

ТЕОРЕТИЧНА ЧАСТИНА З ІНФОРМАТИКИ

Використані технології і архітектура програми

Використані технології

У процесі створення даного проєкту було використано сучасні підходи до програмування та технології, які забезпечують зручність і ефективність розробки. Основною мовою програмування обрано **Python**, що є універсальним інструментом для реалізації математичних розрахунків, побудови графіків і створення графічного інтерфейсу. Python має широкий набір бібліотек і модулів, які суттєво полегшують роботу з математичними операціями, візуалізацією даних та розробкою користувацького інтерфейсу. Також причиною обрання саме цієї мови стала зацікавленість авторки в вивченні математичних бібліотек Python, зважаючи на те, що вона опановує цю мову вже 4 роки і прагне продовжувати розвиток у нових її напрямках.

Віртуальне середовище Python також було використано у роботі для ізоляції проєкту і керування залежностями, що забезпечує стабільність роботи застосунку.

Основні бібліотеки, які було використано у ході розробки додатку:

- **NumPy** — це основний пакет для наукових обчислень на Python. Це бібліотека Python, яка надає об'єкт багатовимірного масиву, різні похідні об'єкти (такі як замасковані масиви та матриці), а також набір процедур для швидких операцій над масивами, включаючи математичні, логічні, маніпуляції формою, сортування, вибір, введення/виведення, дискретне перетворення Фур'є, базова лінійна алгебра, основні статистичні операції, випадкове моделювання та багато іншого більше.
- **SymPy** — це бібліотека Python для символічної математики.
- **Matplotlib** — це комплексна бібліотека для створення статичних, анімованих та інтерактивних візуалізацій.
- **matplotlib.pyplot** — це інтерфейс для matplotlib на основі стану. Він забезпечує неявний, схожий на MATLAB, спосіб побудови. Він також

відкриває фігури на вашому екрані та діє як менеджер графічного інтерфейсу користувача (GUI).

- **CustomTkinter** — це Desktop бібліотека інтерфейсу користувача Python на основі Tkinter, яка надає сучасні та повністю настроювані віджети. За допомогою CustomTkinter ви отримаєте узгоджений вигляд на всіх платформах настільних ПК (Windows, macOS, Linux).
- **Tkinter** — є стандартним Desktop інтерфейсом Python до інструментарію GUI Tcl/Tk. І Tk, і tkinter доступні на більшості платформ Unix, включаючи macOS, а також у системах Windows.
- **PIL** — бібліотека зображень Python додає можливості обробки зображень до інтерпретатора Python. Ця бібліотека забезпечує розширену підтримку форматів файлів, ефективне внутрішнє представлення та досить потужні можливості обробки зображень.
- **Fitz (PyMuPDF)** — це високопродуктивна бібліотека Python для вилучення даних, аналізу, перетворення та обробки документів PDF (та інших).

Архітектура програми

Проект побудовано на основі чітко структурованої архітектури, що дозволяє розділити функціональність за окремими модулями та пакетами, забезпечуючи простоту в обслуговуванні, масштабованість і читабельність коду. Основний пакет програми — **modules**, у якому організовано весь функціонал для виконання задачі.

Поділ коду на модулі та пакети забезпечує:

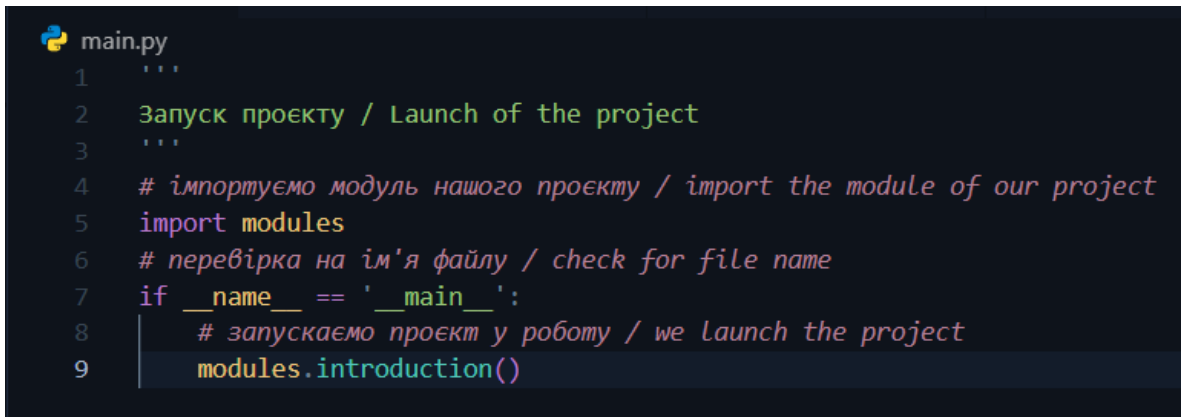
- **Логічну ізоляцію** функцій: кожен компонент відповідає за окремий аспект роботи програми;
- **Масштабованість**: легко додавати нові функції або розширювати існуючі без зміни основної архітектури;
- **Зручність тестування**: кожен модуль можна перевірити окремо, що знижує ризик помилок;

- **Повторне використання коду:** функції, які використовуються в декількох місцях, доступні через окремі модулі;

Основна структура проєкту складається з таких файлів і пакетів:

1. Файл `main.py`

Відповідає за запуск програми та ініціалізацію основного вікна. У ньому підключаються всі головний пакет й забезпечується зв'язок між компонентами:



```
main.py
1  '''
2  Запуск проєкту / Launch of the project
3  '''
4  # імпортуємо модуль нашого проєкту / import the module of our project
5  import modules
6  # перевірка на ім'я файлу / check for file name
7  if __name__ == '__main__':
8      # запускаємо проєкт у роботу / we launch the project
9      modules.introduction()
```

2. Пакет `modules`

У кореневій директорії цього пакета зберігаються:

- **Файли з реалізацією 4 основних вікон** — дозволяють відображати окремі частини функціоналу програми, організуючи її зручний інтерфейс.
- **Файл з константами** — містить глобальні змінні, що використовуються у проєкті. Це спрощує внесення змін у програму, адже всі важливі параметри зосереджені в одному місці. Також створення глобальних змінних саме таким чином допомагає уникнути можливих проблем, помилок і некоректних даних у програмі під час її використання.

3. Пакет style

Призначений для:

- Задавання стилів елементам інтерфейсу customtkinter та tkinter.
 - Забезпечення єдиного дизайну програми, що покращує її візуальну узгодженість.
- Завдяки цьому пакет забезпечується легке оновлення стилів для всього застосування.

4. Пакет main_elements

Містить модулі для створення:

- Основних об'єктів графічного інтерфейсу: вікон, кнопок, чекбоксів, полотен тощо.
 - Компонента FigureCanvasTkAgg з matplotlib.backends.backend_tkagg, який використовується для відображення графіків у вікнах customtkinter.
- Завдяки цьому розділенню полегшується підтримка та модифікація інтерфейсу.

5. Пакет data_calculation

Містить функції для математичних розрахунків і дослідження функцій, зокрема:

- область визначення функції;
- парність чи непарність;
- зростання та спадання;
- локальні максимуми та мінімуми;
- максимальне і мінімальне значення функції;
- нулі функції(x_1, x_2, x_n);
- точки перетину функції з осями абсцис і ординат ($x_0; y_0$);
- проміжок знакосталості;
- точки перегину;
- проміжки опуклості;

- асимптоти;

6. Пакет plotting


Призначений для:

- Побудови графіків функцій та їх дослідження.
- Взаємодії з функціями з пакета data_calculation для відображення результатів дослідження на графіку.
- Інтеграції графіків у GUI через FigureCanvasTkAgg.

Ця структура дозволяє легко додавати нові методи візуалізації.

Компоненти програми взаємодіють через добре структуровану систему імпортів:

- головний пакет **modules** ініціалізує програму, викликаючи модулі та функції з інших пакетів у **__init__.py**;

```
modules >  __init__.py
1  """
2  Ініціалізація усіх файлів до пакету. / Initialization of all files to the package.
3  """
4  from .variables_constants import *
5  # імпортуємо функцію introduction з файлу introduction_window для запуску титульного вікна / import the introduction function from introduction_window
6  from .introduction_window import introduction
7  # імпортуємо з main_window функцію run_main для можливості запуску головного додатку / import the run_main function from main_window
8  from .main_window import run_main, on_close
9  # імпортуємо вікно документації для його роботи і відображення під час використання проекту / import the documentation window
10 from .document_window import *
11 # імпортуємо з plotting усі функції для кнопок або чекбоксів для подальшого використання / import all plotting functions
12 from .plotting import *
13 # імпортуємо з data_calculation усі функції для виконання дослідження графіків функцій / import all data_calculation functions
14 from .data_calculation import *
15 # імпортуємо усі зміни стилів для елементів з файлу style / import all style changes for elements from style
16 from .style import *
17 # імпортуємо функцію functions_window з файлу window_with_fun для запуску вікна для вибору функцій / import the functions_window function from window_with_fun
18 from .window_with_fun import functions_window
19 # імпортуємо вікно помилки для його подальшого відображення при необхідності / import the error window
20 from .error_window import show_error_window
```

- main_elements відповідає за створення елементів інтерфейсу, які використовують функції з **data_calculation** та **plotting** для обчислень і візуалізації;
- **style** забезпечує єдиний дизайн усіх елементів інтерфейсу;
- файл констант дозволяє узгоджено змінювати глобальні налаштування програми;

Вплив структури на проєкт:

- **покращена організація коду:** кожна функція чітко відокремлена і відповідає за свою задачу;
- **легка підтримка:** у разі змін у програмі достатньо змінити конкретний модуль, не впливаючи на решту коду;
- **зручність для роботи в команді:** завдяки поділу на пакети можна працювати над окремими модулями незалежно;

Така архітектура забезпечує надійність і ефективність програми, дозволяючи легко адаптувати її до нових задач або розширювати функціонал.

Алгоритм роботи програми

1. Ініціалізація програми

- Запускається файл `main.py`.
- Імпортуються всі необхідні модулі, пакети та бібліотеки.
- Ініціалізується основне вікно програми (`customtkinter`).
- Налаштовуються стилі інтерфейсу, використовуючи модулі з пакета `style`.

2. Створення графічного інтерфейсу

- У вікні програми створюються основні елементи інтерфейсу:
 - Поля для введення коефіцієнтів функцій.
 - Кнопки для запуску розрахунків, побудови графіків, дослідження функцій.
 - Чекбокси для вибору параметрів (наприклад, відображення похідної або критичних точок).
 - Полотна для візуалізації графіків, з використанням `FigureCanvasTkAgg`.
- Елементи створюються через модулі з пакета `main_elements`.

3. Введення даних користувачем

- Користувач вводить параметри функції (коефіцієнти, тип функції тощо).
- Дані перевіряються на коректність.

4. Розрахунки

- Після натискання кнопки викликаються функції з пакета `data_calculation`:
 - Обчислення похідної функції.
 - Пошук критичних точок (максимумів, мінімумів, точок перегину).
 - Аналіз інтервалів зростання та спадання функції.
 - Визначення особливих точок, таких як вертикальні асимптоти або точки розриву.

5. Побудова графіків

- На основі результатів розрахунків викликаються функції з пакета `plotting`:
 - Створюється графік заданої функції.
 - Накладаються графіки похідних.
 - Відображаються критичні точки та інші особливості графіку (наприклад, точки перетину з осями).
- Графік інтегрується в графічний інтерфейс через полотно (`FigureCanvasTkAgg`).

6. Відображення результатів

- Результати розрахунків виводяться у відповідних `STkLabel` та на графіку.
- Всі графіки та точки дослідження відображаються у візуалізації.

-

7. Інтерактивна робота

- Користувач може змінювати коефіцієнти, щоб побачити, як вони впливають на графік.
- Надається можливість ввімкнути або вимкнути відображення похідних на графіку.

8. Завершення роботи

- Користувач може закрити програму через стандартний механізм закриття вікна.
- При закритті звільняються всі ресурси, і програма завершує свою роботу.

Алгоритм також передбачає гнучкість у додаванні нових функцій (наприклад, інших математичних досліджень або методів візуалізації), оскільки всі компоненти програми чітко ізольовані.