Методичні вказівки
до виконання лабораторних робіт з дисципліни
«Засоби моделювання в електротехнічних системах»
зі спеціальності «Електрична інженерія»

Міністерство освіти і науки України
Вінницький національний технічний університет

Методичні вказівки
до виконання лабораторних робіт з дисципліни «Засоби моделювання в електротехнічних системах» зі спеціальності «Електрична інженерія»

Вінниця
ВНТУ
2025

Рекомендовано до видання Радою з якості освіти Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 3 від 23.10.2025 р.)

Рецензенти:
О. Є. Рубаненко, кандидат технічних наук, професор
О. В. Бабенко, кандидат технічних наук, доцент

Методичні вказівки до виконання лабораторних робіт з дисципліни «Засоби моделювання в електротехнічних системах» зі спеціальності «Електрична інженерія» [Електронний ресурс] / уклад.: О. М. Кравець, Д. Ю. Лебедь. – Вінниця : ВНТУ, 2025. – (PDF, 71 с.)

У методичних вказівках наведено основні теоретичні відомості, порядок та приклади виконання лабораторних робіт, вимоги до звіту, перелік запитань до захисту лабораторних робіт та рекомендовані літературні джерела. Методичні вказівки розроблені відповідно до навчальної програми дисципліни «Засоби моделювання в електротехнічних системах».

ЗМІСТ

ВСТУП

Методичні вказівки відповідають програмі дисципліни «Засоби моделювання в електротехнічних системах» і складаються з шести лабораторних робіт. В лабораторних роботах вказана мета, наведені короткі теоретичні відомості, порядок виконання роботи і обробки результатів дослідження та рекомендована література.

Метою курсу є формування вмінь розв’язувати прикладні інженерні задачі методом чисельного моделювання, обробки й візуалізації експериментальних даних та використання програмних емуляторів електричних кіл. Курс забезпечує поетапне опанування: базової синтаксичної конструкції Python, логіки програм, функціонального моделювання, роботи з масивами і матрицями, побудови графіків, символьних обчислень, обробки даних та застосування табличних редакторів і симуляторів.

У результаті виконання лабораторних робіт студент повинен:

– коректно формулювати алгоритми розв’язання інженерних задач та реалізувати їх мовою Python;

– застосовувати умовні конструкції, цикли та логічні операції для управління виконанням програм;

– розробляти і використовувати функції та опрацьовувати текстові рядки для автоматизації обчислень;

– виконувати операції над матрицями та розв’язувати системи лінійних алгебраїчних рівнянь із застосуванням бібліотеки NumPy;

– побудувати наочні графічні представлення результатів досліджень за допомогою Matplotlib та виконувати базові символьні перетворення із sympy;

– застосовувати методи обробки експериментальних даних: фільтрацію, апроксимацію, статистичний аналіз і візуалізацію;

– використовувати Google Таблиці для зберігання та первинного аналізу табличних даних;

– моделювати електричні кола в програмних емуляторах (на прикладі Falstad) та інтерпретувати результати симуляцій у контексті інженерної задачі.

Методичні вказівки побудовані послідовно: від базової роботи з Python і Colab до прикладних задач обробки даних і моделювання електричних кіл. Така послідовність забезпечує поступове нарощування компетенцій і дає змогу студенту застосовувати набуті знання у проєктних і дослідницьких завданнях.

ЛАБОРАТОРНА РОБОТА № 1

ЗНАЙОМСТВО З PYTHON ТА СЕРЕДОВИЩЕМ COLAB

Мета роботи. Ознайомитися з онлайн-середовищем розробки Google Colaboratory (Colab). Навчитися створювати, редагувати та запускати код Python в Colab.

Хід роботи

1. Змініть текст запиту в функції input(), щоб запитати користувача про його вік.
2. Виведіть на екран повідомлення, яке включає ім’я та вік користувача (наприклад, "Вас звати [ім'я] і вам [вік] років"). Зверніть увагу на перетворення типу даних, якщо потрібно (вік, отриманий від input(), спочатку буде рядком).
3. Оформіть звіт та зробіть висновки з виконаної роботи.

Теоретичні відомості

1. Відкрийте Google Colab.

– У вашому веб-браузері перейдіть за посиланням: https://colab.research.google.com/
– Зайдіть у свій обліковий запис Google (якщо ви ще не увійшли).

2. Створіть новий блокнот Python 3.

– На головній сторінці Colab натисніть "Файл" -> "Створити блокнот" (рис. 1.1).

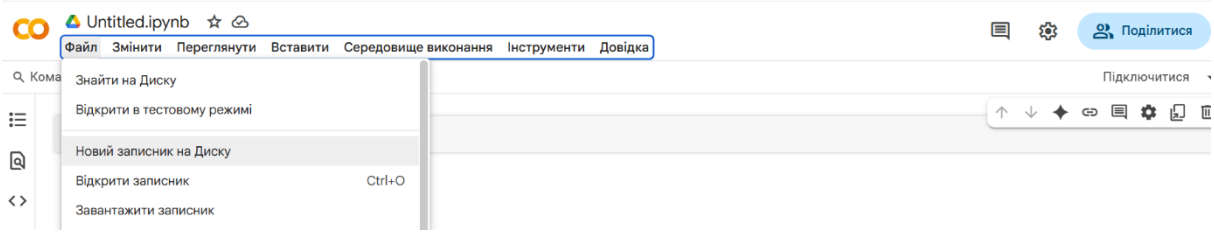


Рисунок 1.1 – Вікно програми Colab

– Переконайтеся, що у верхньому правому куті обрано "Python 3" (або вища версія Python, якщо доступна).

3. Виведення тексту на екран.

У першій клітинці коду (вона з’явиться автоматично) напишіть наступний код:

```
print("Привіт, світе! ")
```

– Натисніть кнопку "Виконати" (виглядає як значок відтворення ▶) зліва від клітинки або натисніть клавіші Shift + Enter.

– Переконайтеся, що під клітинкою з’явився результат виконання: Привіт, світе!.
– Завдання: змініть текст у дужках функції print() на будь-який інший вираз і знову виконайте клітинку. Спробуйте вивести своє ім’я.

4. Змінні та типи даних.

У новій клітинці коду (натисніть "+ Код", щоб створити нову клітинку) напишіть наступний код:

```
name = "Іван" # Створення змінної рядкового типу (string)
age = 20    # Створення змінної цілого типу (integer)
height = 1.75 # Створення змінної типу з плаваючою комою (float)

print("Мене звати ", name)
print("Мені ", age, " років")
print("Мій зріст ", height, " метри")
```

– Виконайте клітинку. Переконайтеся, що вивід відповідає вашим очікуванням.
– Змініть значення змінних name, age та height на свої власні дані.
– Створіть нову змінну, наприклад, city та присвойте їй назву вашого міста. Виведіть значення цієї змінної на екран.
– Спробуйте додати коментарі до коду, щоб пояснити, що робить кожна змінна (коментарі в Python починаються з символу #).

5. Арифметичні операції.

У новій клітинці коду напишіть:

```
number1 = 10
number2 = 5

sum_result = number1 + number2
difference_result = number1 - number2
product_result = number1 * number2
quotient_result = number1 / number2

print("Сума:", sum_result)
print("Різниця:", difference_result)
print("Добуток:", product_result)
print(“Частка:", quotient_result)
```

Виконайте клітинку. Перевірте результати арифметичних операцій.
– Змініть значення number1 та number2 на інші числа і знову виконайте код.
– Спробуйте використати інші арифметичні оператори, наприклад, ділення націло //, залишок від ділення %, піднесення до степеня **. Додайте виведення результатів цих операцій на екран.

6. Введення даних від користувача.

У новій клітинці коду напишіть:

```
user_name = input("Будь ласка, введіть своє ім’я: ")
print("Привіт,", user_name, "!")
```

– Виконайте клітинку. У полі для введення введіть своє ім’я та натисніть Enter.
– Переконайтеся, що програма привіталася з вами, використовуючи введене ім’я.

Контрольні запитання

1. Наведіть приклад команди print() для виведення вашого імені замість фрази "Привіт, світе!".
2. Поясніть різницю між типами даних int, float та str у Python.
3. Які арифметичні оператори відповідають за:
    a) цілочисельне ділення;
    b) операцію піднесення до степеня;
    c) знаходження залишку від ділення?
4. Напишіть код для обчислення і виведення суми, різниці та добутку двох довільних чисел у Python.
5. Як прийняти текстовий ввід від користувача в Python і використати його у функції print() для персоналізованого привітання?
6. Опишіть порядок дій для перевірки результатів змінення значень змінних number1 та number2 і доповнення коду новими арифметичними операціями (//, %, **).

ЛАБОРАТОРНА РОБОТА № 2

ЛОГІЧНІ ВИРАЗИ ТА ЛОГІЧНІ ОПЕРАТОРИ. РОЗГАЛУЖЕННЯ ПРОГРАМИ. НАЙПРОСТІШІ ЦИКЛИ

Мета роботи. Ознайомитися з основними поняттями логічних виразів та операторів в мові програмування Python. Освоїти використання найпростіших циклів (for та while) для організації повторюваних дій у програмі. Ознайомитися із застосуванням мови програмування Python для розв'язання задач аналізу кіл постійного струму. Навчитися використовувати математичні оператори та функції Python для розрахунку параметрів кіл постійного струму.

Хід роботи

Завдання 1.

1. Виконання прикладів. У блоці коду Google Colab введіть та виконайте приклади коду з розділу "Теоретичні відомості" для логічних виразів, умовних операторів та циклів. Переконайтеся, що ви розумієте, як вони працюють.

2. Перевірка парності числа. Напишіть програму на Python, яка запитує користувача ввести ціле число. Програма повинна перевірити, чи є число парним або непарним, і вивести відповідне повідомлення на екран.

3. Пошук найбільшого числа. Напишіть програму, яка приймає на вхід три числа від користувача. Використовуючи умовні оператори, визначте та виведіть на екран найбільше з цих трьох чисел.

4. Калькулятор з основними операціями. Створіть простий калькулятор, який виконує чотири основні арифметичні операції: додавання, віднімання, множення та ділення. Програма повинна запитувати у користувача два числа та операцію, яку потрібно виконати (наприклад, '+', '-', '*', '/'). Використовуючи умовні оператори, виконайте відповідну операцію та виведіть результат. Врахуйте можливість ділення на нуль.

5. Сума чисел до заданого значення з циклом while. Напишіть програму, яка запитує у користувача ввести додатне ціле число n. Використовуючи цикл while, обчисліть та виведіть на екран суму всіх цілих чисел від 1 до n включно.

6. Перевірка результатів. Перевірте результати виконання ваших програм. Переконайтеся, що ваші програми працюють правильно та відповідають умовам завдання.

7. Збереження роботи. Ваш блокнот Google Colab автоматично зберігається на Google Диску. Ви можете завантажити його на свій комп'ютер, вибравши "Файл" -> "Завантажити .ipynb".

Завдання 2.

1. Розрахунок параметрів кола за законом Ома. Виконайте завдання згідно з вашим варіантом. Для кожного завдання створіть новий блок коду в Google Colab. Напишіть програму, яка розраховує невідомий параметр кола постійного струму за законом Ома.

– Варіант 1: Відомі опір та струм. Розрахувати напругу.

– Варіант 2: Відомі напруга та опір. Розрахувати струм.

– Варіант 3: Відомі напруга та струм. Розрахувати опір.

– Варіант 4: Відомі ЕРС джерела, зовнішній та внутрішній опори. Розрахувати струм в колі та напругу на зовнішньому опорі.

2. Еквівалентний опір резисторів.

Напишіть програму для розрахунку еквівалентного опору комбінації резисторів.

– **Варіант 1:** Послідовне з'єднання чотирьох резисторів.

– **Варіант 2:** Паралельне з'єднання чотирьох резисторів.

– **Варіант 3:** Послідовне з'єднання двох груп паралельно з'єднаних резисторів (дві групи по два резистори в кожній). Користувач вводить опори чотирьох резисторів: R_1, R_2, R_3, R_4. R_1 і R_2 з'єднані паралельно, R_3 і R_4 з'єднані паралельно, а потім ці дві паралельні групи з'єднані послідовно.

– **Варіант 4:** Паралельне з'єднання двох груп послідовно з'єднаних резисторів (дві групи по два резистори в кожній). Користувач вводить опори чотирьох резисторів: R_1, R_2, R_3, R_4. R_1 і R_2 з'єднані послідовно, R_3 і R_4 з'єднані послідовно, а потім ці дві послідовні групи з'єднані паралельно.

3. Розподіл струму та напруги в послідовному колі.

Програма має розраховувати розподіл напруги та струму в послідовному колі, що складається з трьох резисторів та джерела напруги.

– Варіант 1: Відомі загальна напруга джерела та опори трьох резисторів. Розрахувати струм в колі та напругу на кожному резисторі.

– Варіант 2: Відомий струм в колі та опори трьох резисторів. Розрахувати загальну напругу джерела та напругу на кожному резисторі.

– Варіант 3: Відома напруга на першому резисторі та опори трьох резисторів. Розрахувати струм в колі, загальну напругу джерела та напруги на другому та третьому резисторах.

– Варіант 4: Відома напруга на третьому резисторі, загальна напруга джерела та опір першого резистора. Розрахувати опори другого та третього резисторів. (Це завдання може вимагати трохи більше алгебраїчних маніпуляцій).

4. Розподіл струму в паралельному колі.

Програма має розраховувати розподіл струмів в паралельному колі, що складається з трьох паралельних гілок та джерела струму (або напруги – оберіть, як простіше для розрахунку).

– Варіант 1: Відома загальна напруга (джерела) та опори трьох паралельних резисторів. Розрахувати струм через кожен резистор та загальний струм.

– Варіант 2: Відомий загальний струм та опори трьох паралельних резисторів. Розрахувати напругу на паралельних гілках та струм через кожен резистор.

– Варіант 3: Відомий струм через перший резистор та опори всіх трьох паралельних резисторів. Розрахувати загальну напругу, загальний струм та струми через другий та третій резистори.

– Варіант 4: Відомий струм через третій резистор, загальний струм та опір першого резистора. Розрахувати опори другого та третього резисторів. (Це завдання може вимагати трохи більше алгебраїчних маніпуляцій).

5. Перевірте результати, переконайтеся в їх відповідності законам електротехніки та логіці електричних кіл. Збережіть блокнот Colab на Google Диску або завантажте у форматі .ipynb.

6. Оформіть звіт та зробіть висновки з проробленої роботи

Теоретичні відомості

1. Логічні вирази та оператори в Python.

Логічні вирази використовуються для перевірки умов та отримання результату True (істина) або False (хибність). Python надає набір логічних операторів для побудови цих виразів [1]-[2]:

- **Оператори порівняння:**
  - == (дорівнює)
  - != (не дорівнює)
  - > (більше)
  - < (менше)
  - >= (більше або дорівнює)
  - <= (менше або дорівнює)
- **Логічні оператори:**
  - and (логічне І) - повертає True тільки якщо обидва операнди True.
  - or (логічне АБО) - повертає True якщо хоча б один з операндів True.
  - not (логічне НЕ) - інвертує логічне значення операнда (з True на False і навпаки).

Приклади логічних виразів:

```
x = 5
y = 10
```

```
print(x == y)     # False (x не дорівнює y)
print(x < y)      # True (x менше y)
print(x > 0 and y < 20) # True (обидві умови істинні)
print(x < 0 or y < 5)  # False (жодна з умов не істинна)
print(not (x == y))  # True (заперечення хибності - істина)
```

2. Розгалуження програми (умовні оператори).

Оператори розгалуження дозволяють виконувати різні блоки коду залежно від виконання певної умови. В Python використовуються наступні умовні конструкції:

- **if (якщо):** Виконує блок коду, тільки якщо умова істинна.

```
if умова:
   # Блок коду, що виконується, якщо умова True
```

- **if-else (якщо-інакше):** Виконує один блок коду, якщо умова істинна, та інший блок коду, якщо умова хибна.

```
if умова:
   # Блок коду, якщо умова True
else:
   # Блок коду, якщо умова False
```

- **if-elif-else (якщо-інакше якщо-інакше):** Дозволяє перевірити декілька умов послідовно. elif (скорочення від "else if") дозволяє додати додаткові умови для перевірки.

```
if умова1:
   # Блок коду, якщо умова1 True
elif умова2:
   # Блок коду, якщо умова2 True
elif умова3:
   # Блок коду, якщо умова3 True
else:
   # Блок коду, якщо жодна з умов не True
```

Приклади розгалуження:

- Приклад 1.

```
age = 20
if age >= 18:
   print("Ви повнолітні")
```

- Приклад 2.

```
number = -5
if number > 0:
   print("Число позитивне")
else:
   print("Число не позитивне")
```

- Приклад 3.

```
grade = 85
if grade >= 90:
   print("Відмінно")
elif grade >= 80:
   print("Дуже добре")
elif grade >= 70:
   print("Добре")
else:
   print("Задовільно")
```

3. Найпростіші цикли в Python.

Цикли дозволяють повторювати блок коду декілька разів. Python пропонує два основні види циклів:

- **for (цикл for):** Використовується для повторення дій задану кількість разів. Часто використовується з функцією range() для створення послідовності чисел.

```
for лічильник in range(кількість_повторень):
   # Блок коду, що виконується задану кількість разів
   # `лічильник` буде змінюватись від 0 до `кількість_повторень - 1`
```

- **Функція range():** Створює послідовність чисел для використання в циклі for.
  - range(stop) - генерує послідовність від 0 до stop-1.

```
# Цикл for з range() - повторення 5 разів
for i in range(5): # від 0 до 4
   print("Повторення:", i)
```

  - range(start, stop) - генерує послідовність від start до stop-1.

```
# Цикл for з range(), починаючи з 1 та до 6 (не включно)
for i in range(1, 6): # від 1 до 5
   print("Число:", i)
```

  - range(start, stop, step) – генерує послідовність від start до stop-1 з кроком step.
- **while (цикл while):** Виконує блок коду до тих пір, поки умова залишається істинною.

```
while умова:
   # Блок коду, що виконується, поки умова True
   # Важливо: Умова повинна стати False в певний момент, щоб уникнути нескінченного циклу.
```

Приклад оновлення циклу.

```
# Цикл while
count = 0
while count < 3:
   print("Лічильник:", count)
   count = count + 1 # Або count += 1
```

4. Основні закони та поняття для кіл постійного струму.

Аналіз кіл постійного струму є фундаментальною частиною електротехніки. Python надає зручні інструменти для виконання розрахунків, необхідних для дослідження таких кіл.

- **Закон Ома для ділянки кола:** $U = I \cdot R$, де U – напруга (В), I – струм (А), R – опір (Ом).
- **Закон Ома для повного кола:** $E = I \cdot (R + r)$, де E – електрорушійна сила (ЕРС) джерела (В), R – зовнішній опір (Ом), r – внутрішній опір джерела (Ом).
- **Потужність електричного струму:** $P = U \cdot I = I^2 \cdot R = U^2/R$ (Вт).
- **Провідність:** $G = 1/R$ (См - Сіменс).
- **Послідовне з'єднання резисторів:**
  - Загальний опір: $R_{заг} = R_1 + R_2 + ... + R_n$.
  - Струм через всі резистори однаковий: $I = I_1 = I_2 = ... = I_n$.
  - Загальна напруга дорівнює сумі напруг на кожному резисторі: $U = U_1 + U_2 + ... + U_n$.
- **Паралельне з'єднання резисторів:**
  - Загальна провідність: $G_{заг} = G_1 + G_2 + ... + G_n$ або загальний опір $1/R_{заг} = 1/R_1 + 1/R_2 + ... + 1/R_n$. Для двох паралельних резисторів: $R_{заг} = (R_1 \cdot R_2) / (R_1 + R_2)$.
  - Напруга на всіх резисторах однакова: $U = U_1 = U_2 = ... = U_n$.
  - Загальний струм дорівнює сумі струмів через кожен резистор: $I = I_1 + I_2 + ... + I_n$.
- **Перший закон Кірхгофа (для вузлів):** Алгебраїчна сума струмів, що сходяться в вузлі, дорівнює нулю. (Сума струмів, що входять у вузол, дорівнює сумі струмів, що виходять з нього).
- **Другий закон Кірхгофа (для контурів):** Алгебраїчна сума ЕРС в замкнутому контурі дорівнює алгебраїчній сумі падінь напруг на опорах цього контуру.
- **Математичні інструменти Python:**
  - **Арифметичні оператори:** +, -, *, /, //, %, **.
  - **Вбудовані функції:** abs(), round(), pow(), min(), max().

Бібліотека math не є критично важливою для більшості розрахунків кіл постійного струму на початковому етапі, але може бути використана для більш складних обчислень або для форматування виводу (наприклад, math.sqrt() для обчислення діагоналей, якщо потрібно, або math.pi, якщо задача включає геометричні елементи).

Контрольні запитання

1. Що таке логічний вираз? Які значення може приймати логічний вираз?
2. Які логічні оператори є в Python? Поясніть їх призначення та наведіть приклади.
3. Для чого використовуються умовні оператори (if, if-else, if-elif-else)?
4. Які типи циклів існують в Python? В чому їх відмінність?
5. Як працює функція range()? Наведіть приклади використання.
6. Які співвідношення для струмів та напруг справедливі при послідовному з’єднанні резисторів?
7. Сформулюйте закон Ома для ділянки кола та для повного кола.
8. Які співвідношення для струмів та напруг справедливі при паралельному з’єднанні резисторів?
9. Сформулюйте перший та другий закони Кірхгофа. Для яких розрахунків вони застосовуються?
10. Які основні математичні оператори Python використовуються для розрахунків в колах постійного струму?

ЛАБОРАТОРНА РОБОТА № 3

РОБОТА ФУНКЦІЯМИ ТА ТЕКСТОВИМИ РЯДКАМИ НА МОВІ ПРОГРАМУВАННЯ PYTHON

Мета роботи. Ознайомитися з концепцією функцій у програмуванні та їх роллю в мові Python. Зрозуміти структуру функції: параметри, тіло функції, оператор повернення значення. Ознайомитися з поняттям текстового рядка як типу даних у мові програмування Python. Освоїти основні операції над рядками: конкатенація, повторення, індексування, зрізи.

Хід роботи

Завдання 1.

1. Прості функції. Напишіть прості функції на Python для виконання вказаних дій.

– Варіант 1: Функція, що приймає два числа як аргументи та повертає їх суму.

– Варіант 2: Функція, що приймає рядок як аргумент та виводить його на екран у верхньому регістрі.

– Варіант 3: Функція, що не приймає аргументів, але повертає поточну дату та час (використовуйте бібліотеку datetime).

– Варіант 4: Функція, що приймає список чисел як аргумент та повертає їх середнє арифметичне.

2. Функції з умовними операторами. Напишіть функції на Python, що використовують умовні оператори (if, elif, else).

– Варіант 1: Функція, що приймає число як аргумент та повертає "Парне", якщо число парне, і "Непарне", якщо число непарне.

– Варіант 2: Функція, що приймає оцінку (число від 0 до 100) як аргумент та повертає текстову оцінку ("Відмінно", "Добре", "Задовільно", "Незадовільно") відповідно до діапазонів оцінок.

– Варіант 3: Функція, що приймає три числа як аргументи та повертає найбільше з них.

– Варіант 4: Функція, що приймає рік як аргумент та повертає True, якщо рік високосний, і False – якщо ні. (Високосний рік ділиться на 4, але не ділиться на 100, якщо не ділиться на 400).

3. Функції з циклами. Напишіть функції на Python, що використовують цикли (for або while).

– Варіант 1: Функція, що приймає ціле число n як аргумент та повертає факторіал числа n (n!).

– Варіант 2: Функція, що приймає ціле число n як аргумент та виводить на екран усі числа Фібоначчі до n-го числа.

– **Варіант 3:** Функція, що приймає рядок як аргумент та повертає кількість голосних літер у рядку (голосні літери: ‘а’, ‘е’, ‘є’, ‘и’, ‘і’, ‘ї’, ‘о’, ‘у’, ‘ю’, ‘я’).

– **Варіант 4:** Функція, що приймає ціле число n як аргумент та повертає True, якщо число є простим, і False – якщо ні. (Просте число ділиться тільки на 1 та на себе).

4. Функції з різними типами аргументів. Напишіть функції на Python, що демонструють використання різних типів аргументів.

– **Варіант 1:** Функція display_info, що приймає два позиційні аргументи: name (ім’я) та city (місто), та виводить повідомлення у форматі "Користувач {name} з міста {city}". (*залишено без змін, демонструє позиційні аргументи та рядкове повернення*)

– **Варіант 2:** Функція calculate_discount, що приймає два аргументи: price (позиційний - ціна товару) та discount (ключовий аргумент зі значенням за замовчуванням 0.1 – знижка 10%), та повертає ціну зі знижкою. (*залишено без змін, демонструє ключові аргументи та аргументи за замовчуванням і числове повернення*)

– **Варіант 3:** Функція is_positive_sum, що приймає **два позиційні аргументи** (num1, num2) та повертає **рядок "Positive Sum"**, якщо їх сума є додатною, та **рядок "Non-positive Sum"** в іншому випадку. (*Спрощено до двох позиційних аргументів та повернення рядка, демонструє позиційні аргументи та умовне повернення рядка.*)

– **Варіант 4:** Функція get_rectangle_area, що приймає два аргументи: width (ширина) та height (висота) – **обидва ключові аргументи зі значеннями за замовчуванням width=1 та height=1**, та повертає **площу прямокутника як число**. (*Спрощено до ключових аргументів зі значеннями за замовчуванням та повернення числа, демонструє ключові аргументи та числове повернення.*)

Завдання 2.

1. Створення та основні операції з рядками. Напишіть програму на Python, яка демонструє створення рядків різними способами та основні операції над ними.

– **Варіант 1:** Створення трьох рядків: в одинарних лапках, в подвійних лапках, багаторядковий рядок. Вивести їх на екран. З’єднати перший та другий рядок за допомогою конкатенації та вивести результат. Повторити третій рядок 2 рази та вивести результат.

– **Варіант 2:** Створити рядок "Python is powerful". Вивести на екран: перший символ, останній символ, символи з 3-го по 7-й індекси (зріз). Перевірити, чи містить рядок підрядок "powerful" (використовуючи оператор in).

– Варіант 3: Запитати у користувача два рядки. З'єднати їх, розділивши пробілом, та вивести результат у верхньому регістрі. Повторити отриманий рядок 3 рази.

– Варіант 4: Створити багаторядковий рядок, що містить вірш (або його частину) на ваш вибір. Вивести його на екран. Вивести на екран третій рядок цього вірша (якщо він є, інакше – останній рядок).

2. Методи рядків для зміни регістру та пошуку. Напишіть програму на Python, яка використовує методи рядків для зміни регістру та пошуку підрядків.

– Варіант 1: Запитати у користувача рядок. Перетворити його у верхній регістр, нижній регістр, зробити першу літеру кожного слова великою (title case), та вивести всі результати на екран.

– Варіант 2: Запитати у користувача рядок та підрядок для пошуку. Знайти індекс першого входження підрядка в рядку (використовуючи .find()) та вивести результат. Також порахувати, скільки разів підрядок зустрічається в рядку (використовуючи .count()) та вивести результат.

– Варіант 3: Створити рядок "Programming на Python – це цікаво!". Перевірити, чи починається рядок зі слова "Programming" та чи закінчується знаком оклику (використовуючи .startswith() та .endswith()). Вивести результати перевірок (True або False).

– Варіант 4: Запитати у користувача речення. Знайти індекс першого входження слова "Python" (або іншого слова на вибір) у реченні. Якщо слово знайдено, вивести повідомлення "Слово знайдено на позиції [індекс]", інакше – "Слово не знайдено".

3. Методи рядків для заміни, розділення та з'єднання. Напишіть програму на Python, яка використовує методи рядків для заміни, розділення та з'єднання рядків.

– Варіант 1: Запитати у користувача рядок. Замінити в ньому всі входження слова "old" на "new" (використовуючи .replace()) та вивести змінений рядок.

– Варіант 2: Створити рядок, що містить декілька слів, розділених комами (наприклад, "яблуко, банан, апельсин, груша"). Розділити рядок на список слів за роздільником-комою (використовуючи .split()). Вивести отриманий список слів на екран.

– Варіант 3: Створити список слів (наприклад, ["Земля", "Місяць", "Сонце"]). З'єднати слова зі списку в один рядок, розділивши їх символом " | " (вертикальна риска з пробілами) (використовуючи .join()). Вивести отриманий рядок.

– Варіант 4: Запитати у користувача рядок з пробілами на початку та в кінці. Очистити рядок від пробільних символів з обох боків (використовуючи .strip()) та вивести очищений рядок. Також вивести довжину вихідного рядка та довжину очищеного рядка.

4. Форматування рядків. Напишіть програму на Python, яка використовує різні способи форматування рядків.

– Варіант 1: Запитати у користувача ім’я та вік. Створити форматоване привітання, використовуючи f-рядки, та вивести його на екран у форматі: "Доброго дня, [ім’я]! Вам [вік] років.".

– Варіант 2: Запитати у користувача назву товару та його ціну. Створити рядок, використовуючи метод .format(), що містить інформацію про товар та ціну у форматі: "Товар: [назва_товару], Ціна: [ціна] грн.". Вивести рядок на екран.

– Варіант 3: Створити шаблон рядка "Результат: %s, значення: %d" та підставити у нього рядок "Успіх" та число 100, використовуючи %-форматування. Вивести результат.

– Варіант 4: Запитати у користувача назву міста та країну. Створити форматоване повідомлення, використовуючи f-рядки, яке виводить назву міста у верхньому регістрі, а назву країни – у нижньому регістрі, у форматі: "[МІСТО] - [країна]". Наприклад, для вводу "Київ" та "Україна" має вивести "КИЇВ - україна".

5. Оформіть звіт та зробіть висновки з проробленої роботи.

Теоретичні відомості

1. Що таке функція?

Функція – це іменований блок коду, який виконує певну задачу. Функції є важливим будівельним блоком програмного коду, дозволяючи:

– **Структурувати код:** розбивати складні програми на менші, логічно завершені блоки.

– **Повторне використання коду:** викликати один і той же блок коду з різних місць програми, не дублюючи його.

– **Покращити читабельність:** зробити код більш зрозумілим та легким для розуміння, використовуючи описові імена функцій.

– **Модульність:** створювати незалежні модулі коду, які можна розробляти та тестувати окремо.

2. Визначення функції в Python.

Функція в Python визначається за допомогою ключового слова def, за яким слідує ім’я функції, дужки () та двокрапка :. Тіло функції (блок коду, який виконується при виклику функції) пишеться з відступом.

```
def function_name(parameter1, parameter2, ...):
    # Function body (code to be executed)
    # ...
    return value  # (Optional) return value
```

– def: Ключове слово для визначення функції.

– **function_name**: Ідентифікатор, унікальне ім’я, за яким функцію можна буде викликати. Імена функцій створюються за тими ж правилами, що й імена змінних.

– **parameter1, parameter2, ...**: Необов’язковий список вхідних параметрів (аргументів), які функція може приймати. Параметри вказуються в дужках через кому. Якщо функція не приймає параметрів, дужки залишаються порожніми.

– **:**: Двокрапка, що позначає початок тіла функції.

– **Тіло функції**: Блок коду, що виконується при виклику функції. Код тіла функції повинен бути написаний з відступом (зазвичай 4 пробіли або 1 табуляція).

– **return value**: Необов’язковий оператор return. Використовується для повернення результату виконання функції. Якщо return не вказано, функція повертає None за замовчуванням.

3. Виклик функції.

Щоб виконати код функції, її потрібно викликати, вказавши її ім’я та передавши необхідні аргументи в дужках (якщо функція їх приймає).

```
function_name(argument1, argument2, ...)
```

Приклад простої функції:

```
def greet(name):
    """Ця функція виводить привітання для заданого імені."""
    print("Привіт, " + name + "!")

# Виклик функції
greet("Іван")
greet("Марія")
```

4. Параметри та аргументи функції:

– Параметри – це імена змінних, перелічені в дужках у визначенні функції. Вони служать "заповнювачами" для значень, які функція очікує отримати при виклику.

– Аргументи – це фактичні значення, які передаються функції під час її виклику. Аргументи підставляються на місце відповідних параметрів.

Типи аргументів:

– Позиційні аргументи: передаються у функцію в тому ж порядку, в якому визначені параметри.

```
def difference(x, y):
    return x - y

result = difference(10, 5)
print(result)
```

– Ключові аргументи: передаються у функцію з вказівкою імені параметра, якому вони відповідають. Ключові аргументи дозволяють передавати аргументи в будь-якому порядку, роблячи виклики функцій більш читабельними.

```
result = difference(y=5, x=10)
print(result)
```

- Аргументи за замовчуванням: значення за замовчуванням можуть бути задані для параметрів функції під час її визначення. Якщо при виклику функції аргумент для параметра з значенням за замовчуванням не передається, буде використане значення за замовчуванням.

```
def power(number, exponent=2):
   return number ** exponent
result1 = power(5)
print(result1)
result2 = power(5, 3)
print(result2)
```

5. Область видимості змінних (Scope).

Область видимості змінної визначає, в якій частині програми змінна є доступною для використання. У Python існують дві основні області видимості:

– Локальна область видимості. Змінні, визначені всередині функції, мають локальну область видимості. Вони доступні лише всередині цієї функції та перестають існувати після завершення її виконання.

```
def example_local_variable():
   local_variable = 10
   print(local_variable)
example_local_variable()
# print(local_variable) # This would cause an error
```

– Глобальна область видимості. Змінні, визначені поза функціями (на верхньому рівні модуля), мають глобальну область видимості. Вони доступні для використання у будь-якій частині програми, в тому числі всередині функцій.

```
global_variable = 20
def example_global_variable():
   print(global_variable)

example_global_variable()
print(global_variable)
```

6. Текстові рядки в Python.

Текстовий рядок (string, str) – це послідовність символів, що використовуються для представлення текстової інформації. Рядки є одним з основних типів даних у Python [3]-[4].

Основні характеристики рядків в Python:

– Не змінні (immutable): після створення рядка, його не можна змінити безпосередньо. Будь-яка операція, що "змінює" рядок, насправді створює новий рядок.

– **Впорядковані послідовності:** символи в рядку розташовані в певному порядку, і до кожного символу можна отримати доступ за його індексом.

– **Unicode:** рядки в Python 3 є Unicode-рядками, що дозволяє представляти символи різних мов та алфавітів.

7. Створення рядків в Python.

Рядки в Python можна створювати різними способами:

– Одинарні лапки (‘...’):

```
string1 = ‘Це рядок в одинарних лапках’
```

– Подвійні лапки ("..."):

```
string2 = "Це рядок в подвійних лапках"
```

Одинарні та подвійні лапки використовуються для створення звичайних однорядкових рядків. Різниця між ними полягає в зручності використання всередині рядка лапок іншого типу (щоб не екранувати їх).

```
string_with_single_quotes = "Рядок, що містить ‘одинарні’ лапки"
string_with_double_quotes = ‘Рядок, що містить "подвійні" лапки’
```

– Потрійні лапки (‘‘‘...’’’ або """..."""):

```
multiline_string = ‘‘‘Це багаторядковий
рядок, що може займати
декілька рядків коду’’’
multiline_string2 = """Ще один варіант
багаторядкового рядка"""
```

Потрійні лапки використовуються для створення багаторядкових рядків, а також для рядків документації (docstrings) у функціях та класах. Всередині потрійних лапок можна використовувати як одинарні, так і подвійні лапки без екранування.

– Екрановані послідовності (escape sequences):

Спеціальні комбінації символів, що починаються з зворотного слешу (\), використовуються для представлення символів, які важко або неможливо ввести безпосередньо в рядку. Деякі корисні екрановані послідовності:

- \n: Новий рядок
- \t: Горизонтальна табуляція
- \\: Зворотний слеш
- \’: Одинарна лапка
- \": Подвійна лапка

```
string_with_newline = "Перший рядок\nДругий рядок"
string_with_tab = "Колонка1\tКолонка2"
```

8. Основні операції над рядками:

– Конкатенація (додавання рядків) (+):

```
string1 = "Hello"
string2 = "World"
result = string1 + " " + string2  # "Hello World"
```

– Повторення рядків (*):

```
string_example = "Привіт! "
repeated_string = string_example * 3  # "Привіт! Привіт! Привіт! "
```

– Індексування (доступ до символу за індексом) ([]):

Індекси в рядках починаються з 0 для першого символу. Від'ємні індекси відраховуються з кінця рядка (-1 – останній символ, -2 – передостанній і т.д.).

```
string_index = "Python"
first_char = string_index[0]  # 'P'
third_char = string_index[2]  # 't'
last_char = string_index[-1] # 'n'
```

– Зрізи (вилучення підрядка) ([:]):

Зрізи дозволяють отримати частину рядка, вказавши діапазон індексів. Синтаксис зрізу: рядок[початок:кінець:крок]. початок та кінець вказують індекси початку та кінця зрізу (кінець **не включається** у зріз). крок визначає крок вибірки символів (за замовчуванням 1).

```
string_slice = "Programming"
substring1 = string_slice[0:7]    # "Program" (символи з індексами 0, 1, 2, 3, 4, 5, 6)
substring2 = string_slice[7:]     # "ming" (символи з індексу 7 до кінця)
substring3 = string_slice[:4]     # "Prog" (символи з початку до індексу 4, не включаючи 4)
substring4 = string_slice[::2]    # "Pormn" (кожен другий символ)
substring5 = string_slice[::-1]   # "gnimmargorP" (рядок у зворотному порядку)
```

– Перевірка на входження підрядка (in, not in):

```
string_contains = "Hello World"
contains_hello = "Hello" in string_contains    # True
not_contains_python = "Python" not in string_contains # True
```

9. Методи рядків:

Python надає велику кількість вбудованих методів для роботи з рядками. Методи викликаються через крапку після імені рядкової змінної. Деякі корисні методи рядків:

- **Зміна регістру:**
  - .upper(): Перетворює всі символи рядка у верхній регістр.
  - .lower(): Перетворює всі символи рядка у нижній регістр.
  - .title(): Перетворює першу літеру кожного слова у верхній регістр, а решту – у нижній.

- .capitalize(): Перетворює першу літеру рядка у верхній регістр, а решту – у нижній.
- .swapcase(): Змінює регістр кожної літери на протилежний.

```
string_case = "приклад рядка"
upper_case = string_case.upper()
lower_case = upper_case.lower()
title_case = string_case.title()
capitalize_case = string_case.capitalize()
swap_case = upper_case.swapcase()
```

– **Пошук підрядків:**
  - .find(підрядок, початок, кінець): Знаходить перше входження підрядка в рядку та повертає індекс початку підрядка. Якщо підрядок не знайдено, повертає -1. початок та кінець (необов'язкові) вказують діапазон пошуку.
  - .rfind(підрядок, початок, кінець): Знаходить останнє входження підрядка (пошук справа наліво).
  - .index(підрядок, початок, кінець): Аналогічно .find(), але якщо підрядок не знайдено, викликає виняток ValueError.
  - .rindex(підрядок, початок, кінець): Аналогічно .rfind(), але викликає виняток ValueError, якщо підрядок не знайдено.
  - .startswith(префікс, початок, кінець): Перевіряє, чи починається рядок з вказаного префікса. Повертає True або False.
  - .endswith(суфікс, початок, кінець): Перевіряє, чи закінчується рядок вказаним суфіксом. Повертає True або False.
  - .count(підрядок, початок, кінець): Підраховує кількість входжень підрядка в рядку.

```
string_find = "Hello World Hello"
index_hello = string_find.find("Hello")
last_index_hello = string_find.rfind("Hello")
count_hello = string_find.count("Hello")
starts_with_hello = string_find.startswith("Hello")
ends_with_world = string_find.endswith("World")
```

– Заміна підрядків:
  - .replace(старий_підрядок, новий_підрядок, кількість): Замінює всі входження старий_підрядок на новий_підрядок. кількість (необов'язковий) обмежує кількість замін.

```
string_replace = "Hello World"
new_string = string_replace.replace("World", "Python")
replace_first_hello = string_replace.replace("Hello", "Hi", 1)
```

– Розділення рядка на частини:
  - .split(роздільник, maxsplit): Розділяє рядок на список підрядків за вказаним роздільником. Якщо роздільник не вказано, розділення відбувається за пробілами. maxsplit (необов'язковий) обмежує кількість розділень.

- .rsplit(роздільник, maxsplit): Розділяє рядок справа наліво.
- .splitlines(keepends): Розділяє рядок на список рядків за символами нового рядка (\n, \r, \r\n). keepends=True (необов'язковий) залишає символи нового рядка в кінці підрядків.

```
string_split = "рядок, розділений, комами"
substrings = string_split.split(", ")
string_split_lines = "рядок з різними\nпереносами\rрядків"
lines = string_split_lines.splitlines()
```

– З'єднання рядків:

- .join(послідовність): З'єднує елементи послідовності (списку, кортежу тощо) в один рядок, використовуючи рядок, для якого викликається метод, як роздільник.

```
words = ["Привіт", "світ", "!"]
string_with_spaces = " ".join(words)
string_with_commas = ", ".join(words)
```

– Видалення пробільних символів з початку та кінця рядка:

- .strip(символи): Видаляє пробільні символи (пробіли, табуляції, нові рядки) або вказані символи з початку та кінця рядка.
- .lstrip(символи): Видаляє символи тільки зліва (з початку).
- .rstrip(символи): Видаляє символи тільки справа (з кінця).

```
string_strip = "   рядок з пробілами з обох боків   "
stripped_string = string_strip.strip()
string_with_chars = "***рядок з зірочками***"
stripped_string2 = string_with_chars.strip("*")
```

– Форматування рядків:

Python надає різні способи форматування рядків для підстановки значень змінних у рядкові шаблони.

f-рядки (f-strings) (починаючи з Python 3.6): Найзручніший та найефективніший спосіб. Використовуйте префікс f перед рядком у лапках та вставляйте змінні або вирази у фігурних дужках {}.

```
name_example = "Іван"
age_example = 25
greeting = f"Привіт, мене звати {name_example}, мені {age_example} років."
```

.format(): Метод рядків для форматування. Використовуйте фігурні дужки {} як заповнювачі та метод .format() для підстановки значень.

```
name_example2 = "Петро"
age_example2 = 30
greeting2 = "Привіт, мене звати {}, мені {} років.".format(name_example2, age_example2)
```

Контрольні запитання

1. Що таке функція в програмуванні? Які переваги використання функцій?
2. Як визначити функцію в Python? Який синтаксис?
3. Як викликати функцію?
4. В чому різниця між параметрами та аргументами функції?
5. Що таке оператор return? Для чого він використовується?
6. Що таке область видимості змінних (scope)? Які типи області видимості існують в Python?
7. Як використовувати глобальні змінні всередині функції?
8. Що таке текстовий рядок в Python? Які його основні характеристики?
9. Які способи створення рядків в Python ви знаєте? В чому відмінність між одинарними, подвійними та потрійними лапками?
10. Які основні операції можна виконувати над рядками в Python? (конкатенація, повторення, індексування, зрізи).
11. Що таке екрановані послідовності? Наведіть приклади.
12. Які методи рядків ви знаєте для зміни регістру?
13. Які методи рядків використовуються для пошуку підрядків? В чому відмінність між .find() та .index()?
14. Як розділити рядок на список підрядків? Як з'єднати список рядків в один рядок?
15. Які способи форматування рядків існують в Python? Який спосіб є найбільш сучасним та рекомендованим?

ЛАБОРАТОРНА РОБОТА № 4

ВИРІШЕННЯ СИСТЕМ ЛІНІЙНИХ АЛГЕБРАЇЧНИХ РІВНЯНЬ ТА МАТРИЧНИХ ОБЧИСЛЕНЬ МОВОЮ ПРОГРАМУВАННЯ PYTHON З ВИКОРИСТАННЯМ БІБЛІОТЕКИ NUMPY

Мета роботи. Навчитися створювати матриці різних типів та розмірностей за допомогою NumPy. Освоїти базові операції матричної алгебри: додавання, віднімання, скалярне множення, матричне множення та транспонування з використанням NumPy. Ознайомитися з можливостями бібліотеки NumPy для вирішення математичних задач, зокрема лінійних рівнянь та систем лінійних алгебраїчних рівнянь (СЛАР). Навчитися використовувати функцію numpy.linalg.solve() для знаходження розв'язків СЛАР.

Хід роботи

Завдання 1.

Імпорт бібліотеки NumPy. На початку блокноту імпортуйте бібліотеку NumPy, використовуючи стандартне скорочення np:

```
import numpy as np
```

1. Базові операції з матрицями. Напишіть програму на Python, яка виконує базові операції з двома матрицями, заданими користувачем.

– Варіант 1: Додавання двох матриць. Програма запитує розмірність матриць (квадратні матриці однакового розміру) та елементи матриць A та B, потім обчислює та виводить їх суму (A + B).

– Варіант 2: Віднімання двох матриць. Програма запитує розмірність матриць (квадратні матриці однакового розміру) та елементи матриць A та B, потім обчислює та виводить їх різницю (A - B).

– Варіант 3: Скалярне множення матриці. Програма запитує розмірність матриці (квадратна матриця) та елементи матриці A, а також скалярне число. Обчислює та виводить результат множення матриці A на скаляр.

– Варіант 4: Елемент-wise множення двох матриць. Програма запитує розмірність матриць (квадратні матриці однакового розміру) та елементи матриць A та B, потім обчислює та виводить результат їх поелементного множення (A * B).

2. Матричне множення. Напишіть програму на Python, яка виконує матричне множення.

– Варіант 1: Множення двох квадратних матриць. Програма запитує розмірність матриць (однаковий розмір) та елементи матриць A та B, потім обчислює та виводить їх матричний добуток (A @ B).

– Варіант 2: Множення матриці на вектор-стовпчик. Програма запитує розмірність матриці A (NxM), елементи матриці A та елементи вектора-стовпчика B (розмірності M). Обчислює та виводить добуток матриці A на вектор B.

– Варіант 3: Транспонування матриці. Програма запитує розмірність квадратної матриці та її елементи. Обчислює та виводить транспоновану матрицю.

– Варіант 4: Множення транспонованої матриці на вихідну. Програма запитує розмірність квадратної матриці та її елементи. Обчислює транспоновану матрицю A.T, а потім обчислює та виводить матричний добуток A.T @ A.

3. Властивості матриць. Напишіть програму на Python для обчислення деяких властивостей матриць.

– Варіант 1: Обчислення визначника матриці. Програма запитує розмірність квадратної матриці та її елементи. Обчислює та виводить визначник матриці.

– Варіант 2: Обчислення сліду матриці. Програма запитує розмірність квадратної матриці та її елементи. Обчислює та виводить слід матриці (суму діагональних елементів).

– Варіант 3: Пошук максимального та мінімального елементів матриці. Програма запитує розмірність матриці (довільної розмірності) та її елементи. Знаходить та виводить максимальний та мінімальний елементи матриці.

– Варіант 4: Обчислення суми елементів матриці. Програма запитує розмірність матриці (довільної розмірності) та її елементи. Обчислює та виводить суму всіх елементів матриці, а також суму елементів кожного рядка та кожного стовпчика.

4. Створення матриць різних типів. Напишіть програму на Python, яка створює матриці різних типів та розмірностей, заданих користувачем.

– Варіант 1: Створення нульової матриці. Програма запитує розмірність матриці (кількість рядків та стовпчиків). Створює та виводить нульову матрицю заданої розмірності.

– Варіант 2: Створення одиничної матриці. Програма запитує розмірність матриці (розмірність N для NxN одиничної матриці). Створює та виводить одиничну матрицю заданої розмірності.

– Варіант 3: Створення матриці з випадковими числами. Програма запитує розмірність матриці та тип розподілу випадкових чисел (рівномірний чи нормальний). Створює та виводить матрицю, заповнену випадковими числами відповідного розподілу.

– Варіант 4: Перетворення одновимірного масиву на матрицю заданої розмірності. Програма запитує кількість елементів для одновимірного масиву, його елементи, та бажану розмірність матриці (кількість рядків та

стовпчиків, перевірити сумісність з кількістю елементів). Створює одновимірний масив, потім перетворює його на матрицю заданої розмірності та виводить результат.

Завдання 2.

1. Розв'язання простих систем 2x2. Розв'язати наступні системи лінійних рівнянь 2x2 за допомогою бібліотеки NumPy. Для кожного варіанту:

– Задати матрицю коефіцієнтів **A** та вектор вільних членів **b** як масиви NumPy.

– Використати функцію numpy.linalg.solve(A, b) для знаходження розв'язку.

– Вивести на екран розв'язки для x та y з точністю до двох знаків після коми.

– Перевірити розв'язок, підставивши знайдені значення у вихідні рівняння, та вивести результат перевірки (наприклад, різницю між лівою та правою частинами рівнянь).

Варіант 1

$$\begin{cases} x + y = 5 \\ x - y = 1 \end{cases}$$

Варіант 2

$$\begin{cases} 2x - y = 8 \\ x + 3y = -3 \end{cases}$$

Варіант 3

$$\begin{cases} 0.5x + 2y = 7 \\ -x + 4y = 2 \end{cases}$$

Варіант 4

$$\begin{cases} -3x + y = -9 \\ 2x + 2y = 10 \end{cases}$$

2. Розв'язання систем 3x3. Розв'язати наступні системи лінійних рівнянь 3x3 за допомогою бібліотеки NumPy. Для кожного варіанту:

– Задати матрицю коефіцієнтів **A** та вектор вільних членів **b** як масиви NumPy.

– Використати функцію numpy.linalg.solve(A, b) для знаходження розв'язку.

– Вивести на екран розв'язки для x, y та z з точністю до двох знаків після коми.

– Перевірити розв'язок, підставивши знайдені значення у вихідні рівняння, та вивести результат перевірки (наприклад, суму абсолютних різниць між лівими та правими частинами рівнянь).

Варіант 1

$$\begin{cases} x + y + z = 6 \\ 2x - y + z = 3 \\ x - 2y - z = -2 \end{cases}$$

Варіант 2

$$\begin{cases} 3x - y + 2z = 11 \\ x + 2y - z = 3 \\ 2x + y + z = 8 \end{cases}$$

Варіант 3

$$\begin{cases} -x + 2y + 3z = 10 \\ 2x - y + z = 1 \\ x + y - 2z = -5 \end{cases}$$

Варіант 4

$$\begin{cases} 4x + y - z = 2 \\ x - 4y + 2z = -7 \\ 2x + 3y + 3z = 1 \end{cases}$$

3. Системи з параметром: Розв'язати наступні системи лінійних рівнянь з параметром p за допомогою бібліотеки NumPy при заданому значенні p. Для кожного варіанту:

– Задати матрицю коефіцієнтів **A** (залежну від параметра p) та вектор вільних членів **b** як функції від p, що повертають масиви NumPy при заданому p.

– Визначити значення параметра p згідно з варіантом.

– Викликати функції для отримання матриці **A** та вектора **b** для заданого p.

– Використати функцію numpy.linalg.solve(A, b) для знаходження розв’язку.

– Вивести на екран значення параметра p та розв’язки системи для x, y (та z, якщо є) з точністю до трьох знаків після коми.

Варіант 1

$$\begin{cases} px + y = 2 \\ x - y = p \\ p = 2.5 \end{cases}$$

Варіант 2

$$\begin{cases} x + py = 3 \\ 2x - y = -p \\ p = -1.5 \end{cases}$$

Варіант 3

$$\begin{cases} x + y + pz = 4 \\ x - y + z = 1 \\ px + y - z = 0 \\ p = 0.8 \end{cases}$$

Варіант 4

$$\begin{cases} px + 2y - z = 5 \\ x - py + 2z = -2 \\ 2x + y + pz = 1 \\ p = -0.5 \end{cases}$$

Приклад 1: Просте лінійне рівняння. Розглянемо рівняння 3x = 12. Матриця коефіцієнтів A буде масивом [[3]], а вектор вільних членів b буде масивом [12]. Розв’яжіть рівняння за допомогою numpy.linalg.solve() та виведіть розв’язок.

```
a = np.array([[3]])
b = np.array([12])
x = np.linalg.solve(a, b)
print(f"Розв’язок рівняння 3x = 12: x = {x}")
```

Приклад 2: Система двох лінійних рівнянь. Розглянемо систему:
2x + y = 5
x - 3y = -1
Задайте матрицю A та вектор b у вигляді масивів NumPy та розв’яжіть систему за допомогою numpy.linalg.solve(). Виведіть розв’язки для x та y.

```
a = np.array([[2, 1], [1, -3]])
b = np.array([5, -1])
x = np.linalg.solve(a, b)
print(f"Розв’язки системи:\nx = {x[0]:.2f}, y = {x[1]:.2f}") # Форматоване виведення
```

Приклад 3: Система трьох лінійних рівнянь. Розглянемо систему:
x + 2y + z = 8
-x + 3y + 2z = 1
2x - y - z = 3
Аналогічно прикладу 2, задайте матрицю A та вектор b та розв’яжіть систему за допомогою numpy.linalg.solve(). Виведіть розв’язки для x, y та z.

```
a = np.array([[1, 2, 1], [-1, 3, 2], [2, -1, -1]])
b = np.array([8, 1, 3])
x = np.linalg.solve(a, b)
print(f"Розв’язки системи:\nx = {x[0]:.2f}, y = {x[1]:.2f}, z = {x[2]:.2f}")
```

Теоретичні відомості

1. Бібліотека NumPy.

NumPy (Numerical Python) – це фундаментальна бібліотека для наукових обчислень у Python. Вона надає підтримку для великих, багатовимірних масивів та матриць, а також велику колекцію математичних функцій для операцій над цими масивами. NumPy є основою для багатьох інших наукових бібліотек Python, таких як SciPy, Matplotlib та Pandas [5].

Основні переваги NumPy для матричних обчислень:

– Ефективність. Операції NumPy реалізовані на C та Fortran, що забезпечує значно вищу продуктивність порівняно зі стандартними списками Python для числових обчислень.

– Зручність. NumPy надає інтуїтивно зрозумілий та лаконічний синтаксис для роботи з масивами та матрицями.

– Багатий функціонал. Бібліотека містить широкий набір функцій для лінійної алгебри, перетворення Фур'є, випадкових чисел та інших математичних задач.

2. Масиви NumPy (ndarray).

Основний об'єкт NumPy – це однорідний багатовимірний масив, який називається ndarray (N-dimensional array). Матриці в NumPy є двовимірними масивами (ndarray з ndim=2).

Створення матриць в NumPy:

– Зі списку (або списку списків):

```
import numpy as np

# Створення матриці зі списку списків
matrix_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
matrix_np = np.array(matrix_list)
print(matrix_np)
```

– За допомогою функцій NumPy:

- np.zeros(shape): Створення матриці, заповненої нулями. shape - це кортеж, що визначає розмірність (наприклад, (3, 3) для матриці 3x3).
- np.ones(shape): Створення матриці, заповненої одиницями.
- np.eye(N): Створення одиничної матриці розмірності NxN.
- np.diag(v): Створення діагональної матриці з елементами вектора v на головній діагоналі.
- np.random.rand(shape): Створення матриці з випадковими числами з рівномірного розподілу [0, 1).
- np.random.randn(shape): Створення матриці з випадковими числами зі стандартного нормального розподілу.
- np.arange(start, stop, step).reshape(shape): Створення матриці з послідовності чисел, отриманої за допомогою np.arange() та зміною розмірності за допомогою .reshape().

3. Основні матричні операції в NumPy:

– Додавання матриць (+):

```
matrix_A = np.array([[1, 2], [3, 4]])
matrix_B = np.array([[5, 6], [7, 8]])
matrix_sum = matrix_A + matrix_B
print(matrix_sum)
```

– Віднімання матриць (-):

```
matrix_diff = matrix_B - matrix_A
print(matrix_diff)
```

– Скалярне множення (*): Множення матриці на число (скаляр).

```
scalar = 2
matrix_scaled = scalar * matrix_A
print(matrix_scaled)
```

– Елемент-wise (поелементне) множення (*): Множення відповідних елементів двох матриць однакової розмірності.

```
matrix_element_mult = matrix_A * matrix_B
print(matrix_element_mult)
```

- Матричне множення (@ або np.dot()): Стандартне матричне множення (рядок на стовпчик).

```
matrix_product = matrix_A @ matrix_B  # Або np.dot(matrix_A, matrix_B)
print(matrix_product)
```

– Транспонування матриці (.T або np.transpose()):

```
matrix_transposed = matrix_A.T  # Або np.transpose(matrix_A)
print(matrix_transposed)
```

4. Інші корисні функції NumPy для матриць:

– np.linalg.det(matrix): Обчислення визначника матриці.

– np.trace(matrix): Обчислення сліду матриці (суми діагональних елементів).

– matrix.min(), matrix.max(), matrix.sum(): Знаходження мінімального, максимального елемента та суми всіх елементів матриці (або по осях – рядок/стовпчик, якщо вказати axis параметр).

– matrix.reshape(new_shape): Зміна розмірності матриці. Кількість елементів повинна залишатися незмінною.

5. Лінійні рівняння та системи лінійних рівнянь.

Лінійне рівняння – це рівняння першого ступеня відносно однієї або кількох змінних. У найпростішому випадку з однією змінною (x) лінійне рівняння має вигляд: $ax = b$, де a і b – відомі коефіцієнти.

Система лінійних алгебраїчних рівнянь (СЛАР) – це набір з двох або більше лінійних рівнянь, що містять спільні змінні. СЛАР можна представити в матричній формі:

$$A \cdot x = b, \qquad (4.1)$$

де A – матриця коефіцієнтів системи;
x – вектор невідомих змінних (стовпець);
b – вектор вільних членів (стовпець).

Наприклад, система двох лінійних рівнянь:

2x + y = 5
x - 3y = -1

У матричній формі запишеться як:

A = [[2, 1], x = [[x], b = [[5],
[1, -3]] [y]] [-1]]

6. Бібліотека NumPy для лінійної алгебри.

NumPy (Numerical Python) – це потужна бібліотека Python, призначена для ефективних числових обчислень, особливо в галузі лінійної алгебри. NumPy надає:

– Багатовимірні масиви (ndarray). Основна структура даних NumPy для представлення векторів, матриць та тензорів. Масиви NumPy є ефективними для зберігання та обробки числових даних.

– Великий набір математичних функцій. NumPy включає широкий спектр функцій для математичних операцій, включаючи лінійну алгебру, перетворення Фур'є, випадкові числа та багато іншого.

– Функції лінійної алгебри в numpy.linalg. Підмодуль numpy.linalg містить функції для вирішення задач лінійної алгебри, зокрема:

- numpy.linalg.solve(a, b): Розв'язує систему лінійних рівнянь **A·x = b**.
- numpy.linalg.det(a): Обчислює визначник матриці.
- numpy.linalg.inv(a): Обчислює обернену матрицю.
- numpy.linalg.eig(a): Обчислює власні значення та власні вектори матриці.
- та інші.

7. Функція numpy.linalg.solve().

Функція numpy.linalg.solve(a, b) є ключовою функцією NumPy для вирішення СЛАР.

– Синтаксис: x = numpy.linalg.solve(a, b)

- a (аргумент): Квадратна матриця коефіцієнтів **A** (масив NumPy). Матриця **A** повинна бути квадратною і невиродженою (визначник не повинен дорівнювати нулю), щоб система мала єдиний розв'язок.
- b (аргумент): Вектор вільних членів **b** (масив NumPy).
- x (повернене значення): Вектор розв'язків **x** (масив NumPy), якщо система має єдиний розв'язок.

– Розв'язки СЛАР: Функція solve() знаходить розв'язок **x**, який задовольняє рівняння **A·x = b**.

– Випадки розв’язків:

- **Єдиний розв’язок.** Якщо матриця **A** квадратна та невироджена, система має єдиний розв’язок, який solve() і повертає.
- **Невизначена система (безліч розв’язків).** Якщо система має безліч розв’язків (наприклад, рівнянь менше ніж змінних або рівняння лінійно залежні), solve() може повернути один з можливих розв’язків, але **не гарантує** знаходження загального розв’язку чи інформування про невизначеність. Для аналізу таких систем можуть знадобитися інші методи лінійної алгебри.
- **Несумісна система (немає розв’язків).** Якщо система не має розв’язків (рівняння суперечливі), solve() **може видати помилку LinAlgError** або повернути розв’язок, що не є точним розв’язком системи у строгому сенсі (може бути результатом чисельної оптимізації для мінімізації нев’язки). Для діагностики таких систем потрібно аналізувати визначник матриці коефіцієнтів та розширену матрицю.

Контрольні запитання

1. Що таке бібліотека NumPy та для чого вона використовується?
2. Який основний об’єкт NumPy для представлення матриць?
3. Які способи створення матриць в NumPy ви знаєте?
4. Які операції поелементного та матричного множення існують в NumPy? В чому їх відмінність?
5. Як транспонувати матрицю в NumPy?
6. Які функції NumPy можна використовувати для обчислення визначника та сліду матриці?
7. Як змінити розмірність матриці в NumPy?
8. Які переваги використання NumPy для матричних обчислень порівняно зі стандартними списками Python?

ЛАБОРАТОРНА РОБОТА № 5

ПОБУДОВА ГРАФІКІВ З ВИКОРИСТАННЯМ БІБЛІОТЕКИ MATPLOTLIB. ОСНОВИ СИМВОЛЬНИХ ОБЧИСЛЕНЬ З ВИКОРИСТАННЯМ БІБЛІОТЕКИ SYMPY

Мета роботи. Ознайомитися з бібліотекою Matplotlib – основним інструментом для візуалізації даних у Python. Навчитися створювати прості графіки: лінійні графіки, точкові діаграми, стовпчасті діаграми, гістограми, кругові діаграми. Освоїти основні елементи графіків Matplotlib: фігура (figure), область побудови (axes), лінії, точки, підписи, легенда, сітка. Вивчити основні можливості бібліотеки SymPy для символьної математики в Python. Навчитися створювати символьні змінні та математичні вирази в SymPy.

Хід роботи

Завдання 1.

Імпорт бібліотеки Matplotlib. Імпортуйте модуль matplotlib.pyplot на початку вашого блокноту:

```
import matplotlib.pyplot as plt
import numpy as np #  Також імпортуйте NumPy, якщо потрібно для даних
```

1. Побудова лінійних графіків. Побудуйте лінійні графіки функцій за допомогою Matplotlib. Для кожного варіанту:

– Створіть масив значень x у заданому діапазоні.

– Обчисліть відповідні значення y за заданою функцією.

– Побудуйте лінійний графік plt.plot(x, y).

– Додайте заголовок графіку, підписи осей, сітку.

– Налаштуйте колір та стиль лінії графіку.

– **Варіант 1:** Функція $y = x^2$ для x від -5 до 5.

– **Варіант 2:** Функція $y = \cos(x)$ для x від 0 до 4π.

– **Варіант 3:** Функція $y = \exp(-x) \cdot \sin(2 \cdot \pi \cdot x)$ для x від 0 до 5. (exp() - експонента, np.exp())

– **Варіант 4:** Дві функції на одному графіку: $y_1 = \sin(x)$ та $y_2 = \cos(x)$ для x від 0 до 2π. Додайте легенду, що відображає мітки для кожної лінії.

2. Побудова точкових діаграм. Побудуйте точкові діаграми для представлення залежності між двома наборами даних. Для кожного варіанту:

– Створіть два масиви даних x та y (наприклад, випадкові дані або дані з певним розподілом).

– Побудуйте точкову діаграму plt.scatter(x, y).

– Додайте заголовок діаграми, підписи осей.

– Налаштуйте колір та стиль маркерів точок, їх розмір.

– **Варіант 1:** Залежність між випадковими значеннями x та y (створити два масиви випадкових чисел однакового розміру).

– **Варіант 2:** Залежність між y = sqrt(x) та x для x від 0 до 100 (обчислити y як квадратний корінь з x).

– **Варіант 3:** Зобразити на діаграмі набір точок з координатами, заданими у варіанті (викладач надає набір координат точок).

- **Варіант 4:** Побудувати точкову діаграму, де колір та/або розмір точок залежить від третьої змінної (наприклад, задати масив z і використовувати c=z або s=z в plt.scatter()).

3. Побудова стовпчастих діаграм та гістограм. Побудуйте стовпчасті діаграми та гістограми для візуалізації розподілу категоріальних та числових даних. Для кожного варіанту:

– Підготуйте дані для стовпчастої діаграми (категорії та відповідні значення) або гістограми (набір числових даних).

– Побудуйте стовпчасту діаграму plt.bar(категорії, значення) або гістограму plt.hist(дані).

– Додайте заголовок діаграми, підписи осей.

– Налаштуйте колір стовпців/гістограми, ширину стовпців, межі стовпців.

– **Варіант 1:** Стовпчаста діаграма для порівняння продажів різних продуктів (назви продуктів та обсяги продажів задані).

– **Варіант 2:** Гістограма розподілу випадкових чисел з нормальним розподілом (створити набір випадкових чисел з np.random.randn()).

– **Варіант 3:** Стовпчаста діаграма для відображення кількості голосних літер у різних словах (слова задані).

– **Варіант 4:** Гістограма розподілу оцінок студентів (набір оцінок заданий). Розбити оцінки на діапазони (bins) для гістограми.

4. Побудова кругових діаграм. Побудуйте кругові діаграми для відображення частин цілого. Для кожного варіанту:

– Підготуйте дані для кругової діаграми (розміри частин та їх мітки).

– Побудуйте кругову діаграму plt.pie(розміри, labels=мітки).

– Додайте заголовок діаграми.

– Налаштуйте кольори секторів, мітки секторів, відображення відсотків, тінь, початок кута повороту діаграми.

– **Варіант 1:** Розподіл часу доби (сон, робота/навчання, дозвілля, інше) у відсотках.

– **Варіант 2:** Розподіл витрат бюджету сім'ї на різні категорії (харчування, комунальні послуги, транспорт, розваги, інше) у відсотках або грошовому вираженні.

– **Варіант 3:** Розподіл населення країни за віковими групами (вікові групи та відсотки задані).

– **Варіант 4:** Кругова діаграма, що відображає частку кожного виду транспорту в загальному обсязі пасажирських перевезень (види транспорту та частки задані).

Завдання 2.

Імпорт бібліотеки SymPy: На початку блокноту імпортуйте бібліотеку SymPy:

```
import sympy
```

Для числових результатів (наприклад, обчислення границь, визначених інтегралів), порівняйте їх з наближеними числовими обчисленнями (за допомогою evalf() або інших методів).

1. Символьні вирази та операції спрощення. Виконайте наступні операції з символьними виразами за допомогою SymPy. Для кожного варіанту:

– Створіть символьні змінні.

– Створіть заданий символьний вираз.

– Використайте функцію sympy.simplify() для спрощення виразу.

– Виведіть на екран початковий та спрощений вирази.

– **(Додатково):** Підставте числові значення для змінних у початковий та спрощений вирази (використовуючи .subs() та .evalf()) та переконайтеся, що їх числові значення збігаються.

– **Варіант 1:** Спростити вираз: $(x^2 - 1) / (x - 1)$.

– **Варіант 2:** Спростити вираз: $\sin(x)^2 + \cos(x)^2$.

– **Варіант 3:** Спростити вираз: $\text{sqrt}((x \cdot y)^2)$ (Врахуйте, що за замовчуванням SymPy не знає, що x та y додатні, тому спрощення може бути не до $x \cdot y$, але можна додати при оголошенні символів positive=True).

– **Варіант 4:** Спростити вираз: $\log(x) - \log(y) + \log(y/x)$.

2. Розкриття дужок та факторизація поліномів. Виконайте розкриття дужок та факторизацію для заданих поліномів за допомогою SymPy. Для кожного варіанту:

– Створіть символьну змінну.

– Створіть заданий поліноміальний вираз.

– Використайте метод .expand() для розкриття дужок.

– Використайте функцію sympy.factor() для факторизації полінома.

– Виведіть на екран початковий, розгорнутий та факторизований вирази.

– **Варіант 1:** Поліном: $(x + 2)^4$.

– **Варіант 2:** Поліном: $(x - 1) \cdot (x + 1) \cdot (x^2 + 1)$.

– **Варіант 3:** Поліном: $x^3 - 6 \cdot x^2 + 11 \cdot x – 6$.

– **Варіант 4:** Поліном: $(x + y)^2 - (x - y)^2$.

3. Символьне диференціювання та інтегрування. Обчисліть похідні та інтеграли для заданих функцій за допомогою SymPy. Для кожного варіанту:

– Створіть символьну змінну.

– Створіть заданий вираз функції.

– Використайте метод .diff() для обчислення похідної за змінною x.

– Використайте функцію sympy.integrate() для обчислення невизначеного інтегралу за змінною x.

– **(Додатково):** Обчисліть визначений інтеграл в заданих межах (наприклад, від 0 до 1) для варіантів 1 та 2.

– Виведіть на екран функцію, її похідну та інтеграл.

– **Варіант 1:** Функція: $x^3 \cdot$sympy.exp(x).

– **Варіант 2:** Функція: sympy.cos(x)/x.

– **Варіант 3:** Функція: sympy.tan(x).

– **Варіант 4:** Функція: sympy.log(x^2 + 1).

4. Розв'язання рівнянь та систем рівнянь. Розв'яжіть задані рівняння та системи рівнянь за допомогою SymPy. Для кожного варіанту:

– Створіть символьні змінні.

– Створіть задане рівняння (використовуючи sympy.Eq) або систему рівнянь (список рівнянь).

– Використайте функцію sympy.solve() для розв'язання рівняння/системи відносно вказаних змінних.

– Виведіть на екран рівняння/систему та знайдені розв'язки.

– **(Додатково):** Перевірте розв'язки, підставивши їх у вихідні рівняння та спростивши отримані вирази (за допомогою .subs() та sympy.simplify()).

– **Варіант 1:** Рівняння: $x^2 - 5 \cdot x + 6 = 0$.

– **Варіант 2:** Рівняння: sympy.cos(x) - x = 0 (Спробуйте знайти символьний розв'язок, якщо можливо, і числовий наближений розв'язок, якщо символьний не знайдено. Зверніть увагу, що деякі рівняння не мають аналітичних розв'язків).

– **Варіант 3:** Система рівнянь:

$$\begin{cases} x + y = 7 \\ x - y = 3 \end{cases}$$

– **Варіант 4:** Система рівнянь:

$$\begin{cases} x^2 + y = 5 \\ 2x - y = 1 \end{cases}$$

Теоретичні відомості

1. Бібліотека Matplotlib.

Matplotlib – це потужна бібліотека Python для створення статичних, інтерактивних та анімованих візуалізацій у Python. Matplotlib широко використовується для наукових обчислень, аналізу даних та машинного навчання для графічного представлення результатів [6]-[7].

Основні модулі Matplotlib:

– **matplotlib.pyplot:** Найбільш часто використовуваний модуль, що надає інтерфейс, подібний до MATLAB, для створення графіків та діаграм. Зазвичай імпортується як plt: import matplotlib.pyplot as plt.

– **matplotlib.figure:** Модуль, що визначає клас Figure – контейнер для всіх елементів графіку, включаючи області побудови, підписи, легенди тощо.

– **matplotlib.axes:** Модуль, що визначає клас Axes – область для побудови графіків, де відображаються дані. Кожна фігура може містити одну або кілька областей Axes.

– **matplotlib.lines, matplotlib.patches, matplotlib.text, matplotlib.image:** Модулі, що містять класи для представлення різних графічних примітивів: лінії, фігури, текст, зображення тощо.

2. Основи побудови графіків за допомогою matplotlib.pyplot.

Основний підхід до побудови графіків за допомогою pyplot полягає у послідовному виклику функцій для додавання елементів на графік та налаштування їхнього вигляду.

2.1 Основні кроки створення графіка:

– Імпорт модуля pyplot: import matplotlib.pyplot as plt

– Підготовка даних: створення даних для графіку у вигляді списків або масивів NumPy (вектори x та y для 2D графіків, або 1D вектори для гістограм, стовпчастих діаграм тощо).

– Створення фігури та області побудови (неявно або явно):

- Багато функцій pyplot (наприклад, plt.plot(), plt.scatter()) автоматично створюють поточну фігуру (Figure) та область побудови (Axes), якщо вони ще не існують.

Явне створення фігури та області побудови:

```
fig, ax = plt.subplots() # Створення фігури (fig) та однієї області побудови (ax)
```

2.2 Побудова графічних елементів. Виклик функцій pyplot для додавання графічних елементів до області побудови:

- plt.plot(x, y, ...): Лінійний графік (або точковий, якщо задати маркери).
- plt.scatter(x, y, ...): Точкова діаграма.
- plt.bar(x, height, ...): Стовпчаста діаграма.
- plt.hist(x, ...): Гістограма.
- plt.pie(data, ...): Кругова діаграма.
- та інші функції для різних типів графіків.

2.3 Налаштування графіку (необов'язково, але рекомендовано). Виклик функцій pyplot для налаштування зовнішнього вигляду та елементів графіку:

- ax.set_title("Заголовок графіку") або plt.title("Заголовок графіку"): Заголовок графіку.
- ax.set_xlabel("Підпис осі X") або plt.xlabel("Підпис осі X"): Підпис осі X.
- ax.set_ylabel("Підпис осі Y") або plt.ylabel("Підпис осі Y"): Підпис осі Y.

- ax.legend() або plt.legend(): Легенда (для графіків з декількома лініями або елементами).
- ax.grid(True) або plt.grid(True): Сітка.
- ax.set_xlim(xmin, xmax), ax.set_ylim(ymin, ymax) або plt.xlim(xmin, xmax), plt.ylim(ymin, ymax): Межі осей.
- Налаштування кольорів, стилів ліній, маркерів, шрифтів та інших властивостей графічних елементів (через аргументи функцій побудови або методи об'єктів Axes, Line2D, Text тощо).

2.4 Відображення графіку: plt.show(). Відображає створений графік у окремому вікні або вбудовано (залежно від середовища виконання).

3. Основні типи графіків Matplotlib та приклади коду.

– Лінійний графік (plot()). Використовується для відображення залежності однієї змінної від іншої у вигляді лінії (рис. 5.1).

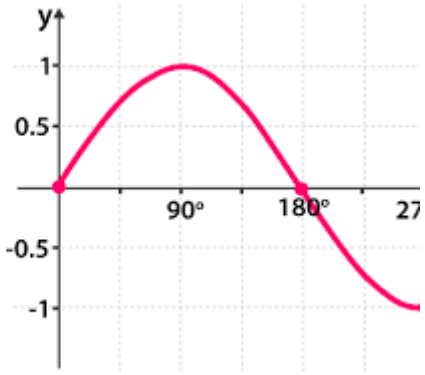


Рисунок 5.1 – Line plot of sine function

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 100) # 100 точок від 0 до 10
y = np.sin(x)

plt.plot(x, y) # Побудова лінійного графіку
plt.title("Графік синуса")
plt.xlabel("Вісь X")
plt.ylabel("Вісь Y")
plt.grid(True)
plt.show()
```

– Точкова діаграма (scatter()). Використовується для відображення окремих точок даних на площині (рис. 5.2).

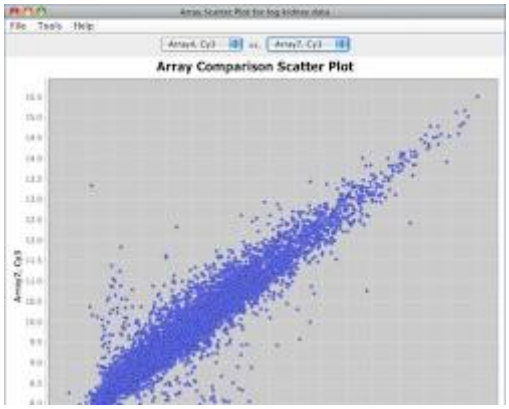


Рисунок 5.2 – Scatter plot of random data

```
import matplotlib.pyplot as plt
import numpy as np

x = np.random.rand(50) # 50 випадкових точок x
y = np.random.rand(50) # 50 випадкових точок y

plt.scatter(x, y) # Побудова точкової діаграми
plt.title("Точкова діаграма випадкових даних")
plt.xlabel("Вісь X")
plt.ylabel("Вісь Y")
plt.show()
```

– **Стовпчаста діаграма (bar()).** Використовується для порівняння значень категорій або груп даних (рис. 5.3).

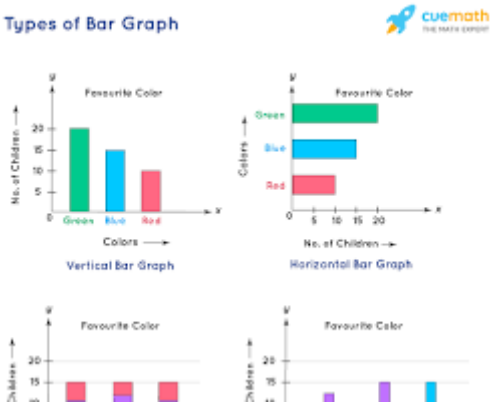


Рисунок 5.3 – Bar chart of categories

```
import matplotlib.pyplot as plt

категорії = [‘A’, ‘B’, ‘C’, ‘D’]
значення = [25, 40, 30, 55]

plt.bar(категорії, значення) # Побудова стовпчастої діаграми
```

```
plt.title("Стовпчаста діаграма категорій")
plt.xlabel("Категорії")
plt.ylabel("Значення")
plt.show()
```

– **Гістограма (hist()).** Використовується для відображення розподілу частот значень у наборі даних (рис. 5.4).

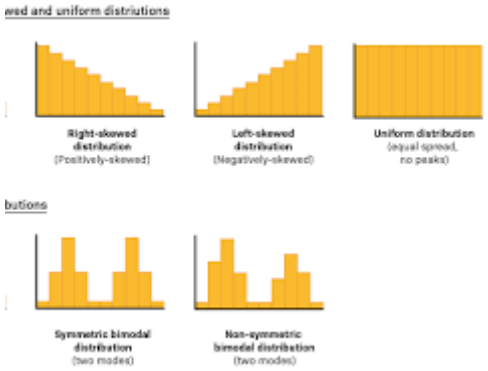


Рисунок 5.4 – Histogram of data distribution

```
import matplotlib.pyplot as plt
import numpy as np

data = np.random.randn(1000) # 1000 випадкових значень з нормальним розподілом

plt.hist(data, bins=30) # Побудова гістограми з 30 стовпцями
plt.title("Гістограма розподілу даних")
plt.xlabel("Значення")
plt.ylabel("Частота")
plt.show()
```

– **Кругова діаграма (pie()).** Використовується для відображення частин цілого, показуючи відносний внесок кожної категорії у загальну суму (рис. 5.5).

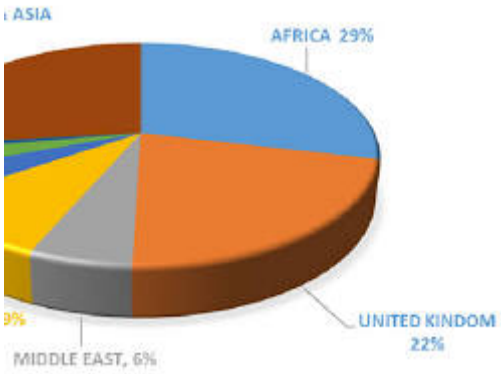


Рисунок 5.5 – Pie chart of fruit distribution

```
import matplotlib.pyplot as plt

sizes = [30, 25, 15, 10, 20]
labels = [‘Яблука’, ‘Банани’, ‘Вишні’, ‘Фініки’, ‘Виноград’]
colors = [‘red’, ‘yellow’, ‘pink’, ‘cyan’, ‘purple’]

plt.pie(sizes, labels=labels, colors=colors, autopct=‘%1.1f%%’, shadow=True, startangle=90) # Побудова кругової діаграми
plt.title("Кругова діаграма розподілу фруктів")
plt.axis(‘equal’) #  Рівні осі для правильного кола
plt.show()
```

4. Налаштування графіків.

Matplotlib надає широкі можливості для налаштування зовнішнього вигляду графіків:

– Колір (color). Задається назвою кольору (наприклад, ‘red’, ‘blue’, ‘green’) або кодом кольору (RGB, HEX).

```
plt.plot(x, y, color=‘red’) # Лінія червоного кольору
plt.bar(категорії, значення, color=[‘blue’, ‘green’, ‘red’, ‘cyan’]) # Різні кольори стовпців
```

– Стиль лінії (linestyle або ls). Задається рядковим кодом стилю (наприклад, ‘-’, ‘--’, ‘:’, ‘-.’, ‘None’).

```
plt.plot(x, y, linestyle=‘--’) # Пунктирна лінія
```

– Товщина лінії (linewidth або lw). Задається числовим значенням товщини в точках.

```
plt.plot(x, y, linewidth=2.5) # Лінія товщиною 2.5 точок
```

– Маркер (marker). Задається символом маркера (наприклад, ‘o’, ‘.’, ‘,’, ‘x’, ‘+’, ‘s’, ‘d’, ‘^’, ‘v’).

```
plt.plot(x, y, marker=‘o’) # Лінія з круглими маркерами
plt.scatter(x, y, marker=‘x’) # Точкові маркери у вигляді хрестиків
```

– Розмір маркера (markersize або ms). Задається числовим значенням розміру в точках.

```
plt.plot(x, y, marker=‘o’, markersize=8) # Великі круглі маркери
```

– Колір маркера (markerfacecolor або mfc), колір контуру маркера (markeredgecolor або mec), товщина контуру маркера (markeredgewidth або mew). Налаштування зовнішнього вигляду маркерів.

```
plt.plot(x, y, marker='o', mfc='yellow', mec='black', mew=1.5) # Жовті маркери з чорним контуром
```

– Шрифт підписів та заголовків (fontdict). Задається словником параметрів шрифту (розмір, колір, стиль, сімейство шрифтів тощо).

```
title_font = {'fontsize': 16, 'fontweight': 'bold', 'color': 'darkblue'}
axis_font = {'fontsize': 14, 'color': 'gray'}
plt.title("Заголовок графіку", fontdict=title_font)
plt.xlabel("Вісь X", fontdict=axis_font)
plt.ylabel("Вісь Y", fontdict=axis_font)
```

– Легенда (legend()). Для відображення легенди потрібно задати мітку (label) для кожного елемента графіку (наприклад, при виклику plt.plot()), а потім викликати plt.legend().

```
plt.plot(x, y1, label='Лінія 1') # Задаємо мітку для першої лінії
plt.plot(x, y2, label='Лінія 2') # Задаємо мітку для другої лінії
plt.legend() # Відображення легенди
```

– Сітка (grid()). Додає сітку на область побудови.

```
plt.grid(True, linestyle='--', alpha=0.5) # Пунктирна сітка напівпрозорого кольору
```

5. Символьні обчислення та бібліотека SymPy.

Символьні обчислення (комп'ютерна алгебра) – це область науки, що займається розробкою алгоритмів та програмного забезпечення для маніпулювання математичними виразами в символьному вигляді. На відміну від числових обчислень, які працюють з наближеними числовими значеннями, символьні обчислення оперують з математичними об'єктами в їх аналітичній формі – символами, змінними, функціями та точними математичними виразами.

Бібліотека SymPy – це потужна бібліотека Python для символьних обчислень. SymPy є вільною бібліотекою з відкритим кодом, написаною на Python, що надає широкий спектр можливостей для символьної математики [8]:

– Символьна алгебра: спрощення виразів, розкриття дужок, факторизація, підстановка, робота з поліномами та раціональними функціями.

– Обчислення: диференціювання, інтегрування, обчислення границь, суми, добутки, ряди Тейлора.

– Розв'язання рівнянь: розв'язання алгебраїчних та диференціальних рівнянь, систем рівнянь.

– **Лінійна алгебра:** матриці, вектори, операції з матрицями, визначники, власні значення, розв'язання систем лінійних рівнянь (символьно).

– **Теорія множин, теорія чисел, комбінаторика, геометрія, фізика та інші розділи математики.**

– **Розширення можливостей:** SymPy легко розширюється користувацькими функціями та типами, що дозволяє використовувати її в різноманітних наукових та інженерних задачах.

6. Основні поняття та операції SymPy:

– **Символи.** В SymPy символьні змінні потрібно оголошувати явно за допомогою функції sympy.symbols().

```
import sympy

x, y, z = sympy.symbols('x y z') # Оголошення символів x, y, z
f = sympy.Symbol('f', function=True) # Оголошення символу функції f
```

– **Вирази.** Математичні вирази в SymPy створюються комбінуванням символів, чисел та математичних операцій (+, -, *, /, **, ** , функції SymPy).

```
expression1 = x**2 + 2*x*y + y**2 # Вираз x^2 + 2xy + y^2
expression2 = sympy.sin(x) + sympy.cos(y) # Вираз sin(x) + cos(y)
expression3 = f(x) + f(y).diff(y) # Вираз f(x) + d(f(y))/dy
```

– **Підстановка значень (subs()).** Метод .subs() дозволяє підставляти числові значення або інші символьні вирази замість символів у виразі.

```
expression_sub = x**2 + y
substitution1 = expression_sub.subs({x: 2, y: 3}) # Підстановка x=2, y=3
substitution2 = expression_sub.subs(x, z+1) # Підстановка x = z+1
```

– **Чисельне обчислення (evalf()).** Метод .evalf() обчислює числове значення символьного виразу з заданою точністю.

```
expression_evalf = sympy.pi**2
numerical_value = expression_evalf.evalf(n=10) # Числове значення Pi^2 з 10 знаками
```

– **Спрощення виразів (simplify()).** Функція sympy.simplify() намагається спростити вираз, використовуючи різні алгебраїчні перетворення.

```
expression_simplify = (x**2 + 2*x + 1) / (x + 1)
simplified_expression = sympy.simplify(expression_simplify) # Спрощення до x + 1
```

– **Розкриття дужок (expand()).** Метод .expand() розкриває дужки у виразі.

```
expression_expand = (x + 1)**3
expanded_expression = expression_expand.expand() # Розкриття дужок: x**3 + 3*x**2 + 3*x + 1
```

– **Факторизація (factor()).** Функція sympy.factor() факторизує поліноміальний вираз.

```
expression_factor = x**3 + 3*x**2 + 3*x + 1
factorized_expression = sympy.factor(expression_factor) # Факторизація: (x + 1)**3
```

– **Диференціювання (diff()).** Метод .diff() обчислює похідну виразу за вказаною змінною.

```
 expression_diff = sympy.sin(x**2)
derivative = expression_diff.diff(x) # Обчислення похідної d/dx(sin(x^2))
```

– **Інтегрування (integrate()).** Функція sympy.integrate() обчислює невизначений або визначений інтеграл виразу за вказаною змінною.

```
expression_integrate = x**2 * sympy.cos(x)
integral = sympy.integrate(expression_integrate, x) # Обчислення інтегралу ∫x^2*cos(x) dx

definite_integral = sympy.integrate(sympy.sin(x), (x, 0, sympy.pi)) # Обчислення ∫[0, Pi] sin(x) dx
```

– **Границі (limit()).** Функція sympy.limit() обчислює границю виразу при наближенні змінної до заданої точки.

```
expression_limit = sympy.sin(x) / x
limit_val = sympy.limit(expression_limit, x, 0) # Обчислення границі lim(x->0) sin(x)/x
```

– **Розв’язання рівнянь (solve()).** Функція sympy.solve() розв’язує рівняння або системи рівнянь.

```
equation = sympy.Eq(x**2 - 4, 0) # Рівняння x^2 - 4 = 0
solutions = sympy.solve(equation, x) # Розв’язання рівняння відносно x

system_equations = [sympy.Eq(x + y, 5), sympy.Eq(x - y, 1)] # Система рівнянь
system_solutions = sympy.solve(system_equations, [x, y]) # Розв’язання системи відносно x, y
```

Контрольні запитання

1. Що таке бібліотека Matplotlib та для чого вона використовується?
2. Який основний модуль Matplotlib використовується для побудови графіків? Як його імпортувати?
3. Які основні кроки створення графіка Matplotlib?
4. Які основні типи графіків можна побудувати за допомогою Matplotlib? Наведіть приклади функцій для кожного типу.
5. Як додати заголовок та підписи осей до графіку Matplotlib?
6. Як налаштувати колір, стиль лінії, маркери та інші параметри графічних елементів у Matplotlib?
7. Як додати легенду до графіку? Для чого вона потрібна?
11. Як оголосити символьні змінні в SymPy?
12. Як створити символьний вираз в SymPy?
13. Які методи використовуються в SymPy для спрощення, розкриття дужок та факторизації виразів?
14. Як обчислити похідну та інтеграл символьного виразу в SymPy?
15. Як обчислити границю функції в SymPy?

ЛАБОРАТОРНА РОБОТА № 6

ОБРОБКА ДОСЛІДНИХ ДАНИХ ЗА ДОПОМОГОЮ МОВИ ПРОГРАМУВАННЯ PYTHON

Мета роботи. Ознайомитися з основними методами обробки дослідних даних за допомогою мови програмування Python. Вивчити можливості бібліотеки NumPy для виконання інтерполяції, апроксимації, фільтрації та статистичного аналізу даних. Закріпити навички візуалізації оброблених даних за допомогою бібліотеки Matplotlib для наочного представлення результатів.

Хід роботи

Завдання 1.

Імпорт бібліотек. На початку блокноту імпортуйте необхідні бібліотеки:

```
import numpy as np
import matplotlib.pyplot as plt
```

1. Інтерполяція даних. Виконайте інтерполяцію заданих даних різними методами за допомогою NumPy. Для кожного варіанту:

– Задайте вихідні дані у вигляді масивів x_data та y_data згідно з варіантом.

– Виконайте лінійну інтерполяцію за допомогою numpy.interp().

– Виконайте поліноміальну інтерполяцію поліномом заданого степеня за допомогою numpy.polyfit() та numpy.poly1d().

– Побудуйте графік, що відображає вихідні дані, лінійну інтерполяцію та поліноміальну інтерполяцію. Додайте легенду, підписи осей, заголовок.

– **Варіант 1:** x_data = [1, 2, 3, 4, 5], y_data = [2.5, 3.1, 3.8, 4.5, 5.2], степінь полінома 2.

– **Варіант 2:** x_data = [0, 1, 2, 3, 4, 5], y_data = [0, 1, 0.5, 2, 0.8, 3], степінь полінома 3.

– **Варіант 3:** x_data = [0.5, 1.5, 2.5, 3.5, 4.5], y_data = [1.8, 2.2, 2.7, 3.3, 4.0], степінь полінома 1.

– **Варіант 4:** x_data = [2, 4, 6, 8, 10], y_data = [5, 3, 6, 4, 7], степінь полінома 4.

2. Апроксимація даних. Виконайте поліноміальну апроксимацію даних, що містять шум, за допомогою NumPy. Для кожного варіанту:

– Згенеруйте набір даних y_data_noisy, додавши випадковий шум до деякої заданої функції y_data_ideal (згідно з варіантом). Використовуйте x_data з відповідного варіанту Завдання 1.

– Виконайте поліноміальну апроксимацію y_data_noisy поліномом заданого степеня за допомогою numpy.polyfit().

– Побудуйте графік, що відображає зашумлені дані, ідеальну функцію (якщо задана) та апроксимуючий поліном. Додайте легенду, підписи осей, заголовок.

– **Варіант 1:** y_data_ideal = x_data**2, додати нормальний шум зі стандартним відхиленням 0.5, степінь полінома для апроксимації 2.

– **Варіант 2:** y_data_ideal = np.sin(x_data), додати рівномірний шум в діапазоні [-0.3, 0.3], степінь полінома для апроксимації 5.

– **Варіант 3:** y_data_ideal = 2*x_data + 1, додати нормальний шум зі стандартним відхиленням 0.2, степінь полінома для апроксимації 1.

– **Варіант 4:** y_data_ideal – задати функцію на власний вибір (наприклад, комбінацію полінома та тригонометричної функції), додати шум за власним вибором, степінь полінома для апроксимації 3.

3. Статистичний аналіз даних: Виконайте статистичний аналіз заданого набору даних за допомогою NumPy. Для кожного варіанту:

– Завантажте або згенеруйте набір даних згідно з варіантом.

– Обчисліть основні статистичні характеристики даних: середнє, медіану, стандартне відхилення, мінімум, максимум, 25-й та 75-й перцентилі. Виведіть ці значення на екран.

– Побудуйте гістограму розподілу даних. Додайте заголовок, підписи осей.

– **(Для варіантів 3 та 4):** Обчисліть коефіцієнт кореляції між двома наборами даних (x та y) та виведіть його значення. Прокоментуйте, чи є кореляція між даними.

– **Варіант 1:** Набір випадкових чисел з нормальним розподілом (розміром 1000).

– **Варіант 2:** Дані про температуру повітря за місяць (завантажити з файлу або задати масивом).

– **Варіант 3:** Два набори випадкових даних з позитивною кореляцією (наприклад, x = np.random.rand(50), y = x + 0.2 * np.random.randn(50)).

– **Варіант 4:** Два набори даних, між якими відсутня кореляція (наприклад, два незалежні набори випадкових чисел).

Теоретичні відомості

1. Обробка дослідних даних та бібліотека NumPy.

Обробка дослідних даних є важливим етапом наукових досліджень та інженерної практики. Вона включає в себе ряд методів, спрямованих на отримання корисної інформації з сирих даних, їх очищення від шуму, виявлення закономірностей та представлення результатів у наочному вигляді.

Бібліотека NumPy (Numerical Python) є фундаментальною бібліотекою Python для наукових обчислень. Вона надає ефективні інструменти для роботи з масивами даних, а також широкий набір функцій для математичних, статистичних та інженерних розрахунків, включаючи обробку даних [9].

2. Інтерполяція та апроксимація даних.

Інтерполяція – це метод знаходження проміжних значень функції за відомими дискретними значеннями. Інтерполяція дозволяє "відновити" функцію між заданими точками, припускаючи певну гладкість або форму залежності. NumPy надає функції для різних видів інтерполяції в модулі numpy.interp та scipy.interpolate (частина бібліотеки SciPy, яку часто використовують разом з NumPy). Найпростіші види інтерполяції [10]:

– Лінійна інтерполяція (numpy.interp()): з'єднує сусідні відомі точки прямолінійними відрізками (рис. 6.1).

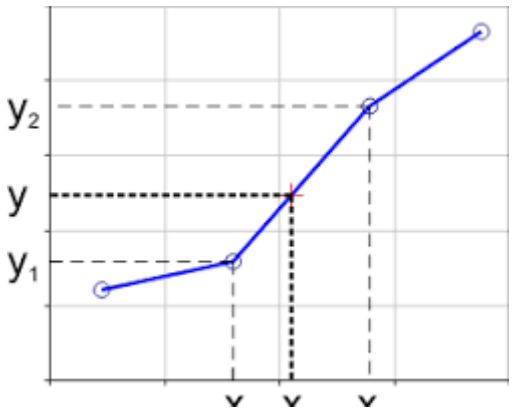


Рисунок 6.1 – Line plot showing original data points and linear interpolation

```
import numpy as np
import matplotlib.pyplot as plt

x_data = np.array([1, 2, 3, 4, 5])
y_data = np.array([2, 4, 1, 3, 5])

x_interp = np.linspace(x_data.min(), x_data.max(), 100) # Точки для інтерполяції
y_interp = np.interp(x_interp, x_data, y_data) # Лінійна інтерполяція

plt.plot(x_data, y_data, 'o', label='Вихідні дані') # Вихідні точки
plt.plot(x_interp, y_interp, '-', label='Лінійна інтерполяція') # Інтерполяція
plt.legend()
plt.grid(True)
plt.show()
```

– Поліноміальна інтерполяція (numpy.polyfit(), numpy.poly1d()): апроксимує дані поліномом заданого степеня, що проходить через усі задані точки (рис. 6.2).

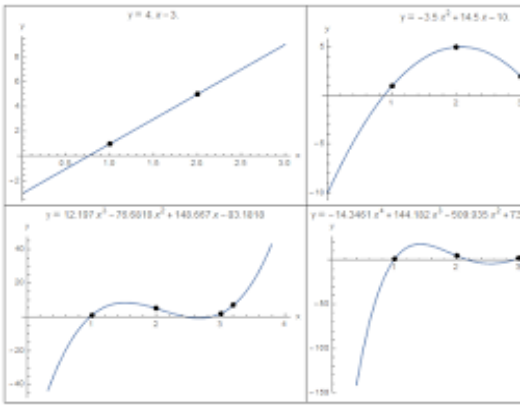


Рисунок 6.2 – Line plot showing original data points and polynomial interpolation

```
import numpy as np
import matplotlib.pyplot as plt

x_data = np.array([1, 2, 3, 4, 5])
y_data = np.array([2, 4, 1, 3, 5])

poly_degree = 3 # Степінь полінома
poly_coefficients = np.polyfit(x_data, y_data, poly_degree) # Коефіцієнти полінома
poly_function = np.poly1d(poly_coefficients) # Поліноміальна функція

x_interp = np.linspace(x_data.min(), x_data.max(), 100)
y_interp = poly_function(x_interp) # Обчислення значень полінома

plt.plot(x_data, y_data, ‘o’, label=‘Вихідні дані’)
plt.plot(x_interp, y_interp, ‘-’, label=f’Поліноміальна інтерполяція, степінь {poly_degree}’)
plt.legend()
plt.grid(True)
plt.show()
```

Апроксимація (наближення) даних – метод знаходження наближеної функції, що "найкраще" описує загальну тенденцію в даних, але не обов’язково проходить через усі задані точки. Апроксимація використовується, коли дані містять шум або похибки вимірювання.

– Поліноміальна апроксимація (метод найменших квадратів, МНК) (numpy.polyfit()): знаходить поліном заданого степеня, який мінімізує суму квадратів відхилень між значеннями полінома та вихідними даними (рис. 6.3).

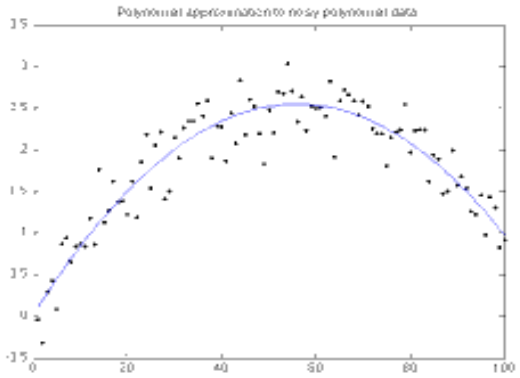


Рисунок 6.3 – Line plot showing noisy data and polynomial approximation

```
import numpy as np
import matplotlib.pyplot as plt

x_data = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
y_data_noisy = np.array([1.5, 3.8, 2.5, 5.2, 4.8, 6.1, 5.9, 7.5, 8.2, 9.1]) # Дані з шумом

poly_degree = 2 # Степінь апроксимуючого полінома
poly_coefficients = np.polyfit(x_data, y_data_noisy, poly_degree) # МНК апроксимація
poly_function = np.poly1d(poly_coefficients)

x_approx = np.linspace(x_data.min(), x_data.max(), 100)
y_approx = poly_function(x_approx) # Значення апроксимуючого полінома

plt.plot(x_data, y_data_noisy, 'o', label='Дані з шумом')
plt.plot(x_approx, y_approx, '-', label=f'Поліноміальна апроксимація, степінь {poly_degree}')
plt.legend()
plt.grid(True)
plt.show()
```

3. Статистична обробка даних за допомогою NumPy.

NumPy надає широкий набір функцій для статистичного аналізу даних:

– Основні статистичні характеристики:

- np.mean(data): Середнє значення.
- np.median(data): Медіана.
- np.std(data): Стандартне відхилення.
- np.var(data): Дисперсія.
- np.min(data), np.max(data): Мінімальне та максимальне значення.
- np.percentile(data, percentile): Перцентиль (заданий відсоток).

```
import numpy as np
```

```
data = np.array([1, 5, 2, 7, 3, 8, 4, 9, 5])
mean_val = np.mean(data)
median_val = np.median(data)
std_dev = np.std(data)
min_val = np.min(data)
max_val = np.max(data)
percentile_75 = np.percentile(data, 75)
print(f"Середнє: {mean_val:.2f}")
print(f"Медіана: {median_val}")
print(f"Стандартне відхилення: {std_dev:.2f}")
print(f"Мінімум: {min_val}")
print(f"Максимум: {max_val}")
print(f"75-й перцентиль: {percentile_75}")
```

Гістограми (numpy.histogram()). Функція numpy.histogram() обчислює гістограму набору даних, розділяючи діапазон значень на задану кількість інтервалів (bins) та підраховуючи кількість точок, що потрапляють в кожен інтервал (рис. 6.4).

CUSTOMER WAITING TIME IN SECONDS (n=32)							
10.4	12.0	18.7	15.9	11.8	12.0	17.5	11.3
10.9	12.4	11.4	10.7	10.2	13.9	13.0	12.7
12.5	14.3	10.4	16.4	11.4	10.6	13.9	11.2
17.3	11.4	11.2	20.3	19.9	20.0	14.2	11.6

Рисунок 6.4 – Histogram of data

```
import numpy as np
import matplotlib.pyplot as plt

data = np.random.randn(1000) # Випадкові дані

hist, bin_edges = np.histogram(data, bins=20) # Гістограма з 20 інтервалами

plt.hist(data, bins=20) # Побудова гістограми за допомогою matplotlib.pyplot.hist()
plt.title("Гістограма даних")
plt.xlabel("Значення")
plt.ylabel("Частота")
plt.show()
```

Кореляція (numpy.corrcoef()). Функція numpy.corrcoef(x, y) обчислює матрицю кореляції Пірсона між двома наборами даних x та y. Діагональні елементи матриці – це кореляція змінної з самою собою (завжди 1), а недіагональні елементи – коефіцієнт кореляції між x та y.

```
import numpy as np

x_data = np.random.rand(50)
y_data = x_data + 0.5 * np.random.randn(50) # y залежить від x + шум
correlation_matrix = np.corrcoef(x_data, y_data)
correlation_coefficient = correlation_matrix[0, 1] # Коефіцієнт кореляції між x та y
print(f"Коефіцієнт кореляції між x та y: {correlation_coefficient:.3f}")
```

Контрольні запитання

1. Для чого використовується інтерполяція даних? Які види інтерполяції ви знаєте, реалізовані в NumPy?
2. В чому відмінність між інтерполяцією та апроксимацією даних? Коли доцільно використовувати апроксимацію?
3. Що таке метод найменших квадратів (МНК)? Як він використовується для поліноміальної апроксимації в NumPy?
4. Для чого потрібна фільтрація даних? Що таке фільтр ковзного середнього? Як він реалізується в NumPy?
5. Які основні статистичні характеристики можна обчислити за допомогою NumPy? Наведіть приклади відповідних функцій.
6. Як побудувати гістограму розподілу даних в Python? Яка інформація відображається на гістограмі?
7. Що таке коефіцієнт кореляції? Як його обчислити за допомогою NumPy? Яку інформацію він надає про зв’язок між двома наборами даних?
8. Які бібліотеки Python використовуються для обробки дослідних даних та їх візуалізації в цій лабораторній роботі?

ЛАБОРАТОРНА РОБОТА № 7

ЗНАЙОМСТВО З ТАБЛИЧНИМ РЕДАКТОРОМ GOOGLE ТАБЛИЦІ

Мета роботи. Ознайомитись з базовим функціоналом редактора Google Таблиці. Оволодіти навиками побудови графіків та діаграм в табличному редакторі Google Таблиці.

Хід роботи

1. Створіть нову електронну таблицю за поданим зразком. Введіть дані про студентів своєї групи. Правильності даних, окрім прізвищ та імен студентів, не вимагається (рис. 7.1).

	A	B	C	D	E	F	G
1			Список студентів				
2							
3			Вінницький національний технічний університет				
4							
5			Факультет:	ФЕЕЕМ		група:	ЕЕ-28б
6							
7			К-ть студентів:	20			
8							
9		№ п/п	П.І.П.	email	телефон	дата народження	Іноземна мова
10		1	Рябенький	riab@vntu.edu.ua	0937894562	06.01.2011	англ.
11		2	Іванов	b@gmail.com	0676543213	08.03.2011	англ.
12		3	Вернидуб	a@gmail.com	0503332255	19.07.2010	нім.
13		4	Козак	c@gmail.com	0669876543	01.03.2011	англ.
14		5	Сидоров	sydorov@ukr.net	0737894563	12.11.2010	нім.
15		6	Петров	petr@ukr.net	0636549872	21.09.2010	англ.
16		7	Крутигора	krut@gmail.com	0971597535	10.09.2010	нім.
17							

Рисунок 7.1 – Приклад введення списку студентів

2. Навчіться сортувати дані. Виділіть список студентів без порядкових номерів та заголовків стовпців та оберіть з меню “Дані – Відсортувати діапазон” необхідний варіант (рис. 7.2).

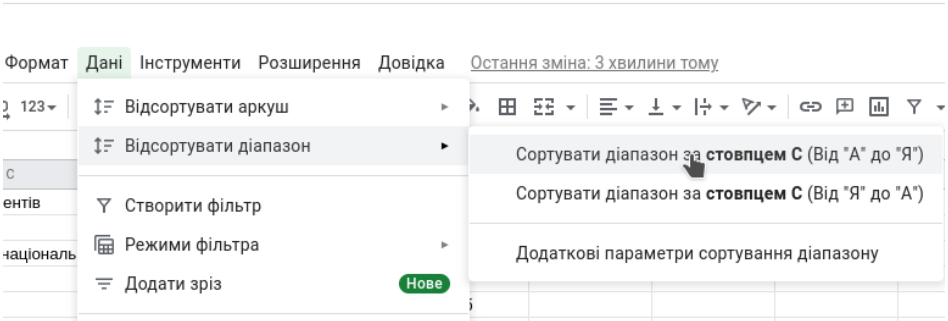


Рисунок 7.2 – Приклад використання сортування даних

3. Відформатуйте таблицю згідно поданого зразка. Використовуйте функції обрамлення комірок, заливки комірок. Навчіться змінювати шрифт за типом, розміром, товщиною. Для оформлення заголовків використайте перенесення тексту в комірках з меню “Формат – Перенесення тексту” (рис. 7.3).

Список студентів

Вінницький національний технічний університет

Факультет: **ФЕЕЕМ** ***група:*** **ЕЕ-28б**

К-ть студентів: **20**

№ п/п	П.І.П.	email	телефон	дата народження	іноземна мова
1	Вернидуб	a@gmail.com	0503332255	19.07.2010	нім.
2	Іванов	b@gmail.com	0676543213	08.03.2011	англ.
3	Козак	c@gmail.com	0669876543	01.03.2011	англ.
4	Крутигора	krut@gmail.com	0971597535	10.09.2010	нім.
5	Петров	petr@ukr.net	0636549872	21.09.2010	англ.
6	Рябенький	riab@vntu.edu.ua	0937894562	06.01.2011	англ.
7	Сидоров	sydorov@ukr.net	0737894563	12.11.2010	нім.

Рисунок 7.3 – Приклад форматування даних

Застосуйте умовне форматування стовпчика “Іноземна мова” таким чином, щоб комірки з різними мовами були заповнені різними кольорами (рис. 7.4).

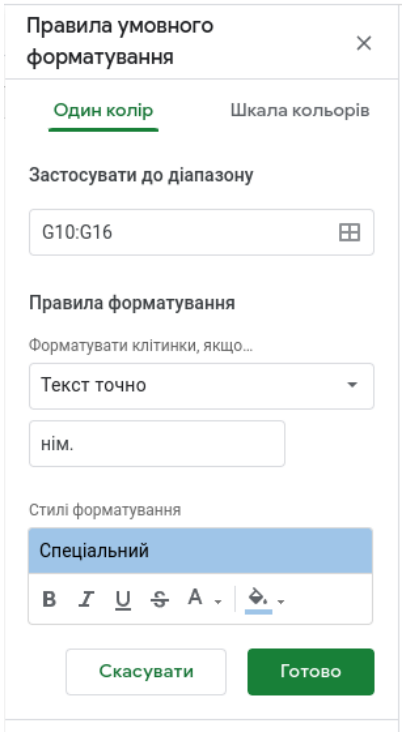


Рисунок 7.4 – Вікно вкладки умовне форматування

4. Налаштуйте перевірку даних у стовпчиках “email”, дата народження” та “іноземна мова” таким чином, щоб при введенні неправильних даних, редактор не дозволяв вставити дані (рис. 7.5).

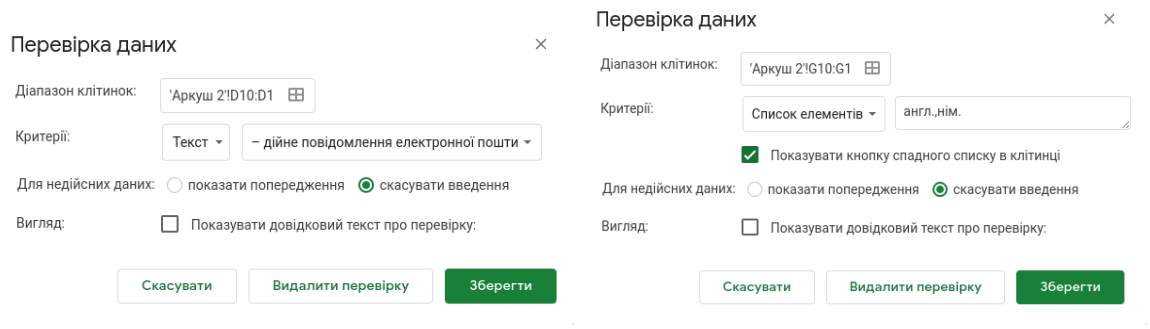


Рисунок 7.5 – Вікно вкладки перевірки даних

Скопіюйте поточний лист електронної таблиці, обравши пункт “Створити копію” в меню аркуша, змініть його назву на “Відомість” та відформатуйте, щоб отримати такий документ (рис. 7.6):

	A	B	C	D	E	F	G	H
1			***Вінницький національний технічний університет***					
2								
3			**Екзаменаційна відомість**					
4				дата екзамену:				
5								
6			***Факультет:***	**ФЕЕЕМ**	***група:***	**ЕЕ-28б**		
7								
8			***К-ть студентів:***	**20**				
9								
10		**№ п/п**	**П.І.П.**	**К-ть балів за семестр**	**К-ть балів за екзамен**	**Загальна к-ть балів**	**Оцінка ECTS**	**Оцінка**
11		*1*	Вернидуб	65	18			
12		*2*	Іванов	47	11			
13		*3*	Козак	72	20			
14		*4*	Крутигора	51	15			
15		*5*	Петров	55	16			
16		*6*	Рябенький	69	13			
17		*7*	Сидоров	38	25			

Рисунок 7.6 – Приклад екзаменаційної відомості

5. В комірки стовпців “К-ть балів за семестр” та “К-ть балів за екзамен” додайте перевірку, яка не дозволить вставляти бали, що менші за 0 чи більші за максимально дозволені (за семестр - 75, за екзамен - 25).

Додайте формули для підрахунку загальної кількості балів, а також для вирахування оцінки за шкалою ECTS та за національною шкалою згідно з правилами, поданих в табл. 7.1.

Таблиця 7.1 – Кількість балів за шкалою ECTS.

К-ть балів	Оцінка ECTS	Національна оцінка
90-100	A	відмінно
82-89	B	добре
75-81	C	
64-74	D	задовільно
60-63	E	
35-59	FX	незадовільно
0-34	F	

При визначенні оцінок використовуйте функцію IF, яка вкладена в іншу IF.

Наприклад:

=IF(F11>=90;"A";IF(F11>=82;"B";IF(F11>=75;"C";....)))

Знизу таблиці додайте комірки, в які внесіть формули для вираховування кількості кожної оцінки (рис. 7.7).

Вінницький національний технічний університет

Екзаменаційна відомість

дата екзамену:

Факультет: ФЕЕЕМ *група:* ЕЕ-28б

К-ть студентів: 20

№ п/п	П.І.П.	К-ть балів за семестр	К-ть балів за екзамен	Загальна к-ть балів	Оцінка ECTS	Оцінка
1	Вернидуб	55	13	68	D	задовільно
2	Іванов	41	23	64	D	задовільно
3	Козак	66	23	89	B	добре
4	Крутигора	71	20	91	A	відмінно
5	Петров	51	11	62	E	задовільно
6	Рябенький	74	20	94	A	відмінно
7	Сидоров	74	18	92	A	відмінно

Оцінка	К-ть
A	3
B	1
C	0
D	2
E	1
FX	0
F	0

Рисунок 7.7 – Приклад заповненої екзаменаційної відомості

Для підрахування кількості оцінок використайте функцію =COUNTIF(). Правила застосування цієї функції дізнайтесь з довідки.

6. На листі відомості вставте стовпчасту діаграму, яка показує загальну результативність групи (рис. 7.8).

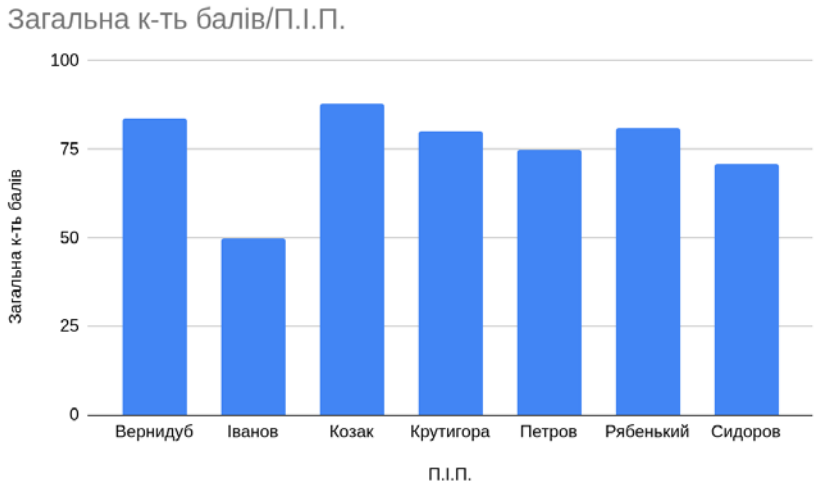


Рисунок 7.8 – Приклад стовпчастої діаграми

Створіть схожу стовпчасту діаграму з накопиченням (рис. 7.9).

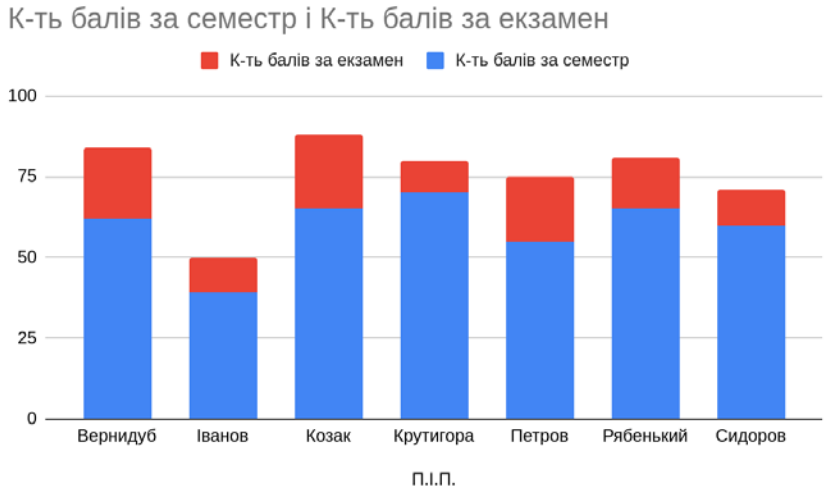


Рисунок 7.9 – Різновид стовпчастої діаграми

Використовуючи таблицю з кількістю оцінок, створіть кругову діаграму (рис. 7.10).

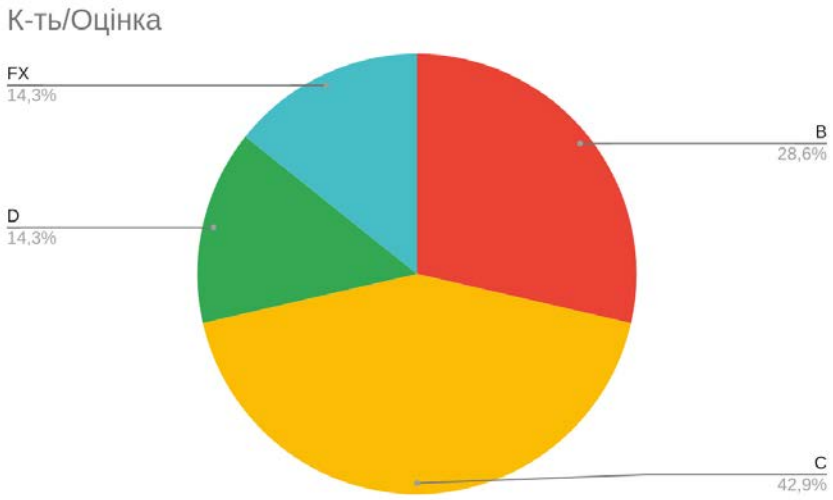
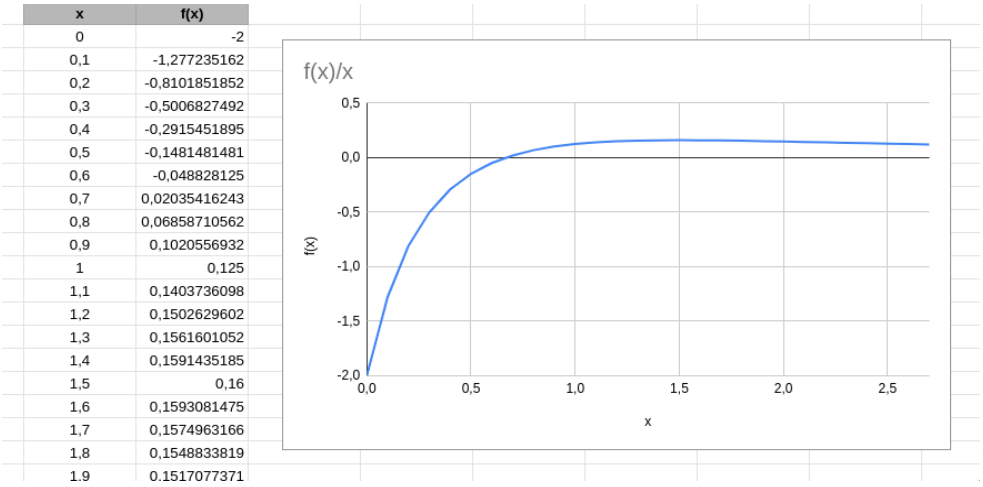


Рисунок 7.10 – Приклад кругової діаграми

7. На новому аркуші створіть таблицю, в якій обчисліть значення функції при зміні аргументу від 0 до 10. Побудуйте графік цієї функції (рис. 7.11). Функцію для побудови візьміть з табл. 7.2 згідно зі своїм варіантом.



x	f(x)
0	-2
0,1	-1,277235162
0,2	-0,8101851852
0,3	-0,5006827492
0,4	-0,2915451895
0,5	-0,1481481481
0,6	-0,048828125
0,7	0,02035416243
0,8	0,06858710562
0,9	0,1020556932
1	0,125
1,1	0,1403736098
1,2	0,1502629602
1,3	0,1561601052
1,4	0,1591435185
1,5	0,16
1,6	0,1593081475
1,7	0,1574963166
1,8	0,1548833819
1,9	0,1517077371

Рисунок 7.11 – Приклад точкового графіка

Таблиця 7.2 – Варіанти завдань

Варіант	**Функція**	**Варіант**	**Функція**
1	$\frac{8(x-1)}{(x+2)^2}$	9	$\left(\frac{x+2}{x-1}\right)^2$
2	$\frac{5x}{x^2+3}$	10	$(x+1)\cdot e^{x+2}$
3	$\frac{x^3-27x+54}{x^3}$	11	$\frac{x^2-6x+9}{(x-1)^2}$

Продовження таблиці 7.2.

4	$-\dfrac{5x}{x^2+2}$	12	$(x+4)\cdot e^{-x-3}$
5	$(2x+4)\cdot e^{2(x+2)}$	13	$\dfrac{2x^2+1}{x^2+3}$
6	$2-\dfrac{3x}{x^2+3}$	14	$(2x-1)\cdot e^{2(x-1)}$
7	$-(x+4)\cdot e^{-x-3}$	15	$\dfrac{5x^2}{x^2+3}$
8	$\left(2+\dfrac{1}{x}\right)^2$	16	$\dfrac{3x-2}{(x+1)^3}$

Створіть форму, в якій ви можете задавати діапазон зміни аргумента для побудови графіка. Кількість точок для побудови графіка задайте не менше 50.

Вставте новий графік функції (рис. 7.12). Переконайтесь, що при зміні діапазону функції змінюється графік.

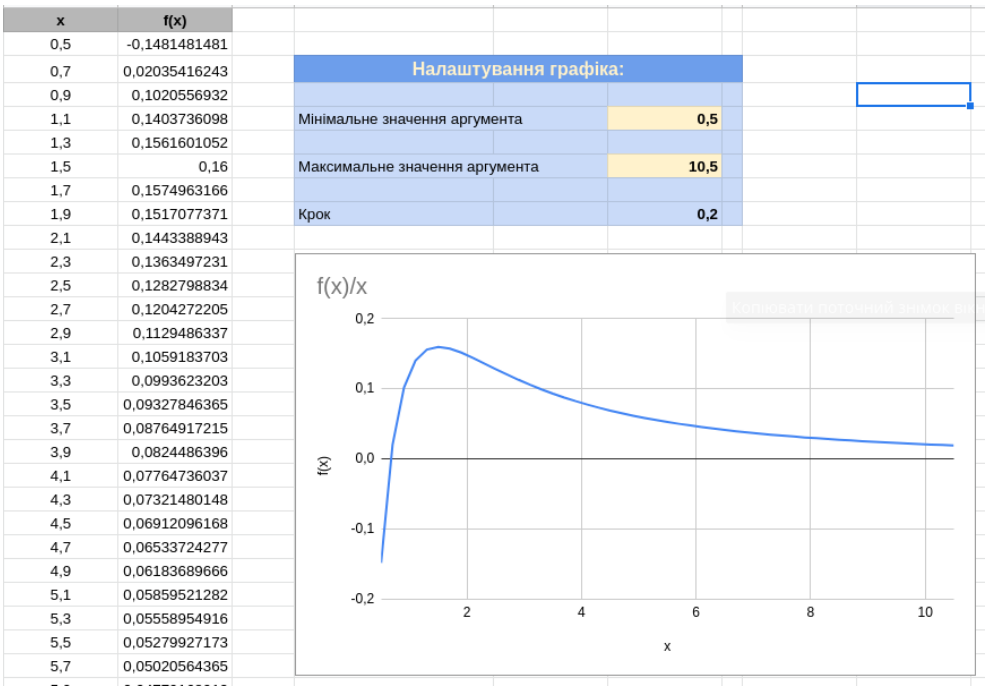


Рисунок 7.12 – Приклад зміни діапазону відображення графіка

Додайте до таблиці обчислене значення похідної від заданої функції. Похідну обчисліть як відношення зміни функції до зміни аргументу. Зміну значення вираховуйте як різницю наступного та попереднього значень. Побудуйте в одних осях графік функції та графік похідної (рис. 7.13).

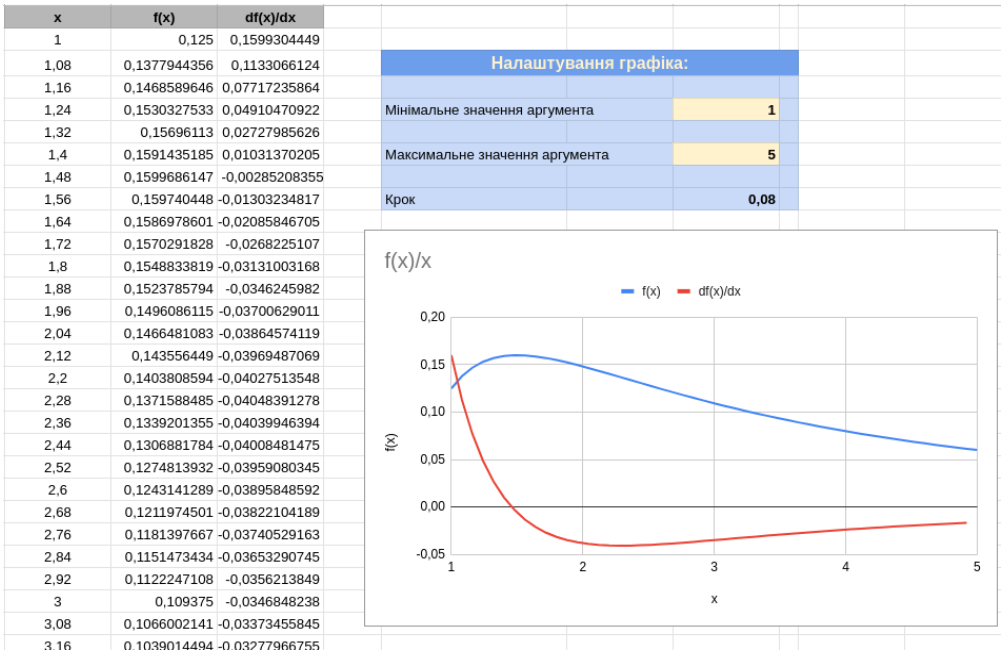


Рисунок 7.13 – Приклад побудови графіка похідної функції

Зайдіть в редактор діаграм, двічі клацнувши на графік. Змініть його оформлення, задавши колір фону, колір осей, товщину лінії графіка тощо (рис. 7.14).

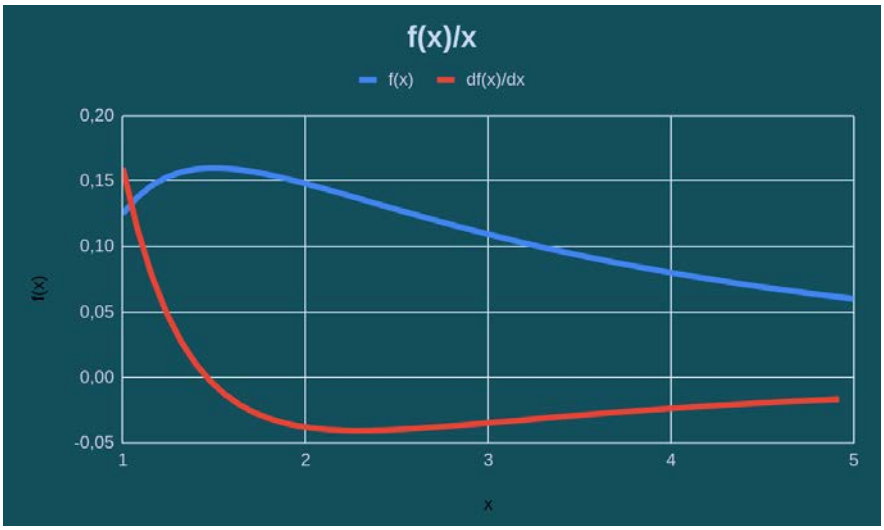


Рисунок 7.14 – Приклад форматування графіка

8. Зробіть висновки про виконану роботу.

Теоретичні відомості

Google Таблиці — додаток для роботи з електронними таблицями, що входить до складу безкоштовного вебпрограмного пакету програмного забезпечення, пропонованого компанією Google у межах служби Google Диск. Сервіс доступний онлайн, а також як мобільний додаток для Android, iOS, Windows, BlackBerry, а також як настільний додаток у Google ChromeOS. Інтерфейс додатка схожий з інтерфейсом Microsoft Excel, що

входить до складу Microsoft Office та сумісний з форматами файлів Microsoft Excel. В режимі реального часу користувач має можливість самостійно працювати з таблицями (редагувати, форматувати, тощо), а також надати спільний доступ іншим користувачам для спільної роботи. Правки відстежуються користувачем, а історія редагувань представляє зміни. Положення редактора виділяється певним для редактора кольором та курсором, а система дозволів регулює, що користувачі можуть робити [11].

Завантажте інтернет браузер Google Chrome або Mozilla Firefox. Зайдіть у свій аккаунт Google. Завантажте вебдодаток Google Диск. Ознайомтесь з основними можливостями цього додатку. Створіть теку, спробуйте надати до неї доступ своїм одногрупникам. Доступ можна надать конкретному користувачеві, задавши його електронну пошту, а можна – будь-кому, хто має посилання. Також можна виставляти рівень доступу: лише перегляд; коментування; редагування.

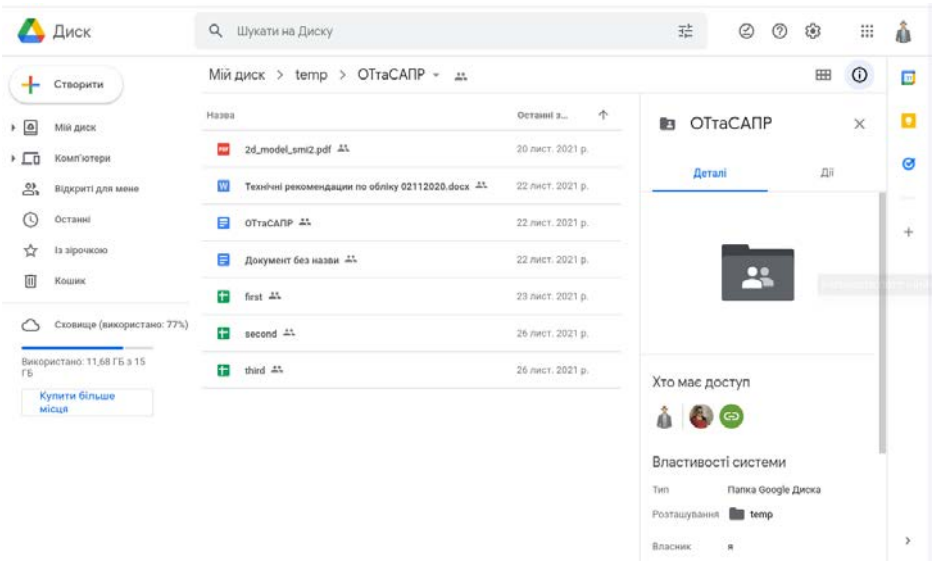


Рисунок 7.15 – Інтерфейс вебдодатка Google Диск.

Контрольні запитання

1. Чим інтерфейс Google Таблиць схожий на інтерфейс Microsoft Excel і які формати файлів забезпечують взаємну сумісність?
2. Які типи доступу до загальної папки Google Диск можна надати користувачам і які дії вони відповідно можуть виконувати?
3. Яким чином у Google Таблицях відстежуються та відображаються зміни, внесені різними користувачами в режимі реального часу?
4. Як система кольорового виділення курсора редактора допомагає відрізнити дії різних учасників спільної роботи?
5. Яким чином система дозволів Google Диск обмежує або розширює можливості користувачів при роботі з електронними таблицями?

ЛАБОРАТОРНА РОБОТА № 8

МОДЕЛЮВАННЯ ЕЛЕКТРИЧНИХ КІЛ В ПРОГРАМНИХ ЕМУЛЯТОРАХ (НА ПРИКЛАДІ ОНЛАЙН СИСТЕМИ FALSTAD)

Мета роботи. Ознайомлення з інтерфейсом та можливостями онлайн-емулятора Falstad Circuit Simulator для аналізу різних типів кіл. Дослідження режимів роботи резистивних кіл постійного струму та кіл змінного струму з ємнісними та індуктивними елементами шляхом комп'ютерного моделювання.

Хід роботи

1. Відкрити вебсторінку емулятора: https://www.falstad.com/circuit/.
2. Ознайомитися з інтерфейсом програми.
3. Для Завдання 1 (коло постійного струму):
   – Зібрати електричне коло постійного струму відповідно до варіанту завдання.
   – Встановити параметри джерел ЕРС та резисторів.
   – Провести моделювання та виміряти вказані струми та напруги. Використовувати режим "DC" (навести курсор на елемент/вузол).
   – Занести результати (схему, параметри, виміряні значення) до звіту.
4. Для Завдання 2 (коло змінного струму):
   – Зібрати електричне коло змінного струму відповідно до варіанту завдання.
   – Встановити параметри джерела змінної напруги (амплітуду, частоту) та пасивних елементів (R, L, C).
   – Провести моделювання. Використовувати інструмент "Scope" для візуалізації та вимірювання параметрів сигналів (амплітуди, RMS значення, часові зсуви). Виміряти вказані величини.
   – Занести результати (схему, параметри, осцилограми, виміряні значення) до звіту.
5. Виконати теоретичний розрахунок параметрів кіл та порівняти з результатами моделювання.
6. Зробити загальні висновки по роботі.
7. Підготувати відповіді на контрольні запитання.

Теоретичні відомості

1. Комп'ютерне моделювання електричних кіл: Моделювання електричних кіл – це процес створення математичної моделі електричного кола та її дослідження за допомогою комп'ютерних програм (симуляторів або емуляторів). Це дозволяє аналізувати поведінку кола, визначати

напруги, струми, потужності та інші параметри в різних режимах роботи (постійний струм, змінний струм, перехідні процеси) без необхідності фізичної побудови макету [12]-[13].

Переваги моделювання:

– **Економія часу та ресурсів:** не потрібні реальні компоненти та вимірювальні прилади на етапі проєктування та аналізу.

– **Безпека:** можливість дослідження режимів, які можуть бути небезпечними для реальних компонентів (наприклад, короткі замикання, перевантаження).

– **Наочність:** візуалізація процесів (протікання струму, зміна напруги, форми сигналів), що полегшує розуміння роботи кола.

– **Гнучкість:** легкість зміни параметрів компонентів та структури кола для дослідження їх впливу.

– **Доступність:** існує багато програмних продуктів, включаючи безкоштовні та онлайн-системи.

2. Онлайн-емулятор Falstad Circuit Simulator (www.falstad.com/circuit/). Falstad Circuit Simulator – це безкоштовний інтерактивний онлайн-симулятор електронних схем, написаний на Java (і доступний у вигляді JavaScript-апплету), що дозволяє створювати та моделювати електричні кола безпосередньо у веб-браузері [14].

Основні можливості та елементи інтерфейсу:

– **Робоча область:** місце для побудови схеми шляхом розміщення та з’єднання компонентів.

– **Панель компонентів:** доступна через меню "Draw". Містить резистори, конденсатори, котушки індуктивності, джерела напруги/струму (постійні, змінні, імпульсні), діоди, транзистори, логічні елементи, вимірювальні інструменти (вольтметр, амперметр) та інше.

– **Редагування параметрів:** подвійний клік або правий клік на компоненті дозволяє змінити його параметри (опір, ємність, індуктивність, напругу, частоту тощо).

– **Візуалізація:** симулятор відображає напрямок та інтенсивність струму (рухомі точки), рівень напруги (колір провідників), а також дозволяє будувати графіки напруг та струмів у часі (Scope).

– **Керування симуляцією:** можна змінювати швидкість симуляції, скидати стан кола.

– **Збереження/Завантаження:** схеми можна зберігати у вигляді текстових файлів або посилань.

3. Основи теорії електричних кіл.

Для успішного виконання роботи необхідно пригадати основні закони та методи аналізу електричних кіл:

– **Закон Ома:** $U=I\cdot R$, $I=R\cdot U$, $R=I\cdot U$.

– **Закони Кірхгофа:** Перший (для вузлів, $\sum I_{вх} = \sum I_{вих}$) та другий (для контурів, $\sum E = \sum U = \sum (I \cdot R)$).

– **Розрахунок кіл постійного струму:** методи еквівалентних перетворень, контурних струмів, вузлових потенціалів.

– **Розрахунок кіл змінного синусоїдального струму:** комплексний метод, векторні діаграми, поняття повного, активного та реактивного опорів (Z, R, X_L, X_C), резонанс [13].

– **Перехідні процеси:** Аналіз поведінки кіл з реактивними елементами (C, L) при комутаціях, поняття постійної часу ($\tau = R \cdot C$ або $\tau = L/R$).

Варіанти завдань

Варіант 1.

Завдання 1: Моделювання кола постійного струму.

1. Зібрати схему (Рис. 8.1).

2. Параметри: E = 24 В, R_1 = 10 Ом, R_2 = 20 Ом, R_3 = 30 Ом, R_4 = 40 Ом. Виміряти: загальний струм I, струми I_2 (через R_2) та I_3 (через R_3), напругу U_{R4} на резисторі R_4.

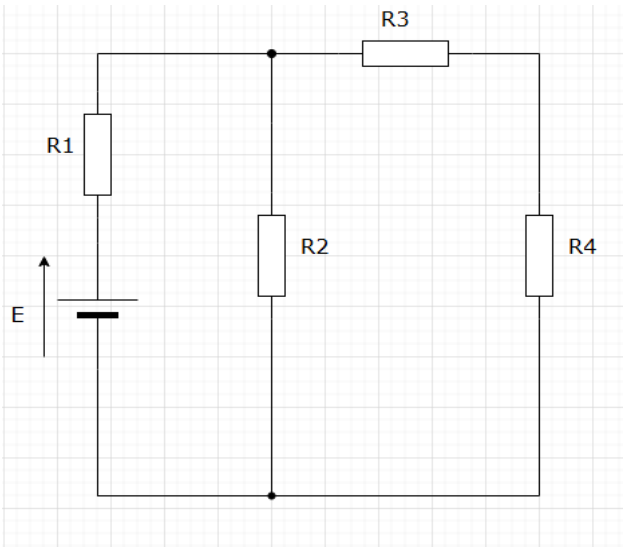


Рисунок 8.1 – Схема до завдання 1

Завдання 2: Моделювання кола змінного струму (RC-фільтр).

1. Зібрати схему (Рис.–8.2).

2. Параметри: Джерело E(t) – синусоїдальне, амплітуда E_{max} = 10 В, частота f = 100 Гц. R = 1,5 кОм, C = 1 мкФ.

3. Використовуючи "Scope", побудувати осцилограми вхідної напруги E(t) та вихідної напруги $U_C(t)$ (на конденсаторі).

4. Виміряти амплітудне значення вихідної напруги U_{Cmax}.

5. Визначити зсув фаз ($\Delta\phi$) між E(t) та $U_C(t)$ (в градусах або радіанах). Вказати, який сигнал випереджає/відстає.

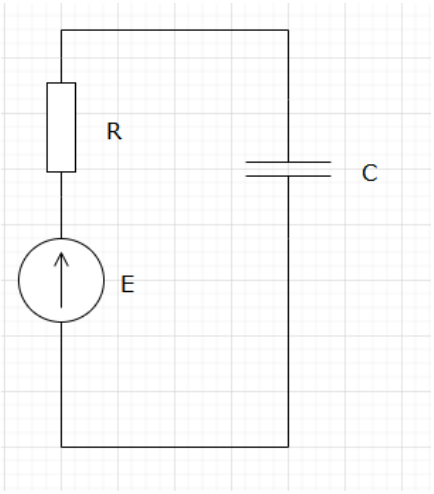


Рисунок 8.2 – Схема до завдання 2

Варіант 2.

Завдання 1: Моделювання кола постійного струму (Міст Уітстона).

1. Зібрати схему (Рис. 8.3).

2. Параметри: E = 10 В, R_1 = 100 Ом, R_2 = 150 Ом, R_3 = 200 Ом, R_4=250 Ом, R_G = 50 Ом (опір гальванометра).

3. Виміряти: струм I_G через резистор R_G (вказати напрямок), потенціали точок A та B відносно землі.

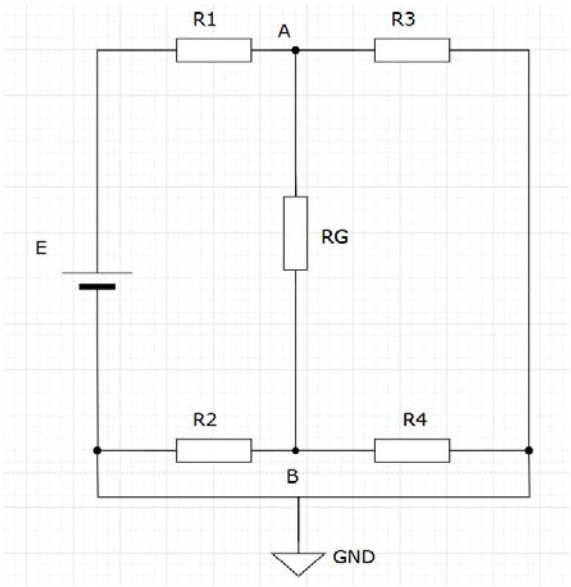


Рисунок 8.3 – Схема до завдання 1

Завдання 2: Моделювання кола змінного струму (Послідовне RL-коло).

1. Зібрати схему (Рис. 8.4).

2. Параметри: Джерело E(t) – синусоїдальне, амплітуда E_{max} = 20 В, частота f = 50 Гц. R = 30 Ом, L = 150 мГн.

3. Використовуючи "Scope", побудувати осцилограми напруги джерела E(t) та струму в колі I(t).

4. Виміряти амплітудне значення струму Imax.

5. Визначити зсув фаз (Δϕ) між E(t) та I(t). Вказати, який сигнал випереджає/відстає.

6. Обчислити повний опір кола |Z|=Emax/Imax.

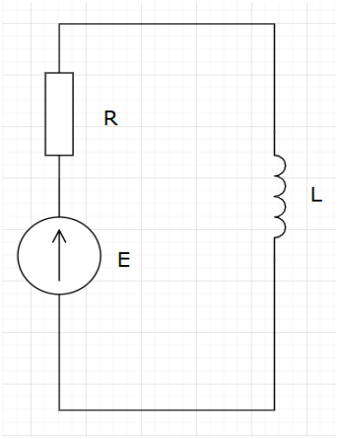


Рисунок 8.4 – Схема до завдання 2

Варіант 3.

Завдання 1: Моделювання кола постійного струму (Два джерела).

1. Зібрати схему (Рис. 8.5).

2. Параметри: E_1 = 12 В, E_2 = 6 В, R_1 = 5 Ом, R_2 = 10 Ом, R_3 = 15 Ом.

3. Виміряти: струм I_3 через резистор R_3 (вказати напрямок), напругу U_{R2} на резисторі R_2.

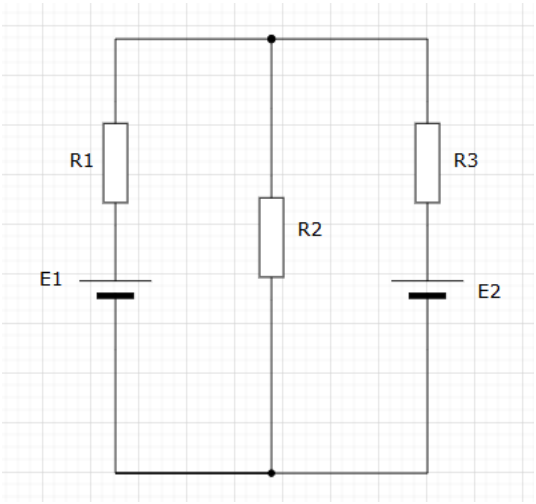


Рисунок 8.5 – Схема до завдання 1

Завдання 2: Моделювання кола змінного струму (Послідовне RLC-коло).

1. Зібрати схему (Рис. 8.6).

2. Параметри: Джерело E(t) – синусоїдальне, амплітуда E_{max} = 10 В, частота f = 1 кГц. R = 20 Ом, L = 10 мГн, C = 2 мкФ.

3. Використовуючи "Scope", побудувати осцилограми напруги джерела E(t) та струму I(t).

4. Виміряти амплітудні значення струму Imax, напруги на резисторі URmax, котушці ULmax та конденсаторі U_{Cmax}.

5. Визначити зсув фаз (Δϕ) між E(t) та I(t).

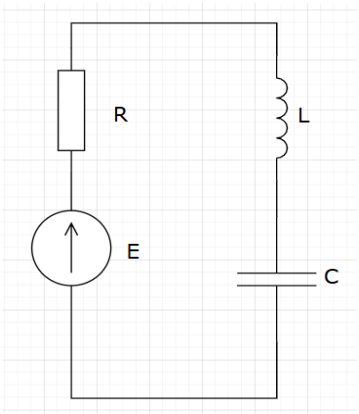


Рисунок 8.6 - Схема до завдання 2

Варіант 4.

Завдання 1: Моделювання кола постійного струму

1. Зібрати схему (Рис. 8.7).

2. Параметри: E = 30 В, R_1 = 100 Ом, R_2 = 200 Ом, R_3 = 300 Ом, R_4= 400 Ом, R_5 = 500 Ом.

3. Виміряти: еквівалентний опір кола відносно джерела $R_{екв}$ = $E/I_{заг}$, струм I4 через R_4, потужність P_5, що виділяється на R_5.

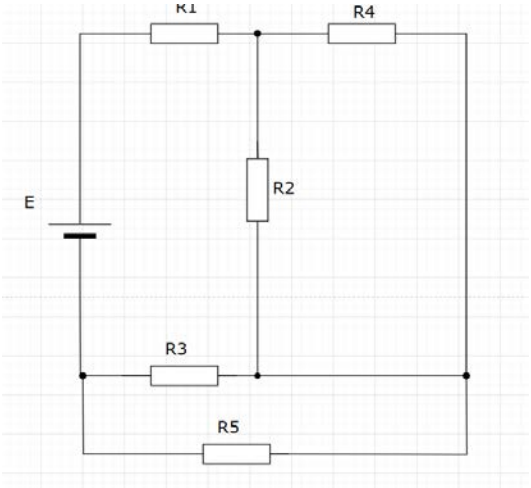


Рисунок 8.7 - Схема до завдання 1

Завдання 2: Моделювання кола змінного струму.

1. Зібрати схему (Рис. 8.8). Використати джерело змінного струму "AC Current Source".

2. Параметри: Джерело I(t) – синусоїдальне, амплітуда I_{max}= 1А, частота f = 500 Гц. L = 50 мГн, C = 1 мкФ. (Додайте невеликий резистор послідовно з джерелом струму, наприклад 1 Ом, для стабільності симуляції, якщо потрібно).

3. Використовуючи "Scope", побудувати осцилограми струму джерела I(t), струму через котушку $I_L(t)$ та струму через конденсатор $I_C(t)$.

4. Виміряти амплітудні значення струмів I_{Lmax} та I_{Cmax}.

5. Визначити зсув фаз між $I_L(t)$ та $I_C(t)$.

6. Виміряти амплітуду напруги U_{LCmax} на паралельному контурі.

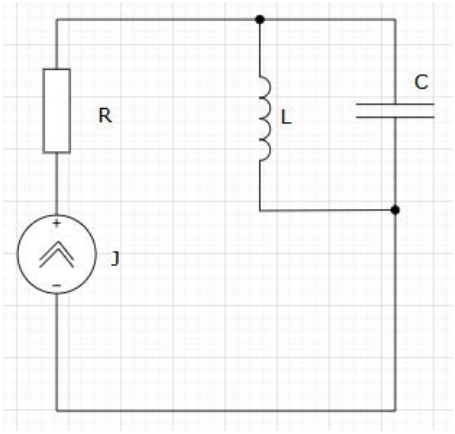


Рисунок 8.8 – Схема з джерелом струму

Контрольні запитання

1. Сформулюйте закон Ома та закони Кірхгофа для кіл постійного струму.

2. Як визначити еквівалентний опір послідовного та паралельного з’єднання резисторів?

3. Що таке імпеданс? Як розраховуються реактивні опори котушки індуктивності (X_L) та конденсатора (X_C)?

4. Як представити синусоїдальну напругу або струм за допомогою комплексного числа?

5. Поясніть поняття зсуву фаз між напругою та струмом в колах змінного струму з R, L, C елементами. Коли струм випереджає напругу, а коли відстає?

6. Опишіть інтерфейс симулятора Falstad: як додати елементи, змінити їх параметри, з’єднати їх?

7. Як у симуляторі Falstad виміряти напругу у вузлі та струм у гілці для кола постійного струму?

8. Як за допомогою інструменту "Scope" в Falstad відобразити та проаналізувати осцилограми напруг/струмів у колі змінного струму? Як визначити амплітуду та зсув фаз за осцилограмами?

9. Які переваги використання програмних симуляторів для аналізу електричних кіл порівняно з фізичним експериментом?

10. Наведіть приклад задачі з вашої галузі (промислова автоматизація), де моделювання електричних/електронних кіл може бути корисним.

ЛІТЕРАТУРА

1. Beazley D., Jones B. K. Python Cookbook: Recipes for Mastering Python 3. 3rd ed. Sebastopol ; CA, USA : O'Reilly Media, 2013. 706 p.

2. Sweigart A. Automate the Boring Stuff with Python: Practical Programming for Total Beginners. 2nd ed. San Francisco, CA, USA : No Starch Press, 2019. 504 p. Available: https://automatetheboringstuff.com. (Accessed: Sep. 1, 2025).

3. McKinney W. Python for Data Analysis: Data Wrangling with pandas, NumPy, and IPython. 2nd ed. Sebastopol ; CA, USA : O'Reilly Media, 2018.

4. Kiusalaas J. Numerical Methods in Engineering with Python 3. Cambridge, UK : Cambridge University Press, 2013.

5. Oliphant T. E. Guide to NumPy. 2nd ed. CreateSpace Independent Publishing Platform, 2015.

6. Hunter J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering.* 2007. Vol. 9, no. 3. P. 90–95. DOI: 10.1109/MCSE.2007.55. Available: https://ieeexplore.ieee.org. (Accessed: Sep. 9, 2025).

7. Johansson R. Numerical Python: Scientific Computing and Data Science Applications with NumPy, SciPy and Matplotlib. New York, NY, USA : Apress, 2016.

8. SymPy Development Team, SymPy – Python library for symbolic mathematics. Available: https://www.sympy.org/docs. (Accessed: Sep. 1, 2025).

9. Langtangen H. P. A Primer on Scientific Programming with Python. 2nd ed. Cham, Switzerland : Springer, 2016.

10. Oppenheim A. V., Willsky A. S., Nawab S. H. Signals and Systems. 2nd ed. Upper Saddle River, NJ, USA : Prentice Hall, 1996.

11. Google, Google Workspace – Google Sheets: Learning Center. Available: https://support.google.com/docs/topic/. (Accessed: Sep. 1, 2025).

12. Nilsson J. W., Riedel S. A. Electric Circuits. 11th ed. Upper Saddle River, NJ, USA : Pearson, 2019.

13. Alexander C. K., Sadiku M. N. O. Fundamentals of Electric Circuits. 6th ed. New York, NY, USA : McGraw-Hill Education, 2017. 903 p.

14. Falstad P. Circuit Simulator (CircuitJS/Falstad). Available: https://www.falstad.com/circuit/. (Accessed: Sep. 1, 2025).

Електронне навчальне видання

Олександр Миколайович Кравець
Денис Юрійович Лебедь

Методичні вказівки до виконання лабораторних робіт з дисципліни «Засоби моделювання в електротехнічних системах» зі спеціальності «Електрична інженерія»

Рукопис оформив: Д. Лебедь

Редактор: Н. Кравчук

Оригінал-макет виготовлено в РВВ ВНТУ

Підписано до видання 11.11.2025
Гарнітура Times New Roman.
Зам. № Р2025-163.
Видавець та виготовлювач
Вінницький національний технічний університет,
Редакційно-видавничий відділ.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95,
м. Вінниця, 21021.
press.vntu.edu.ua;
Email: rvv.vntu@gmail.com
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.