНИХ ОБ'ЄКТІВ УКРАЇНИ. *Молодий вчений*, (2 (66)), 333-336.

22. Верес, М. (2020). Сучасний стан українських земель та їх екологічна безпека.

23. Крутько, В. В. Налагодження системи спостережень і контролю за забрудненням водних об'єктів.

24. Хомутова, А. Ю. Проблема забруднення водних ресурсів України.

25. Lemenkova, P. (2020). Python Libraries Matplotlib, Seaborn and Pandas for Visualization Geo-spatial Datasets Generated by QGIS. *Analele stiintifice ale Universitatii" Alexandru Ioan Cuza" din Iasi-seria Geografie*, *64*(1), 13-32.

26. Roubeyrie, L., & Celles, S. (2018). Windrose: A Python Matplotlib, Numpy library to manage wind and pollution data, draw windrose. *Journal of Open Source Software*, *3*(29), 268.

27. Никонорова, Л. И., Тимофеев, М. Г., & Кузнецова, А. П. (2019). Python как современный язык программирования. *Наука и образование*, *2*(2).

28. Чорна, А. В., & Таганова, Д. В. (2018). Використання алгоритмів розв'язування графічних задач засобами Python. *Інформаційні технології в освіті та науці: зб. наук. пр.*, (10), 310-315.

29. Стряпков, А. В., Демченко, С. А., & Прянишникова, Л. И. (2019, December). Визуализация графиков с помощью библиотек Python. In *Студенческий научный форум: материалы Международной студенческой научной конференции, Москва* (Vol. 1, pp. 67-70).