Міністерство освіти і науки України
Запорізький національний університет

С. І. Гоменюк, С. М. Гребенюк, Є. В. Панасенко

ЗАСТОСУВАННЯ СУЧАСНИХ МОВ ПРОГРАМУВАННЯ ДО РОЗВ'ЯЗАННЯ МАТЕМАТИЧНИХ ЗАДАЧ

Методичні рекомендації до лабораторних занять
для здобувачів ступеня вищої освіти магістра
спеціальності «Математика»
освітньо-професійної програми «Математика»

Затверджено
вченою радою ЗНУ
Протокол №10
від 29.04.2025

Запоріжжя
2025

УДК 004.43:51(076.5)
Г641

Гоменюк С. І., Гребенюк С. М., Панасенко Є. В. Застосування сучасних мов програмування до розв’язання математичних задач : методичні рекомендації до лабораторних занять для здобувачів ступеня вищої освіти магістра спеціальності «Математика» освітньо-професійної програми «Математика». Запоріжжя : Запорізький національний університет, 2025. 73 с.

У методичних рекомендаціях в систематизованому вигляді подано матеріал для виконання лабораторних робіт з дисципліни «Застосування сучасних мов програмування до розв’язання математичних задач». У виданні представлено теоретичний матеріал з кожної теми дисципліни, хід виконання та рекомендації щодо виконання лабораторних робіт, які пропонуються при вивченні даного курсу. Кожна лабораторна робота супроводжується необхідними прикладами та поясненнями до них, певними практичними завданнями, виконання яких поглиблює сприйняття матеріалу та контрольними запитаннями.

Методичні рекомендації призначено для здобувачів ступеня вищої освіти магістра спеціальності «Математика» освітньо-професійної програми «Математика».

Рецензент
А.О. Лісняк, кандидат фізико-математичних наук, доцент, завідувач кафедри програмної інженерії

Відповідальний за випуск
С.М. Гребенюк, доктор технічних наук, професор, завідувач кафедри фундаментальної та прикладної математики

ЗМІСТ

ВСТУП

Інтенсивна математизація різних галузей знань, а також потреби практики та бурхливий розвиток обчислювальної техніки вимагають постійного вдосконалення математичних методів досліджень, розробки питань математичного забезпечення, що в свою чергу вимагає умінь розробки алгоритмів, використовуючи сучасні мови програмування та математику. Тому й виникла потреба вивчати магістрами математики курс «Застосування сучасних мов програмування до розв'язання математичних задач».

На сучасному етапі розвитку математики різко зростає її прикладне застосування у різних галузях. Ось кілька суто математичних задач, які можна включити до курсу: чисельне розв'язування нелінійних рівнянь та систем, реалізуйте програм для обчислення визначених інтегралів, чисельне розв'язування звичайних диференціальних рівнянь і розробка програм для розв'язування задачі Коші для системи диференціальних рівнянь, задача на пошук коренів рівняння за допомогою метода дихотомії, реалізація програми для розв'язування задачі лінійного програмування, реалізація програми для знаходження оберненої і псевдооберненої матриці, реалізація алгоритму для знаходження числових характеристик масиву даних, побудова графіків. Ці задачі можна реалізувати за допомогою сучасних мов програмування, таких як Python, C++, або Java, використовуючи відповідні бібліотеки для роботи з матрицями, чисельними методами та математичними обчисленнями (наприклад, NumPy для Python).

Метою вивчення навчальної дисципліни «Застосування сучасних мов програмування до розв'язання математичних задач» є засвоєння систематичних знань із сучасних мов програмування: Python та R; оперування інструментарієм сучасних мов програмування для автоматизації вирішення математичних задач та обробки даних; отримання досвіду з програмування систем, зокрема аналізу та моделюванню процесів та явищ у галузі математики та статистики.

Python має широкий набір бібліотек, які активно використовуються в математиці та наукових обчисленнях. Ось деякі з найпопулярніших:

- Math: стандартний модуль Python, який надає базові математичні функції, такі як обчислення факторіалу, логарифмів, тригонометричних функцій та інших.
- NumPy: бібліотека для роботи з багатовимірними масивами та матрицями, яка також містить велику кількість математичних функцій для обробки цих структур. Вона є основою для багатьох інших наукових бібліотек у Python.
- SciPy: розширює можливості NumPy, надаючи додаткові модулі для чисельного інтегрування, оптимізації, обробки сигналів та інших задач, пов'язаних з науковими обчисленнями.

– SymPy: бібліотека для символічних обчислень, яка дозволяє виконувати алгебраїчні маніпуляції, спрощення виразів, обчислення похідних та інтегралів у символічній формі.
– Pandas: призначена для роботи з табличними даними та часовими рядами. Вона забезпечує зручні структури даних, такі як DataFrame, та інструменти для аналізу і маніпуляції даними.
– Matplotlib: бібліотека для створення статичних, анімованих та інтерактивних візуалізацій даних у Python. Вона дозволяє будувати різноманітні графіки та діаграми для представлення математичних даних.
– Seaborn: побудована на основі Matplotlib, ця бібліотека спрощує створення статистичних графіків та надає стильніші та інформативніші візуалізації.

Використання цих бібліотек дозволяє ефективно вирішувати широкий спектр математичних задач, від базових обчислень до складного аналізу даних та візуалізації.

Основними **завданнями** вивчення дисципліни «Застосування сучасних мов програмування до розв'язання математичних задач» є:

– виробити навички із застосування сучасних мов Python та R до розв'язання математичних задач;
– набути навички із використання модулів Python;
– виробити навички використовувати сучасні мови програмування при організації наукових досліджень;
– набути навички із використання прикладних та спеціалізованих бібліотек з метою їх запровадження у професійній діяльності;
– набути навички із аналізу і візуалізації даних у R.

У результаті вивчення навчальної дисципліни студент повинен набути таких програмних компетентностей:

– здатність застосовувати знання у практичних ситуаціях;
– здатність до пошуку, обробки та аналізу інформації з різних джерел, в тому числі іноземною мовою;
– здатність користуватися існуючими програмними засобами для пошуку інформації, проведення обчислень, статистичного аналізу та візуалізації даних;
– здатність працювати з великими обсягами даних та аналізувати їх.

У результаті вивчення навчальної дисципліни студент повинен набути таких програмних результатів навчання:

– знати теоретичні основи і застосовувати основні методи математичної статистики для перевірки гіпотез, обробки реальних даних, застосовувати комп'ютерні засоби статистичного аналізу даних;
– вміти застосовувати сучасні мови програмування до розв'язання математичних задач.

Курс «Застосування сучасних мов програмування до розв'язання математичних задач» є логічним продовженням курсів: «Статистична обробка

даних», «Моделювання динамічних систем і процесів», «Математичні основи машинного навчання». Набуті знання при вивченні курсу «Застосування сучасних мов програмування до розв’язання математичних задач» необхідні для подальшого вивчення курсів: «Наближені методи розв’язання крайових задач» і «Розробка та аналіз алгоритмів» та підготовки кваліфікаційної роботи магістра.

1. Лабораторна робота. Застосування мови Python до наближених обчислень у математичному аналізі

Мета: засвоїти можливості роботи мови Python до застосування розв'язування задач математичного аналізу.

Теоретичні відомості та методичні рекомендації

Використання Python у математичному аналізі робить процес обчислень більш ефективним та зручним, дозволяючи зосередитися на суті задачі, а не на технічних деталях реалізації алгоритмів.

У Python доступний широкий набір арифметичних операторів для виконання математичних обчислень. Основні з них дивись у таблиці 1.1.

Таблиця 1.1 – Арифметичні операції.

Операція	Опис
`x+y`	Додавання. Складає два числа.
`x-y`	Віднімання. Віднімає одне число від іншого
`x*y`	Множення. Множить два числа.
`x/y`	Ділення. Ділить одне число на інше, результатом є число з плаваючою комою.
`x//y`	Цілочисельне ділення. Ділить одне число на інше, повертаючи лише цілу частину результату.
`x%y`	Остача від ділення. Повертає залишок від ділення одного числа на інше.
`x**y`	Піднесення до степеня. Підносить число до вказаного степеня.

Порядок виконання цих операцій відповідає математичним правилам: спочатку виконуються операції піднесення до степеня, потім множення, ділення, цілочисельне ділення та остача від ділення, і нарешті додавання та віднімання. Для зміни порядку виконання операцій можна використовувати круглі дужки.

Окрім основних арифметичних операцій та функцій Python надає програмісту математичну бібліотеку `math`, яка містить додаткові математичні функції та сталі. Для використання бібліотеки її необхідно імпортувати командою:

```
import math
```

Основні функції модуля `math`:

Арифметичні функції:

- `ceil(x)`: повертає найменше ціле число, більше або рівне `x`.
- `floor(x)`: повертає найбільше ціле число, менше або рівне `x`.
- `fabs(x)`: повертає абсолютне значення `x`.
- `factorial(x)`: повертає факторіал числа `x`.
- `fmod(x, y)`: повертає остачу від ділення `x` на `y`.

– `trunc(x)`: відкидає дробову частину числа `x`, повертаючи його цілу частину.

Степеневі та логарифмічні функції:

– `exp(x)`: повертає значення експоненти.
– `log(x[, base])`: повертає логарифм числа `x` за основою `base`. За замовчуванням основа дорівнює e.
– `log10(x)`: повертає десятковий (за основою 10) логарифм числа `x`.
– `pow(x, y)`: повертає значення `x` у степені `y`.
– `sqrt(x)`: повертає квадратний корінь з `x`.

Тригонометричні функції:

– `sin(x)`: повертає синус кута `x`, заданого в радіанах.
– `cos(x)`: повертає косинус кута `x`, заданого в радіанах.
– `tan(x)`: повертає тангенс кута `x`, заданого в радіанах.
– `asin(x)`: повертає арксинус `x`.
– `acos(x)`: повертає арккосинус `x`.
– `atan(x)`: повертає арктангенс `x`.
– `degrees(x)`: конвертує кут `x` з радіанів у градуси.
– `radians(x)`: конвертує кут `x` з градусів у радіани.

Гіперболічні функції:

– `sinh(x)`: повертає гіперболічний синус `x`.
– `cosh(x)`: повертає гіперболічний косинус `x`.
– `tanh(x)`: повертає гіперболічний тангенс `x`.
– `asinh(x)`: повертає зворотний гіперболічний синус `x`.
– `acosh(x)`: повертає зворотний гіперболічний косинус `x`.
– `atanh(x)`: повертає зворотний гіперболічний тангенс `x`.

Спеціальні функції:

– erf(x): повертає функцію помилки для x.
– gamma(x): повертає гамма-функцію для x.

Константи:

– `pi`: число π (приблизно 3.14159).
– `e`: число e (приблизно 2.71828).

Розширений набір функцій для роботи з комплексними числами містить бібліотека `cmath`. Бібліотека містить додаткові функції і сталі. Для позначення уявної частини, поруч з її значенням використовують символ `j`. Комплексний тип у Python, позначають `complex`.

Наприклад,

```
import math

# Приклад використання деяких функцій
print(math.sqrt(16))       # Виведе: 4.0
print(math.log10(100))     # Виведе: 2.0
print(math.sin(math.pi/2)) # Виведе: 1.0
```

У Python списки (`list`) є фундаментальним типом даних, який представляє собою впорядковану змінювану колекцію елементів довільних типів. Вони дозволяють зберігати набір значень, до яких можна звертатися за індексом, змінювати, додавати або видаляти елементи.

Створення списків.

Списки можна створювати кількома способами. Перший спосіб – це за допомогою квадратних дужок.

Власне потрібно просто перерахувати елементи списку у квадратних дужках через кому:

```
# Порожній список
empty_list = []

# Список з елементами
numbers = [1, 2, 3, 4, 5]
fruits = ['яблуко', 'банан', 'вишня']
mixed = [1, 'два', 3.0, [4, 5]]
```

Другий – за допомогою функції `list()`:

```
# Створення списку з ітерованого об'єкта
chars = list('python')  # ['p', 'y', 't', 'h', 'o', 'n']
```

Список, це послідовність, у якій всі елементи занумеровані, починаючи з 0. У Python існує механізм по-елементної роботи зі списком. Для того, щоб звернутися до відповідного елемента списку, необхідно вказати номер цього елементу у квадратних дужках. Функція `len(list)` повертає кількість елементів у списку.

Завдання до лабораторної роботи

1. Записати до одновимірного масиву значення функції $f(x)$ на відрізку $[a; b]$ з заданим шагом h. Значення функції округлити до 4 знаків після коми. На екран вивести перші 10 значень з отриманого масиву. Знайти найменший і найбільший елемент масиву:

1) $f(x) = e^{-5x}(x^3 - 10x + 9) + \frac{\log_5^3 x}{x^2}$, $x \in [0{,}2; 2]$, $h = 0{,}2$.
2) $f(x) = \sin\left(\frac{3x}{2} + \pi\right) + x^3 - 5x^2 + 7x$, $x \in [0{,}01; 1]$, $h = 0{,}01$.
3) $f(x) = \sqrt{\frac{x^2+9x-1}{x+6}} + 7\cos(5\pi + x)$, $x \in [0{,}02; 4]$, $h = 0{,}02$.
4) $f(x) = \ln(1 + 5x) - \frac{e^{3x}+4}{x^3+8x+9}$, $x \in [0{,}04; 2]$, $h = 0{,}02$.
5) $f(x) = \frac{(x+5)^3}{x^5+x^2+2} - \operatorname{arctg}\left(\frac{x}{7}\right)$, $x \in [0{,}01; 1]$, $h = 0{,}01$.
6) $f(x) = e^{9x}(-x^5 + 9x - 12) + \frac{\log_8^3 2x}{x^4}$, $x \in [0{,}2; 2]$, $h = 0{,}2$.
7) $f(x) = \cos\left(\frac{5x}{2} + \pi\right) + x^5 - 8x^2 + 3x$, $x \in [0{,}01; 1]$, $h = 0{,}01$.

8) $f(x) = \sqrt{\frac{x^2+3x-1}{x+4}} + 10\sin(3\pi - 2x)$, $x \in [0{,}02; 4]$, $h = 0{,}02$.

9) $f(x) = \ln(1 + 10x) - \frac{e^{5x}+9}{x^3+10x+11}$, $x \in [0{,}04; 2]$, $h = 0{,}02$.

10) $f(x) = \frac{(x+2)^5}{x^3+3x^2+1} - \operatorname{arcctg}\left(\frac{x}{8}\right)$, $x \in [0{,}01; 1]$, $h = 0{,}01$.

2. Написати програму для наближеного обчислення:

1) e^{15}.
2) $\sin 2$.
3) $\cos 4$.
4) $\ln 3$.
5) $\operatorname{sh} 1$.
6) e^{17}.
7) $\sin 6$.
8) $\cos 9$.
9) $\ln 11$.
10) $\operatorname{sh} 4$.

3. Написати програми для обчислення:

1) суми всіх елементів матриці, що належать головній діагоналі;
2) кількості нульових елементів матриці;
3) суми всіх елементів матриці, що належать побічній діагоналі;
4) суми всіх недіагональних елементів матриці;
5) кількості ненульових елементів матриці.

4. Обчислити наближено визначений інтеграл

1) Від функції $f(x) = e^{-5x^2}$ на проміжку $[0; 1]$.
2) Від функції $f(x) = \cos 6x^2$ на проміжку $[0; 1]$.
3) Від функції $f(x) = \sin 3x^2$ на проміжку $[0; 1]$.
4) Від функції $f(x) = \frac{1}{\ln 2x}$ на проміжку $[0; 1]$.
5) Від функції $f(x) = \frac{\cos 7x}{x}$ на проміжку $[0; 1]$.
6) Від функції $f(x) = \frac{\sin 5x}{x}$ на проміжку $[0; 1]$.
7) Від функції $f(x) = e^{-12x^2}$ на проміжку $[0; 1]$.
8) Від функції $f(x) = \frac{1}{\ln 7x}$ на проміжку $[0; 1]$.
9) Від функції $f(x) = \frac{\cos 2x}{x}$ на проміжку $[0; 1]$.
10) Від функції $f(x) = \frac{\sin 8x}{x}$ на проміжку $[0; 1]$.

5. Обчислити наближено похідну

1) Від функції $f(x) = \frac{12x^7-8x^3-x}{x^4+x+1}$ у точці $x_0 = 2$.
2) Від функції $f(x) = \frac{\cos^5 x}{\sqrt{x}}$ у точці $x_0 = 5$.
3) Від функції $f(x) = (8x^3 - 3x)\sin 7x^2$ у точці $x_0 = -2$.

4) Від функції $f(x) = \ln\left(\frac{1}{x}\right)$ у точці $x_0 = 1$.
5) Від функції $f(x) = x \cdot \operatorname{arctg} x^3$ у точці $x_0 = \pi$.
6) Від функції $f(x) = x^{5x-9} + x^{20} - e^{3x}$ у точці $x_0 = -3$.
7) Від функції $f(x) = \operatorname{ctg}\left(\frac{1}{x^2}\right)$ у точці $x_0 = 6$.
8) Від функції $f(x) = e^{3\sin x + 8\cos x}$ у точці $x_0 = \frac{\pi}{2}$.
9) Від функції $f(x) = \frac{\operatorname{arctg} 4x}{16x^2+1}$ у точці $x_0 = 1$.
10) Від функції $f(x) = \log_4(9 \arcsin x^2)$ у точці $x_0 = 4$.

Питання для самоконтролю

1. Наведіть порядок виконання операцій у Python.
2. Перерахуйте арифметичні операції у Python.
3. Охарактеризуйте математичну бібліотеку `math`.
4. Наведіть функції модуля `math`, які відповідають за тригонометричні функції.
5. Наведіть функції модуля `math`, які відповідають за гіперболічні функції.
6. Наведіть функції модуля `math`, які відповідають за спеціальні функції і константи. Приведіть приклади.
7. Охарактеризуйте математичну бібліотеку `cmath`. Яка причина наявності двох модулів?
8. Охарактеризуйте комплексний тип у Python.
9. Наведіть способи створення списків.
10. За допомогою якої функція можна визначити кількість елементів у списку?

2. Лабораторна робота. Застосування мови Python до статистичних задач у теорії ймовірностей

Мета: засвоїти можливості роботи мови Python до застосування розв'язування статистичних задач теорії ймовірностей.

Python є потужним інструментом для розв'язання статистичних задач завдяки бібліотекам, таким як NumPy, Pandas, SciPy, Statsmodels, Scikit-learn, Matplotlib та Seaborn. Нижче наведені приклади задач зі статистики, які можна вирішити за допомогою Python.

Методичні рекомендації до написання коду

1. Описова статистика

Задача. Обчислити середнє значення, медіану, моду, дисперсію та стандартне відхилення.

```
import numpy as np
from scipy import stats

data = [12, 15, 14, 10, 8, 15, 18, 20]

mean = np.mean(data)
median = np.median(data)
mode = stats.mode(data)
variance = np.var(data, ddof=1)
std_dev = np.std(data, ddof=1)

print(f"Середнє: {mean}, Медіана: {median}, Мода: {mode.mode}, Дисперсія: {variance}, Стандартне відхилення: {std_dev}")
```

Результат:

Середнє: 14.0, Медіана: 14.5, Мода: 15, Дисперсія: 15.714285714285714, Стандартне відхилення: 3.9641248358604595

2. Тестування статистичних гіпотез

Задача: Виконати *t*-тест для перевірки середнього значення вибірки.

```
from scipy.stats import ttest_1samp

data = [2.3, 2.5, 2.8, 3.0, 2.9, 3.1]
t_stat, p_value = ttest_1samp(data, 3.0)  # Нульова гіпотеза: середнє значення = 3.0

print(f"T-статистика: {t_stat}, P-значення: {p_value}")
```

Результат:

T-статистика: -1.8576071235199598, P-значення: 0.12234647325914844

Задача: Виконати t-тест для двох вибірок.

```
from scipy.stats import ttest_ind

group1 = [23, 20, 22, 24, 25]
group2 = [30, 28, 27, 25, 26]
t_stat, p_value = ttest_ind(group1, group2)

print(f"T-статистика: {t_stat}, P-значення: {p_value}")
```

Результат:

T-статистика: -3.616777720717859, P-значення: 0.006814354918278612

3. Кореляція та регресія

Задача: Обчислити коефіцієнт кореляції між двома наборами даних.

```
from scipy.stats import pearsonr, spearmanr

x = [10, 20, 30, 40, 50]
y = [12, 24, 36, 48, 60]

pearson_corr, _ = pearsonr(x, y)
spearman_corr, _ = spearmanr(x, y)

print(f"Кореляція Пірсона: {pearson_corr}, Кореляція Спірмана: {spearman_corr}")
```

Результат:

Кореляція Пірсона: 0.9999999999999998, Кореляція Спірмана: 0.9999999999999999

Задача: Виконати лінійну регресію.

```
from scipy.stats import linregress

x = [1, 2, 3, 4, 5]
y = [2, 4, 5, 4, 5]

slope, intercept, r_value, p_value, std_err = linregress(x, y)

print(f"Нахил: {slope}, Перетин: {intercept}, R-квадрат: {r_value**2}, P-значення: {p_value}")
```

Результат:

Нахил: 0.6000000000000001, Перетин: 2.1999999999999997, R-квадрат: 0.6000000000000002, P-значення: 0.12402706265755441

4. Візуалізація статистичних даних

Задача: Побудувати гістограму (див. рис. 2.1) та ящик boxplot (див. рис. 2.2).

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns

data = [12, 15, 14, 10, 8, 15, 18, 20]

plt.hist(data, bins=5, color='blue', alpha=0.7)
plt.title('Гістограма')
plt.xlabel('Значення')
plt.ylabel('Частота')
plt.show()

sns.boxplot(data)
plt.title('Boxplot')
plt.show()
```

Результат:

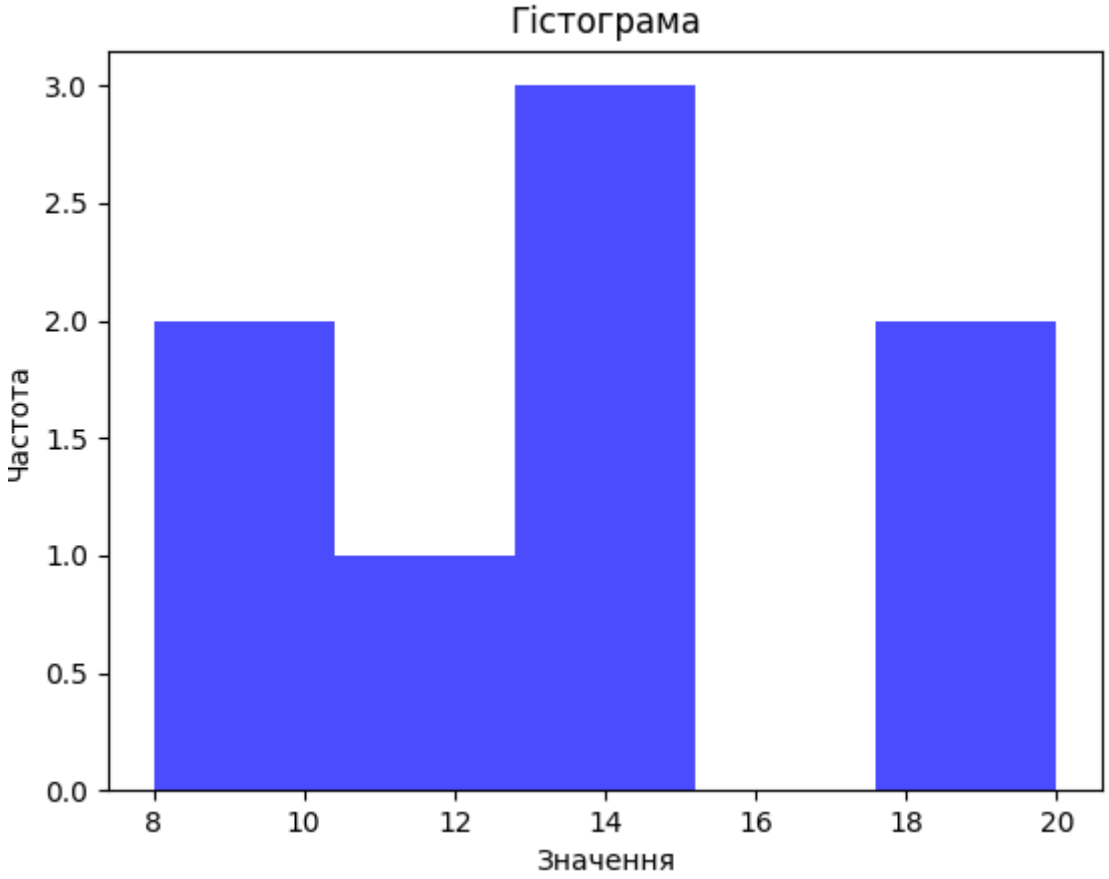

Рисунок 2.1 – Вивід гістограми засобами `matplotlib.pyplot`

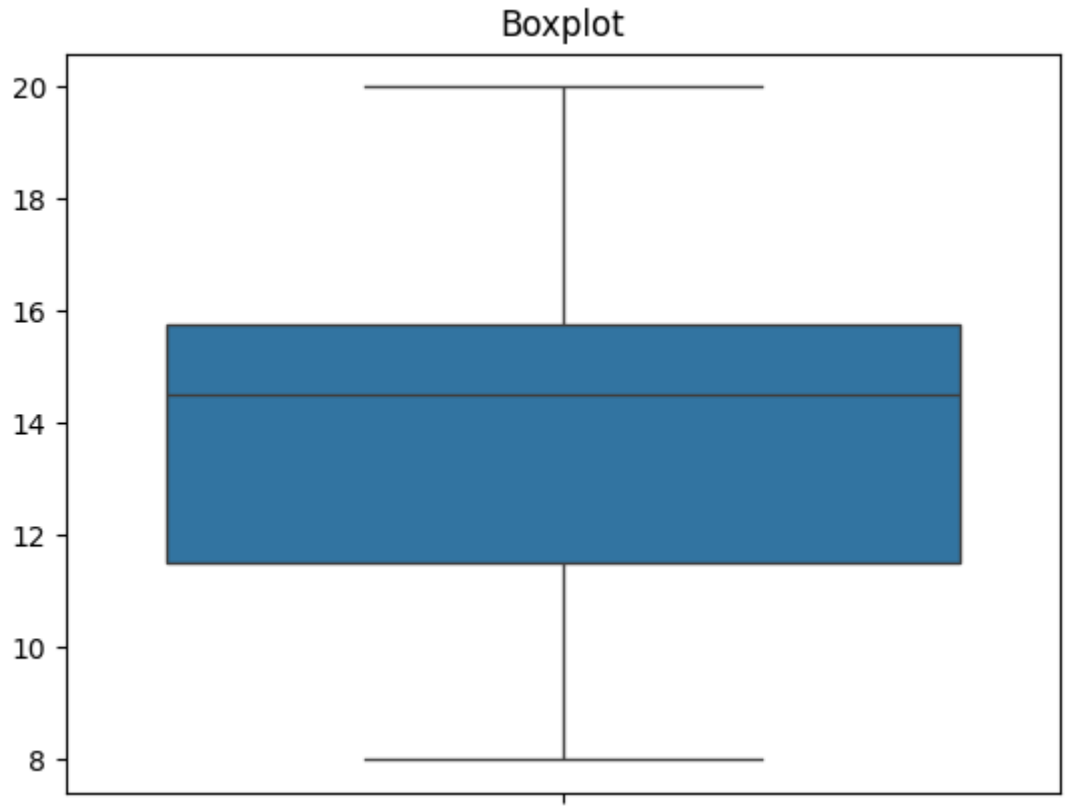

Рисунок 2.2 – Вивід boxplot

5. Аналіз розподілу

Задача: Перевірити, чи відповідає вибірка нормальному розподілу.

```
from scipy.stats import shapiro

data = [12, 15, 14, 10, 8, 15, 18, 20]
stat, p = shapiro(data)
print(f"Статистика  Шапіро-Уілка:  {stat},  P-значення:
{p}")
if p > 0.05:
    print("Дані відповідають нормальному розподілу.")
else:
    print("Дані не відповідають нормальному розподілу.")
```

Результат:

Статистика Шапіро-Уілка: 0.9781675297855071, P-значення: 0.9532741883063235

Дані відповідають нормальному розподілу.

6. Статистичне моделювання

Задача: Генерація вибірок та моделювання розподілів (див. рис. 2.3).

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

# Генерація нормального розподілу
```

```
data = np.random.normal(loc=0, scale=1, size=1000)

plt.hist(data, bins=30, density=True, alpha=0.6,
color='g')
plt.title('Нормальний розподіл')
plt.show()
```

Результат:

Рисунок 2.3 – Вивід нормального розподілу

7. Аналіз варіації (ANOVA)

Односторонній дисперсійний аналіз (ANOVA) – це статистичний метод, який використовується для перевірки значущих відмінностей між середніми значеннями груп даних. Він зазвичай використовується в експериментальних дослідженнях для порівняння впливу різних методів лікування або втручань на певний результат.

Основна ідея ANOVA полягає в тому, щоб розділити загальну варіабельність даних на два компоненти: варіабельність між групами (внаслідок лікування) і варіабельність всередині кожної групи (внаслідок випадкової варіабельності та індивідуальних відмінностей). Тест ANOVA обчислює F-статистику, яка є відношенням міжгрупової варіації до внутрішньогрупової варіації.

Односторонній дисперсійний аналіз – це статистичний тест, який використовується для визначення того, чи існують значущі відмінності між середніми значеннями двох або більше незалежних груп. Він використовується для перевірки нульової гіпотези про те, що середні всіх груп рівні, проти

альтернативної гіпотези про те, що принаймні одне середнє відрізняється від інших.

Задача: Провести однофакторний ANOVA-аналіз.

```
from scipy.stats import f_oneway

group1 = [23, 20, 22, 24, 25]
group2 = [30, 28, 27, 25, 26]
group3 = [20, 21, 22, 23, 24]

f_stat, p_value = f_oneway(group1, group2, group3)
print(f"F-статистика: {f_stat}, P-значення: {p_value}")
```

Результат:

F-статистика: 11.878787878787882, P-значення: 0.0014284961417153712

8. Байєсівський аналіз

Задача: Оцінити ймовірності за допомогою формули Байєса.

```
# Ймовірність A та B
p_a = 0.3  # P(A)
p_b = 0.4  # P(B)
p_b_given_a = 0.7  # P(B|A)

# P(A|B)
p_a_given_b = (p_b_given_a * p_a) / p_b
print(f"P(A|B): {p_a_given_b}")
```

Результат:

P(A|B): 0.5249999999999999

9. Кластеризація даних

Scikit-learn – це модуль Python для машинного навчання, створений на основі SciPy і розповсюджується за ліцензією 3-Clause BSD.

Задача: Виконати кластеризацію методом k-means.

```
from sklearn.cluster import KMeans
import numpy as np

data = np.array([[1, 2], [1, 4], [1, 0],
                 [10, 2], [10, 4], [10, 0]])

kmeans = KMeans(n_clusters=2, random_state=0).fit(data)
print(f"Кластери: {kmeans.labels_}")
```

Результат:

Кластери: [1 1 1 0 0 0]

Ці задачі демонструють можливості Python для аналізу, обробки даних і вирішення статистичних проблем, забезпечуючи точні результати та можливість легкої візуалізації.

Завдання до лабораторної роботи

1. Згенерувати випадкову вибірку випадкових величин, об’ємом 100 елементів (цілі числа). Використовуючи бібліотеку NumPy, обчислити середнє значення, медіану, моду, дисперсію та стандартне відхилення.

2. Згенерувати випадкову вибірку випадкових величин, об’ємом 50 елементів (дробові числа, які містять один знак після коми). Виконати *t*-тест для перевірки середнього значення вибірки. Пояснити отриманий результат.

3. Згенерувати дві випадкові вибірки випадкових величин, об’ємом 20 елементів (цілі числа). Обчислити коефіцієнт кореляції між двома наборами даних (кореляція Пірсона, кореляція Спірмана). Пояснити отриманий результат.

4. Згенерувати випадкову вибірку випадкових величин, об’ємом 25 елементів (цілі числа). Зробити візуалізація статистичних даних за допомогою бібліотек Matplotlib і Seaborn.

5. Згенерувати випадкову вибірку випадкових величин, об’ємом 75 елементів (цілі числа). Перевірити, чи відповідає вибірка нормальному розподілу. Пояснити отриманий результат.

6. Згенерувати три випадкові вибірки випадкових величин, об’ємом 20 елементів (цілі числа). Провести односторонній дисперсійний аналіз (ANOVA-аналіз). Пояснити отриманий результат.

Питання для самоконтролю

1. Наведіть приклади сучасних бібліотек мови Python, які використовуються для розв’язання статистичних задач. Охарактеризуйте їх можливості.
2. Сформулюйте означення таких понять: середнє значення вибірки, медіана, мода, дисперсія та стандартне відхилення вибірки.
3. Як можна обчислити середнє значення, медіану, моду, дисперсію та стандартне відхилення вибірки?
4. Що називають коефіцієнтом кореляції між двома наборами даних?
5. Як робиться візуалізація статистичних даних за допомогою бібліотек Matplotlib і Seaborn?
6. Охарактеризуйте коефіцієнт кореляції Пірсона та коефіцієнт кореляції Спірмана. Що вони означають?
7. Що таке односторонній дисперсійний аналіз?

3. Лабораторна робота. Застосування мови Python до аналітичних обчислень символьної математики

Мета: засвоїти можливості роботи мови Python до аналітичних обчислень, таких як диференціювання, інтегрування, обчислення границь, роботу з матрицями та графіко.

Теоретичні відомості та методичні рекомендації

SymPy – це бібліотека символьної математики мови Python. Вона є реальною альтернативою таким математичним пакетам як Mathematica або Maple, і володіє дуже простим та легко розширюваним кодом. SymPy написана виключно на мові Python і не вимагає ніяких сторонніх бібліотек. Ця бібліотека є потужним інструментом для тих, хто працює з математичними обчисленнями в Python

Після встановлення бібліотеки SymPy, її треба підключити за допомогою інструкції `import`. Наприклад, використовуючи псевдонім «`sm`»:

```
import sympy as sm
```

На відміну від інших систем комп'ютерної алгебри, в SymPy можна в явному вигляді задавати символьні змінні:

```
x = sm.Symbol('x')
```

або так:

```
y, z = sm.symbols('y z')
```

Після їх завдання з ними можна робити різні маніпуляції з використанням деяких операторів мови Python, а саме, арифметичних та логічних. Наприклад, використовуючи функцію `pprint(expression, use_unicode=True)` з бібліотеки SymPy, де `expression` – математичний вираз для виводу, `use_unicode=True` – параметр, що вмикає юнікод-символи (за замовчуванням `True`). Якщо встановити `False`, буде використовуватись ASCII (див. рис. 3.1):

```
sm.pprint(x ** 2 + y + sm.sqrt(z), use_unicode=False)
```

```
 2      ___
x  + y + \/ z
```

Рисунок 3.1 – Вивід результату у консоль (Unicode) за допомогою функцію `pprint()`

Для кращого візуального відображення алгебраїчних виразів на екрані у бібліотеці SymPy використовується функція `init_printing()`. `Init_printing` – це функція з бібліотеки SymPy, яка налаштовує вивід математичних виразів у красивому форматі. Вона робить представлення символьних обчислень більш читабельним, наприклад, у вигляді LaTeX або юнікод-символів. Функція `init_printing()` автоматично налаштовує найкращий доступний метод візуалізації залежно від середовища (Jupyter Notebook, консоль, тощо). Якщо працюєте в Jupyter Notebook і хочете ще гарніший вивід, краще використовувати `init_printing()`.

У бібліотеці SymPy знаходити границі функцій можна за допомогою функції `limit()`. Вона дозволяє обчислювати границі виразів при заданому значенні змінної, включаючи однобічні границі. Синтаксис такий:

`limit(expression, variable, point, dir='+'),`

де `expression` – вираз, границю якого потрібно знайти, `variable` – змінна, за якою обчислюється границя, `point` – точка, до якої прямує змінна, `dir` – необов'язковий параметр для знаходження односторонніх границь (`'+'` для правосторонньої, `'-'` для лівосторонньої).

Наприклад,

```
import sympy as sm
x = sm.Symbol('x')
expr = x / (x + 1)
result = sm.limit(expr, x, float('inf'))
sm.pprint(result, use_unicode=False)
```

Результат, 1.

У бібліотеці SymPy для обчислення похідних використовується функція `diff()`. Вона дозволяє знаходити похідні будь-якого порядку від алгебраїчних, тригонометричних та інших функцій. Синтаксис такий:

`diff(function, variable, order),`

де `function` – функція, яку треба продиференціювати, `variable` – ім'я змінної, по якій проводиться диференціювання; `order` – порядок похідної (за замовчуванням це перша похідна). Зрозуміло, що якщо функція має кілька змінних, то можна брати часткові похідні за допомогою цієї ж функції.

Для обчислення невизначених інтегралів використовується метод:

`integrate(function, variable),`

де `function` – функція, яку треба проінтегрувати, `variable` – ім'я змінної інтегрування.

Для обчислення визначених інтегралів використовується метод:

`integrate(function, (variable, a, b)),`

де `function` – функція, яку треба проінтегрувати, `variable` – ім'я змінної інтегрування; `a, b` – межі інтегрування.

Для створення матриці з елементами, які задані у вигляді списку `list`, у бібліотеці SymPy використовується клас `Matrix(list)`.

Наприклад, матриця 2×2 задається командою:

```
A = Matrix([[1, 2], [3, 4]])  # 2×2 матриця
```

Клас `Matrix` має багато корисних методів для роботи з матрицями: обчислення визначників, знаходження обернених матриць, транспонування, рангу, власних значень тощо. `Matrix` у SymPy дозволяє легко працювати з матрицями в Python.

SymPy дозволяє будувати графіки функцій заданих явно, заданих неявно, заданих параметрично та будувати поверхні, а також зображати області між

функціями. Для побудови графіка функції, яка задано явно у декартовій системі координат, використовується метод

```
plot(function, (variable, a, b)),
```

де `function` – функція, яку треба побудувати, `variable` – ім'я змінної; `a, b` – межі по осі *OX*.

Наприклад, графік поверхні $y = x \cdot \sin x$ має вигляд (див. рис. 3.2):

```
import sympy as sm
from sympy.plotting import plot3d
x = sm.Symbol('x')
plot3d(x * sm.sin(x), (x, -10, 10), (y, -8, 8))
```

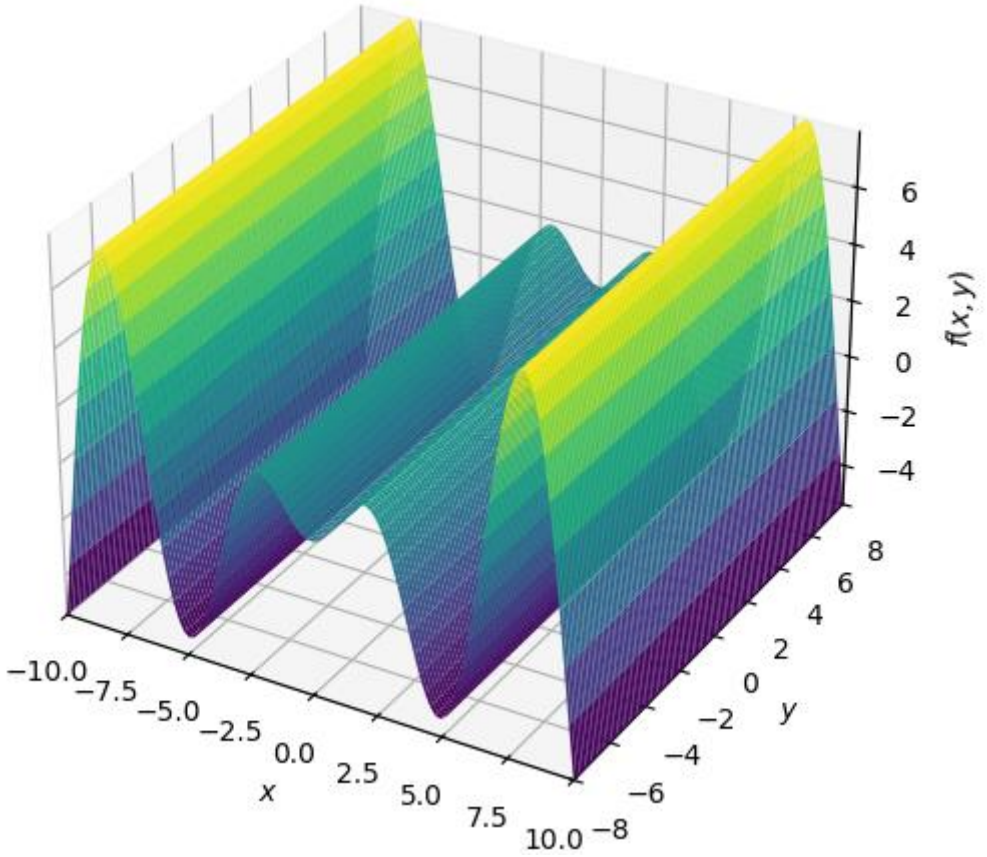

Рисунок 3.2 – Результат роботи `plot3d()`

Завдання до лабораторної роботи

1. Обчислити границі функції:

1) a) $\lim\limits_{x\to 1}(1-x)\operatorname{tg}\frac{\pi x}{2}$; b) $\lim\limits_{x\to\infty}\frac{2x-1}{3x^2+4x+5}$.

2) a) $\lim\limits_{x\to 1-0}\operatorname{arctg}\frac{1}{1-x}$; b) $\lim\limits_{x\to\infty}\frac{x\sqrt[3]{5x^2}+\sqrt[4]{9x^8+1}}{x^5+\sqrt{x}}$.

3) a) $\lim\limits_{x\to 1+0}\operatorname{arctg}\frac{1}{1-x}$; b) $\lim\limits_{x\to 0}\frac{\sin 3x}{\sin 5x}$.

4) a) $\lim\limits_{x\to 0}\frac{6x}{2-e^{3x}-\cos 4x}$; b) $\lim\limits_{x\to\infty}\left(\frac{x+2}{x-1}\right)^{x^2}$.

5) a) $\lim\limits_{x\to 0}\frac{\ln(x^2+1)}{\sin 2x}$; b) $\lim\limits_{x\to\infty}\frac{5x^2+3x+1}{3x^2+4x+5}$.

6) a) $\lim\limits_{x\to 0}\frac{e^{4x}-\cos 2x}{\sin 2x}$; b) $\lim\limits_{x\to\infty}\frac{\sqrt[4]{x^2+2}-5x^2}{x-\sqrt{x^4-x+1}}$.

7) a) $\lim\limits_{x\to 1}\frac{\sin 7x}{3x^2}$; b) $\lim\limits_{x\to\infty}\left(\frac{x+6}{x-2}\right)^{x+1}$.

8) a) $\lim\limits_{x\to 1}\frac{\ln(x^2+1)}{\sin 9x}$; b) $\lim\limits_{x\to\infty}\frac{x(x+36)}{4x^2+1}$.

9) a) $\lim\limits_{x\to 1}\frac{x+\sqrt[4]{81x^2-1}}{x+4\sqrt{x^5}}$; b) $\lim\limits_{x\to 0}\frac{\sin^2 2x}{x\sin 2x}$.

10) a) $\lim\limits_{x\to\infty}\left(\frac{1-2x}{3-2x}\right)^{4x}$; b) $\lim\limits_{x\to 0}\frac{e^{12x}-\cos 5x}{\sin 4x}$.

2. Обчислити похідну. Знайти y'' та y'''. Знайти всі частинні похідні:

1) a) $y=\frac{\sin 3x}{\operatorname{arctg} 4x}$; b) $y=\ln^5(x+1)$; c) $u=\sqrt{\frac{y^3}{x}}$.

2) a) $y=\frac{1}{4}\operatorname{ctg}^2(2x-1)-\operatorname{arctg}\frac{5}{x}+3$; b) $y=x^5+\sqrt{x}$; c) $u=x^{\frac{1}{y}}$.

3) a) $y=e^{6x-1}\cdot\arccos 2x$; b) $y=2-x^2+\sqrt{x}$; c) $u=e^{\frac{x}{y}}$.

4) a) $y=\sin^2 2x-\operatorname{tg}\sqrt{x-2}$; b) $y=\sin(-2x)$; c) $u=\sqrt{x^2+y^2}$.

5) a) $y=7\sqrt{6x^4+1}\cdot\sin 6x$; b) $y=\frac{1}{x}$; c) $u=\frac{\cos x^2}{3y}$.

6) a) $y=x^2-\ln(3+x^3)$; b) $y=\cos\sqrt[5]{7x^2}$; c) $u=\ln\left(\frac{x}{y}\right)-\sin y^3$.

7) a) $y=\frac{2\operatorname{tg}6x}{e^{3x}+2}$; b) $y=x-\operatorname{arctg}\sqrt{x}-\ln\sqrt[3]{1+x}$; c) $u=\ln\left(\frac{y}{x^2}\right)$.

8) a) $y=\ln x^3-\operatorname{tg}x-\sqrt[2]{x}$; b) $y=\frac{\sqrt{1+x^3}}{3\arcsin 2x}$; c) $u=\arccos\sqrt{\frac{x}{y^3}}$.

9) a) $y=\frac{2x^5-1}{3\operatorname{tg}4x}$; b) $y=\sin\sqrt{3x}+\frac{1}{3}\sin\frac{3}{x}-\operatorname{ctg}6x$; c) $u=e^{-\frac{y}{x^2}}$.

10) a) $y=x^3-\arcsin\sqrt{x}+\sqrt{1-x^2}$; b) $y=\sqrt{1-4x}$; c) $u=\frac{y^2}{x}$.

3. Знайти значення інтеграла (невизначеного, визначеного, невласного):

1) a) $\int\cos x\cos 2x\cos 3x\,dx$; b) $\int_0^{\frac{\pi}{2}}x\cos x\,dx$; c) $\int_1^{+\infty}\frac{dx}{\sqrt{x}}$.

2) a) $\int\frac{3x^4+4}{x^2(x^2+1)^3}dx$; b) $\int_{\frac{\pi}{6}}^{\frac{\pi}{4}}\frac{\cos x}{x}dx$; c) $\int_0^{+\infty}\frac{\sin x\cos x dx}{x}$.

3) a) $\int x^3\sin x\,dx$; b) $\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\frac{dx}{1+\cos x}$; c) $\int_1^{+\infty}\frac{dx}{1+x^2}$.

4) a) $\int\frac{x^3-6}{x^4+6x^2+8}dx$; b) $\int_0^{\frac{\pi}{2}}x^3\cos x\,dx$; c) $\int_0^{+\infty}\frac{1-e^{-2x^2}}{xe^{x^2}}dx$.

5) a) $\int\operatorname{tg}^2 x\,dx$; b) $\int_0^{\frac{\pi}{2}}\frac{dx}{5-4\sin x+3\cos x}$; c) $\int_1^{+\infty}\frac{\operatorname{arctg}4x}{x^2}dx$.

6) a) $\int\frac{dx}{\cos^2 x\sin^2 x}$; b) $\int_0^{\frac{\pi}{2}}\frac{\sin x\cos x}{(\cos^2 x+\sin^2 x)^2}dx$; c) $\int_1^{+\infty}\frac{\ln(x^2+1)}{x}dx$.

7) a) $\int\frac{dx}{\cos 2x+\sin^2 x}$; b) $\int_0^{\pi}x^3\sin x\,dx$; c) $\int_0^{+\infty}e^{-2x}dx$.

8) a) $\int\frac{\cos 2x dx}{\cos^2 x\sin^2 x}$; b) $\int_0^{e}\ln^3 x\,dx$; c) $\int_0^{+\infty}\frac{dx}{x^2+4}$.

9) a) $\int \frac{3x-1}{x^2+9}dx$; b) $\int_4^9 \frac{x-1}{\sqrt{x}+1}dx$; c) $\int_0^{+\infty} e^{-x}\sqrt{x}dx$.

10) a) $\int \frac{dx}{\sqrt{1+x}+\sqrt{x-1}}$; b) $\int_{\frac{\pi}{4}}^{\frac{\pi}{3}} \frac{x}{\sin^2 x}dx$; c) $\int_1^{+\infty} \frac{x^3+1}{x^4}dx$.

4. Для матриць

$$A = \begin{pmatrix} 2 & 8-n & h-7 \\ -6 & 2h+n & n+5 \\ -3 & 3h-n & -2 \end{pmatrix}, \quad B = \begin{pmatrix} -9 & -n & -h+1 \\ 1 & h & 5-n \\ 6 & n & 9 \end{pmatrix},$$

де n – номер варіанту, h – остання цифра номеру вашої групи, знайти: 1) AB; 2) BA; 3) $nA - hB$; 4) det$A$ та det$B$; 5) A^{-1} та B^{-1}; 6) ранг матриці A та ранг матриці B; 7) власні значення матриці A та матриці B.

5. Побудувати графіки функцій у декартовій системі координат:

1) a) $y = \frac{x^5}{(x^2-1)^2}$; b) $y = \sin x + \frac{1}{2}\sin 2x + \frac{1}{3}\sin 3x$.

2) a) $y = \sqrt[3]{x(3-x)^2} - x$; b) $y = \left(\frac{x+1}{x-1}\right)^4$.

3) a) $y = \frac{20x^2}{(x-1)^3}$; b) $y = \frac{\ln^2 x}{x}$.

4) a) $y = 8x^2(x^2-1)^3$; b) $y = \frac{x}{2} - \arccos\frac{2x}{1+x^2}$.

5) a) $y = \frac{x^5-8}{x^4}$; b) $y = \arccos\frac{1-x^2}{1+x^2}$.

6) a) $y = \frac{(x-5)^3}{(x-7)^2}$; b) $y = \frac{x}{\ln x}$.

7) a) $y = \frac{x^3-2x^2-x+2}{x}$; b) $y = \frac{x^2\sqrt{x^2-1}}{2x^2-1}$.

8) a) $y = \frac{x^3}{x^2-1}$; b) $y = (x^2-2)e^{-2x}$.

9) a) $y = \frac{1+x^2}{1+(x-2)^2}$; b) $y = \ln\left|\frac{1-x}{1+x}\right| + \frac{6}{1+x}$.

10) a) $y = \frac{x^2+2x-3}{x} - e^{\frac{1}{x}}$; b) $y = \frac{x}{2} + 2\,\mathrm{arctg}\,x$.

Питання для самоконтролю

1. Охарактеризуйте можливості символьної математики SymPy мови Python.
2. Де використовується SymPy? Наведіть приклади.
3. Які функції використовуються у SymPy для знаходження границь, похідних та інтегралів?
4. Охарактеризуйте методи класу Matrix.
5. Які операції можна робити з матрицями у SymPy?
6. Які функції дозволяють побудови графіки функцій у SymPy?
7. У яких системах координат можна побудувати графік функції у SymPy?

4. Лабораторна робота. Застосування бібліотеки NumPy до задач лінійної алгебри

Мета: засвоїти можливості роботи бібліотеки NumPy до математичних обчислень, таких як операції з багатовимірними масивами, операції з матрицями, розв'язання систем лінійних рівнянь, використання вбудованих математичних функцій.

Теоретичні відомості та методичні рекомендації

Бібліотека NumPy (Numerical Python) є основним інструментом для наукових обчислень у Python. Вона забезпечує ефективні методи роботи з багатовимірними масивами та містить велику кількість функцій для виконання математичних операцій. Бібліотека NumPy значно спрощує математичні обчислення, роблячи їх ефективнішими, швидшими та зручнішими для реалізації. Вона незамінна у чисельному аналізі, машинному навчанні, моделюванні, інженерії та багатьох інших галузях.

NumPy має велику кількість підмодулів. Зазвичай для звичайного використання NumPy потрібен лише основний простір імен і менший набір підмодулів. Решта – це підмодулі спеціального призначення.

Рекомендовані простори імен для загального використання:

- numpy;
- numpy.exceptions;
- numpy.fft;
- numpy.linalg;
- numpy.polynomial;
- numpy.random;
- numpy.strings;
- numpy.testing;
- numpy.typing.

Простори імен спеціального призначення:

- numpy.ctypeslib;
- numpy.dtypes;
- numpy.emath;
- numpy.lib;
- numpy.rec;
- numpy.version.

Застарілі простори імен, які бажано не використовувати для нового коду:

- numpy.char;
- numpy.distutils;
- numpy.f2py;
- numpy.ma;
- numpy.matlib.

В NumPy основним об’єктом для зберігання та обробки числових даних є багатовимірний масив (`numpy.ndarray`). Найчастіше це одномірна послідовність або двовимірна таблиця, заповнені елементами одного типу, як правило, числами, які проіндексовані кортежом додатних цілих чисел. У NumPy елементи цього кортежу називаються осями (див. рис. 4.1), а число осей рангом. Цей об’єкт значно ефективніший за стандартні списки Python, оскільки зберігає всі елементи одного типу та підтримує векторизовані операції.

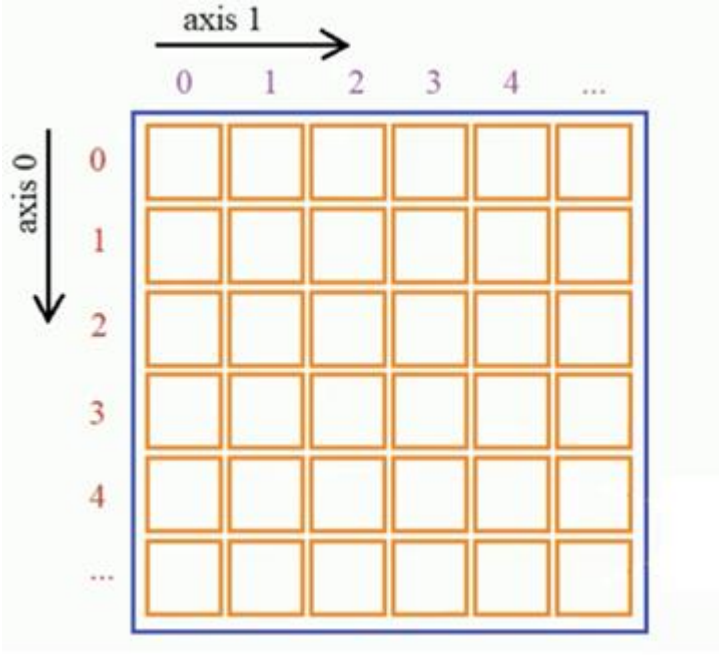

Рисунок 4.1 – Напрямок осей у двовимірному масиві у NumPy

Масив `numpy.ndarray` можна створити за допомогою функції `np.array(object, dtype)`, де `object` – список або інша структура даних, що конвертується в `ndarray`, `dtype` – (необов’язково) тип даних масиву (`int`, `float`, `complex`, `bool`, тощо). Вона дозволяє конвертувати звичайні списки Python у масиви NumPy, які більш ефективні для числових обчислень.

Наприклад,

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])
print(arr)  # [1 2 3 4 5]
print(type(arr))

arr_float = np.array([1, 2, 3], dtype=float)
print(arr_float)  # [1. 2. 3.]

arr_bool = np.array([0, 1, 2], dtype=bool)
print(arr_bool)  # [False  True  True]

arr_2d = np.array([[1, 2, 3], [4, 5, 6]])
```

```
print(arr_2d)  # [[1 2 3]  #  [4 5 6]]
```

Кожен `ndarray` має корисні методи:

- `shape` – розмірність `ndarray`;
- `size` – загальна кількість елементів `ndarray`;
- `ndim` – кількість вимірів `ndarray`;
- `dtype` – тип даних `ndarray`;
- `itemsize` – розмір одного елемента (в байтах) `ndarray`.

Наприклад,

```
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

print(arr.shape)  # (2, 3) - розмірність (рядки, стовпці)
print(arr.size)   # 6 - загальна кількість елементів
print(arr.ndim)   # 2 - кількість вимірів
print(arr.dtype)  # int32 або int64 - тип даних
print(arr.itemsize)  # розмір одного елемента (в байтах)
```

NumPy підтримує матричні операції. Нехай треба задати матрицю 3×2. Треба вказати зовнішній список у квадратних дужках `[]`. А в середині вкладені списки. У результаті ми отримаємо *двовимірний масив* розмінністю 3 рядка і 2 стовпчика:

```
import numpy as np

a = np.array([[1, 2], [3, 4], [5, 6]])
print(a)
```

В результаті отримаємо матрицю 3×2:

```
[[1 2]
 [3 4]
 [5 6]]
```

NumPy містить кілька вбудованих функцій для генерації матриць різних типів:

- `np.zeros(shape, dtype=float)` – створює матрицю нулів заданої форми;
- `np.ones(shape, dtype=float)` – створює матрицю одиниць заданої форми;
- `np.full(shape, fill_value)` – створює матрицю, заповнену певним числом `fill_value`;
- `np.eye(N, M=None, k=0)` – створює одиничну матрицю (або діагональну, якщо `M` не `None`, `k` – номер діагоналі);
- `np.random.rand(N, M)` – створює матрицю випадкових чисел з діапазону `[0, 1)`;

– `np.arange(start, stop, step)` – створює масив чисел у заданому діапазоні;
– `np.linspace(start, stop, num)` – створює масив рівномірно розподілених чисел.

Ці функції корисні для лінійної алгебри, моделювання, машинного навчання та інших задач.

NumPy містить кілька функцій для формування масивів на основі існуючих даних. Вони дозволяють змінювати структуру, розмірність і вміст масивів без необхідності створювати їх з нуля.

Функції копіювання та конвертації:

– `np.array(object, copy=True)` – створює новий масив з існуючих даних;
– `np.copy(arr)` – створює повну копію масиву.

Зміна форми масиву:

– `np.reshape(arr, new_shape)` – змінює форму масиву без зміни даних;
– `arr.flatten()` та `arr.ravel()` – перетворюють багатовимірний масив в одновимірний.

Додавання нових осей:

– `np.expand_dims(arr, axis)` – додає новий вимір;
– `arr[:, np.newaxis]` – альтернативний спосіб.

Об’єднання масивів:

– `np.concatenate((arr1, arr2), axis)` – об’єднує масиви вздовж заданої осі;
– `np.hstack()` та `np.vstack()` – горизонтальне з’єднання та вертикальне з’єднання.

Розбиття масивів:

– `np.split(arr, sections, axis)` – розбиває масиву на рівні частини;
– `np.hsplit()` та `np.vsplit()` – спрощені варіанти попередніх команд.

Повторення елементів:

– `np.tile(arr, reps)` – повторює масив певну кількість разів;
– `np.repeat(arr, repeats, axis)` – повторює кожен елемент певну кількість разів.

NumPy надає широкий набір матричних операцій, які дозволяють виконувати базові та розширені лінійні алгебраїчні обчислення (використовуючи підмодуль `linalg`):

– + і – додавання та віднімання;
– – множення елементів (елементне множення);
– `@` або `np.dot()` – матричне множення;

– `T` – Транспонування;
– `inv` – обернена матриця;
– `det` – визначник;
– `trace` – слід матриці;
– `matrix_rank` – ранг матриці;
– `eig` – власні значення;
– `svd` – сингулярний розклад;

Наприклад, знайдемо транспоновану і обернену матрицю:

```
import numpy as np

A = np.array([(1, -2, 3), (4, 5, -6), (-7, 8, 9)],
'float64')
print(A)

A_T = A.T
print(A_T)

A_inv = np.linalg.inv(A)
print(A_inv)
```

У результаті отримаємо:

```
[[ 1. -2.  3.]
 [ 4.  5. -6.]
 [-7.  8.  9.]]
[[ 1.  4. -7.]
 [-2.  5.  8.]
 [ 3. -6.  9.]]
[[ 0.32978723  0.14893617 -0.0106383 ]
 [ 0.0212766   0.10638298  0.06382979]
 [ 0.23758865  0.0212766   0.04609929]]
```

Системи $Ax = b$ алгебраїчних рівнянь можна розв'язувати, використовуючи метод `solve` з підмодуля `linalg`.

Завдання до лабораторної роботи

1. Створити масиви

$$A = \begin{pmatrix} -8 & 10-n & h-2 \\ 5 & 2h+n & n+8 \\ -2 & 3h-n & -1 \end{pmatrix},\ B = \begin{pmatrix} -7 & -n & -h+1 \\ 4 & h & 6-n \\ 3 & n & 7 \end{pmatrix},\ C = \begin{pmatrix} n \\ h \\ n+h \end{pmatrix},$$

де n – номер варіанту, h – остання цифра номеру вашої групи.

1) Перевірити, однакові чи неоднакові масиви A та B.
2) Знайти добуток матриць $A \cdot B$.
3) Замінити знак елементів у масиві A, значення яких знаходяться між 1 та 10.

4) Замінити знак елементів у масиві B, значення яких знаходяться між 5 та 8.
5) Знайти мінімальний та максимальний елемент масивів A та B.
6) Перетворити масив A із float в int.
7) Знайти діагональні елементи добутку матриць $A \cdot B$.
8) Поміняти місцями 2 рядки у матриці B.
9) Обчислити ранги матриць A та B.
10) Знайти 3 найбільших значень у масиві A. Знайти 2 найбільших значень у масиві B.
11) Розв'язати систему лінійних рівнянь: $Ax = C$ двома способами.

2. Створити одновимірний масив даних, в якому буде вказана зарплата вчителя по місяцям за поточний рік (брати у діапазоні від 7500 гривень до 15500 гривень). Нехай податок на доходи фізичних осіб обчислюється так. Базова ставка становить 12%. Якщо в якомусь місяці заробіток вчителя за рік складе більше 50000 гривень, то на частину року яка залишилася (не включаючи цей місяць) встановлюється ставка в 20%. Наприклад, якщо вчитель заробляє щомісяця 8000 гривень, то до липня вчитель отримає $8000 \cdot 7 = 56000$ гривень і починаючи з серпня податок на доходи фізичних осіб нараховуватиметься за ставкою 20%. Написати функцію `calculate()`, яка приймає на вхід масив, що містить дохід за кожен місяць року, починаючи з першого та повертає загальну суму податку, який належить заплатити за рік.

Питання для самоконтролю

1. Охарактеризуйте можливості бібліотеки NumPy мови Python.
2. Що таке numpy.ndarray?
3. Наведіть корисні корисні методи `ndarray`.
4. Розкажіть про функції для формування масивів на основі існуючих даних у NumPy.
5. Чи можна виконувати матричні дії NumPy?
6. Охарактеризуйте особливості роботи з матрицями у NumPy.
7. Як виконується елементне множення у NumPy?
8. Якими способами можна розв'язати системи алгебраїчних рівнянь у NumPy?

5. Лабораторна робота. Застосування бібліотеки Matplotlib до візуалізації функцій

Мета: засвоїти можливості роботи бібліотеки Matplotlib до візуалізації функцій у різних системах координат та візуалізації для аналізу даних.

Теоретичні відомості та методичні рекомендації

Бібліотека Matplotlib – це одна з найпопулярніших бібліотек для візуалізації даних у мові Python. Matplotlib – це бібліотека для візуалізації даних двовимірною та тривимірною графікою. Matplotlib є гнучким, легко конфігурованим пакетом, який разом з NumPy, SciPy і IPython надає можливості, подібні до MatLab.

Основні можливості Matplotlib:

- побудова лінійних графіків, гістограм, діаграм розсіювання, кругових діаграм тощо;
- гнучке налаштування кольорів, підписів, осей та стилів;
- можливість збереження графіків у різних форматах (png, pdf, svg);
- підтримка роботи з іншими бібліотеками (NumPy, Pandas, Seaborn).

Основні компоненти (див. рис. 5.1):

- pyplot — основний модуль, що містить функції для побудови графіків;
- Figure — основний об’єкт для побудови графіків;
- Axes — область побудови графіка всередині Figure;
- plot — безпосередньо сам графік (лінії, точки, гістограми тощо).

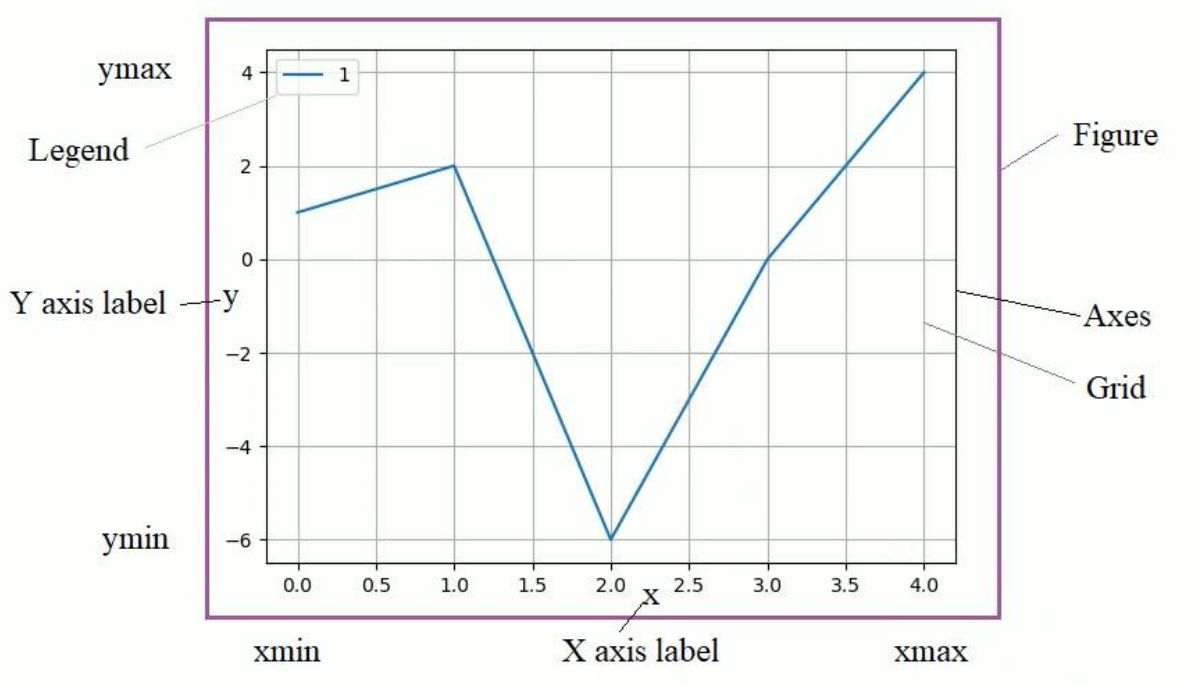

Рисунок 5.1 – Базові компоненти бібліотека Matplotlib

У процесі роботи бібліотека Matplotlib виводить графічну інформацію на екран, використовуючи пакет Tkinter для роботи з користувацьким інтерфейсом. Пакет Tkinter поставляється з самою мовою Python. Виконаємо

метод `get_backend()`, який покаже, що використовується Tkinter у якості backend за замовченням:

```
import matplotlib as plt

print(plt.get_backend())
```

у консолі з'явиться **TkAgg**. Якщо за якимось причинами TkInter не встановлено, то будуть проблеми при відображенні графіків. При бажанні можна переключитися на інший backend, наприклад, на **Qt5Agg**. Для цього треба виконати інструкцію `plt.use('Name_backend')`.

Бібліотека Matplotlib підтримує широкий спектр типів графіків для візуалізації даних. Ось основні з них:

1. Лінійний графік (Line plot).

Використовується для відображення змін величин у часі або інших послідовностях (див. рис. 5.2).

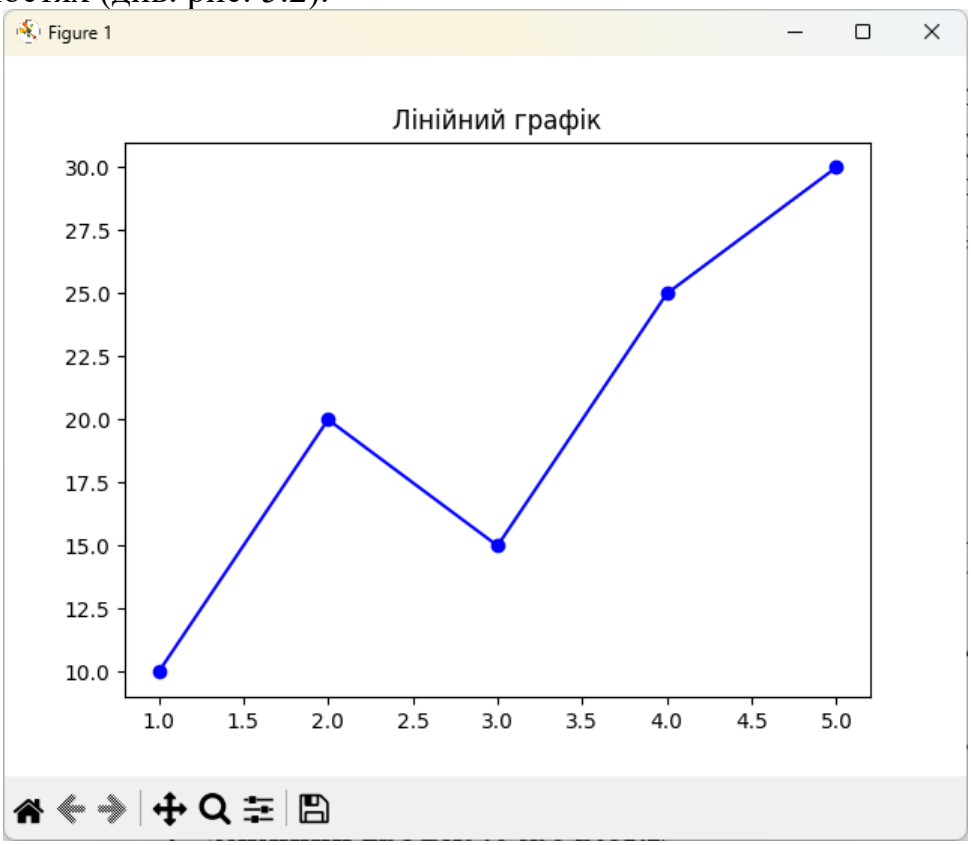

Рисунок 5.2 – Лінійний графік у Matplotlib

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [10, 20, 15, 25, 30]

plt.plot(x, y, marker='o', linestyle='-', color='b')
plt.title("Лінійний графік")
plt.show()
```

2. Точковий графік (Scatter plot).

Корисний для аналізу взаємозв'язку між двома змінними (див. рис. 5.3).

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

x = np.random.rand(50)
y = np.random.rand(50)

plt.scatter(x, y, color='r')
plt.title("Точковий графік")
plt.show()
```

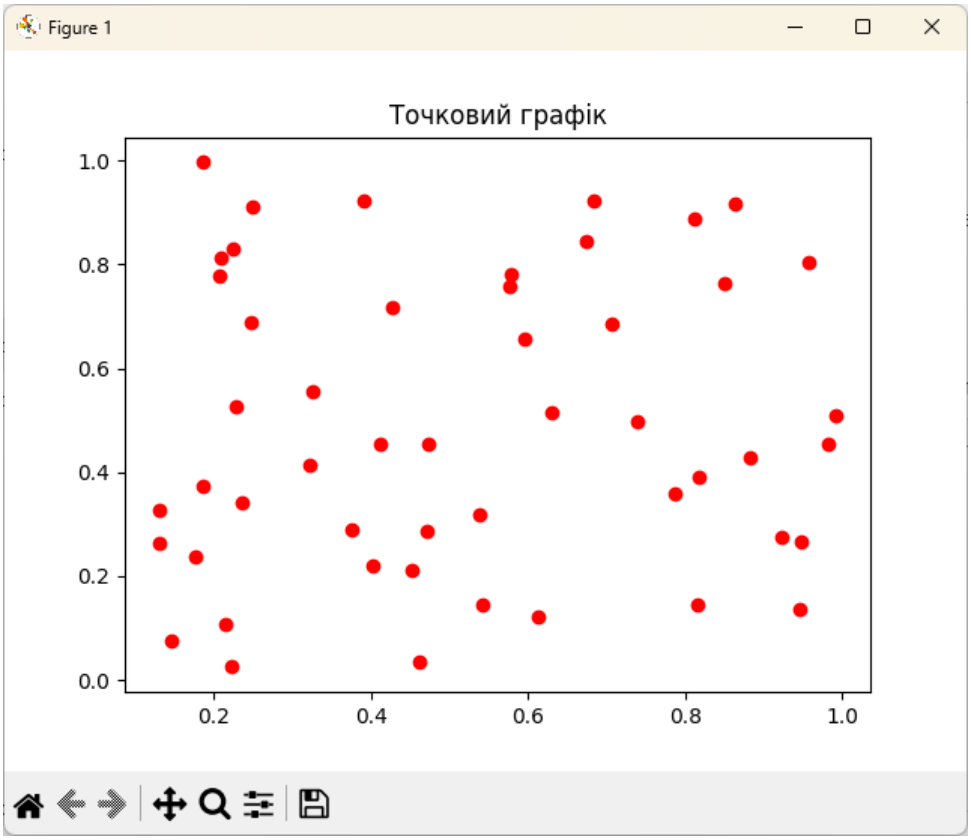

Рисунок 5.3 – Точковий графік у Matplotlib

3. Гістограма (Histogram).

Використовується для аналізу розподілу даних (див. рис. 5.4).

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

data = np.random.randn(1000)

plt.hist(data, bins=30, color='g', alpha=0.7)
plt.title("Гістограма")
plt.show()
```

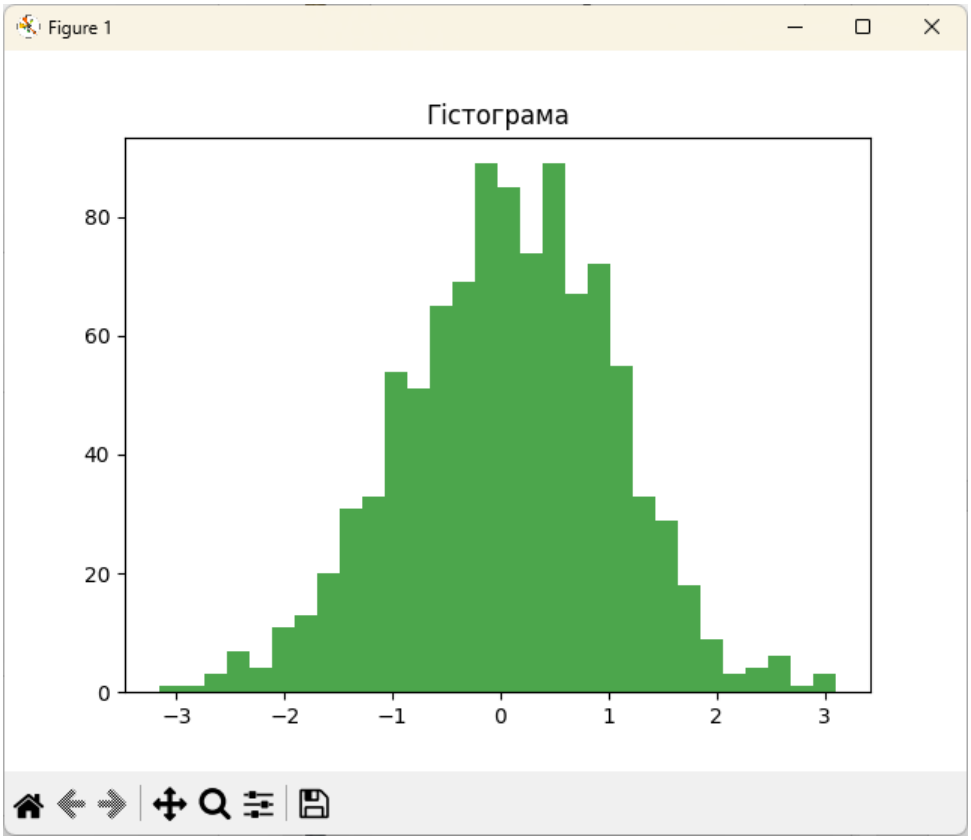

Рисунок 5.4 – Гістограма у Matplotlib

4. Стовпчаста діаграма (Bar chart).

Підходить для порівняння категорій (див. рис. 5.5).

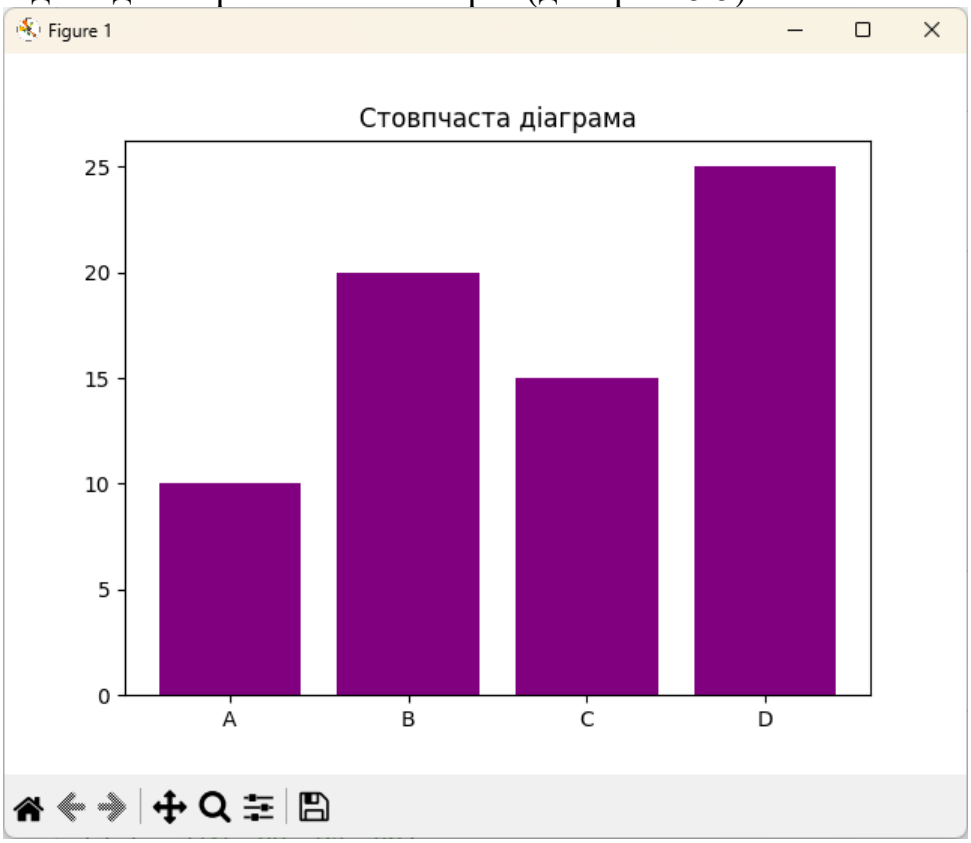

Рисунок 5.5 – Діаграма у Matplotlib

```
import matplotlib.pyplot as plt

labels = ['A', 'B', 'C', 'D']
values = [10, 20, 15, 25]

plt.bar(labels, values, color='purple')
plt.title("Стовпчаста діаграма")
plt.show()
```

5. Кругова діаграма (Pie chart).

Корисна для представлення часток у загальній сумі (див. рис. 5.6).

```
import matplotlib.pyplot as plt

sizes = [30, 20, 25, 25]
labels = ['A', 'B', 'C', 'D']

plt.pie(sizes, labels=labels, autopct='%1.1f%%',
colors=['blue', 'red', 'green', 'yellow'])
plt.title("Кругова діаграма")
plt.show()
```

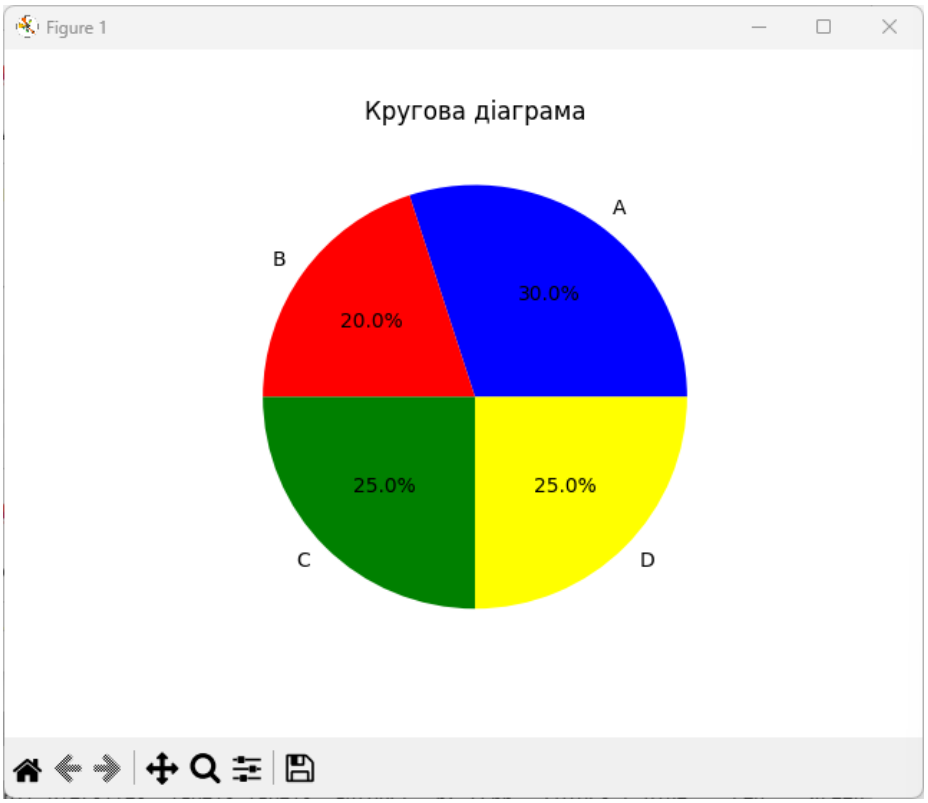

Рисунок 5.6 – Кругова діаграма у Matplotlib

6. Графік із заповненою площею (Area plot).

Використовується для аналізу накопичених даних. Наприклад, $y = \sin x$, $x \in [0; 2\pi]$ (див. рис. 5.7).

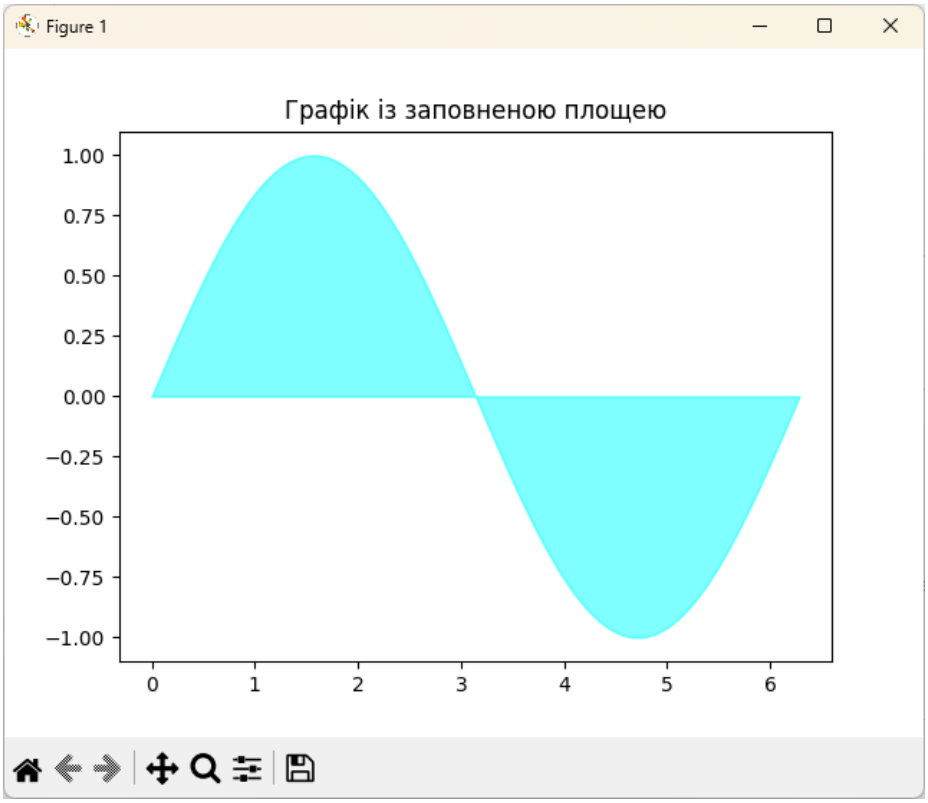

Рисунок 5.7 – Графік із заповненою площею у Matplotlib

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(0, 2*np.pi, 0.001)
y = np.sin(x)

plt.fill_between(x, y, color='cyan', alpha=0.5)
plt.title("Графік із заповненою площею")
plt.show()
```

7. Боксплот (Box plot).

Допомагає виявити розкид даних та аномальні значення (див. рис. 5.8).

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

data = [np.random.randn(100) for _ in range(4)]

plt.boxplot(data, labels=['A', 'B', 'C', 'D'])
plt.title("Боксплот")
plt.show()
```

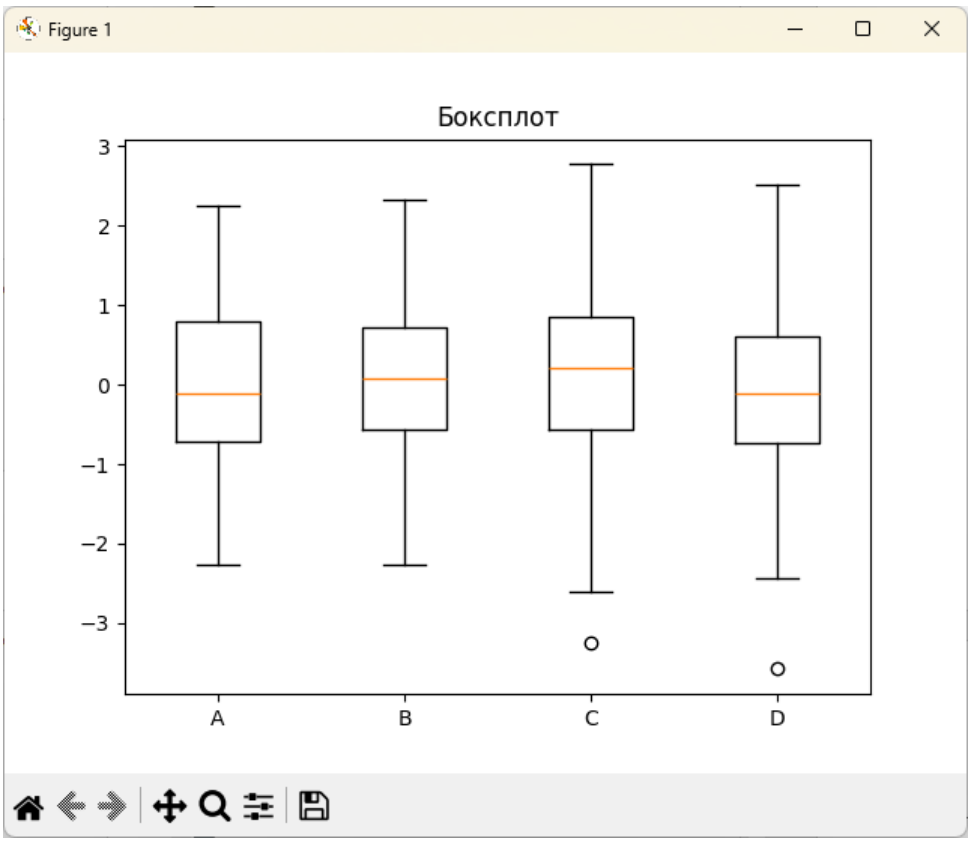

Рисунок 5.8 – Боксплот у Matplotlib

Matplotlib також підтримує 3D-графіки.

8. 3D-лінійний графік (див. рис. 5.9).

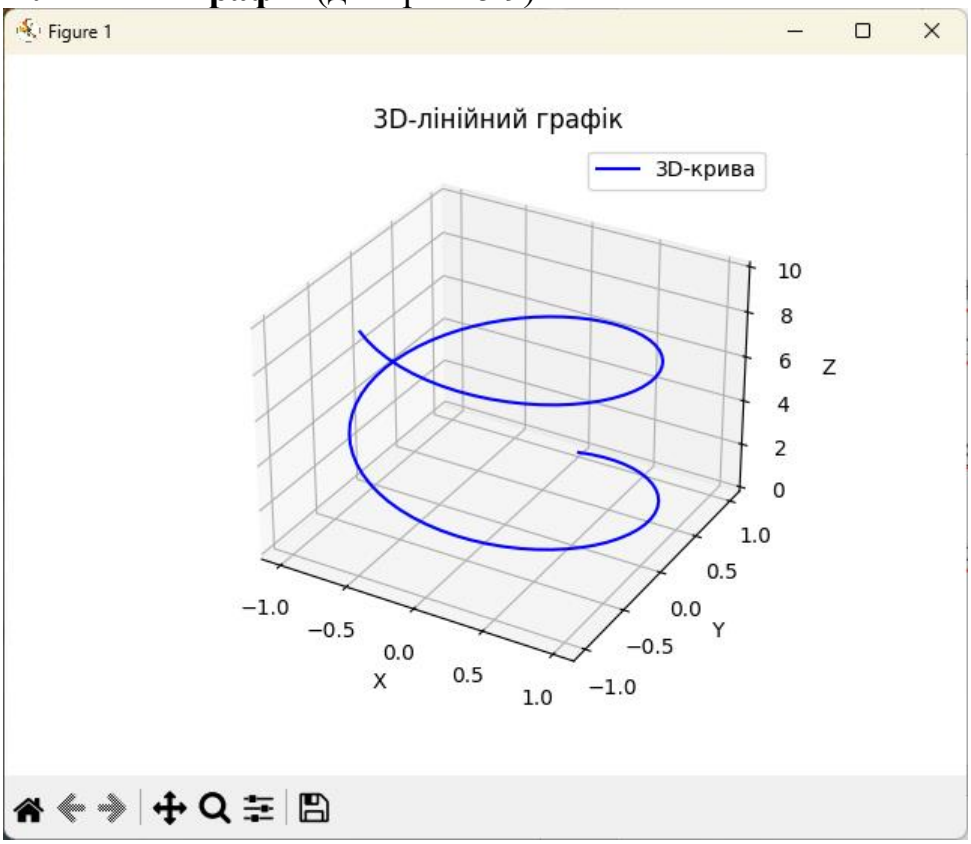

Рисунок 5.9 – 3D-лінійний графік у Matplotlib

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

# Створюємо 3D-фігуру
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Дані
t = np.linspace(0, 10, 100)
x = np.sin(t)
y = np.cos(t)
z = t

# Побудова 3D-лінії
ax.plot(x, y, z, color='b', label='3D-крива')

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('3D-лінійний графік')
plt.legend()
plt.show()
```

9. 3D-точковий графік (Scatter plot) (див. рис. 5.10).

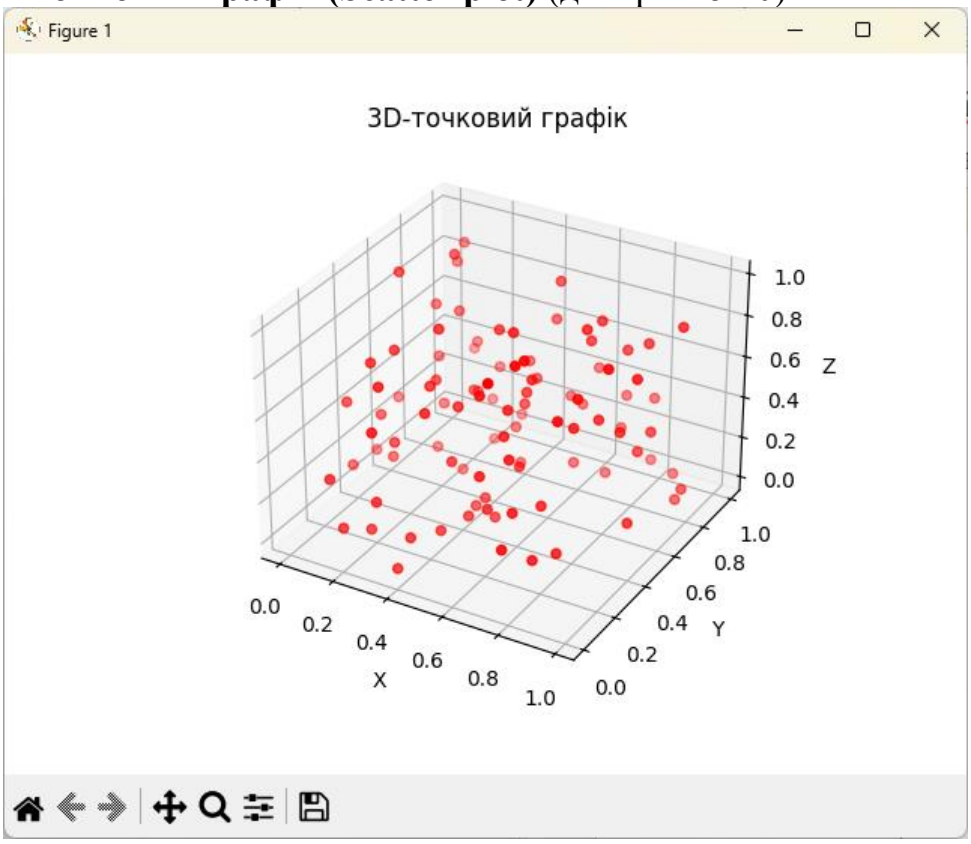

Рисунок 5.10 – 3D-точковий графік у Matplotlib

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

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Генерація випадкових точок
x = np.random.rand(100)
y = np.random.rand(100)
z = np.random.rand(100)

# Побудова точкового графіка
ax.scatter(x, y, z, c='r', marker='o')

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('3D-точковий графік')

plt.show()
```

10. Поверхні (Surface plot).

Наприклад, побудуємо функцію $z = \sin\sqrt{x^2 + y^2}$ у площині *XYZ* (див. рис. 5.11).

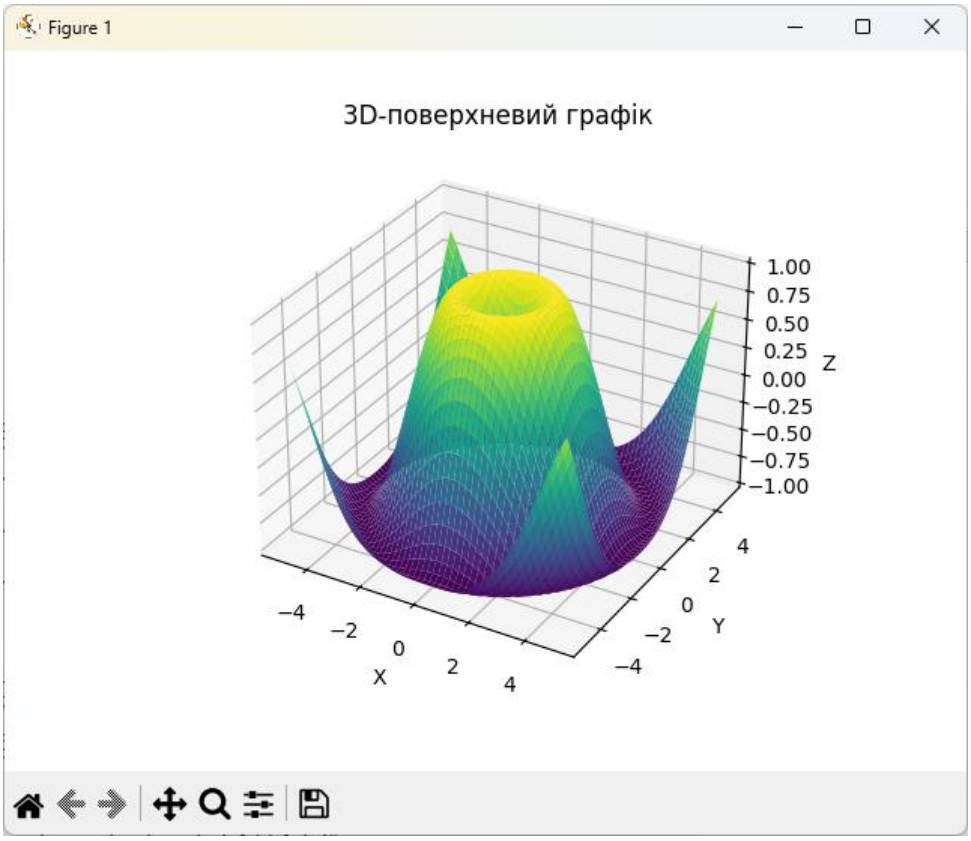

Рисунок 5.11 – Поверхня у Matplotlib

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

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Створення сітки значень
x = np.linspace(-5, 5, 50)
y = np.linspace(-5, 5, 50)
X, Y = np.meshgrid(x, y)
Z = np.sin(np.sqrt(X**2 + Y**2))

# Побудова поверхні
ax.plot_surface(X, Y, Z, cmap='viridis')

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('3D-поверхневий графік')
plt.show()
```

У Matplotlib можна відобразити зображення у форматі png, jpg, bmp, tiff тощо за допомогою модуля `imshow()`. Наприклад, виведемо на екран зображення у форматі png, яке лежить у корені проєкту (див. рис. 5.12).

Якщо треба завантажити зображення з URL та відобразити його в Matplotlib, то можна використати бібліотеку requests або PIL.

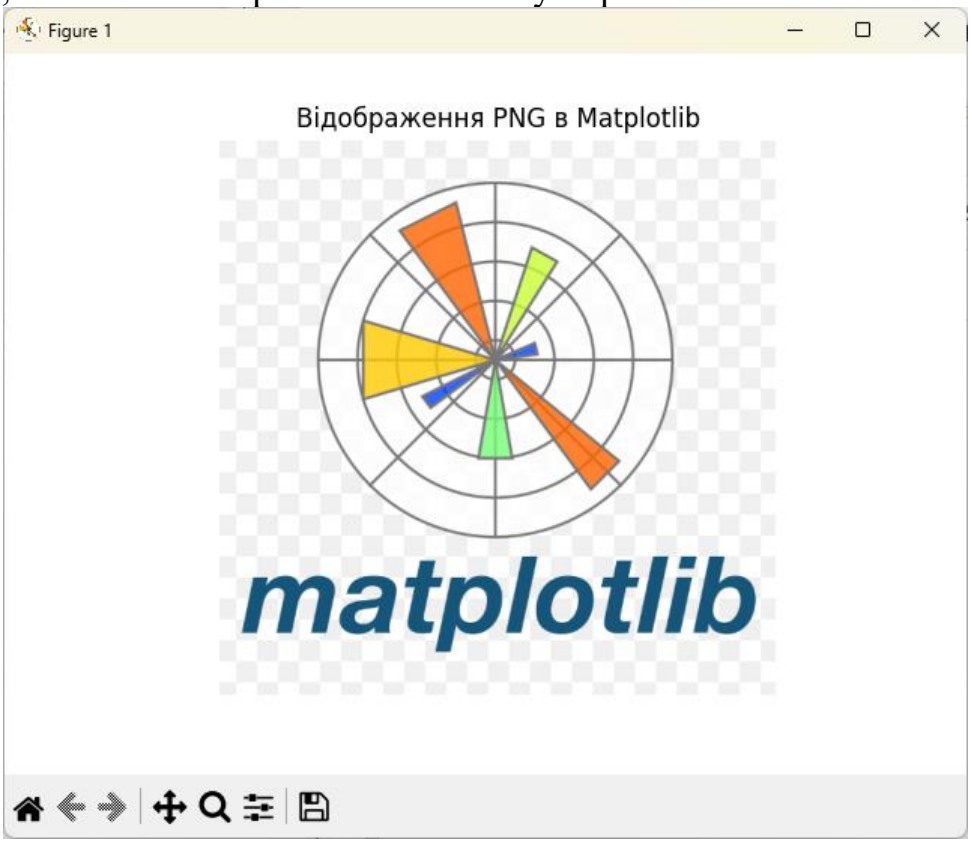

Рисунок 5.12 – Відображення зображення у форматі png у Matplotlib

Завдання до лабораторної роботи

1. Зобразити графіки функцій на одному рисунку різними кольорами, підписати кожну функцію та зберегти рисунок у файл з розширенням .png:

1) $y = \sin(x) + \frac{1}{2}\sin(2x) + \frac{1}{3}\sin(3x), y = \sqrt[7]{x^3 - 3x^2 + 5}$.
2) $y = 8x^2(x^2 - 1)^3, y = \mathrm{tg}\left(9x - \frac{\pi}{2}\right)$.
3) $y = \frac{x}{\ln x}, y = x^5 + 6x^4 - \frac{1}{x}$.
4) $y = \frac{x^3 - 2x^2 - x + 2}{x}, y = \mathrm{ctg}\left(3x + \frac{\pi}{2}\right)$.
5) $y = (x^2 - 2)e^{-2x}, y = \frac{x^5 - 8x^4 + 3x - 5}{\sqrt{x}}$.
6) $y = \frac{x^3}{x^2 - 1}, y = \sin\left(5x - \frac{\pi}{2}\right)$.
7) $y = \sin x - \ln(\sin x), y = e^{2x+3}$.
8) $y = \cos x + \frac{1}{2}\cos(2x) + \frac{1}{3}\cos(3x), y = \sqrt[3]{7x^6 - 5}$.
9) $y = \sqrt[3]{x(7 - x)^2} - \frac{x}{2}, y = \cos\left(2x + \frac{\pi}{2}\right)$.
10) $y = \left(\frac{x+2}{x-2}\right)^2$.

2. Зобразити графіки функції у полярній системі координат та зберегти рисунок у файл з розширенням .png:

1) $\rho = 5\sin\varphi$.
2) $\rho = 3\cos\varphi$.
3) $\rho = 4(1 + \cos\varphi)$.
4) $\rho = 5\sin^2 4\varphi$.
5) $\rho = 2\sin^2 3\varphi$.
6) $\rho = 3\sin 4\varphi$.
7) $\rho = 7\sin 5\varphi$.
8) $\rho = 2(2 + \cos\varphi)$.
9) $\rho = 6\cos 2\varphi$.
10) $\rho = 7\cos 4\varphi$.

3. Зобразити графік поверхні функції та зберегти рисунок у файл з розширенням .png:

1) $z = 4x^2 + 9y^2$.
2) $z = 6 - (x^2 + y^2)$.
3) $z = 9 - x^2$.
4) $z = \frac{y^2}{15}$.
5) $z = 3x^2 - 2y^2$.
6) $z = 2x^2 - 3y^2$.
7) $z = 5x^2 - 4$.

8) $z = 4x^2 + \frac{y^2}{10} + 1.$

9) $z = 9x^2 + \frac{y^2}{9} + 1.$

10) $z = 16x^2 - y^2.$

4. Зобразити кругову діаграму для компанії, в якій працює $(500 \cdot n)$ співробітників, де n – номер варіанта, з яких: 4% говорить португальською мовою, 5% – українською, 8% – арабською, 14% – японською, 16% – іспанською, 22% – французькою, 31% – англійською.

5. Зобразити гістограму зарплати вчителя по місяцям з попередньої лабораторної роботи за останні два роки.

Питання для самоконтролю

1. Яке призначення бібліотеки Matplotlib?
2. Як додати до графіка сітку?
3. Чи можна одночасно виводити декілька графіків на одному зображенні?
4. Як змінюється стиль відображення лінії?
5. Чи можна змінювати маркери у точках лінії?
6. Як зробити заливку області між графіком і прямою $y = 0$?
7. Чи можливе відображення декількох графіків на різних координатних осях у Matplotlib?
8. Чи можна на зображення у форматі png додати графік функції?
9. Що у Matplotlib у якості backend за замовченням використовується? Наведіть приклади інших підтримуваних Matplotlib backend.
10. Як діаграми можна будувати у Matplotlib?
11. Охарактеризуйте задання логарифмічних і полярних координат.
12. Чи можна реалізувати вбудовування двовимірного рисунка у тривимірний?
13. Чи можна створювати тривимірні стовпчикові діаграми?
14. Охарактеризуйте можливості побудови поверхонь у Matplotlib.
15. Чи можна робити візуалізацію двовимірних векторних полів?

6. Лабораторна робота. Застосування бібліотеки Pandas до задач аналізу даних

Мета: засвоїти можливості роботи бібліотеки Pandas до аналізу та обробки даних.

Теоретичні відомості та методичні рекомендації

Бібліотека Pandas – це одна з найпопулярніших бібліотек для

Pandas – це високорівнева бібліотека Python для аналізу та обробки даних. Вона забезпечує зручні структури даних і функції для роботи з табличними даними, подібними до електронних таблиць (Excel) або реляційних баз даних.

Основні можливості Pandas:

1. Основні структури даних:
   - Series – одномірний масив, схожий на стовпчик у таблиці або список;
   - DataFrame – двовимірна таблиця з рядками та стовпцями (основний об'єкт Pandas, який є табличною структурою даних).
2. Імпорт та експорт даних:
   - зчитування даних із csv, Excel, sql, json;
   - запис у файли різних форматів.
3. Обробка та маніпуляції з даними:
   - фільтрація, сортування, групування, агрегування;
   - обробка відсутніх значень.
4. Аналіз даних та статистика:
   - обчислення середнього, медіани, стандартного відхилення тощо;
   - робота з часовими рядами.
5. Візуалізація:
   - інтеграція з бібліотекою Matplotlib для побудови графіків.

Наприклад, створимо DataFrame, у якому міститься така інформація: ім'я, вік і оцінка студента. Треба вивести студентів, вік яких більше 22 та обчислити середню оцінку студентів.

```
import pandas as pd

# Створення DataFrame
data = {
    "Ім'я": ["Анна", "Богдан", "Віктор"],
    "Вік": [23, 25, 22],
    "Оцінка": [90, 85, 88]
}

df = pd.DataFrame(data)

# Виведення таблиці
print(df)
```

```
# Вибір рядків за умовою
print(df[df["Вік"] > 22])

# Обчислення середньої оцінки
print("Середня оцінка:", df["Оцінка"].mean())
```

У результаті отримаємо:

```
Ім'я  Вік  Оцінка
0    Анна   23      90
1  Богдан   25      85
2  Віктор   22      88
     Ім'я  Вік  Оцінка
0    Анна   23      90
1  Богдан   25      85
Середня оцінка: 87.66666666666667
```

Бібліотека Pandas розроблялася для роботи з табличними даними. Популярними типами файлів для їх зберігання є:

- csv – текстовий формат, у якому значення в стовпцях відокремлені один від одного роздільником, часто комою;
- xlsx або xls – формати файлів електронних таблиць Microsoft Excel;
- json-файли – це текстовий формат, призначений для зберігання структурованих даних.

Для читання та запису таблиць зазначених форматів у Pandas існують спеціальні методи:

- для читання файлів csv використовується метод `pd.read_csv()`;
- для читання файлів xlsx використовується метод `pd.read_excel()`;
- для читання файлів json використовується метод `pd.read_json()`;
- для читання бази даних SQL використовується метод `pd.read_sql()`;
- для читання HTML-таблиць використовується метод `pd.read_html()`;
- для читання великих файлів використовується метод `pd.read_parquet()`;
- для читання ZIP-архівів використовується метод з додатковим параметром `pd.read_csv("data.zip", compression="zip")`.

Метод `pd.read_csv("data.csv")` – найпоширеніший метод для завантаження даних у Pandas. Корисні параметри:

- `sep=";"` – змінює роздільник (якщо не кома);
- `header=None` – якщо немає заголовків;
- `usecols=["Ім'я", "Вік"]` – читає лише вибрані стовпці;
- `encoding="utf-8"` – задає кодування (важливо для українських текстів);

– `dtype={"Оцінка": float}` – задає тип даних для стовпця;
– `parse_dates=["Дата"]` – автоматично конвертує стовпець у формат дати.

У Pandas є багато методів для перегляду та аналізу DataFrame.

1. Основні методи для перегляду DataFrame.

Виведення перших або останніх рядків:

– `df.head(n)` – показує перші n рядків (за замовчуванням 5);
– `df.tail(n)` – показує останні n рядків.

Основна інформація про DataFrame:

– `df.info()` – загальна інформація про стовпці, типи даних та пропущені значення;
– `df.shape` – розмірність DataFrame (кількість рядків і стовпців);
– `df.columns` – список назв стовпців;
– `df.index` – інформація про індекси.

Перегляд статистичних характеристик:

– `df.describe()` – основна статистика (середнє, медіана, мін., макс.);
– `df.describe(include="all")` – статистика для всіх типів даних (не лише числових);
– `df["Оцінка"].mean()` – середнє значення конкретного стовпця.

Вибір випадкових рядків: `df.sample(n)` – випадковий вибір n рядків.

2. Перегляд вмісту DataFrame.

Вибір конкретного стовпця:

– `df["Ім'я"]` – вибір одного стовпця;
– `df[["Ім'я", "Оцінка"]]` – вибір кількох стовпців.

Вибір рядків за номером (iloc):

– `df.iloc[0]` – перший рядок;
– `df.iloc[1:4]` – рядки з 1 по 3 (не включаючи 4).

Вибір рядків за умовою:

– `df[df["Вік"] > 22]` – вибір рядків, де "Вік" більше 22;
– `df[df["Ім'я"] == "Анна"]` – вибір рядків, де ім'я "Анна".

Перевірка наявності пропущених значень: `df.isnull().sum()` – кількість пропущених значень у кожному стовпці.

3. Методи для роботи зі структурою DataFrame.

Перейменування стовпців, наприклад:

```
df.rename(columns={"Оцінка": "Середній бал"}, inplace=True)
```

Сортування, наприклад:

– `df.sort_values("Вік")` – сортує за віком (за зростанням).
– `df.sort_values("Вік", ascending=False)` – сортує за спаданням.

Видалення стовпців або рядків, наприклад:

– `df.drop(columns=["Оцінка"])` – видаляє стовпець.

– `df.drop(index=[0, 2])` – видаляє рядки за індексом.

Таким чином, бібліотека Pandas спрощує обробку даних, має інтуїтивно зрозумілий синтаксис, добре інтегрується з іншими бібліотеками Python (NumPy, Matplotlib, Seaborn), підходить для великих обсягів даних.

Завдання до лабораторної роботи

1. Завантажити з https://www.kaggle.com/datasets Public Datasets (csv-файл з даними). Провести аналіз даних:

1) Перетворити csv-дані в об'єкт DataFrame.

2) Вивести на екран початок та кінець файлу.

3) Вивести на екран останні 15 рядків файлу.

4) Вивести на екран декілька довільних стовпчиків.

5) Додати новий стовпчик Total і присвоїти йому суму або кількість деяких значень з інших стовпчиків.

6) Перейменувати колонки, які містять два і більше слів.

7) Зробити перезапис DataFrame.

2. За допомогою методу `read_html()` виконайте парсінг таблички з вебсторінки, наприклад, з офіційної сторінки українського банку з курсом валют.

1) Визначить кількість отриманих таблиць.

2) Отримайте DataFrame з курсами валют.

3. Використовуючи базові фільтри, відфільтруйте дані, за числовими даними. Збережіть отриманий результат у інший csv-файл з даними. Зробіть висновки.

4. Виконайте розділення csv-файлу на частини (по 15 рядків) та об'єднайте отримані частини в один DataFrame.

Питання для самоконтролю

1. Що представляє собою у бібліотеці Pandas об'єкт Series?
2. Що представляє собою у бібліотеці Pandas об'єкт DataFrame?
3. Які існують методи DataFrame читання тестових форматів?
4. Розкажіть про основні методи для перегляду DataFrame.
5. Розкажіть про способи перегляду вмісту DataFrame.
6. Охарактеризуйте методи для роботи зі структурою DataFrame.

7. Лабораторна робота. Застосування бібліотеки Pandas до групування та агрегування даних

Мета: засвоїти можливості роботи бібліотеки Pandas до групування даних, агрегуванню даних.

Теоретичні відомості та методичні рекомендації

DataFrameGroupBy – це об'єкт у Pandas, який створюється після виклику методу `groupby()`. Він дозволяє групувати дані за одним або кількома стовпцями і виконувати агрегатні операції, такі як підрахунок, сума, середнє значення тощо.

Синтаксис:

```
df.groupby("стовпець")
```

Приклад використання groupby():

```
import pandas as pd

# Створення DataFrame
data = {
    "Група": ["A", "B", "A", "B", "A", "C"],
    "Студент": ["Анна", "Богдан", "Віктор", "Ганна",
"Денис", "Єва"],
    "Оцінка": [90, 85, 88, 92, 75, 80]
}

df = pd.DataFrame(data)

# Групування за "Група" та знаходження середньої оцінки
grouped = df.groupby("Група")["Оцінка"].mean()
print(grouped)
```

Результат:

```
Група
A    84.333333
B    88.500000
C    80.000000
Name: Оцінка, dtype: float64
```

Агрегуючі методи groupby():

- `mean()` – середнє значення;
- `sum()` – сума значень;
- `count()` – кількість елементів у групі;
- `min()` – мінімальне значення;
- `max()` – максимальне значення;
- `median()` – медіана;
- `std()` – стандартне відхилення;

– `describe()` – повна статистика групи.

Наприклад, сумування всіх числових даних в DataFrame (див. рис. 7.1).

```
In [14]: df.groupby('Day_name').sum()
```

Out[14]:

Day_name	ProductID	Quantity	Price	Total
Friday	906881	6	33973.80	33973.80
Monday	765576	6	23435.39	38971.89
Saturday	143185	1	4334.33	4334.33
Sunday	767407	5	37180.35	43145.85
Thursday	3925651	10	41439.05	59075.05
Tuesday	641762	11	25618.08	64991.43
Wednesday	717171	6	32873.79	37488.07

Рисунок 7.1 – Агрегація по дням тижня

Очевидно, що `groupby()` – це потужний інструмент для аналізу даних.

Завдання до лабораторної роботи

1. Завантажити з https://www.kaggle.com/datasets Public Datasets (csv-файл з даними). Провести аналіз даних.
2. Виберіть колонки з числовими даними. Дайте характеристику вибраним даним.
3. Виконайте групування та агрегування даних у файлі. Надайте пояснення.

Питання для самоконтролю

1. Охарактеризуйте об’єкт DataFrameGroupBy з бібліотеки Pandas.
2. Наведіть агрегуючі методи.
3. Чи можливе групування за кількома стовпцями?
4. Який метод дозволяє залишити тільки групи, які відповідають певній умові?
5. Чи можливо застосовувати кілька агрегуючих функцій?

8. Лабораторна робота. Застосування бібліотеки Pandas до візуалізації даних

Мета: засвоїти можливості роботи бібліотеки Pandas до групуванню даних, агрегуванню даних, візуалізації даних.

Теоретичні відомості та методичні рекомендації

У бібліотеці Pandas є вбудовані засоби для візуалізації даних, які базуються на бібліотеці Matplotlib. Вони дозволяють швидко будувати графіки без додаткових налаштувань.

Основні методи візуалізації у Pandas:

1. Лінійний графік line plot. За замовчуванням `df.plot()` будує лінійний графік (див. рис. 8.1). Деякі властивості:

- `kind="line"` – вказує, що будуємо лінійний графік;
- `marker="o"` – додає точки на лінії.

```
import pandas as pd
import matplotlib.pyplot as plt

# Створюємо DataFrame
data = {"Місяць": ["Січ", "Лют", "Бер", "Кві", "Тра"],
        "Продажі": [100, 150, 130, 170, 200]}

df = pd.DataFrame(data)

# Будуємо графік
df.plot(x="Місяць", y="Продажі", kind="line", marker="o",
figsize=(8, 5))

plt.title("Динаміка продажів")
plt.xlabel("Місяць")
plt.ylabel("Продажі")
plt.grid(True)
plt.show()
```

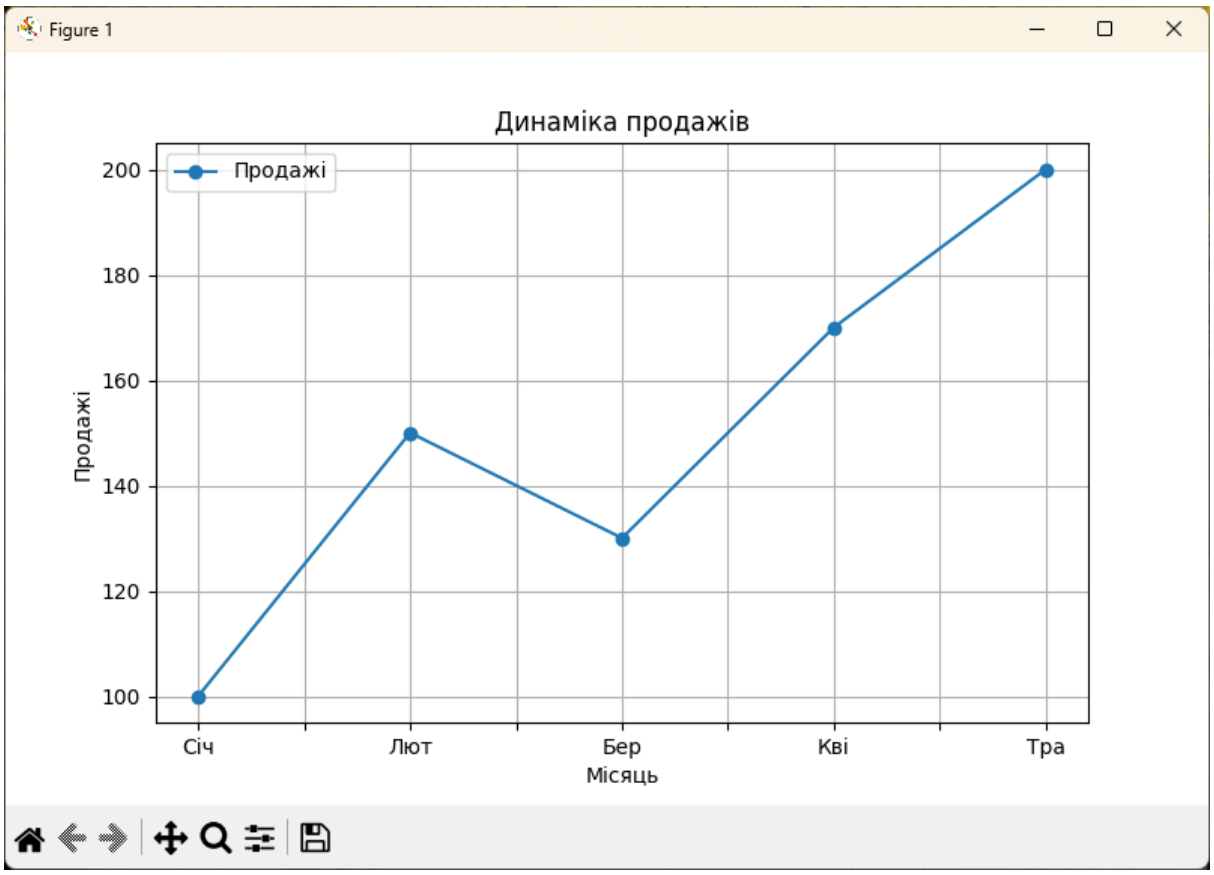

Рисунок 8.1 – Лінійний графік

2. Стовпчикова діаграма bar plot (див. рис. 8.2).

```
import pandas as pd
import matplotlib.pyplot as plt

data = {"Місяць": ["Січ", "Лют", "Бер", "Кві", "Тра"],
        "Продажі": [100, 150, 130, 170, 200]}

df = pd.DataFrame(data)

df.plot(x="Місяць",        y="Продажі",        kind="bar",
color="skyblue", figsize=(8, 5))

plt.title("Продажі за місяцями")
plt.xlabel("Місяць")
plt.ylabel("Продажі")
plt.show()
```

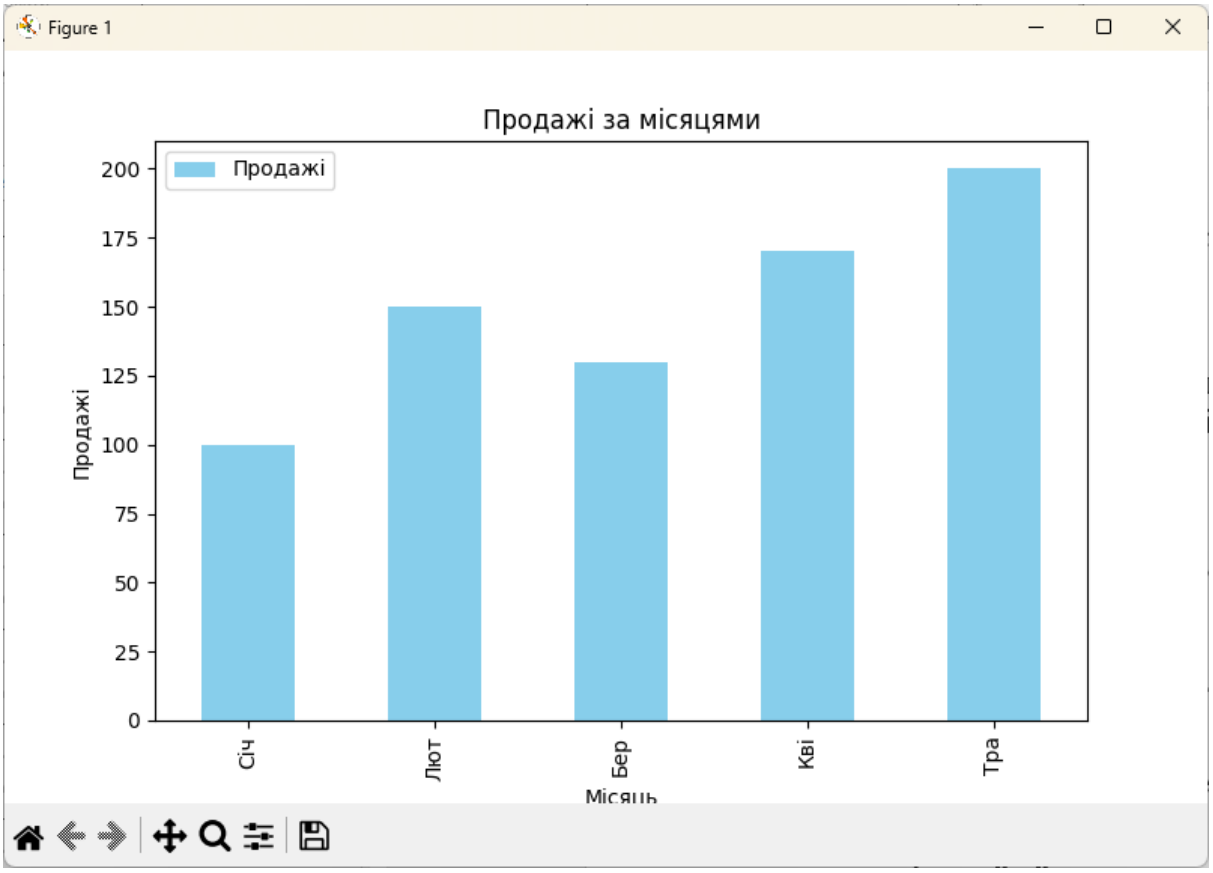

Рисунок 8.2 – Стовпчикова діаграма

3. Кругова діаграма pie chart (див. рис. 8.3).

```
import pandas as pd
import matplotlib.pyplot as plt

data = {"Місяць": ["Січ", "Лют", "Бер", "Кві", "Тра"],
        "Продажі": [100, 150, 130, 170, 200]}

df = pd.DataFrame(data)

df.plot(kind="pie",   y="Продажі",   labels=df["Місяць"],
autopct="%1.1f%%", legend=False)

plt.title("Розподіл продажів")
plt.ylabel("")  # Прибираємо підпис осі Y
plt.show()
```

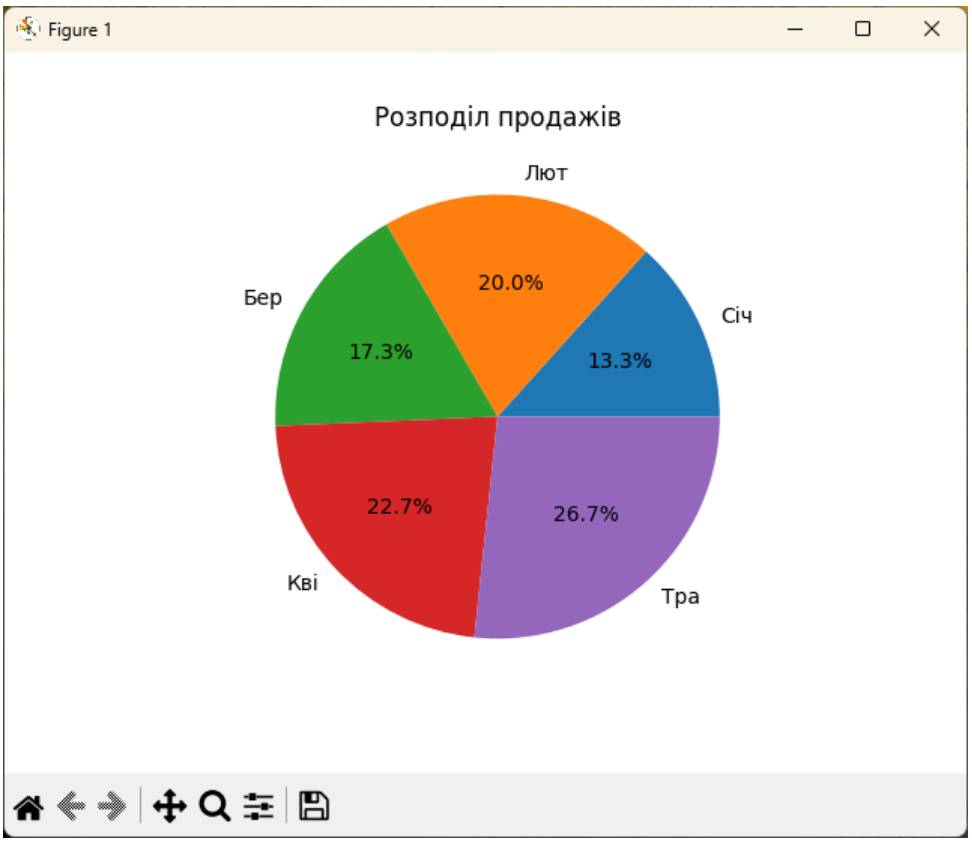

Рисунок 8.3 – Кругова діаграма

4. Графік розсіювання scatter plot (див. рис. 8.4).

```
import pandas as pd
import matplotlib.pyplot as plt

data = {"Місяць": ["Січ", "Лют", "Бер", "Кві", "Тра"],
        "Продажі": [100, 150, 130, 170, 200]}

df = pd.DataFrame(data)

df.plot(kind="scatter",      x="Місяць",      y="Продажі",
color="red")

plt.title("Зв'язок між місяцем і продажами")
plt.xlabel("Місяць")
plt.ylabel("Продажі")
plt.show()
```

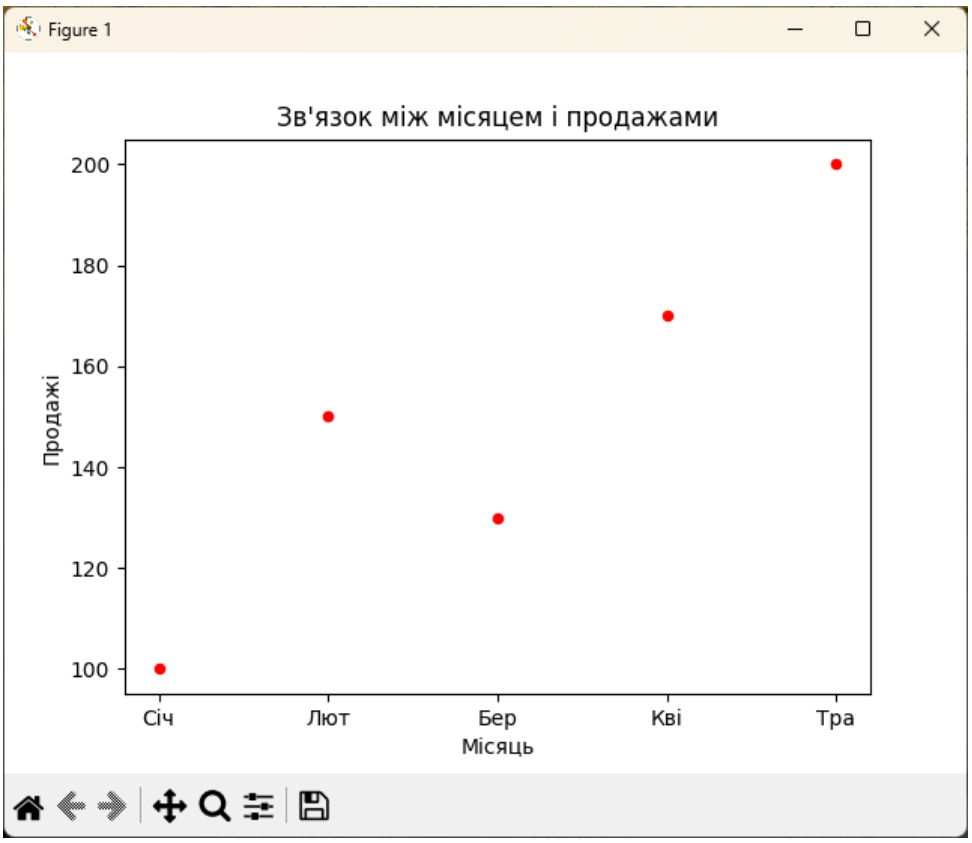

Рисунок 8.4 – Графік розсіювання

Крім того, у бібліотеці Pandas можна візуалізувати дані штатними можливостями, не використовуючи бібліотеку Matplotlib. Для побудови гістограми існує метод `plot.bar()`, а для побудови кругової діаграми існує метод `plot. pie()` (див. рис. 8.5). Отже, Pandas має вбудовані методи візуалізації, які достатні для базового аналізу.

```
In [22]: plot = res.plot.pie(y='sum', figsize=(7, 7))
```

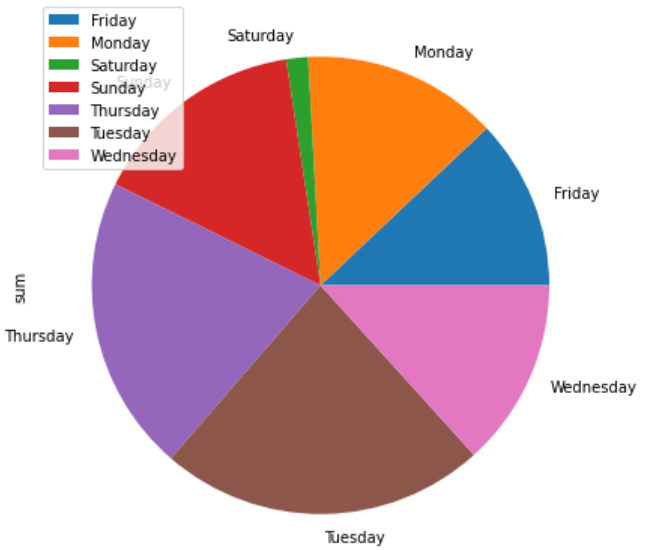

Рисунок 8.5 – Візуалізація

Завдання до лабораторної роботи

1. Завантажити з https://www.kaggle.com/datasets Public Datasets (csv-файл з даними). Провести аналіз даних.

2. Виберіть колонки з числовими даними. Дайте характеристику вибраним даним.

3. Побудуйте стовпчикову діаграми, використовуючи метод `bar()`. Зробіть висновки.

4. Побудуйте кругову діаграму, використовуючи метод `plot.pie()`. Зробіть висновки.

Питання для самоконтролю

1. Який метод використовується для створення графіків у Pandas?
2. Як побудувати лінійний графік у Pandas?
3. Який параметр kind потрібно вказати, щоб створити стовпчикову діаграму?
4. Як змінити розмір графіка в Pandas?
5. Як додати заголовок до графіка?
6. Як зробити точки на лінійному графіку більш помітними?
7. Чим відрізняється bar від barh у Pandas?
8. Як побудувати гістограму (графік розподілу)?
9. Як змінити колір графіка у Pandas?
10. Як зробити підписи у відсотках на круговій діаграмі?
11. Як побудувати графік розсіювання у Pandas?
12. Що таке bins у гістограмі?
13. Як змінити підписи осей на графіку?
14. Як можна покращити оформлення графіків у Pandas за допомогою Matplotlib?
15. Чи можна будувати кілька графіків одночасно? Якщо так, як?
16. Як вибрати певні стовпці для візуалізації?

9. Лабораторна робота. Алгебраїчні обчислення мовою R

Мета: засвоїти можливості роботи мови R до застосування розв'язування задач лінійної алгебри.

Теоретичні відомості та методичні рекомендації

R – є самою популярною у світі мовою для статистичних обчислень. R – це потужна інтерпретована мова сценаріїв для обробки та аналізу статистичних даних, графічної візуалізації даних. Вона широко використовується серед статистиків, дослідників даних та аналітиків завдяки своїй потужності, гнучкості та наявності великої кількості пакетів для вирішення різноманітних завдань.

Ядро R можна завантажити з офіційної сторінки CRAN (Comprehensive R Archive Network) за посиланням для Windows https://cran.r-project.org/bin/windows/base/. Крім того, ядро R доступно для інших операційних систем: OS та Linux. Одразу бажано встановити додаткові інструменти для створення пакетів R з вихідного коду в Windows https://cran.r-project.org/bin/windows/Rtools/ (графічного інтерфейсу немає). Для написання коду можна використовувати програму RStudio IDE (див. рис. 9.1), яка є вільно розповсюдженим IDE-редактором (див. рис. 9.2) коду для мови R https://posit.co/download/rstudio-desktop/.

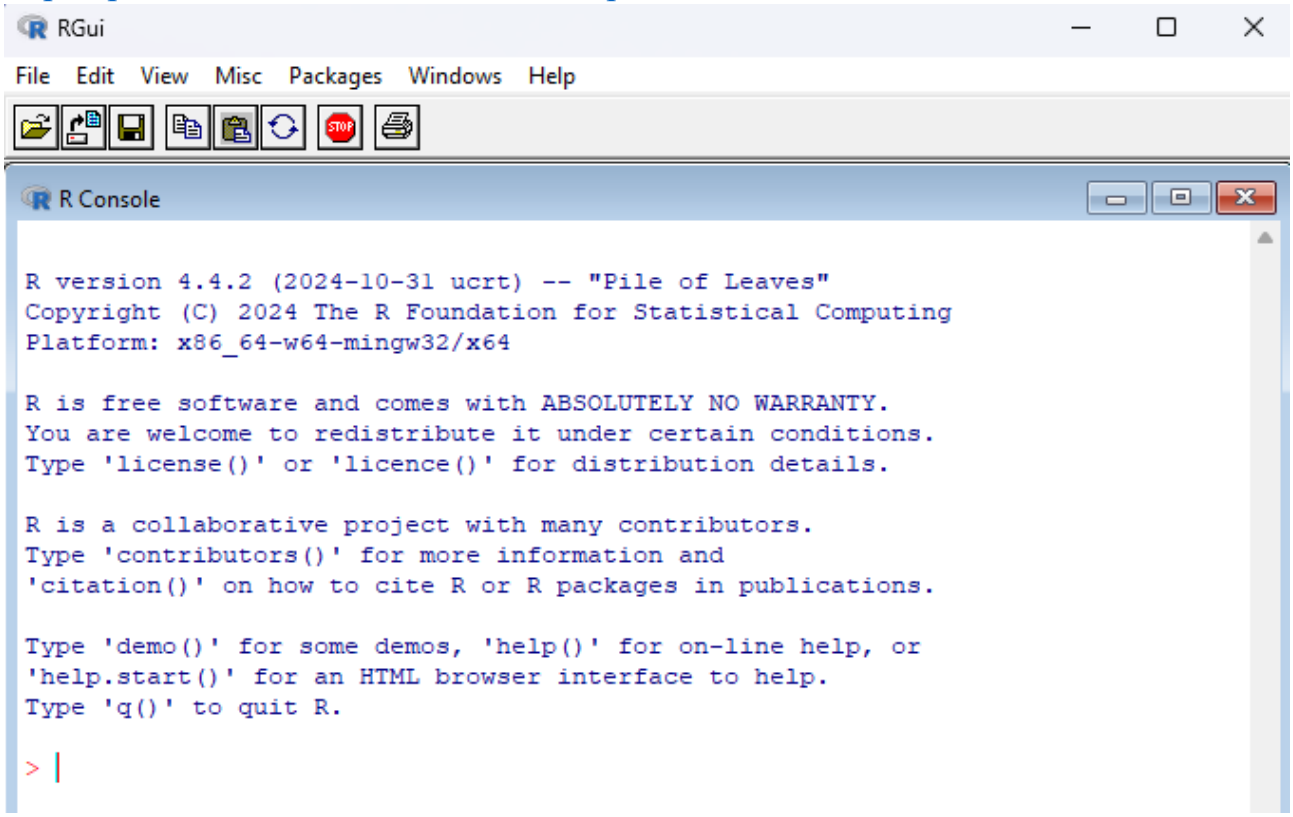

Рисунок 9.1 – Вікно R Console

R Console (рис.) можна використати, наприклад, для оновлення ядра R. Для цього потрібно встановити пакет `installr`, виконавши команди у консолі:

```
install.packages('installr')
library("installr")
updateR()
```

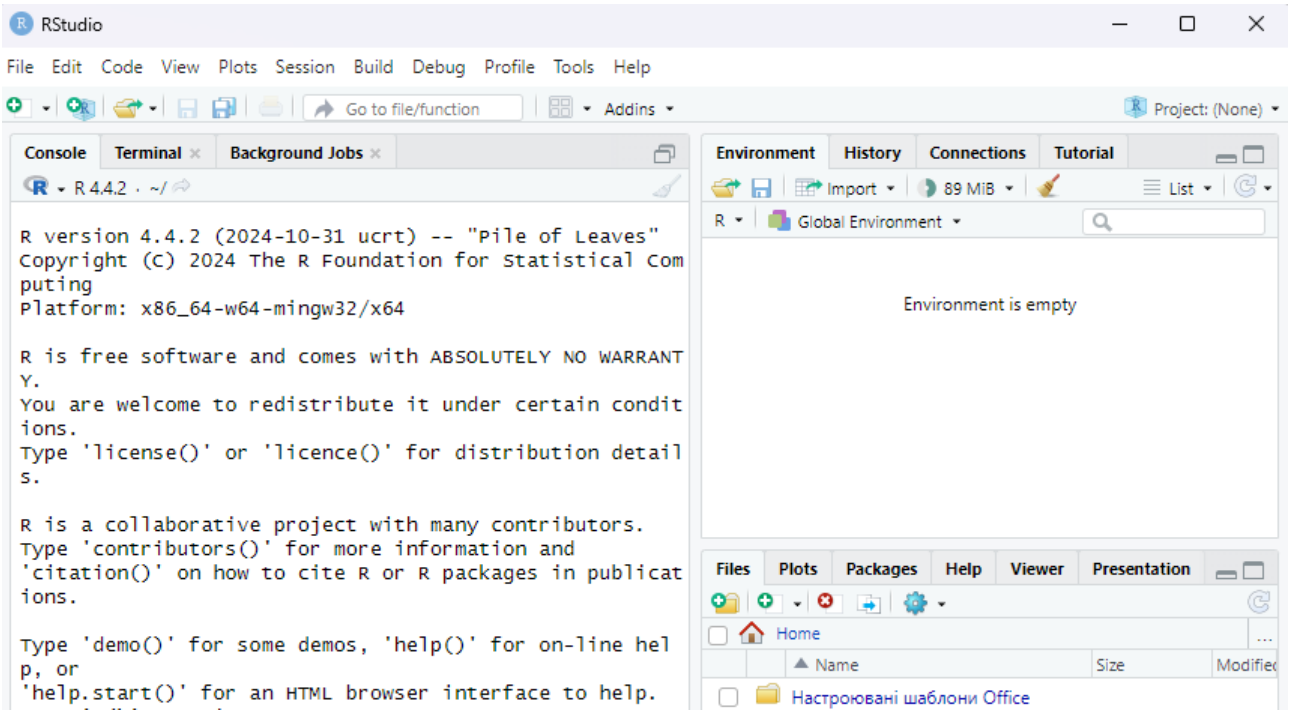

Рисунок 9.2 – Вікно RStudio IDE

Основні характеристики мови R:

- Спрямованість на статистику: R створений спеціально для аналізу даних і включає багато вбудованих функцій для виконання статистичних операцій, таких як регресія, кластеризація, часові ряди тощо.
- Візуалізація даних: R дозволяє створювати якісні графіки, такі як гістограми, діаграми розсіювання, коробкові діаграми, теплові карти тощо. Популярні пакети, як-от ggplot2, значно розширюють можливості візуалізації.
- Розширюваність: R має тисячі пакетів, доступних через репозиторій CRAN (Comprehensive R Archive Network). Ці пакети дозволяють працювати з машинним навчанням, біоінформатикою, аналізом тексту, геопросторовими даними та багатьма іншими задачами.
- Відкрите програмне забезпечення: R є безкоштовним і має відкритий вихідний код, що робить його доступним для всіх користувачів і дозволяє спільноті активно розвивати середовище.
- Кросплатформеність: R працює на Windows, MacOS та Linux.
- Скрипти, написані на мові R, зберігаються у форматі `namefile.R`. Команди вводяться користувачем у консолі (командному вікні) після символу запрошення, що має вигляд «>».

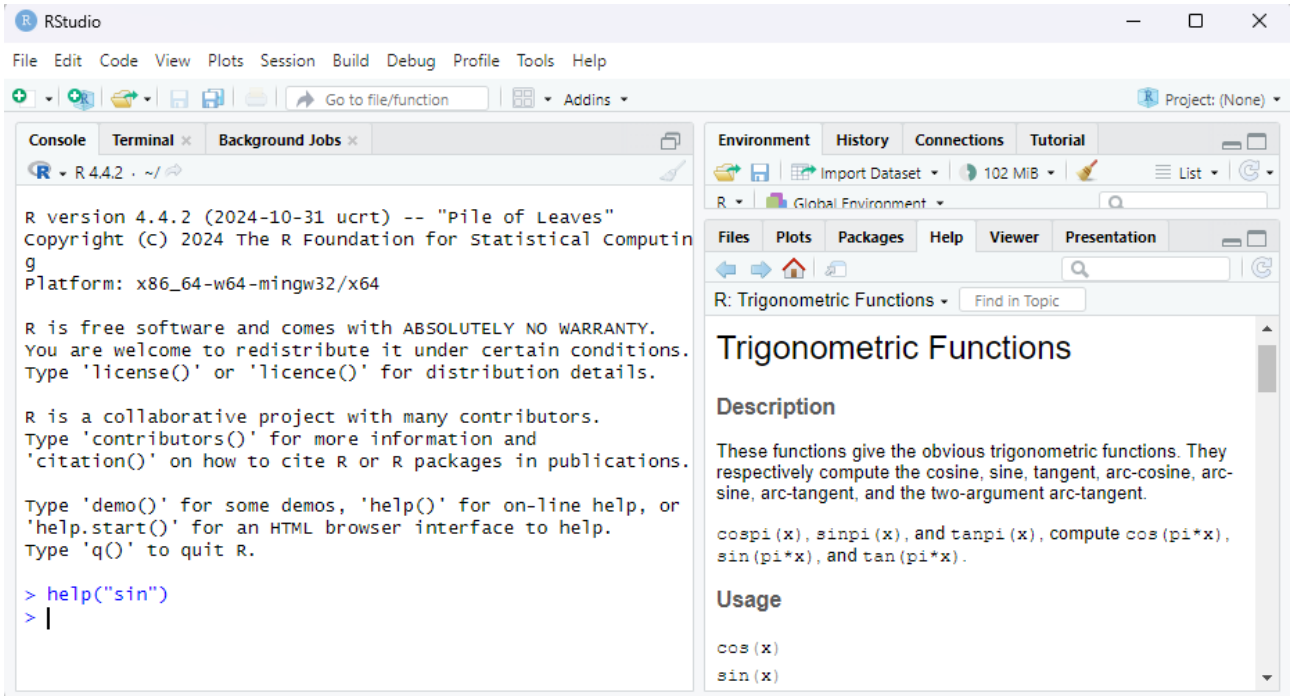

Рисунок 9.3 – Виклик довідки для тригонометричної функції

R чутливий до регістру клавіатури. Всі команди вводяться у командному рядочку. Наприклад, щоб отримати довідку (див. рис. 9.3) за тригонометричною функцією $\sin x$, у командному рядочку треба ввести наступне:

`help("sin")` або `help.search("sin")`

У правому нижньому куті на вкладці Help відобразиться інформація щодо виклику функції, аргументу функції та прикладом використання функції.

Основний функціонал програми реалізується за допомогою вбудованих та створених користувачем функцій. Всі об’єкти і набори даних зберігаються у пам’яті до завершення інтерактивної сесії. Основні функції доступні за замовченням. Інші функції містяться у пакетах, які можуть бути активовані у ході роботи з програмою за необхідністю. Командні рядочки складаються з функцій і присвоєнь. R використовує символ <− для присвоєння замість звичайного =. Коментарі пишуться після символу #.

Основні поняття:

- **об’єкти**: у R все є об’єктом, включаючи числа, вектори, матриці, списки, дані тощо. Наприклад, матрицю 2×3 можна побудувати за допомогою функції `matrix(1:6, nrow=2, ncol=3);`
- **вектори**: основний будівельний блок даних в R. Вектори можуть містити числа, символи або логічні значення. Вектори в системі R формуються функції конкатенації `c();`
- **функції**: R має багато вбудованих функцій і дозволяє користувачам створювати свої власні.

Популярні пакети:

- `ggplot2` — для візуалізації даних;

– `dplyr` — для маніпуляції даними;
– `tidyr` — для трансформації даних;
– `caret` — для машинного навчання;
– `shiny` — для створення інтерактивних вебдодатків.

У мові програмування R існують різні типи даних, які дозволяють зберігати і обробляти різноманітну інформацію. Ось основні типи даних у R:

Прості типи даних:

– `numeric` (числовий). Об'єкти даного класу діляться на цілочислові (`integer`) дійсні (`double`);
– `logical` (логічний). Використовується у логічних умовах. Має два можливих значення: `TRUE` або `FALSE`;
– `complex` (комплексний). Використовується для представлення комплексних чисел. Наприклад: `3 + 2i` (де `i` – уявна одиниця);
– `character` (символьний). Представляє текст або рядки. Рядки записуються у лапках (або в подвійних, або в одинарних). Наприклад: `"hello"`, `'data'`.

Складні типи даних:

– `vector` (вектор). Одновимірна структура, яка містить елементи одного типу. Приклад: `c(1, 2, 3)` (числовий вектор), `c("apple", "banana")` (символьний вектор);
– `matrix` (матриця). Двовимірна структура, що містить елементи одного типу. Створюється функцією `matrix()`. Наприклад, `matrix(1:6, nrow = 2, ncol = 3)`;
– `array` (масив). Багатовимірна структура, що містить елементи одного типу. Створюється функцією `array()`. Наприклад, `array(1:8, dim = c(2, 2, 2))`;
– `list` (список). Неструктуровані дані, які можуть містити елементи різних типів (числа, рядки, вектори, матриці тощо). Створюється функцією `list()`. Наприклад, `list(1, "text", TRUE, c(1, 2, 3))`;
– `data Frame` (фрейм даних). Таблиця, де стовпці можуть містити дані різних типів. Це основний тип для роботи з даними в R. Створюється функцією `data.frame(data1, data2, ...)`. При цьому всі вектори `data1, data2, ...` повинні мати однакову довжину. Наприклад,

```
data.frame(
  Name = c("Alice", "Bob"),
  Age = c(25, 30),
  Married = c(TRUE, FALSE)
)
```

– `factor` (фактор). Використовується для представлення категоріальних змінних. Фактор – це векторний об'єкт, що кодує категоріальні дані

(класи). Значення факторів зберігаються як числа, але мають відповідні рівні (labels). Наприклад, `factor(c("low", "medium", "high", "low"))`.

Інші типи:

- `NULL`. Позначає відсутність або неіснування значення. Наприклад, `NULL`.
- `NA` (Not Available). Використовується для позначення пропущених значень у даних. Наприклад, `NA`;
- `Inf` і `-Inf`. Позначають нескінченність. Наприклад, $\frac{1}{0}$ дасть `Inf`;
- `NaN` (Not a Number). Позначає результат, який не є числом (наприклад, $\frac{0}{0}$). Наприклад, `NaN`.

Ці типи (див. рис. 9.4) є основою роботи з даними в R і допомагають ефективно аналізувати та обробляти інформацію.

```
> num <- 3.14
> class(num)
[1] "numeric"
> int <- 42L
> class(int)
[1] "integer"
> log <- TRUE
> class(log)
[1] "logical"
> char <- "Hello, R!"
> class(char)
[1] "character"
> fac <- factor(c("low", "medium", "high"))
> class(fac)  # "factor"
[1] "factor"
> levels(fac)
[1] "high"   "low"    "medium"
>
```

Рисунок 9.4 – Приклад визначення типу даних

Арифметичні операції над матрицями здійснюються по компонентно, тому, наприклад, щоб додати дві матриці, вони повинні мати однакові розміри. Функція `A = cbind(A, B)` створює матрицю, приписуючи до `A` справа `B` (для цього кількість рядків у `A` і `B` має співпадати). Функція `A = rbind(A, B)` створює матрицю, приписуючи до `A` знизу `B` (для цього кількість стовпців у `A` і `B` має співпадати).

Операції над матрицями.

- `A+B` – додавання;
- `A-B` – віднімання;
- `A%*%B` – множення матриць;
- `t(A)` – транспонування матриці `A`;
- `det(A)` – визначник матриці `A`;
- `solve(A)` – обернена матриця `A`;
- `eigen(A)` – знаходження власних значень матриці `A`.

Приклад створення матриці (див. рис. 9.5):

```
MatrixData_A <- c(1, 2, 3, 4)
A <- matrix(MatrixData_A, nrow=2, ncol=2, byrow = TRUE)
A
```

```
> A
     [,1] [,2]
[1,]    1    3
[2,]    2    4
>
```

Рисунок 9.5 – Результат виконання скрипту виведення матриці у консолі

Завдання до лабораторної роботи

Дано матриці A, $B = \frac{1}{v}A$, C згідно варіанту, де v – номер варіанту. Напишіть скрипт мовою R, який буде виконувати завдання.

1. Обчислити: $A + B; AB; A^{-1}; A^T; \det(A)$.
2. Обчисліть власні значення та власні вектори матриці A.
3. Створіть матрицю AB, приписуючи до A справа B та приписуючи до A знизу B.
4. Розв'язати матричне рівняння $AX = B$ матричним методом.
5. Згенерувати цілочисловий вектор M, що складається з 50 випадкових чисел, що підкоряються нормальному розподілу з математичним сподіванням 25 і середнім квадратичним відхиленням 10.
6. Визначити максимальний, мінімальний елементи і їхні індекси у векторі M.
7. Визначити середнє геометричне й медіану значень вектору M.
8. Визначити моду значень вектору M.

Варіанти завдань

1) $A = \begin{pmatrix} -1{,}7 & 6{,}8 & 9{,}4 & 1{,}3 & 4{,}6 \\ 2{,}8 & 0{,}7 & -5{,}2 & -9{,}5 & 12{,}5 \\ -6{,}1 & 9{,}3 & 12{,}0 & 7{,}8 & 6{,}7 \\ 0{,}5 & -5{,}2 & -4{,}1 & -3{,}4 & -8{,}4 \\ 1{,}1 & 11{,}3 & -5{,}5 & 2{,}2 & 7{,}9 \end{pmatrix}; C = \begin{pmatrix} 25{,}14 \\ -12{,}56 \\ 31{,}22 \\ -17{,}84 \\ 20{,}25 \end{pmatrix}.$

2) $A = \begin{pmatrix} 3{,}2 & 3{,}3 & 4{,}3 & 1{,}8 & -8{,}2 \\ -1{,}3 & -4{,}5 & -5{,}2 & 2{,}5 & -1{,}4 \\ 4{,}9 & 6{,}3 & 2{,}0 & 0{,}8 & 1{,}9 \\ 0{,}2 & 8{,}9 & -14{,}1 & 6{,}4 & 0{,}4 \\ -5{,}7 & -1{,}3 & 6{,}1 & 9{,}3 & -0{,}8 \end{pmatrix}; C = \begin{pmatrix} 12{,}57 \\ -14{,}01 \\ 26{,}48 \\ 13{,}01 \\ 0{,}85 \end{pmatrix}.$

3) $A = \begin{pmatrix} 0{,}2 & 5{,}3 & -7{,}3 & 1{,}5 & 4{,}2 \\ 8{,}5 & -4{,}5 & 3{,}8 & -3{,}8 & -5{,}5 \\ 6{,}2 & -0{,}2 & -13{,}2 & 0{,}2 & 6{,}7 \\ -2{,}7 & -9{,}4 & 10{,}8 & 1{,}4 & -3{,}1 \\ -0{,}9 & 6{,}5 & 7{,}3 & -2{,}3 & -0{,}1 \end{pmatrix}; C = \begin{pmatrix} 8{,}32 \\ 5{,}17 \\ -11{,}26 \\ 21{,}09 \\ 28{,}05 \end{pmatrix}.$

4) $A = \begin{pmatrix} 7{,}3 & -1{,}2 & 0{,}8 & 3{,}8 & 7{,}0 \\ 5{,}9 & -3{,}1 & -1{,}8 & -1{,}9 & 5{,}2 \\ 5{,}7 & 11{,}9 & 17{,}7 & 6{,}1 & 6{,}3 \\ 10{,}5 & -2{,}3 & -3{,}8 & 1{,}8 & 4{,}6 \\ 1{,}6 & 15{,}1 & -4{,}9 & 13{,}1 & 12{,}7 \end{pmatrix}; C = \begin{pmatrix} -18{,}91 \\ 12{,}10 \\ -15{,}45 \\ 22{,}18 \\ 17{,}12 \end{pmatrix}.$

5) $A = \begin{pmatrix} 2{,}6 & 13{,}7 & -3{,}5 & 11{,}4 & -4{,}4 \\ 6{,}5 & 5{,}6 & 11{,}9 & 2{,}5 & 10{,}6 \\ 7{,}7 & 2{,}1 & -1{,}2 & 13{,}2 & 2{,}1 \\ 3{,}6 & -1{,}9 & 7{,}4 & 1{,}8 & -7{,}6 \\ -5{,}9 & 7{,}8 & 5{,}3 & -1{,}7 & 7{,}3 \end{pmatrix}; C = \begin{pmatrix} 23{,}68 \\ 4{,}18 \\ -15{,}70 \\ 4{,}62 \\ -10{,}98 \end{pmatrix}.$

6) $A = \begin{pmatrix} -12{,}7 & 10{,}1 & 15{,}4 & -0{,}5 & -2{,}6 \\ 11{,}3 & -3{,}2 & -7{,}7 & 12{,}1 & 5{,}9 \\ 5{,}7 & 14{,}6 & 0{,}6 & 4{,}7 & 12{,}2 \\ 6{,}4 & 0{,}8 & -0{,}1 & 11{,}6 & 2{,}7 \\ 13{,}5 & -0{,}3 & 14{,}4 & 11{,}1 & 9{,}6 \end{pmatrix}; C = \begin{pmatrix} 14{,}63 \\ -6{,}70 \\ -17{,}16 \\ 7{,}66 \\ 1{,}37 \end{pmatrix}.$

7) $A = \begin{pmatrix} 2{,}9 & 4{,}1 & 12{,}5 & -3{,}6 & 6{,}3 \\ 6{,}5 & 1{,}4 & 10{,}3 & -2{,}3 & -0{,}5 \\ -4{,}1 & 0{,}3 & 7{,}8 & 6{,}2 & 5{,}3 \\ 9{,}4 & -3{,}9 & 14{,}2 & -3{,}6 & 0{,}2 \\ -0{,}4 & 1{,}6 & 7{,}5 & 11{,}2 & 10{,}6 \end{pmatrix}; C = \begin{pmatrix} -12{,}92 \\ -2{,}75 \\ 24{,}07 \\ 3{,}58 \\ 23{,}06 \end{pmatrix}.$

8) $A = \begin{pmatrix} 11{,}2 & 7{,}9 & -4{,}1 & 2{,}3 & 13{,}6 \\ 14{,}1 & 9{,}6 & 4{,}1 & 11{,}7 & -9{,}3 \\ -12{,}4 & -0{,}2 & 10{,}3 & 14{,}9 & 8{,}8 \\ 2{,}5 & 1{,}1 & 6{,}6 & 5{,}9 & 3{,}4 \\ -1{,}9 & 1{,}8 & 9{,}5 & 4{,}7 & 8{,}2 \end{pmatrix}; C = \begin{pmatrix} 10{,}46 \\ 12{,}11 \\ -9{,}42 \\ -16{,}01 \\ 10{,}24 \end{pmatrix}.$

9) $A = \begin{pmatrix} 8{,}8 & 14{,}6 & -2{,}2 & 9{,}1 & 4{,}7 \\ 4{,}8 & 0{,}7 & -3{,}7 & 10{,}2 & -0{,}3 \\ -4{,}2 & 12{,}9 & 10{,}1 & 7{,}8 & 8{,}9 \\ 5{,}9 & 5{,}1 & 14{,}7 & 2{,}6 & 13{,}6 \\ 1{,}7 & 9{,}5 & 4{,}7 & 4{,}2 & -2{,}6 \end{pmatrix}; C = \begin{pmatrix} 18{,}04 \\ 5{,}78 \\ -7{,}45 \\ -15{,}36 \\ 18{,}25 \end{pmatrix}.$

10) $A = \begin{pmatrix} -2{,}2 & 6{,}4 & 13{,}4 & 8{,}5 & 3{,}9 \\ 3{,}7 & 2{,}1 & -5{,}5 & 7{,}2 & 7{,}1 \\ 12{,}7 & 8{,}2 & -9{,}1 & 8{,}9 & 3{,}4 \\ 2{,}1 & 15{,}6 & -3{,}6 & 11{,}4 & 12{,}2 \\ -14{,}1 & 7{,}9 & -5{,}3 & 15{,}2 & 11{,}7 \end{pmatrix}; C = \begin{pmatrix} 4{,}98 \\ -13{,}61 \\ -17{,}36 \\ 6{,}59 \\ -9{,}82 \end{pmatrix}.$

Питання для самоконтролю

1. Для чого використовують мову R?
2. Для чого потрібен Rtools?
3. Які Ви знаєте популярні пакети в R. Охарактеризуйте їх.
4. Охарактеризуйте команди введення\виведення в R.
5. Охарактеризуйте типи даних в R.
6. Наведіть приклади основних команд для роботи з типами даних.
7. Охарактеризуйте структури даних в R.

8. Наведіть приклади використання основних команд для роботи зі структурами даних.
9. Опишіть способи організації функцій в R.
10. Наведіть приклади Data Frame.
11. Як задаються матриці на мові R?
12. Охарактеризуйте операції над матрицями.

10. Лабораторна робота. Візуалізація даних мовою R

Мета: засвоїти можливості роботи мови R до застосування розв'язування задач візуалізації даних.

Теоретичні відомості та методичні рекомендації

R є однією з найкращих мов програмування для візуалізації даних завдяки вбудованим графічним можливостям та потужним бібліотекам. Візуалізація у R дозволяє аналізувати великі обсяги даних, знаходити закономірності та ефективно презентувати результати.

1. Базова графіка в R. Мова R має вбудовані засоби для побудови простих графіків без встановлення додаткових пакетів.

Наприклад, лінія по точках (див. рис. 10.1).

```
x <- c(1, 2, 3, 4, 5)
y <- c(2, 4, 6, 8, 10)

plot(x, y, type="o", col="blue", main="Простий графік",
xlab="X-вісь", ylab="Y-вісь")
```

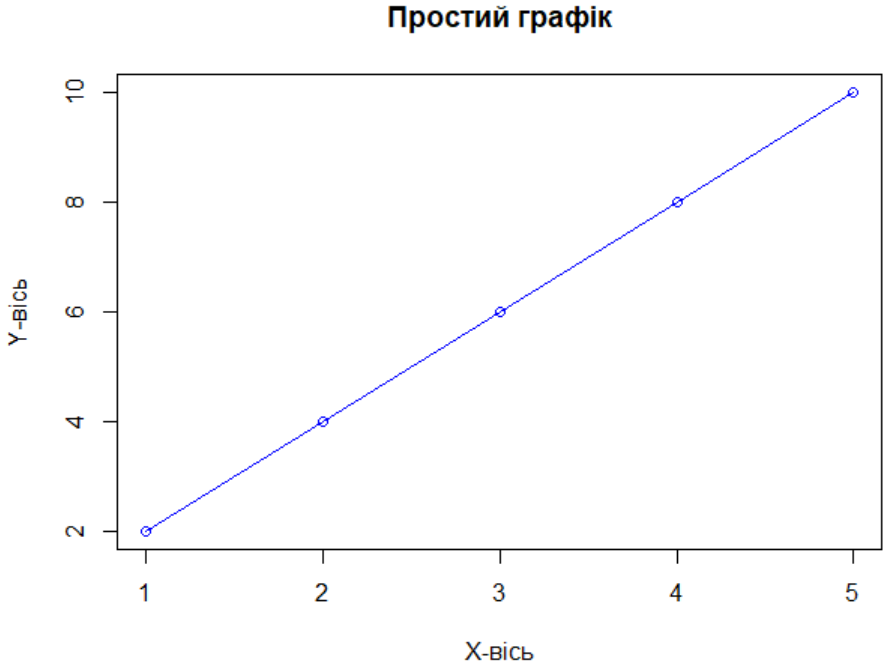

Рисунок 10.1 – Лінія по точках

Тут

- `type="o"` – лінія з точками.
- `col="blue"` – колір лінії.
- `main` – заголовок.
- `xlab, ylab` – підписи осей.

Наприклад, побудуємо графік функції $y = \sin x$ (див. рис. 10.2).

```
# Генеруємо значення x від -π до π
x <- seq(-pi, pi, length.out = 100)
```

```
# Обчислюємо значення y = sin(x)
y <- sin(x)

# Будуємо графік
plot(x, y, type="l", col="blue", lwd=2,
     main="Графік функції y = sin(x)",
     xlab="x", ylab="sin(x)")

# Додаємо сітку
grid()
```

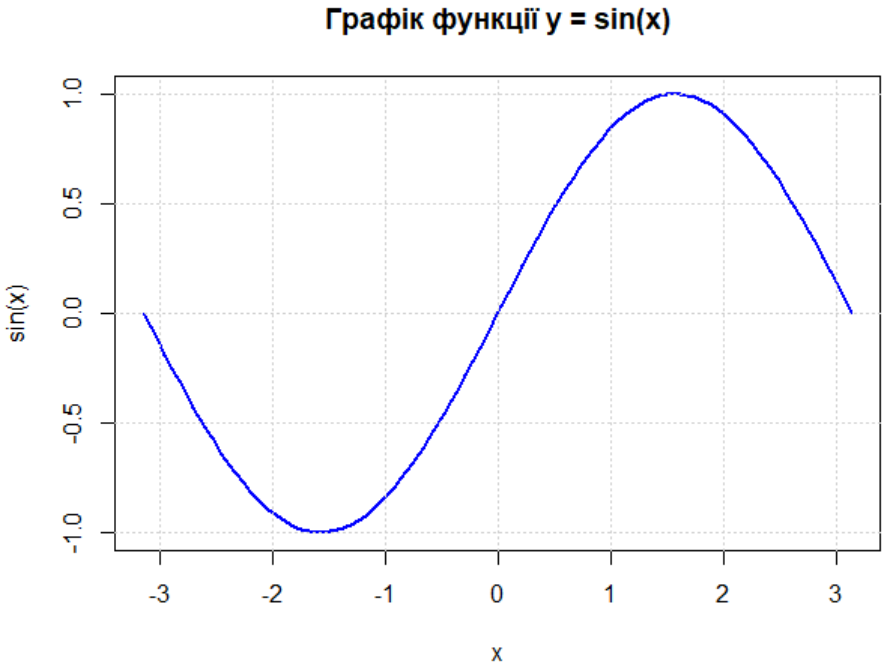

Рисунок 10.2 – Графік функції $y = \sin x$

Тут

- `seq(-pi, pi, length.out = 100)` – створює 100 точок від $-\pi$ до π.
- `sin(x)` – обчислює значення функції.
- `plot(..., type="l")` – будує лінійний графік.
- `col="blue"` – задає колір синусоїди.
- `lwd=2` – збільшує товщину лінії.
- `grid()` – додає сітку.

Мова R має вбудовані засоби для побудови стовпчикових гістограм за допомогою функції `hist()` (див. рис. 10.3).

```
data <- rnorm(1000)  # Генеруємо 1000 випадкових значень
hist(data, col="skyblue", main="Гістограма",
xlab="Значення", breaks=30)  # breaks=30 — кількість
інтервалів
```

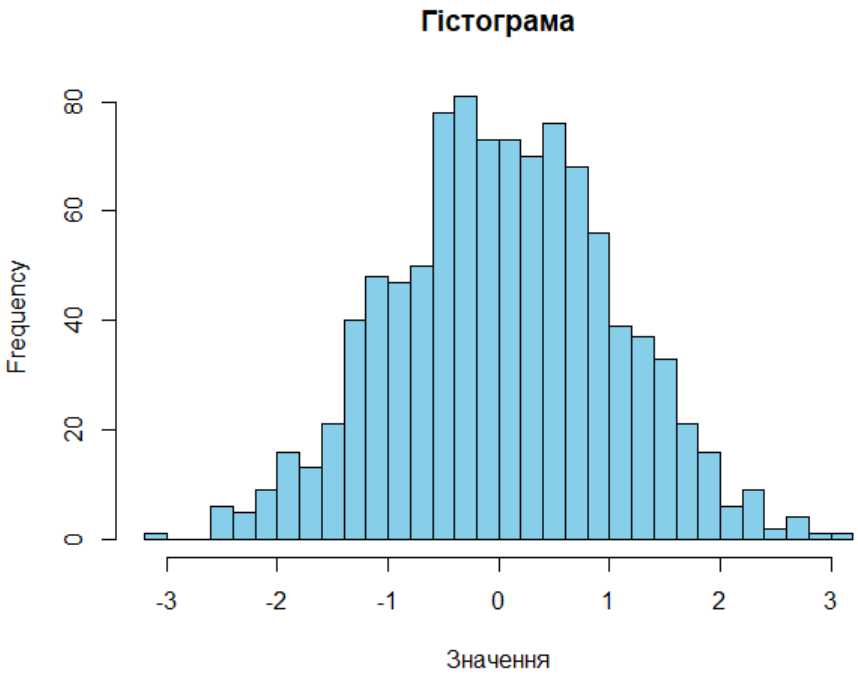

Рисунок 10.3 – Стовпчикова гістограма `hist()`

Коробковий графік (boxplot) показує розподіл даних, медіану та квартилі (див. рис. 10.4).

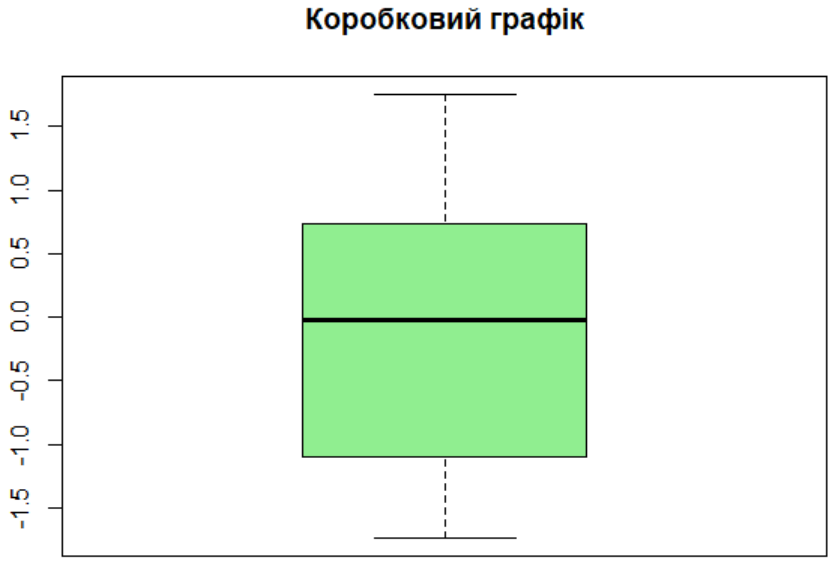

Рисунок 10.4 – Коробковий графік

```
values <- rnorm(50)
boxplot(values, main="Коробковий графік",
col="lightgreen")
```

Кругова діаграма використовується для візуалізації часток категорій у загальному обсязі даних. У R можна створити кругову діаграму використовуючи функцію `pie()` (Base R Graphics);

Наприклад, (див. рис. 10.5).

```
# Дані
categories <- c("A", "B", "C", "D")
values <- c(25, 30, 20, 25)  # Частки у відсотках
```

```
# Побудова кругової діаграми
pie(values, labels = categories, col = rainbow(length(values)), main = "Кругова діаграма")
```

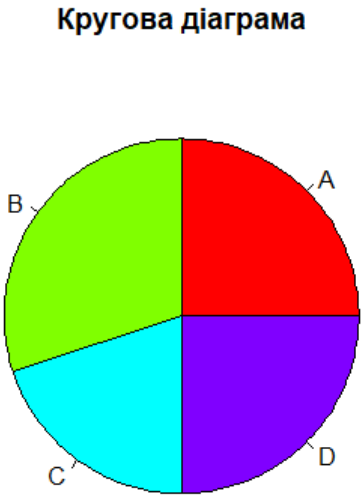

Рисунок 10.5 – Кругова діаграма

Тут

- `values` – вектор часток;
- `labels = categories` – підписи для секторів;
- `col = rainbow(length(values))` – кольори секторів;
- `main = "Кругова діаграма"` – заголовок.

2. Графіка в R за допомогою бібліотеки ggplot2. Бібліотека ggplot2 – це одна з найпопулярніших бібліотек для візуалізації даних у R. Вона базується на концепції Grammar of Graphics (граматика графіки), що дозволяє легко створювати складні графіки шляхом додавання різних шарів.

Основні переваги ggplot2:

- гнучкість та налаштовуваність;
- професійний вигляд графіків;
- можливість поєднувати різні типи візуалізацій;
- можливість додавання інших графічних елементів;
- інтуїтивний синтаксис.

Якщо бібліотека ще не встановлена, то потрібно виконати команду у консолі:

```
install.packages("ggplot2")
```

Після встановлення пакета ви можете завантажити його за допомогою функції R `library("ggplot2")`. Наприклад, (див. рис. 10.6).

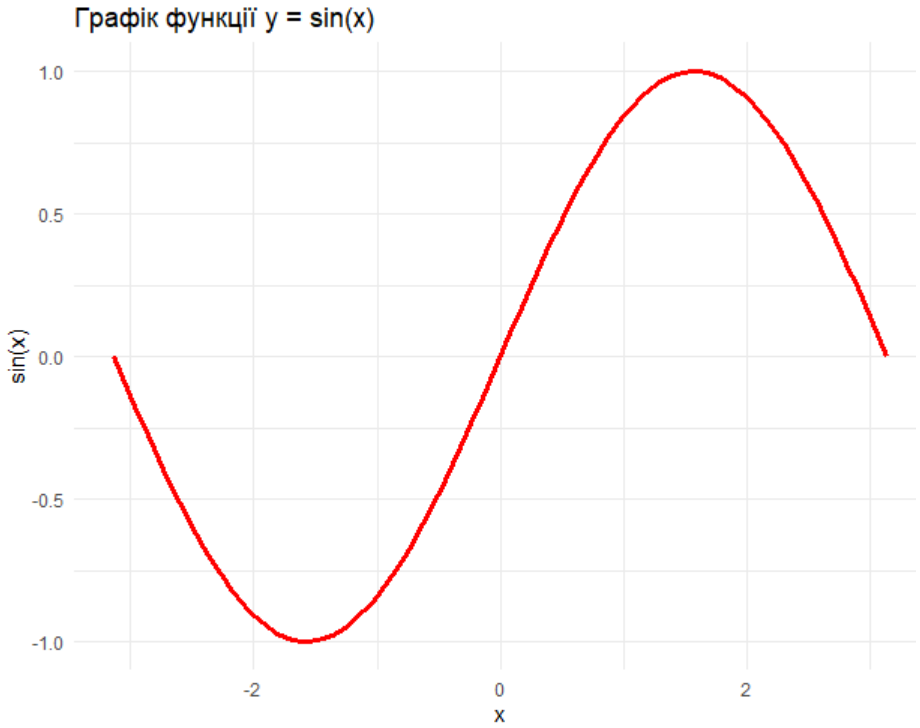

Рисунок 10.6 – Графік функції $y = \sin x$ за допомогою бібліотеки ggplot2

```
# Завантажуємо бібліотеку ggplot2
library(ggplot2)

# Створюємо датафрейм для ggplot2
df <- data.frame(x = seq(-pi, pi, length.out = 100))
df$y <- sin(df$x)

# Будуємо графік за допомогою ggplot2
ggplot(df, aes(x=x, y=y)) +
  geom_line(color="red", size=1.2) +
  ggtitle("Графік функції y = sin(x)") +
  xlab("x") + ylab("sin(x)") +
  theme_minimal()
```

Можна вивести на екран дві функції. Наприклад, додамо функцію $y = \cos x$, отримаємо (див. рис. 10.7).

```
df$cos_y <- cos(df$x)

ggplot(df, aes(x=x)) +
  geom_line(aes(y=y, color="sin(x)"), size=1.2) +
  geom_line(aes(y=cos_y, color="cos(x)"), size=1.2) +
  ggtitle("Графіки y = sin(x) та y = cos(x)") +
  xlab("x") + ylab("Значення") +
  scale_color_manual(name="Функції",
values=c("sin(x)"="red", "cos(x)"="blue")) +
  theme_minimal()
```

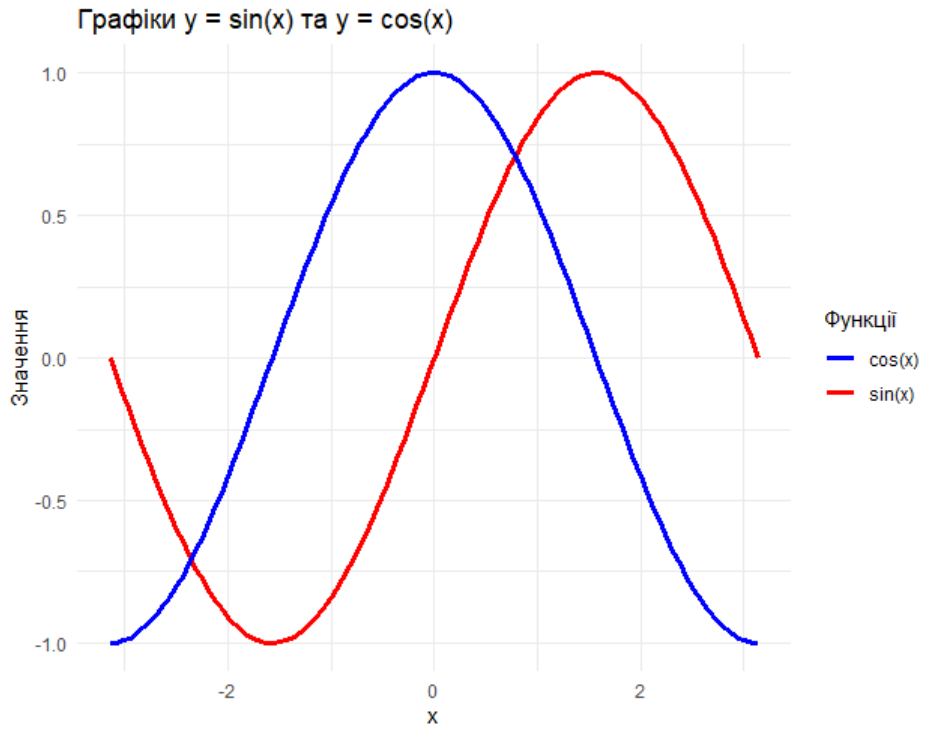

Рисунок 10.7 – Графіки функцій $y = \sin x$ і $y = \cos x$

Приклад створення гістограми за допомогою функції `geom_histogram()` з бібліотеки ggplot2 (див. рис. 10.8).

```
df <- data.frame(values = rnorm(1000))
ggplot(df, aes(x = values)) +
  geom_histogram(binwidth = 0.5, fill="blue", color="black") +
  ggtitle("Гістограма") +
  theme_classic()
```

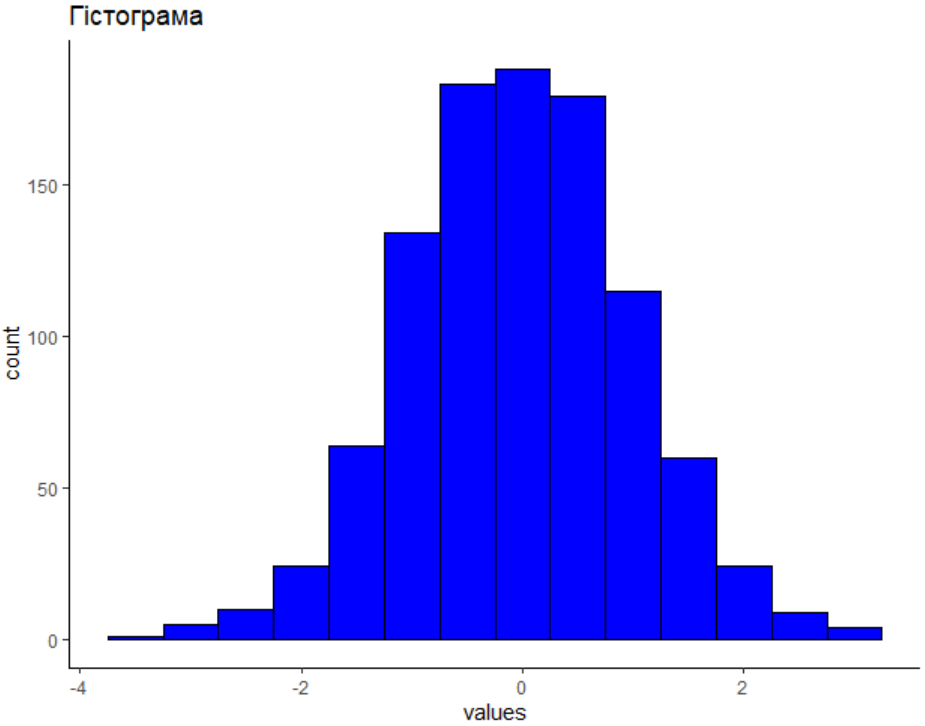

Рисунок 10.8 – Гістограма у ggplot2

Приклад створення кругової діаграми за допомогою функції `geom_bar()` з бібліотеки ggplot2 (див. рис. 10.9).

Рисунок 10.9 – Кругова діаграма у ggplot2

```
library(ggplot2)

# Дані
df <- data.frame(category = c("A", "B", "C", "D"),
                 values = c(25, 30, 20, 25))

# Побудова кругової діаграми
ggplot(df, aes(x = "", y = values, fill = category)) +
  geom_bar(stat = "identity", width = 1) +   # Створюємо
стовпчики
  coord_polar("y", start = 0) +   # Полярна координатна
система
  ggtitle("Кругова діаграма") +
  theme_void()  # Видаляємо осі та сітку
```

Тут

- `geom_bar(stat="identity", width=1)` – створює стовпчики, які потім перетворюються в кругові сектори;
- `coord_polar("y", start=0)` – змінює систему координат на полярну, утворюючи кругову діаграму;
- `theme_void()` – прибирає сітку та осі;
- `fill = category` – задає кольори для секторів.

Отже, R має широкий вибір інструментів для візуалізації даних, що робить його ідеальним для аналізу та презентації інформації. Якщо потрібно швидко створити кругову діаграму – використовуйте функцію `pie()`, а для професійної візуалізації – пакет ggplot2.

Завдання до лабораторної роботи

1. Побудова базових графіків у R.

1) Створити випадковий набір даних з 50 чисел за допомогою функції `rnorm()`.
2) Побудувати графік розсіювання через функцію `plot()`.
3) Побудувати гістограму через функцію `hist()`.
4) Побудувати коробковий графік через функцію `boxplot()`.
5) Побудувати стовпчикову діаграму через функцію `barplot()`.

2. Побудувати графіки функцій. Використати ggplot2 для побудови графіків.

1) $y = \sin(x) + \frac{1}{2}\sin(2x) + \frac{1}{3}\sin(3x), y = \sqrt[7]{x^3 - 3x^2 + 5}$.
2) $y = 8x^2(x^2 - 1)^3, y = \text{tg}\left(9x - \frac{\pi}{2}\right)$.
3) $y = \frac{x}{\ln x}, y = x^5 + 6x^4 - \frac{1}{x}$.
4) $y = \frac{x^3 - 2x^2 - x + 2}{x}, y = \text{ctg}\left(3x + \frac{\pi}{2}\right)$.
5) $y = (x^2 - 2)e^{-2x}, y = \frac{x^5 - 8x^4 + 3x - 5}{\sqrt{x}}$.
6) $y = \frac{x^3}{x^2 - 1}, y = \sin\left(5x - \frac{\pi}{2}\right)$.
7) $y = \sin x - \ln(\sin x), y = e^{2x+3}$.
8) $y = \cos x + \frac{1}{2}\cos(2x) + \frac{1}{3}\cos(3x), y = \sqrt[3]{7x^6 - 5}$.
9) $y = \sqrt[3]{x(7 - x)^2} - \frac{x}{2}, y = \cos\left(2x + \frac{\pi}{2}\right)$.
10) $y = \left(\frac{x+2}{x-2}\right)^2$.

3. Побудувати гістограму та зробити оцінки розподілу даних. Побудувати гістограму з кривою густини, що показує нормальний розподіл.

1) Згенерувати 1000 випадкових чисел за нормальним розподілом, використовуючи функцію `rnorm()`.
2) Побудувати гістограму `hist()` та `ggplot2::geom_histogram()`.
3) Додати криву густини, використовуючи функцію `density()`.

4. Ознайомитися з побудовою кругових діаграм. Побудувати кругові діаграми.

1) Створити набір даних із категоріями ("A", "B", "C", "D") та значеннями.
2) Побудувати кругову діаграму за допомогою функції `pie()`.
3) Побудувати аналогічну кругову діаграму з використанням `ggplot2::coord_polar()`.
4) Додати відсоткові підписи.

5. Візуалізація часових рядів.

1) Згенерувати набір даних для часових рядів, використовуючи функцію `ts()`.

2) Побудувати лінійний графік зміни значень у часі за допомогою `plot()`.
3) Використати `ggplot2::geom_line()` для візуалізації тих самих даних.

Питання для самоконтролю

1. Охарактеризуйте можливості для візуалізації у мові R.
2. Охарактеризуйте можливості бібліотеки ggplot2.
3. Розкажіть про способи побудови стовпчикових гістограм.
4. Розкажіть про способи побудови кругових діаграм.
5. Охарактеризуйте особливості візуалізації часових рядів.

ВИКОРИСТАНА ЛІТЕРАТУРА

1. Васильєв О. Програмування в PYTHON. Теорія і практика. Київ : Ліра-К, 2023. 462 с.
2. Гнатюк В. Вступ до R на прикладах. Харків : Харківський Національний Економічний Університет, 2010. 107 с.
3. Зеленяк О. Програмування мовою Python. Алгоритмічні структури і стратегії. Теорія та практика. Київ : Ліра-К, 2025. 338 с.
4. Julien G. Python Programming for Mathematics. Publisher : CRC Press, 2025. 248 p.
5. Johansson R. Numerical Python: Scientific Computing and Data Science Applications with Numpy, SciPy and Matplotlib Third Edition. Publisher : Apress, 2024. 512 p.
6. Maindonald J., John Braun W. Data Analysis and Graphics Using R. Publisher : Cambridge University Pres, 2013. 549 p.
7. McKinney W. Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter 3rd Edition. Publisher : O'Reilly Media, 2022. 579 p.
8. Molin S. Data Analysis with Pandas: A Python data science handbook for data collection, wrangling, analysis, and visualization 2rd Edition. Publisher : Packt Publishing, 2021. 788 p.
9. Nelli F. Python Data Analytics: With Pandas, NumPy, and Matplotlib 3rd ed. Edition. 2023, Publisher : Apress. 466 p.
10. Pajankar A. Matplotlib. Learn Plotting and Visualizations with Python 3. 1st Ed. Publisher : Apress, 2022. 300 p.
11. Math – Mathematical functions. URL : https://docs.python.org/uk/3.13/library/math.html?utm_source=chatgpt.com
12. Cmath – Mathematical functions for complex numbers. URL : https://docs.python.org/uk/3.13/library/cmath.html#module-cmath
13. Matplotlib. URL : https://matplotlib.org
14. NumPy. URL : https://numpy.org/
15. Pandas. URL : https://pandas.pydata.org/
16. SciPy. URL : https://scipy.org/
17. Seaborn. URL : https://seaborn.pydata.org/
18. SymPy. URL : https://www.sympy.org/en/index.html
19. Scikit-learn. URL : https://scikit-learn.org/stable/
20. Ядро R. URL : https://cran.r-project.org/bin/windows/base/
21. RTools: URL : https://cran.r-project.org/bin/windows/Rtools/
22. RStudio IDE. URL : https://posit.co/download/rstudio-desktop/

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Васильєв О. Програмування в PYTHON. Теорія і практика. Київ : Ліра-К, 2023. 462 с.
2. Зеленяк О. Програмування мовою Python. Алгоритмічні структури і стратегії. Теорія та практика. Київ : Ліра-К, 2025. 338 с.
3. Johansson R. Numerical Python: Scientific Computing and Data Science Applications with Numpy, SciPy and Matplotlib Third Edition. Publisher : Apress, 2024. 512 p.
4. McKinney W. Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter 3rd Edition. Publisher : O'Reilly Media, 2022. 579 p.
5. Nelli F. Python Data Analytics: With Pandas, NumPy, and Matplotlib 3rd ed. Edition. Publisher : Apress, 2023. 466 p.
6. Molin S. Data Analysis with Pandas: A Python data science handbook for data collection, wrangling, analysis, and visualization 2rd Edition. Publisher : Packt Publishing, 2021. 788 p.
7. Pajankar A. Matplotlib. Learn Plotting and Visualizations with Python 3. 1st Ed. Publisher : Apress, 2022. 300 p.

Навчальне видання
(*українською мовою*)

Гоменюк Сергій Іванович
Гребенюк Сергій Миколайович
Панасенко Євген Валерійович

ЗАСТОСУВАННЯ СУЧАСНИХ МОВ ПРОГРАМУВАННЯ ДО РОЗВ'ЯЗАННЯ МАТЕМАТИЧНИХ ЗАДАЧ

Методичні рекомендації до лабораторних занять
для здобувачів ступеня вищої освіти магістра
спеціальності «Математика»
освітньо-професійної програми «Математика»

Рецензент *А.О. Лісняк*
Відповідальний за випуск *С.М. Гребенюк*
Коректор *Є.В. Панасенко*