МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
УКРАЇНСЬКО-УГОРСЬКИЙ НАУКОВО-НАВЧАЛЬНИЙ ІНСТИТУТ
КАФЕДРА ФІЗИКО-МАТЕМАТИЧНИХ ДИСЦИПЛІН

ВИКОРИСТАННЯ ІМПОРТОВАНИХ БІБЛІОТЕК PYTHON У РОЗВ'ЯЗУВАННІ ПРИКЛАДНИХ ЗАДАЧ МАТЕМАТИКИ, ФІЗИКИ ТА ІНФОРМАТИКИ

методичні рекомендації до навчально обчислювальної практики з програмування для студентів спеціальностей А4.08 Середня освіта. Фізика та астрономія, А4.04 Середня освіта. Математика, F Комп'ютерні науки

УЖГОРОД – 2025

Використання імпортованих бібліотек Python у розв'язуванні прикладних задач математики, фізики та інформатики: методичні рекомендації до навчально обчислювальної практики з програмування для студентів спеціальностей А4.08 Середня освіта. Фізика та астрономія, А4.04 Середня освіта. Математика, F Комп'ютерні науки / М.М. Повідайчик, М.І. Шафраньош, О.С. Лукач. Ужгород: Видавництво «Шарк», 2025. 53 с.

Рецензенти:
Мулеса П.П., доктор педагогічних наук, завідувач кафедри кібернетики і прикладної математики ДВНЗ «УжНУ»;
Брила А.Ю., кандидат фізико-математичних наук, доцент кафедри системного аналізу ДВНЗ «УжНУ».

Розглянуто і схвалено на засіданні кафедри фізико-математичних дисциплін
(протокол №1 від 27.08.2025 р.)

Рекомендовано до друку науково-методичною комісією українсько-угорського науково-навчального інституту
(протокол №1 від 30.09.2025 р.)

ЗМІСТ

ВСТУП

Сучасний етап розвитку освіти характеризується активним упровадженням інформаційно-комунікаційних технологій у навчальний процес, що зумовлює необхідність інтеграції знань із різних галузей – насамперед математики, фізики та інформатики. В умовах формування інформаційного суспільства та цифрової економіки особливої ваги набуває підготовка педагогічних кадрів, здатних ефективно застосовувати сучасні програмні засоби для візуалізації, моделювання та аналізу математичних об’єктів і процесів.

Одним із потужних інструментів, що поєднує можливості програмування, математичного моделювання та аналітичних обчислень, є мова програмування Python. Завдяки своїй простоті, відкритості, кросплатформності та широкій базі наукових бібліотек Python поступово перетворюється на універсальний засіб навчання у сфері STEM-освіти (Science, Technology, Engineering, Mathematics). Використання імпортованих математичних пакетів, зокрема NumPy, SciPy та SymPy, відкриває перед учнями та студентами широкі можливості для дослідження математичних закономірностей, виконання складних обчислень, побудови моделей та розв’язування прикладних задач.

Застосування зазначених бібліотек у навчальному процесі сприяє формуванню в учнів і студентів дослідницьких, алгоритмічних і критичних мисленнєвих умінь, розвиває навички роботи з великими обсягами даних, стимулює інтерес до природничих наук і програмування. Це відповідає компетентнісному підходу, задекларованому в Державному стандарті базової середньої освіти України [5], який передбачає інтеграцію навчальних галузей та розвиток умінь застосовувати знання в реальних життєвих ситуаціях.

Водночас у практиці викладання математики у середній школі спостерігається недостатня реалізація потенціалу сучасних цифрових інструментів, що ускладнює формування міжпредметних зв’язків між математикою та

інформатикою. У цьому контексті опанування та методично обґрунтоване використання бібліотек Python під час навчання математики є актуальним завданням педагогічної науки й практики.

Таким чином, дослідження можливостей застосування імпортованих математичних бібліотек Python – NumPy, SciPy, SymPy – у процесі підготовки майбутніх учителів математики, фізики та інформатики, а також у навчанні учнів середньої школи, є своєчасним і значущим для вдосконалення змісту математичної освіти, підвищення її практичної спрямованості та розвитку цифрової компетентності всіх учасників освітнього процесу.

РОЗДІЛ 1
ІМПОРТОВАНІ ПАКЕТИ PYTHON: СУТНІСТЬ, РОЛЬ І КЛАСИФІКАЦІЯ

1.1. Роль імпортованих пакетів у розвитку мови Python.

1.1.1. Основні поняття.

Однією з ключових особливостей мови програмування Python є її модульна структура, яка забезпечує високу гнучкість, розширюваність і можливість повторного використання коду. Модульність – це принцип організації програмного забезпечення, за якого велика програма поділяється на окремі функціональні частини – модулі, кожен з яких виконує певну логічно завершену функцію.

Модуль у Python – це файл із розширенням .py, який містить змінні, функції, класи або інші об’єкти, що можуть бути використані в інших програмах [15]. Завдяки цьому розробник може структурувати свій код, підвищуючи його читабельність і спрощуючи процес тестування та супроводу. Імпорт модуля здійснюється за допомогою ключового слова import (рис. 1.1).

```
import math
print(math.sqrt(16))   # Виведе 4.0
print(math.pi)         # Константа π
```

```
4.0
3.141592653589793
```

Рис. 1.1. Використання стандартного модулю math.

У наведеному прикладі використано стандартний модуль math, який містить функції для виконання математичних обчислень.

На основі модулів формується поняття пакета. Пакет у Python – це структурований каталог, що містить набір модулів, які логічно об’єднані за певною тематикою або призначенням. Кожен пакет може включати підпакети, утворюючи ієрархічну систему модулів. Пакет розпізнається за наявністю у його каталозі

спеціального файлу, який сигналізує інтерпретатору Python про те, що каталог слід розглядати як пакет.

Наприклад, бібліотека SciPy є пакетом, який складається з багатьох підпакетів:

- scipy.linalg – для лінійної алгебри,
- scipy.integrate – для інтегрування,
- scipy.stats – для статистичних обчислень,
- scipy.optimize – для розв’язування оптимізаційних задач тощо.

Користувач може імпортувати як увесь пакет, так і окремі його підпакети або модулі (рис. 1.2).

```
from scipy import integrate
result, error = integrate.quad(lambda x: x**2, 0, 1)
print(result)
```

```
0.3333333333333337
```

Рис. 1.2. Використання підпакету scipy.integrate.

Завдяки такій структурі Python підтримує концепцію повторного використання коду (code reuse) та інкапсуляції функціональності, що є важливою педагогічною і методичною перевагою під час навчання програмування у шкільному курсі інформатики. Використання модулів і пакетів не лише спрощує створення складних програм, але й формує в учнів уміння логічно структурувати інформацію, аналізувати взаємозв’язки між елементами програмного середовища, що безпосередньо пов’язано з розвитком алгоритмічного та системного мислення.

Таким чином, модулі та пакети в Python становлять основу його архітектури, забезпечуючи ефективну організацію програмного коду та сприяючи інтеграції численних зовнішніх бібліотек, зокрема математичних, що істотно розширюють навчальні й дослідницькі можливості цього середовища.

1.1.2. Стандартна бібліотека Python і зовнішні (імпортовані) пакети.

Однією з переваг мови програмування Python є її значна стандартна бібліотека, що містить велику кількість модулів, які охоплюють різні напрями застосування – від базових операцій з файлами та обробки тексту до роботи з мережею, системним оточенням і математичними обчисленнями. Завдяки цьому Python часто називають мовою з «батарейками в комплекті» (batteries included), що підкреслює її готовність до виконання більшості типових завдань без потреби встановлення додаткових інструментів.

До складу стандартної бібліотеки входять такі модулі, як:

- math – для виконання базових математичних операцій (корені, степені, тригонометричні функції, константи π, e тощо);
- random – для генерації випадкових чисел і моделювання випадкових подій;
- statistics – для обчислення середніх значень, дисперсії, медіани та інших статистичних показників;
- datetime – для роботи з датами й часом;
- os і sys – для взаємодії з операційною системою;
- json, csv, re – для роботи з текстовими форматами даних та регулярними виразами.

Наприклад, використання стандартного модуля statistics дає змогу легко виконувати обчислення, пов’язані з аналізом даних (рис. 1.3).

```
import statistics as st
data = [3, 7, 7, 2, 9, 10, 4]
print("Середнє значення:", st.mean(data))
print("Медіана:", st.median(data))
print("Стандартне відхилення:", round(st.stdev(data), 2))
```

```
Середнє значення: 6
Медіана: 7
Стандартне відхилення: 3.06
```

Рис. 1.3. Використання стандартного модуля statistics.

У наведеному прикладі видно, що Python уже містить базові можливості для математичних розрахунків, достатні для розв’язування навчальних задач середнього рівня складності.

Проте у процесі розвитку мови й зростання вимог до наукових, інженерних і освітніх застосувань з’явилася потреба у більш потужних і спеціалізованих бібліотеках, здатних виконувати складні математичні, статистичні, графічні та символьні операції. Такі бібліотеки не входять до стандартної поставки Python і встановлюються окремо – за допомогою менеджера пакетів pip або через спеціальні середовища, наприклад Anaconda. Їх називають зовнішніми (імпортованими) пакетами.

Використання зовнішніх бібліотек суттєво розширює функціональні можливості Python. Зокрема, до найпоширеніших математичних та наукових пакетів належать:

- NumPy – основа для чисельних розрахунків, роботи з багатовимірними масивами та матрицями;
- SciPy – бібліотека для наукових і технічних обчислень, яка включає засоби для інтегрування, оптимізації, статистичного аналізу тощо;
- SymPy – інструмент для символьних (аналітичних) обчислень;
- pandas – для обробки та аналізу табличних даних;
- matplotlib – для побудови графіків і візуалізації результатів;
- scikit-learn, TensorFlow, PyTorch – для машинного навчання та штучного інтелекту.

Встановлення таких пакетів виконується командою у терміналі:

pip install numpy scipy sympy

Після інсталяції пакети можна імпортувати у програму для подальшого використання (рис. 1.4).

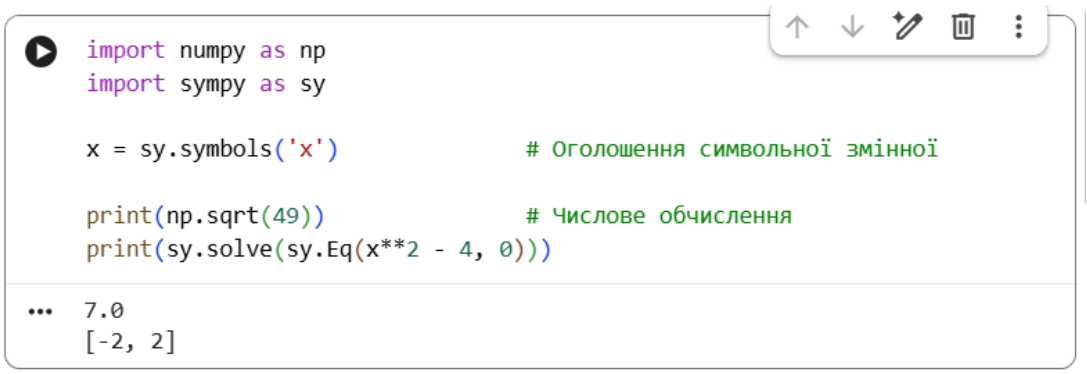

Рис. 1.4. Використання пакетів NumPy, SymPy.

Для навчального процесу важливо, що такі пакети є безкоштовними, відкритими й підтримуються активною науковою спільнотою, що дозволяє вчителям і учням використовувати їх на будь-якому рівні підготовки без додаткових витрат. Крім того, зовнішні бібліотеки Python активно застосовуються у вищій освіті, дослідницькій діяльності, інженерії та аналітиці даних, що сприяє профорієнтації школярів і розвитку практичної математичної компетентності.

Таким чином, поєднання стандартної бібліотеки Python із зовнішніми (імпортованими) пакетами створює універсальне середовище для навчання, дослідження та моделювання, а також забезпечує підґрунтя для формування міжпредметних зв’язків між математикою, інформатикою та природничими науками у сучасній школі.

1.1.3. Встановлення пакетів через pip, робота з середовищами (venv, Anaconda).

Ефективне використання мови програмування Python у навчальному процесі неможливе без розуміння механізмів встановлення та керування бібліотеками. Для цього у Python передбачено вбудовані інструменти, які дозволяють користувачеві швидко встановлювати, оновлювати та видаляти зовнішні пакети, необхідні для виконання навчальних або наукових завдань.

Найпоширенішим засобом є менеджер пакетів pip (Python Installer Package), який входить до складу стандартної інсталяції Python, починаючи з версії 3.4. Його

основним призначенням є встановлення програмних бібліотек із центрального репозиторію PyPI (Python Package Index) – офіційного сховища, де розміщено понад 500 000 відкритих пакетів.

Базовий синтаксис команди pip має вигляд:

pip install назва_пакета

Наприклад, для встановлення математичних бібліотек NumPy, SciPy і SymPy використовуються команди:

pip install numpy

pip install scipy

pip install sympy

Після встановлення пакети зберігаються в середовищі Python і стають доступними для імпорту у будь-якій програмі. Для перевірки наявності пакета можна скористатися командою:

pip list

а для оновлення:

pip install --upgrade назва_пакета

Робота з віртуальними середовищами (venv). Для зручності організації навчальних і дослідницьких проєктів у Python використовується механізм віртуальних середовищ (virtual environments), що дозволяє створювати окремі ізольовані простори для встановлення пакетів. Це дає змогу уникнути конфліктів між різними версіями бібліотек і забезпечити стабільність роботи програм.

Віртуальне середовище створюється за допомогою стандартного модуля venv:

python -m venv myenv

де myenv – назва середовища.

Для активації середовища використовується команда:

- у Windows:

myenv\Scripts\activate

- у Linux / macOS:

source myenv/bin/activate

Після активації середовище функціонує незалежно від глобальної інсталяції Python, що є зручним під час виконання лабораторних чи проєктних робіт, коли для кожного завдання можуть бути потрібні різні бібліотеки або їх версії.

Деактивація середовища здійснюється командою:

deactivate

Середовище Anaconda. Для навчальних і наукових цілей особливо зручним є використання дистрибутива Anaconda, який поєднує у собі Python, менеджер пакетів conda, а також велику кількість попередньо встановлених бібліотек (NumPy, SciPy, SymPy, pandas, matplotlib, Jupyter Notebook тощо). Anaconda забезпечує зручний графічний інтерфейс (Anaconda Navigator) і дозволяє створювати окремі середовища для різних проєктів, аналогічно до venv, але з розширеним функціоналом.

Розглянемо покрокову інструкцію створення середовища у Anaconda.

1. Завантажити Anaconda (https://www.anaconda.com/download).

2. Запустити інсталятор.

3. Запустити Anaconda Prompt (термінал)

4. Створити навчальне середовище (працює лише локально!)

conda create --name school_env python=3.12

5. Активувати середовище

conda activate school_env

6. Встановити потрібні бібліотеки

conda install numpy scipy sympy

або

pip install numpy scipy sympy

(у conda-середовищі pip також працює)

7. Запустити Jupyter Notebook або JupyterLab

Запуск Jupyter Notebook:

jupyter notebook

або JupyterLab:

jupyter lab

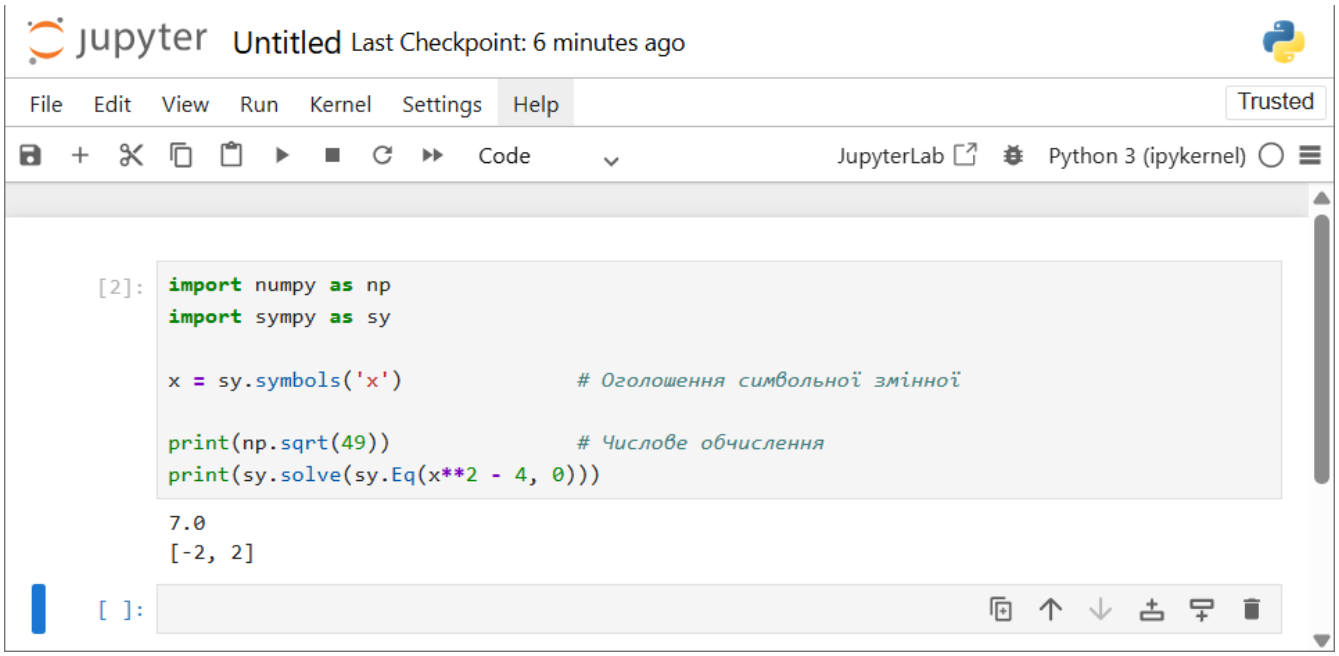

Рис. 1.5. Використання Jupyter Notebook.

Завдяки простоті використання, сумісності з різними операційними системами та наявності інструментів для інтерактивного програмування (наприклад, Jupyter Notebook), Anaconda є рекомендованим середовищем для навчання Python у закладах освіти.

Педагогічна доцільність використання pip, venv та Anaconda. Для вчителя інформатики або математики розуміння принципів роботи з цими інструментами має не лише технічне, а й методичне значення.

- Використання pip дозволяє ознайомити учнів із сучасною культурою роботи з відкритим програмним забезпеченням і формує базові навички самостійного встановлення програмних ресурсів.
- Робота з віртуальними середовищами (venv, conda) сприяє розвитку в учнів інформаційної культури, уміння структурувати навчальні проєкти, логічно

організовувати робочий простір і відповідально ставитися до використання ресурсів.

- Застосування середовища Anaconda забезпечує зручність для освітніх цілей, дозволяє швидко організувати демонстрації, виконання практичних робіт і міні-проєктів без складної попередньої конфігурації системи.

Отже, знання принципів роботи з менеджером пакетів pip, віртуальними середовищами venv та освітньо-науковим дистрибутивом Anaconda є важливим компонентом цифрової компетентності майбутнього вчителя математики й інформатики, оскільки забезпечує технічну основу для ефективного використання імпортованих бібліотек Python у навчальному процесі.

1.1.4. Класифікація пакетів Python за напрямами використання.

Мова програмування Python відзначається своєю гнучкістю та широким спектром застосування, що зумовлено наявністю великої кількості зовнішніх бібліотек і пакетів, розроблених для розв’язання завдань різних галузей науки, освіти та техніки. Сучасна класифікація пакетів Python базується на їх функціональному призначенні, що дозволяє ефективно орієнтуватися у виборі необхідних інструментів для навчальних і дослідницьких цілей.

У педагогічному контексті, особливо у підготовці майбутніх учителів математики та інформатики, така класифікація має не лише теоретичне, але й практичне значення. Вона допомагає сформувати у здобувачів системне уявлення про екосистему Python і навчити їх обирати оптимальні засоби для розв’язання конкретних навчальних або методичних завдань.

Математичні та наукові бібліотеки.

Ця група пакетів забезпечує проведення розрахунків, моделювання, обробку експериментальних даних та реалізацію алгоритмів вищої математики.

До найпоширеніших належать:

- NumPy – основа наукових обчислень у Python, що забезпечує роботу з багатовимірними масивами та високопродуктивні операції над ними.

- SciPy – бібліотека для розширених наукових і технічних обчислень: інтегрування, оптимізація, статистика, обробка сигналів.
- SymPy – система символьної математики, призначена для виконання аналітичних обчислень, таких як спрощення виразів, диференціювання, інтегрування, розв’язування рівнянь.
- Matplotlib, Seaborn, Plotly – бібліотеки для візуалізації результатів обчислень у вигляді графіків, діаграм, функціональних поверхонь.

Ці інструменти широко застосовуються у навчанні алгебри, аналізу, геометрії, теорії ймовірностей та статистики, а також при дослідженні математичних моделей.

Освітні та дидактичні бібліотеки.

До цієї групи належать пакети, створені спеціально для освітніх цілей або для підтримки процесу навчання програмуванню та математиці:

- Turtle – візуальна бібліотека для навчання алгоритмізації через побудову графічних об’єктів; активно використовується у шкільних курсах інформатики.
- Jupyter Notebook – інтерактивне середовище для виконання коду, тексту та візуалізацій, зручне для створення навчальних матеріалів і презентацій.
- Manim – бібліотека для створення анімованих математичних пояснень (використовується, зокрема, у популярних навчальних відео).

Такі пакети сприяють розвитку в учнів візуального та алгоритмічного мислення, а також дозволяють поєднати математичний зміст із практичними навичками програмування.

Пакети для роботи з даними та штучним інтелектом.

Python є однією з провідних мов у сфері аналізу даних і машинного навчання. До відповідних бібліотек належать:

- pandas – для обробки та аналізу табличних даних;
- scikit-learn – для реалізації алгоритмів машинного навчання;

- TensorFlow, PyTorch – для побудови нейронних мереж і глибинного навчання;
- Statsmodels – для статистичного моделювання.

У шкільній освіті ці інструменти можуть використовуватись у спрощеній формі для ознайомлення учнів з поняттями «дані», «модель», «прогнозування», що є важливими складовими сучасної цифрової грамотності.

Пакети для комп'ютерної графіки, моделювання та симуляцій.

До цієї категорії належать:

- Matplotlib та Plotly – побудова дво- та тривимірних графіків;
- Pygame – створення інтерактивних програм та освітніх ігор;
- VPython – візуалізація фізичних процесів і математичних моделей у 3D;
- SimPy – моделювання дискретних систем (черги, обслуговування, рух потоків).

Такі бібліотеки надають можливість створювати наочні приклади для пояснення абстрактних математичних понять, що підвищує рівень розуміння матеріалу учнями.

Інструменти для автоматизації та розробки навчальних проєктів.

У процесі навчання Python може також використовуватись для автоматизації рутинних дій (обробка файлів, тестування знань, створення звітів). Для цього існують пакети:

- os, shutil, pathlib – для роботи з файловою системою;
- openpyxl, csv, json – для обробки навчальних даних і таблиць;
- Flask, Streamlit – для створення простих вебінтерфейсів або інтерактивних навчальних застосунків.

Використання таких бібліотек у проєктній діяльності сприяє формуванню в учнів практичних умінь, пов'язаних з обробкою інформації та розробкою цифрових освітніх продуктів.

Таким чином, класифікація пакетів Python за напрямами використання дозволяє не лише систематизувати навчальний матеріал, але й продемонструвати

студентам майбутнім учителям широкий потенціал мови Python як міждисциплінарного інструмента. Особливе значення для освітньої галузі «Середня освіта (Математика. Інформатика)» мають математичні пакети – NumPy, SciPy, SymPy, які доцільно розглядати детально як засіб формування професійних компетентностей майбутнього вчителя.

1.2. NumPy як базова бібліотека для обчислювальних задач.

1.2.1. Загальна характеристика бібліотеки NumPy.

NumPy (Numerical Python) – це одна з найважливіших бібліотек Python, призначена для ефективного виконання числових і наукових обчислень. Вона забезпечує підтримку багатовимірних масивів (об’єктів ndarray) та надає широкий спектр функцій для операцій над ними – від простих арифметичних дій до складних лінійних, статистичних і тригонометричних обчислень.

NumPy розроблена як високооптимізований інструмент, який дозволяє працювати з великими обсягами числових даних у десятки разів швидше, ніж стандартні структури Python, завдяки реалізації на мові C. Саме ця властивість робить її основою для більшості сучасних наукових бібліотек, зокрема SciPy, pandas, Matplotlib, scikit-learn тощо.

З педагогічного погляду, використання NumPy у курсах математики та інформатики має значний потенціал. Вона дозволяє:

- ілюструвати поняття масиву, матриці, вектора;
- демонструвати властивості лінійних операцій;
- виконувати обчислення у темах, що стосуються систем рівнянь, векторної алгебри, статистики;
- формувати алгоритмічне та логічне мислення учнів через роботу з даними.

1.2.2. Створення та основні операції з масивами.

Базовим елементом у NumPy є об’єкт ndarray (n-dimensional array) – багатовимірний масив. Його можна створити кількома способами. На рис. 1.6 приведено декілька способів створення масиву.

```
import numpy as np
a = np.array([1, 2, 3, 4, 5])
print(a)
b = np.array([[1, 2, 3], [4, 5, 6]])
print(b)
c=np.zeros((2, 3))   # масив із нулів
print(c)
d=np.ones((3, 3))    # масив із одиниць
print(d)
e=np.arange(0, 10, 2)  # послідовність чисел від 0 до 8 із кроком 2
print(e)
f=np.linspace(0, 1, 5) # п’ять рівновіддалених чисел між 0 і 1
print(f)
```

```
[1 2 3 4 5]
[[1 2 3]
 [4 5 6]]
[[0. 0. 0.]
 [0. 0. 0.]]
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
[0 2 4 6 8]
[0.   0.25 0.5  0.75 1.  ]
```

Рис. 1.6. Створення масиву.

1.2.3. Арифметичні операції та властивості масивів.

На відміну від списків у Python, операції між масивами NumPy виконуються поелементно, тобто кожен елемент обробляється незалежно. На рис. 1.7 приведено приклади операцій над масивами.

```python
import numpy as np

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

print(x + y)  # [5 7 9]
print(x * y)  # [ 4 10 18]
print(x ** 2) # [1 4 9]

print(x + 10)   # [11 12 13]
print(x / 2)    # [0.5 1.  1.5]

print(x.mean())  # середнє значення
print(x.std())   # стандартне відхилення
print(x.sum())   # сума елементів
```

```
[5 7 9]
[ 4 10 18]
[1 4 9]
[11 12 13]
[0.5 1.  1.5]
2.0
0.816496580927726
6
```

Рис. 1.7. Приклади операцій над масивами.

1.2.4. Лінійна алгебра в NumPy.

NumPy має вбудований модуль numpy.linalg для виконання операцій лінійної алгебри (рис. 1.8).

```python
import numpy as np
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
# Матричне множення
C = np.dot(A, B)
print(C)
# Обчислення визначника та оберненої матриці:
det = np.linalg.det(A)
inv = np.linalg.inv(A)
print(det)
print(inv)
```

```
[[19 22]
 [43 50]]
-2.0000000000000004
[[-2.   1. ]
 [ 1.5 -0.5]]
```

Рис. 1.8. Приклади операцій над матрицями.

Такі приклади є надзвичайно цінними в освітньому процесі, коли необхідно не лише навчити здобувачів виконувати дії з матрицями, а й показати, як ці обчислення реалізуються комп'ютерними засобами.

1.2.5. Візуалізація даних.

Для інтеграції з візуалізаційними інструментами, зокрема з бібліотекою Matplotlib, NumPy забезпечує швидку передачу даних у вигляді масивів (рис. 1.9).

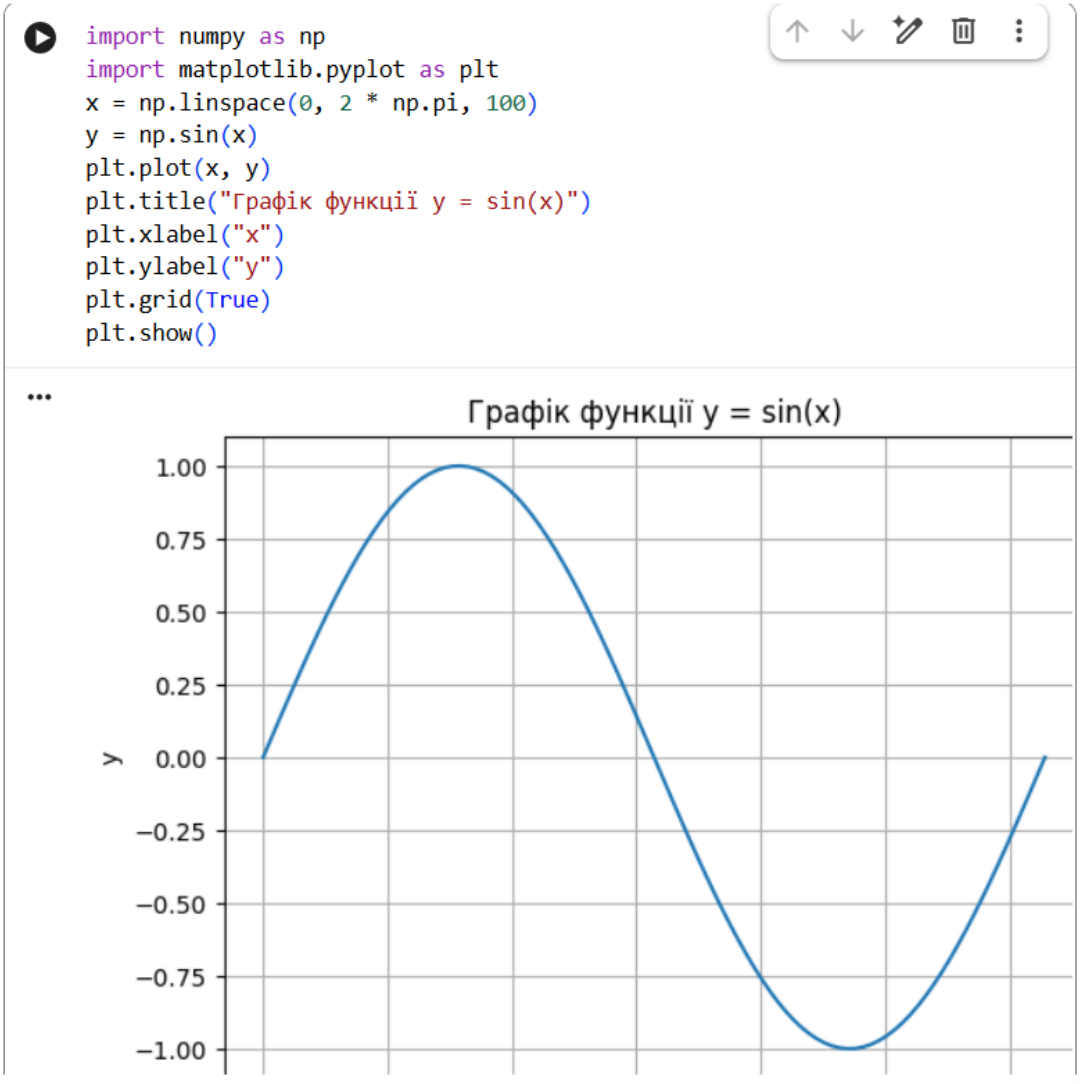

Рис. 1.9. Побудова графіка функції $y = \sin(x)$.

Застосування такого підходу на уроках математики дозволяє візуалізувати тригонометричні функції, параболи, експоненти, що сприяє кращому розумінню властивостей функцій та розвитку аналітичного мислення учнів.

1.2.6. Методичні аспекти використання NumPy в освітньому процесі.

Використання бібліотеки NumPy у шкільному курсі математики та інформатики сприяє:

- практичній спрямованості навчання – учні бачать приклад застосування математики в програмуванні;
- інтеграції дисциплін – поєднання математичних понять (матриці, функції, статистика) з інформатичними навичками (структури даних, алгоритми);
- розвитку критичного мислення – учні аналізують результати обчислень і порівнюють їх із теоретичними знаннями;
- моделюванню процесів – створенню цифрових моделей для дослідження математичних закономірностей.

Таким чином, бібліотека NumPy виступає не лише потужним інструментом наукових обчислень, але й ефективним засобом формування математичних і цифрових компетентностей учнів середньої школи. Її інтеграція в освітній процес відповідає сучасним вимогам STEM-освіти, сприяє реалізації міжпредметних зв’язків і підвищує мотивацію до вивчення як математики, так і інформатики.

1.3. Бібліотека SciPy: функціональні можливості та освітнє значення у навчанні математики.

1.3.1. Загальна характеристика.

SciPy (Scientific Python) – це бібліотека для розширених наукових і технічних обчислень, побудована на основі NumPy. Вона розширює можливості роботи з масивами, додаючи функціонал для інтегрування, оптимізації, інтерполяції, статистики, лінійної алгебри, обробки сигналів і зображень.

У навчальному контексті SciPy виконує роль інструмента, який дозволяє поєднати математичну теорію з практичними обчисленнями, сприяє розвитку аналітичного мислення та вміння застосовувати знання для розв’язання прикладних задач. Завдяки простому синтаксису та інтеграції з іншими бібліотеками (NumPy,

Matplotlib, SymPy), SciPy є придатною для використання у старших класах і на факультативах із поглибленого вивчення математики та інформатики.

Порівняно з базовими можливостями Python та NumPy, SciPy містить спеціалізовані високорівневі функції для виконання алгоритмів:

- чисельного інтегрування;
- диференціювання та розв’язування диференціальних рівнянь;
- оптимізації;
- інтерполяції даних;
- статистичного аналізу;
- обробки сигналів і зображень.

Завдяки цьому SciPy є важливим інструментом для вивчення методів прикладної математики, які входять до змісту профільного шкільного навчання та поглиблених STEM-курсів.

1.3.2. Структура та функціональні модулі SciPy.

Бібліотека складається з низки підмодулів, кожен з яких реалізує певний клас чисельних методів. Найчастіше у навчальних цілях застосовують такі модулі:

Таблиця 1.1.

Функціональні модулі SciPy

Підмодуль	**Призначення**
scipy.integrate	Чисельне інтегрування, розв’язування диференціальних рівнянь
scipy.optimize	Пошук мінімумів/максимумів функцій, розв’язування рівнянь та систем
scipy.interpolate	Інтерполяція та апроксимація експериментальних даних
scipy.stats	Статистичні розподіли, перевірка гіпотез, параметричні та непараметричні методи
scipy.linalg	Розширені алгоритми лінійної алгебри, розклад матриць
scipy.signal	Обробка сигналів, фільтрація, перетворення Фур’є

Наявність такої функціональної структури дозволяє використовувати SciPy як навчальний інструмент для демонстрації прикладних розрахунків, які важко або неможливо реалізувати на дошці під час уроків.

1.3.3. Приклади використання SciPy у навчанні математики.

Модуль integrate використовується для чисельного інтегрування. На рис. 1.10 приведено результат апроксимації відомого аналітичного значення

$$\int_0^{\pi} \sin^2(x)\,dx = \pi/2.$$

```
import numpy as np
from scipy import integrate

f = lambda x: np.sin(x) ** 2
result, error = integrate.quad(f, 0, np.pi)
print("∫ sin²(x) dx від 0 до π =", result)
```

```
∫ sin²(x) dx від 0 до π = 1.5707963267948966
```

Рис. 1.10. Чисельне інтегрування.

Педагогічний ефект полягає у можливості порівняння чисельних та аналітичних методів інтегрування.

На рис. 1.11 приведено використання модуля optimize для пошуку мінімуму функції.

```
from scipy import optimize
f = lambda x: (x - 3)**2 + 2
result = optimize.minimize(f, x0=0)
print("Мінімум функції:", result.fun)
```

```
Мінімум функції: 2.000000000000001
```

Рис. 1.11. Мінімум функції.

Використання модуля optimize під час дослідження функції на максимум/мінімум дозволяє учням опанувати відповідні цифрові інструменти. Учні можуть експериментувати зі зміною початкового наближення x_0, аналізуючи, як це впливає на швидкість і точність розв'язку.

Наступний приклад демонструє розв'язання системи лінійних рівнянь (рис. 1.12).

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

```python
from scipy.linalg import solve
import numpy as np

A = np.array([[2, 1],
              [1, -1]])
b = np.array([5, 1])

x = solve(A, b)
print(x)
```

```
[2. 1.]
```

Рис. 1.12. Розв’язування системи рівнянь.

Цей фрагмент коду може бути використаний на уроці для демонстрації того, як комп’ютер реалізує відомі методи розв’язування систем рівнянь.

Для статистичного аналізу даних використовується модуль stats (рис. 1.13).

```python
from scipy import stats
import numpy as np

data = np.array([2, 4, 4, 4, 5, 5, 7, 9])

mean = np.mean(data)
mode = stats.mode(data)
std = np.std(data)

print("Середнє:", mean)
print("Мода:", mode.mode)
print("Стандартне відхилення:", std)
```

```
Середнє: 5.0
Мода: 4
Стандартне відхилення: 2.0
```

Рис. 1.13. Характеристики вибірки.

Модуль stats доцільно застосовувати при вивченні теорії ймовірності та математичної статистики.

Інтерполяція та апроксимація. У математичному моделюванні часто виникає потреба побудови функції, яка проходить через задані точки – інтерполяційна функція. SciPy містить модуль interpolate для таких задач (рис. 1.14).

```
from scipy import interpolate
import numpy as np
import matplotlib.pyplot as plt
x = np.array([0, 1, 2, 3, 4])
y = np.array([1, 3, 2, 5, 7])
f = interpolate.interp1d(x, y, kind='cubic')
x_new = np.linspace(0, 4, 100)
y_new = f(x_new)
plt.plot(x, y, 'o', x_new, y_new, '-')
plt.title("Інтерполяція даних (кубічний сплайн)")
plt.show()
```

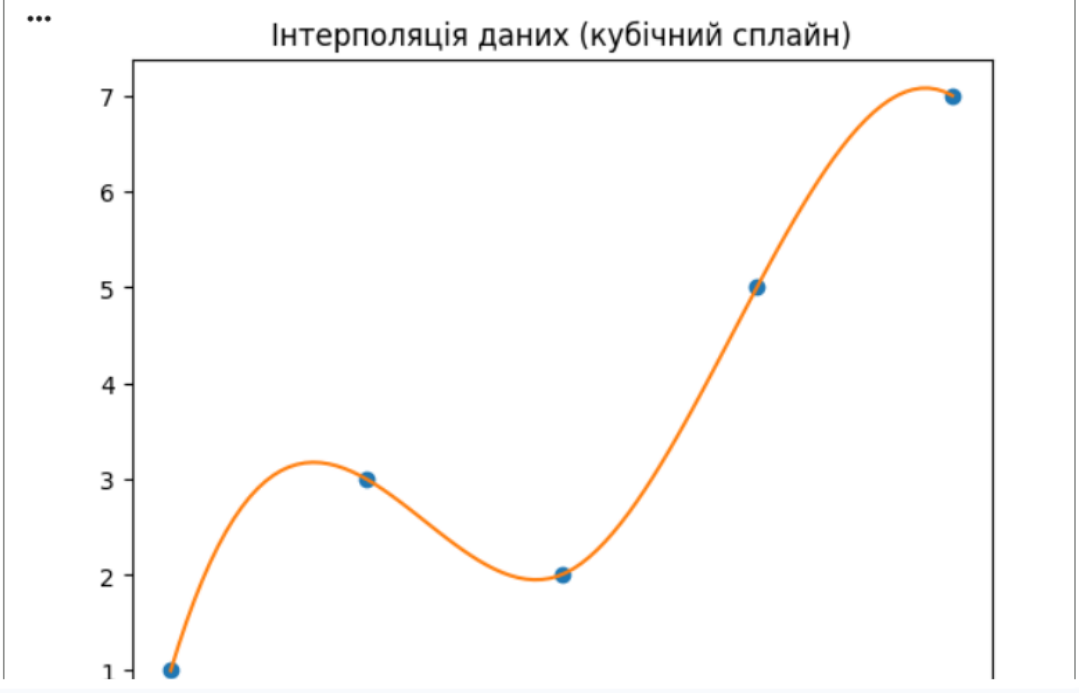

Рис. 1.14. Побудова інтерполяційного полінома.

Учні можуть бачити, як на основі кількох емпіричних даних формується неперервна функція – це наочна демонстрація методів наближеного моделювання.

1.3.4. Методичний потенціал використання SciPy у шкільній освіті.

Інтеграція бібліотеки SciPy у процес вивчення математики та інформатики має значний практико-орієнтований потенціал:

- дозволяє ілюструвати чисельні методи, які часто розглядаються лише теоретично;
- сприяє розвитку дослідницьких умінь – учні можуть моделювати процеси, порівнювати теоретичні та експериментальні результати;
- підтримує проектну діяльність – наприклад, створення міні-моделей руху тіла, коливань, зростання популяцій тощо;
- формує алгоритмічне мислення через побудову послідовностей обчислювальних кроків.

Особливо корисним є застосування SciPy у міжпредметних проєктах, які поєднують фізику, інформатику та математику. Наприклад, розрахунок траєкторії руху об’єкта з урахуванням сили опору або апроксимація експериментальних даних у лабораторній роботі з фізики.

У табл. 1.2 приведено можливості застосування SciPy у процесі викладання математики.

Таблиця 1.2.

Педагогічний ефект застосування SciPy.

Освітній результат	Педагогічний ефект
Формування дослідницьких умінь	учні навчаються ставити гіпотези, перевіряти їх, формувати висновки
Міжпредметна інтеграція математики та інформатики	завдання поєднують алгоритмічне мислення з математичним аналізом
Візуалізація та моделювання	можна застосовувати графічні засоби для інтерпретації результатів
Підготовка до STEM/олімпіадної діяльності	SciPy використовується у реальних наукових дослідженнях

Отже, бібліотека SciPy є важливим інструментом сучасного комп’ютерно орієнтованого вивчення математики. Вона дозволяє реалізувати прикладний, дослідницький і STEM-орієнтований підходи до навчання, розширює можливості шкільного курсу, підвищує мотивацію та сприяє формуванню компетентностей,

передбачених Новою українською школою та Концепцією розвитку математичної освіти.

1.4. Бібліотека SymPy для виконання символьних обчислень.

1.4.1. Загальна характеристика SymPy.

SymPy – це бібліотека Python для виконання символьних (аналітичних) математичних обчислень. Вона забезпечує широкий спектр операцій, зокрема алгебраїчні перетворення, спрощення виразів, диференціювання, інтегрування та аналітичне розв'язування рівнянь різних типів. На відміну від числових пакетів, таких як NumPy чи SciPy, що працюють із числами з плаваючою комою, SymPy оперує математичними виразами у символічному вигляді: рівняннями, змінними, поліномами, інтегралами, матрицями тощо.

Застосування SymPy дає можливість поєднувати традиційні математичні методи з інструментами комп'ютерної алгебри, що сприяє формуванню в учнів навичок математичного моделювання (рис. 1.15).

```
from sympy import symbols, Eq, solve, diff, integrate, sin, pi

x = symbols('x')
expr = x**2 + 2*x + 1
derivative = diff(expr, x)
integral = integrate(sin(x), (x, 0, pi))

print("Похідна:", derivative)
print("Інтеграл sin(x) від 0 до π =", integral)
```

```
Похідна: 2*x + 2
Інтеграл sin(x) від 0 до π = 2
```

Рис. 1.15. Приклад використання бібліотеки SymPy.

SymPy може бути ефективно використана під час вивчення алгебри, математичного аналізу та елементів геометрії. Зокрема, під час опрацювання теми «Похідна функції» учні мають можливість спостерігати процес автоматичного диференціювання та порівнювати його з ручними обчисленнями. Такий підхід

підвищує мотивацію, сприяє усвідомленню алгоритмічної природи диференціювання та підтримує розвиток дослідницького мислення.

Бібліотека написана повністю на Python і не потребує зовнішніх джерел, що робить її доступною для використання на освітніх ноутбуках з обмеженими ресурсами.

Встановлення SymPy:

- Через pip

pip install sympy

- У середовищі Anaconda

conda install sympy

SymPy можна розглядати як альтернативу *Mathematica*, *Maple* або *MATLAB Symbolic Toolbox*.

1.4.2. Ключові можливості SymPy.

Символьна алгебра. SymPy дозволяє створювати символьні змінні (symbols, Symbol), виконувати перетворення алгебраїчних виразів, розкладати на множники, спрощувати вирази (рис. 1.16).

```
from sympy import symbols, expand, factor, simplify

x, y = symbols('x y')

expr = (x + y)**3
print(expand(expr))       # розкриття дужок
print(factor(x**2 - 9))   # розклад на множники
print(simplify((x**2 - 1)/(x - 1)))
```

```
x**3 + 3*x**2*y + 3*x*y**2 + y**3
(x - 3)*(x + 3)
x + 1
```

Рис. 1.16. Алгебраїчні перетворення.

Розв’язування рівнянь і систем. SymPy надає функції solve() – для алгебраїчних рівнянь, solveset() – новіший інтерфейс, solve_univariate_inequality – для нерівностей (рис. 1.17).

```
from sympy import symbols, Eq, solve

x = symbols('x')
solution = solve(Eq(x**2 - 4, 0), x)
print(solution)
```

```
[-2, 2]
```

Рис. 1.17. Розв’язування рівняння.

Диференціювання та інтегрування. SymPy підтримує похідні будь-якого порядку, визначені та невизначені інтеграли, границі функцій (рис. 1.18).

```
from sympy import symbols, diff, integrate, sin, cos

x = symbols('x')
print(diff(sin(x)*x**2, x))          # похідна
print(integrate(cos(x), x))          # інтеграл
print(integrate(x**2, (x, 0, 2)))    # визначений інтеграл
```

```
x**2*cos(x) + 2*x*sin(x)
sin(x)
8/3
```

Рис. 1.18. Символьне диференціювання та інтегрування.

Робота з матрицями. SymPy має власний об’єкт Matrix із підтримкою визначників, обернених матриць, рангу, власних значень (рис. 1.19).

```
from sympy import Matrix

A = Matrix([[1, 2], [3, 4]])
print(A.det())
print(A.inv())
print(A.eigenvals())
```

```
-2
Matrix([[-2, 1], [3/2, -1/2]])
{5/2 - sqrt(33)/2: 1, 5/2 + sqrt(33)/2: 1}
```

Рис. 1.19. Робота з матрицями.

Символьні ряди, поліноми, комбінації. SymPy дозволяє обчислювати ряди Тейлора (series()), поліноми та їх властивості (Poly()), комбінаторні об’єкти (перестановки, факторіали, числа Белла тощо).

Можливість числових обчислень. SymPy має функцію evalf(), що дозволяє отримувати числову апроксимацію (рис. 1.20).

```
from sympy import pi
print(pi.evalf(50))    # число π з 50 знаками
```

```
3.1415926535897932384626433832795028841971693993751
```

Рис. 1.20. Число π.

1.4.3. Переваги SymPy в освітньому середовищі.

Відкритість і доступність. SymPy безкоштовна, з відкритим кодом та легко встановлюється, що дозволяє застосовувати її на шкільних та університетських комп’ютерах без додаткових ліцензій.

Інтерактивність навчання. У поєднанні з Jupyter Notebook SymPy дозволяє демонструвати учням покрокові виведення, поєднувати код, формули та пояснення, показувати, як алгебра чи аналіз працюють у вигляді алгоритмів.

Розвиток алгоритмічного мислення. Учні бачать, як математичні операції реалізуються комп’ютером у символьній алгебрі, системах рівнянь, похідній й інтегралах. Це сприяє формуванню STEM-компетентностей здобувачів.

Застосування у методиці навчання. SymPy корисна для пояснення алгоритмізації обчислювальних процесів; для демонстрації складних математичних перетворень; як інструмент перевірки правильності розв’язань; для створення інтерактивних лабораторних робіт. Учні можуть досліджувати, а не просто обчислювати.

Використання математичних бібліотек Python у закладах загальної середньої освіти має значну педагогічну цінність, оскільки сприяє розвитку інформаційної культури учнів; підсилює міжпредметні зв’язки між математикою, фізикою,

інформатикою; створює умови для проєктно-дослідницької діяльності; забезпечує диференціацію навчання та індивідуальні траєкторії.

Таким чином, Python є цінним засобом формування математичної грамотності та цифрових компетентностей.

Математичні бібліотеки Python – NumPy, SciPy та SymPy – є ефективним інструментом підвищення якості математичної освіти. Їх використання дозволяє візуалізувати складні математичні поняття; моделювати реальні процеси; формувати дослідницький стиль мислення; організовувати STEM-проєкти та інші інноваційні форми навчальної діяльності.

Застосування цих бібліотек сприяє формуванню інформаційно-математичної компетентності та підвищує рівень підготовки учнів до майбутньої діяльності у STEM-сфері.

1.5. Порівняльна характеристика бібліотек SymPy, NumPy та SciPy.

У екосистемі Python для математичних обчислень найбільш поширеними є три бібліотеки: SymPy, NumPy та SciPy. Хоча вони належать до спільного інструментарію наукового програмування, їх функціональні можливості, методологічні підходи та сфери застосування істотно відрізняються. Нижче наведено систематизовану характеристику їхніх особливостей у контексті навчального процесу та наукових обчислень.

1.5.1. Концептуальні засади та призначення.

SymPy є бібліотекою для символьної математики, яка оперує математичними об’єктами у вигляді символічних виразів. Вона забезпечує аналітичні обчислення: спрощення виразів, розв’язування рівнянь, обчислення похідних, інтегралів, границь тощо. Результати подаються у формі точних математичних виразів, а не наближених числових значень.

NumPy – основна бібліотека Python для числових обчислень з масивами. Її структура базується на багатовимірних масивах (ndarray), що оптимізовані для

високої швидкості, у тому числі за рахунок реалізації на C та векторизованих операцій. Вона не виконує аналітичних обчислень і працює виключно із числовими значеннями.

SciPy є надбудовою над NumPy, що призначена для наукових і інженерних обчислень. Вона містить числові методи для оптимізації, інтегрування, статистики, лінійної алгебри, інтерполяції, перетворень Фур'є, диференціальних рівнянь та інших задач.

1.5.2. Рівень математичної точності.

SymPy демонструє *аналітичну точність*:

- похідна від $\sin(x)$ залишається $\cos(x)$;
- інтеграл від $\frac{1}{x}$ – $\ln|x| + C$;
- рівняння розв'язуються у символічній формі.

NumPy та SciPy працюють з числовими апроксимаціями, обмеженими точністю машинного подання (числа з плаваючою комою).

Це дає різні наслідки:

- похідні обчислюються через різницеві схеми;
- інтегралами керує чисельне інтегрування;
- рівняння розв'язуються наближеними методами.

1.5.3. Продуктивність.

За швидкістю виконання операцій:

- NumPy – найшвидший для роботи з масивами та векторизованими обчисленнями; його часто використовують у машинному навчанні, обробці сигналів та моделюванні.
- SciPy також швидкий, але залежить від складності методу (наприклад, чисельне інтегрування може бути ресурсоємним).
- SymPy істотно повільніший, оскільки маніпулює символьними структурами, а не числами.

У задачах, де швидкість критична, SymPy зазвичай не застосовується. У табл. 1.3 приведено коротку порівняльну характеристику бібліотек.

Таблиця 1.3.

Порівняння бібліотек SymPy, NumPy, SciPy

Характеристика	SymPy	NumPy	SciPy
Тип обчислень	Символьні	Числові	Числові (наукові методи)
Точність	Абсолютна (аналітична)	Обмежена floating-point	Обмежена floating-point
Швидкість	Низька	Висока	Висока–середня
Сфера застосування	Алгебра, аналіз, теорія	Дані, масиви, лінійна алгебра	Інтегрування, диференціальні рівняння, оптимізація
Підходить для школи	Так	Так	Так (старша школа/університет)
Потреба у зовнішніх залежностях	Немає	Мінімальні	Залежить від модулів

1.5.4. Використання в освіті.

SymPy корисний для демонстрації математичних перетворень; автоматичної перевірки задач; ілюстрації аналітичних методів; підтримки навчання алгебри та аналізу.

NumPy і SciPy корисні для курсів прикладної математики; моделювання фізичних процесів; обробки експериментальних даних; статистики та математичного моделювання.

У STEM-освіті ці бібліотеки часто використовуються разом: SymPy – для аналітики, NumPy – для числової реалізації, SciPy – для дослідження складних моделей.

РОЗДІЛ 2
ВИКОРИСТАННЯ PYTHON НА УРОКАХ МАТЕМАТИКИ В ЗАКЛАДАХ СЕРЕДНЬОЇ ОСВІТИ

2.1. Теоретичні засади інтеграції математики та інформатики.

Python посідає важливе місце в сучасній математичній освіті завдяки своїй універсальності, доступності та широкому спектру спеціалізованих бібліотек. Для вчителя математики Python виступає ефективним інструментом візуалізації абстрактних понять, автоматизації обчислень та моделювання математичних процесів, що сприяє глибшому розумінню учнями ключових тем алгебри, аналізу та статистики. Використання таких бібліотек, як NumPy, SymPy та SciPy, дозволяє продемонструвати взаємозв'язок між математичними структурами та їх комп'ютерною реалізацією, тим самим формуючи алгоритмічне мислення та STEM-компетентності. Python забезпечує можливість створення інтерактивних завдань, дослідницьких проєктів і візуальних експериментів, що значно підвищує мотивацію учнів і сприяє розвитку навичок самостійного аналізу, критичного мислення та математичного моделювання. Таким чином, Python перетворюється на сучасний педагогічний інструмент, який поєднує традиційні методи навчання математики з цифровими технологіями XXI століття.

Ефективне впровадження Python можливе за дотримання таких умов:

- Поступове введення інструментів від простих команд до бібліотек.
- Використання Jupyter Notebook як інтерактивного зошита – комбінації тексту, формул, графіків та коду.
- Створення проблемних ситуацій, у яких цифровий інструмент дає перевагу над рутинними обчисленнями.
- Формування цифрової та математичної компетентностей, у відповідності до Державного стандарту базової і профільної освіти.

- Використання тих можливостей Python, які учень може засвоїти на рівні середньої школи.

У табл. 2.1 приведені компетентності, які формуються у здобувачів середньої освіти засобами Python.

Таблиця 2.1.

Компетентності, що формуються засобами Python

Компетентність	Можливості Python
Математичне моделювання	Створення моделей функцій, чисельні експерименти, наближення
Алгебраїчне мислення	Символьні перетворення у SymPy
Геометрична компетентність	Побудова графіків, аналіз похідних та кривих
Статистична грамотність	Аналіз даних, гістограми, середнє, медіана, дисперсія
Алгоритмічне мислення	Написання програм для обчислень
Робота з даними	Масиви NumPy, табличні структури
Інженерне мислення	Моделювання реальних ситуацій (рух, приріст, економіка)

Отже, Python є ефективним засобом підтримки шкільного курсу математики, зокрема для візуалізації, моделювання та дослідження математичних об’єктів. Його використання сприяє формуванню ключових компетентностей НУШ та розвитку мислення учнів, наближаючи навчання до реальної наукової діяльності.

2.2. Використання Python як інструменту вчителя математики.

Розглянемо низку задач, де, на нашу думку, доцільно використовувати Python для виконання рутинних задач, обчислення проміжкових результатів, а також візуалізації даних. У цьому випадку Python виступає як інструмент вчителя математики, що дозволяє йому зосередитися на більш складних проблемах, а учням – ефективніше використовувати аудиторну роботу.

Приклад 1. Розглянемо розв’язання задачі на роботу:

Увесь басейн наповнюється водою через першу трубу на a *год* швидше, ніж через другу. Через скільки годин буде наповнений увесь басейн через першу трубу, якщо при одночасному відкритті обох труб він наповнюється за b *год*?

Розв’язання. Запишемо коротку умову задачі:

	Час, *год*	Робота, *бас.*	Продуктивність, *бас./год*
I труба	x	1	$\frac{1}{x}$
II труба	$x+a$	1	$\frac{1}{x+a}$

Згідно умови задачі отримуємо дробово-раціональне рівняння:

$$\frac{1}{x}+\frac{1}{x+a}=\frac{1}{b}. \qquad (2.1)$$

Зауважимо, що розв’язання рівняння (2.1) вимагає відносно багато часу, у той же час задачі на роботу, на нашу думку, у першу чергу орієнтовані на аналіз умови та складання математичної моделі – рівняння, нерівності чи їх системи. Отже, вчитель може використати Python для знаходження розв’язку рівняння – технічної частини завдання, а зекономлений час використати для більш інтелектуальної роботи – розроблення математичної моделі задачі.

Нехай $a=1;\ b=1{,}2$. Тоді розв’язок рівняння отримаємо за допомогою коду:

Лістинг 2.1. Розв’язання рівняння (2.1).

```
from sympy import symbols, Eq, solve
# Задання змінних
x = symbols('x')
a = 1
b = 1.2
# Рівняння: 1/x + 1/(x+a) = 1/b
equation = Eq(1/x + 1/(x + a), 1/b)
# Розв’язання рівняння
solutions = solve(equation, x)
print("Розв’язок рівняння:", solutions)
```

У результати виконання коду отримаємо два розв’язки

$$x_1 = -0{,}6; x_2 = 2.$$

Оскільки $x_1 < 0$, то умову задачі задовольняє лише $x_2 = 2$.

Приклад 2. Розглянемо розв’язання задачі на знаходження ламаної найменшої довжини:

Знайдіть абсцису точки B на осі Ox, якщо $A(0; a), C(1; c)$, а ламана ABC має найменшу довжину.

Приведену задачу можна розв’язати двома способами:

- з використанням апарату мінімізації функції (довжини ламаної);
- з використанням нерівності трикутника.

У першому випадку довжина ламаної виражається формулою

$$L(x) = |AB| + |BC| = \sqrt{(x-0)^2 + (0-a)^2} + \sqrt{(1-x)^2 + (c-0)^2}, \qquad (2.2)$$

де $B(x; 0)$.

У цьому випадку знаходження похідної, критичних точок та аналізу функції $L(x)$ є досить громіздким процесом, тому використаємо для технічних обчислень бібліотеку numpy. Нехай $a = 2, c = 3$. Тоді наступний код дозволяє знайти точку мінімуму функції (2.2), тобто розв’язання задачі.

Лістинг 2.2. Точка мінімуму функції (2.2).

```
import numpy as np
from scipy.optimize import minimize_scalar
# Точки
A = (0, 2)
C = (1, 3)
def L(x):
    B = (x, 0)
    # відстані AB та BC
    AB = np.hypot(B[0] - A[0], B[1] - A[1])
    BC = np.hypot(C[0] - B[0], C[1] - B[1])
    return AB + BC
# мінімізуємо по x у розумних межах
res = minimize_scalar(L, bounds=(-10, 10), method='bounded')
print("Мінімальне значення L =", res.fun)
print("Координата x точки B =", res.x)
```

У результати виконання коду отримаємо $x = 0.4$.

Використаємо 2-й підхід для наочної ілюстрації отриманого результату (рис. 2.1). Візьмемо точку $A_1(1;-2)$, яка симетрична до $A(1;2)$ відносно осі Ox. Оскільки довжини ламаних ABC та A_1BC співпадають, то найменша довжина ламаної A_1BC і, відповідно, ABC досягається, якщо B – точка перетину прямої A_1C та осі Ox (це випливає із нерівності трикутника). Знаючи координати точок A_1 та C, легко побудувати рівняння прямої A_1C: $y = 5x - 2$. Звідси, при $y = 0$, отримуємо $x = 0{,}4$.

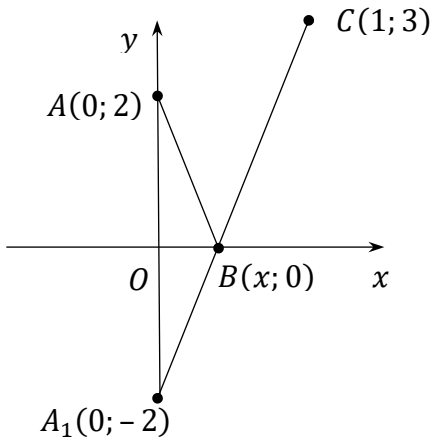

Рис. 2.1. Геометричне розв’язання задачі.

На кінець, використання бібліотеки numpy дозволяє також побудувати криву (2.2).

Лістинг 2.3. Точка мінімуму функції (2.2).

```
import numpy as np
import matplotlib.pyplot as plt
# Точки
A = (0, 2)
C = (1, 3)
# Довжина ламаної L(x)
def L(x):
    B = (x, 0)
    AB = np.hypot(B[0] - A[0], B[1] - A[1])
    BC = np.hypot(C[0] - B[0], C[1] - B[1])
    return AB + BC
```

```
# Діапазон x
xs = np.linspace(-5, 6, 400)
Ls = [L(x) for x in xs]
# Побудова графіка
plt.plot(xs, Ls)
plt.xlabel("x")
plt.ylabel("L(x)")
plt.title("Графік довжини ламаної L(x)")
plt.grid(True)
plt.show()
```

Рис. 2.2. Графік функції $L(x)$.

Приклад 3. Розглянемо розв'язання задачі підвищеного рівня складності на систему рівнянь:

Функції $u(x)$ та $v(x)$ для всіх дійсних значень x задовольняють систему рівнянь

$$\begin{cases} a_1 u(x) + b_1 v(x) = c_1 x + d_1, \\ a_2 u(x) + b_2 v(x) = c_2 x + d_2. \end{cases}$$

Обчисліть значення $u(1) \cdot v(1)$.

Дана задача зводиться до звичайної системи рівнянь, якщо замість x підставити значення 1. Проте її можна розв'язати у загальному вигляді, наприклад, методом підстановки. Нехай

$$a_1 = 2; b_1 = 1; c_1 = 3; d_1 = 4;$$
$$a_2 = 1; b_2 = -2; c_2 = 4; d_2 = -3.$$

Тоді систем рівнянь буде мати вигляд:

$$\begin{cases} 2u(x) + v(x) = 3x + 4, \\ u(x) - 2v(x) = 4x - 3.' \end{cases} \quad (2.3)$$

Для розв'язання системи (2.3) використаємо бібліотеку sympy.

Лістинг 2.4. Розв'язання системи рівнянь.

```
import sympy as sp
x = sp.symbols('x')
u, v = sp.symbols('u v')
solution = sp.solve([
    2*u + v - (3*x + 4),
    u - 2*v - (4*x - 3)
], [u, v])
print(solution)
```

У результати виконання коду отримаємо розв'язок системи

$$u(x) = 2x + 1; v(x) = 2 - x.$$

Тоді $u(1) \cdot v(1) = (2 \cdot 1 + 1) \cdot (2 - 1) = 3$.

Приклад 4. Розглянемо розв'язання рівняння з модулем

$$|x - a| + |x - b| = c,$$

де a, b, c – деякі числа ($a < b$).

Використовуючи метод проміжків (рис. 2.3), дане рівняння можна розглянути як сукупність трьох систем (2.4).

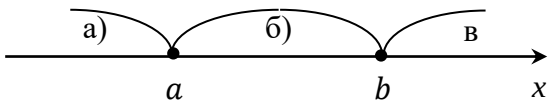

Рис. 2.3. Метод проміжків розв'язування рівняння з модулем.

$$\begin{cases} x < a, \\ -(x-a)-(x-b)=c; \end{cases} \quad \begin{cases} x \in [a;b], \\ x-a-(x-b)=c; \end{cases} \quad \begin{cases} x > b, \\ x-a+x-b=c. \end{cases} \tag{2.4}$$

Проте, найкраще учні зрозуміють суть методу на основі графіку функції

$$y = |x-a| + |x-b|.$$

Тому використання Python для побудови графіка дозволить як зекономи час, так і продемонструвати різні варіанти розв'язання задачі, залежно від параметра c. Розглянемо декілька варіантів.

1) $a = -1, b = 1, c = 6$, тобто рівняння має вигляд:

$$|x+1| + |x-1| = 6.$$

Побудуємо графік, використавши код:

Лістинг 2.5. Побудова графіку функції $y = |x+1| + |x-1|$.

```
import sympy as sp
# Оголошуємо символ
x = sp.symbols('x')
# Задаємо функцію
y = abs(x + 1) + abs(x - 1)
# Побудова графіка
sp.plot(
    y,
    (x, -5, 5),
    xlabel="x",
    ylabel="y",
    title="Графік y = |x+1| + |x-1|")
```

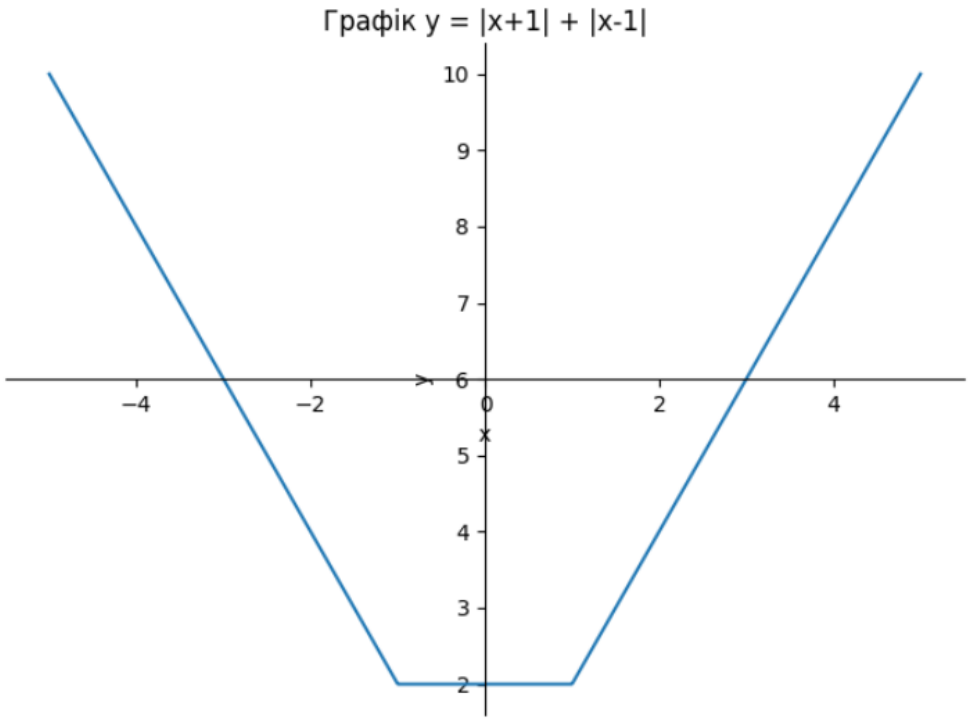

Рис. 2.4. Графік функції, побудований за допомогою бібліотеки sympy.

На основі графіку (рис. 2.4) учні розуміють, що рівняння

$$|x + 1| + |x - 1| = 6$$

має 2 розв’язки. Використовуючи сукупність (2.4) знаходимо:

$$x_1 = -3; x_2 = 3.$$

Графік на рис. 2.4 дозволяє наочно продемонструвати розв’язки рівняння. Тобто використовуючи Python вчитель має інструмент для візуалізації, а також має можливість зекономити час при побудові графіку.

2) $a = -1, b = 1, c = 2$, тобто рівняння має вигляд:

$$|x + 1| + |x - 1| = 2.$$

У цьому випадку розв’язками рівняння є проміжок $x \in [-1; 1]$, що також зручно продемонструвати за допомогою рис. 2.4.

3) $a = -1, b = 1, c = 1$, тобто рівняння має вигляд:

$$|x + 1| + |x - 1| = 1.$$

У цьому випадку рівняння не має розв'язків. Використання рис. 2.4 підтверджує відсутність точок перетину графіків функцій $y = |x + 1| + |x - 1|$ та $y = 1$.

4) $a = -1, b = 1$, тобто розглядається рівняння з параметром c:

$$|x + 1| + |x - 1| = c.$$

Використання попередніх випадків (1-3), а також графіка на рис. 2.4 дозволяє знайти розв'язки сукупності (2.4), залежно від параметра c:

якщо $c < 2$, то рівняння розв'язків не має;

якщо $c = 2$, то розв'язками рівняння є проміжок $x \in [-1; 1]$;

якщо $c > 2$, то рівняння має 2 розв'язки

$$x_1 = -\frac{c}{2}; x_2 = \frac{c}{2}.$$

Приклад 5. Розглянемо також використання бібліотеки math для розв'язання трикутників. У залежності від заданих даних, можна розглянути декілька підзадач:

- дано: три сторони a, b, c трикутника, знайти: три кути A, B, C трикутника;
- дано: дві сторони a, b та кут C трикутника, знайти: сторону c та кути A, B трикутника;
- дано: сторону a та кути B, C трикутника, знайти: сторони b, c та кут A трикутника.

Для розв'язання цих підзадач використовуються теореми синусів та косинусів:

$$\frac{a}{\sin\alpha} = \frac{b}{\sin\beta} = \frac{c}{\sin\gamma}; \quad (2.5)$$

$$c^2 = a^2 + b^2 - 2ab \cdot \cos\gamma. \quad (2.6)$$

Зауважимо, якщо синуси чи косинуси не табличні, то їхнє обчислення вимагає застосування інженерних калькуляторів, тому доцільно використати програму на Python, яка дозволить перевірити правильність обчислень.

Лістинг 2.6. Розв'язання трикутників.

```
import math
def solve_triangle(a=None, b=None, c=None, A=None, B=None, C=None):
    def d2r(x): return x * math.pi / 180
```

```
def r2d(x): return x * 180 / math.pi
# --- FULL-ANGLE CASE: if 2 angles known ---
if A and B and not C:
    C = 180 - A - B
if A and C and not B:
    B = 180 - A - C
if B and C and not A:
    A = 180 - B - C
# If A+B+C != 180 -> no triangle
if A and B and C and abs(A + B + C - 180) > 1e-6:
    return "Кути не утворюють трикутник."
# --- SSS ---
if a and b and c:
    A = r2d(math.acos((b*b + c*c - a*a) / (2*b*c)))
    B = r2d(math.acos((a*a + c*c - b*b) / (2*a*c)))
    C = 180 - A - B
# --- SAS ---
elif a and b and C:
    C_rad = d2r(C)
    c = math.sqrt(a*a + b*b - 2*a*b*math.cos(C_rad))
    A = r2d(math.asin(a * math.sin(C_rad) / c))
    B = 180 - A - C
elif a and c and B:
    B_rad = d2r(B)
    b = math.sqrt(a*a + c*c - 2*a*c*math.cos(B_rad))
    A = r2d(math.asin(a * math.sin(d2r(B)) / b))
    C = 180 - A - B
elif b and c and A:
    A_rad = d2r(A)
    a = math.sqrt(b*b + c*c - 2*b*c*math.cos(A_rad))
    B = r2d(math.asin(b * math.sin(d2r(A)) / a))
    C = 180 - A - B
# --- ASA / AAS (fixed) ---
elif A and B and a:
    A_rad = d2r(A); B_rad = d2r(B); C_rad = d2r(180 - A - B)
    b = a * math.sin(B_rad) / math.sin(A_rad)
    c = a * math.sin(C_rad) / math.sin(A_rad)
elif A and C and a:
    A_rad = d2r(A); C_rad = d2r(C); B_rad = d2r(180 - A - C)
    b = a * math.sin(B_rad) / math.sin(A_rad)
    c = a * math.sin(C_rad) / math.sin(A_rad)
elif B and C and b:
    B_rad = d2r(B); C_rad = d2r(C); A_rad = d2r(180 - B - C)
```

```
        a = b * math.sin(A_rad) / math.sin(B_rad)
        c = b * math.sin(C_rad) / math.sin(B_rad)

    elif A and C and b:
        A_rad = d2r(A); C_rad = d2r(C); B_rad = d2r(180 - A - C)
        a = b * math.sin(A_rad) / math.sin(B_rad)
        c = b * math.sin(C_rad) / math.sin(B_rad)
    elif A and B and c:
        A_rad = d2r(A); B_rad = d2r(B); C_rad = d2r(180 - A - B)
        a = c * math.sin(A_rad) / math.sin(C_rad)
        b = c * math.sin(B_rad) / math.sin(C_rad)
    # --- SSA ambiguous case ---
    elif a and b and A:
        A_rad = d2r(A)
        h = b * math.sin(A_rad)
        solutions = []
        if a < h:
            return "Немає розв'язків."
        # 1st triangle
        B1 = r2d(math.asin(b * math.sin(A_rad) / a))
        C1 = 180 - A - B1
        c1 = a * math.sin(d2r(C1)) / math.sin(A_rad)
        solutions.append((a, b, c1, A, B1, C1))
        # 2nd triangle
        B2 = 180 - B1
        C2 = 180 - A - B2
        if C2 > 0:
            c2 = a * math.sin(d2r(C2)) / math.sin(A_rad)
            solutions.append((a, b, c2, A, B2, C2))
        return solutions
    # --- compute area ---
    A_rad = d2r(A)
    area = 0.5 * b * c * math.sin(A_rad)
    return {
        "a": a, "b": b, "c": c,
        "A": A, "B": B, "C": C,
        "Площа": area
    }
# ----------------------------
# Приклад використання:
result = solve_triangle(c=10, A=53, B=37)
print(result)
```

Лістинг 2.6 демонструє розв'язання трикутника, якщо сторона $c = 10$, а кути $A = 53°, B = 37°$. Тоді сторони $a \approx 8; b \approx 6$, а кут $C = 90°$, тобто задано прямокутний трикутник. Також програма знаходить площу $S \approx 24$ трикутника.

2.3. Використання Python у середовищі Microsoft Excel 365 при виконанні лабораторних робіт з фізики.

Сучасний освітній процес потребує впровадження цифрових інструментів, які сприяють розвитку в учнів навичок аналізу даних, алгоритмічного мислення та міжпредметної інтеграції. Одним із таких інструментів є Python у середовищі Microsoft Excel 365, що поєднує можливості електронних таблиць і мови програмування.

Функціонал Python in Excel дозволяє виконувати математичні обчислення, опрацьовувати експериментальні дані та аналізувати результати, наприклад, фізичних вимірювань без встановлення додаткового програмного забезпечення. Учні можуть працювати з табличними даними, застосовувати фізичні формули у програмному вигляді та порівнювати результати, отримані за допомогою стандартних засобів Excel і Python.

Приклад 1. За заданим законом руху $s = x(t)$ знайдіть $v(t_0)$.

І спосіб. Для знаходження миттєвої швидкості знайдемо похідну:

$$v(t) = \frac{dx}{dt}. \tag{2.7}$$

Нехай

$$x(t) = 2t^3 - 5t^2 + 4t, t_0 = 2. \tag{2.8}$$

Використаємо бібліотеку sympy.

Лістинг 2.7. Знаходження миттєвої швидкості.

```
import sympy as sp
t = sp.symbols('t')
x = 2*t**3 - 5*t**2 + 4*t
v = sp.diff(x, t)             # похідна
v_t0 = v.subs(t, xl("B1"))    # швидкість у t0 = 2 с
float(v_t0)
```

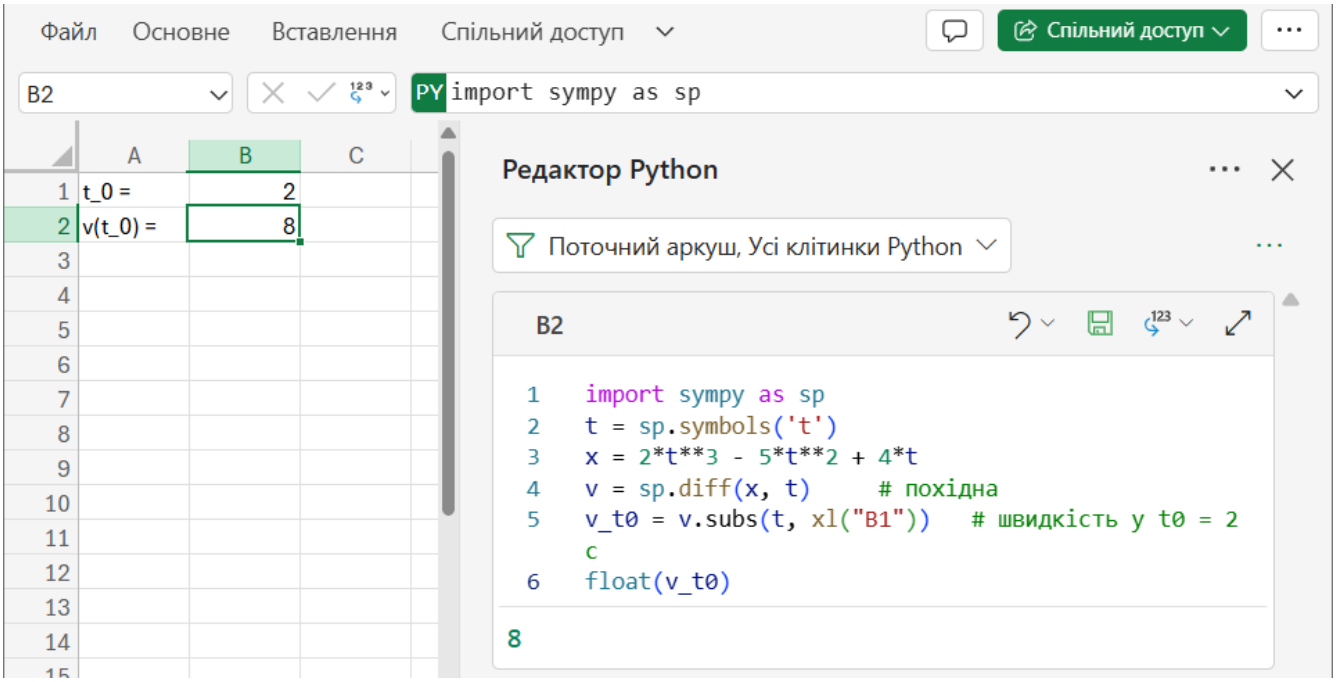

Рис. 2.5. Знаходження миттєвої швидкості.

II спосіб. Для знаходження швидкості у момент t_0 обчислимо:

$$v(t_0) \approx \frac{\Delta x}{\Delta t}. \tag{2.9}$$

Використаємо бібліотеку numpy для (2.8).

Лістинг 2.8. Наближене знаходження миттєвої швидкості.

```
import numpy as np
def x(t):
    return 2*t**3 - 5*t**2 + 4*t
t0 = xl("B1")
dt = 0.001
v_t0 = (x(t0 + dt) - x(t0)) / dt
v_t0
```

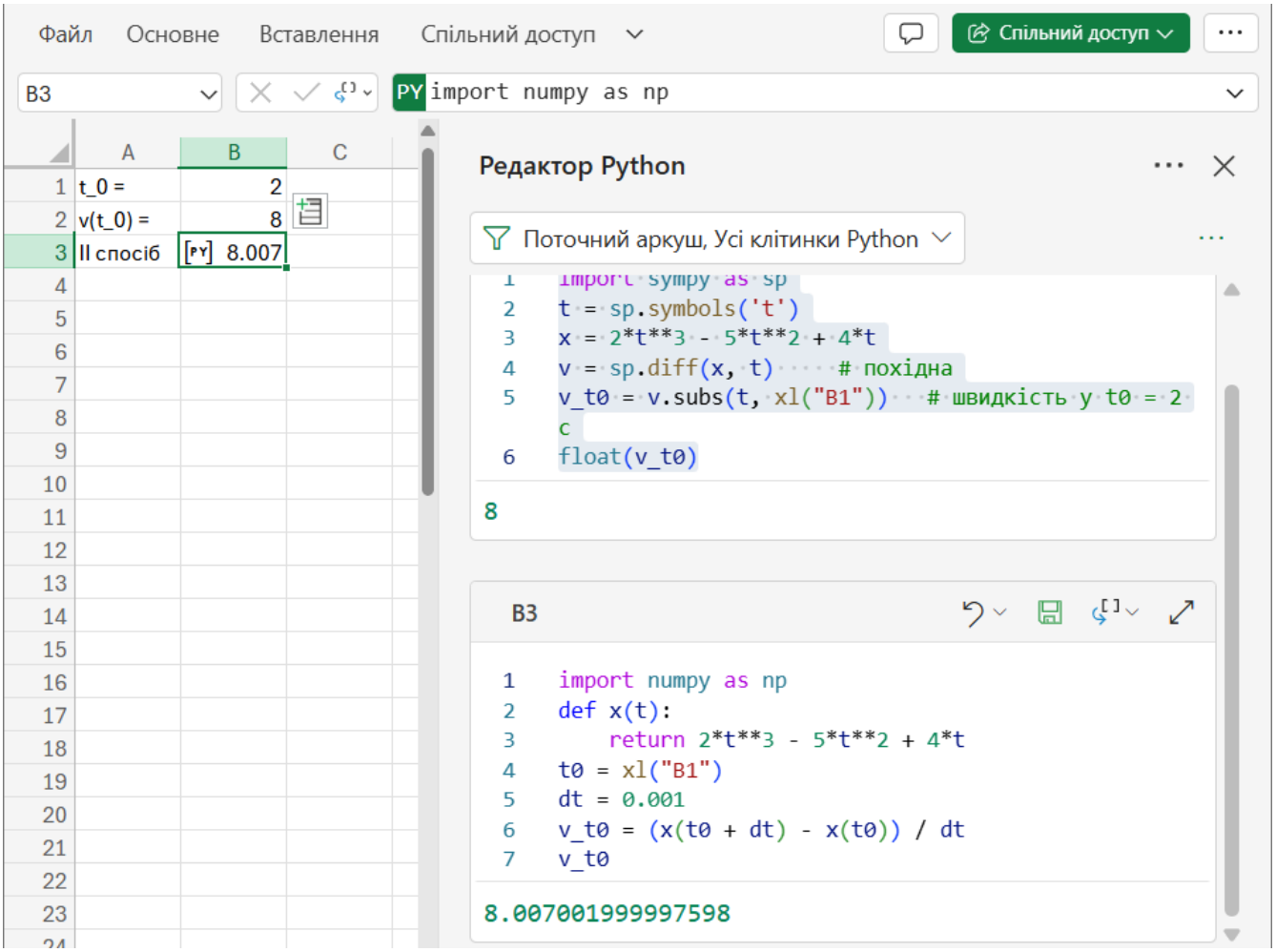

Рис. 2.6. Знаходження миттєвої швидкості 2-ма способами.

Приклад 2. Знайдіть шлях, який проходить матеріальна точка з моменту t_1 до t_2, швидкість якої задано законом $v(t)$.

I спосіб. Використаємо аналітичний метод через визначений інтеграл:

$$s = \int_{t_1}^{t_2} v(t)dt. \tag{2.10}$$

Нехай

$$v(t) = 3t^2 - 4t + 1, t_1 = 1, t_2 = 3. \tag{2.11}$$

Використаємо бібліотеку sympy.

Лістинг 2.9. Знаходження шляху матеріальної точки.

```
import sympy as sp
t = sp.symbols('t')
t1 = xl("B1")
```

```
t2 = xl("B2")
v = 3*t**2 - 4*t + 1
float(sp.integrate(v, (t, t1, t2)))
```

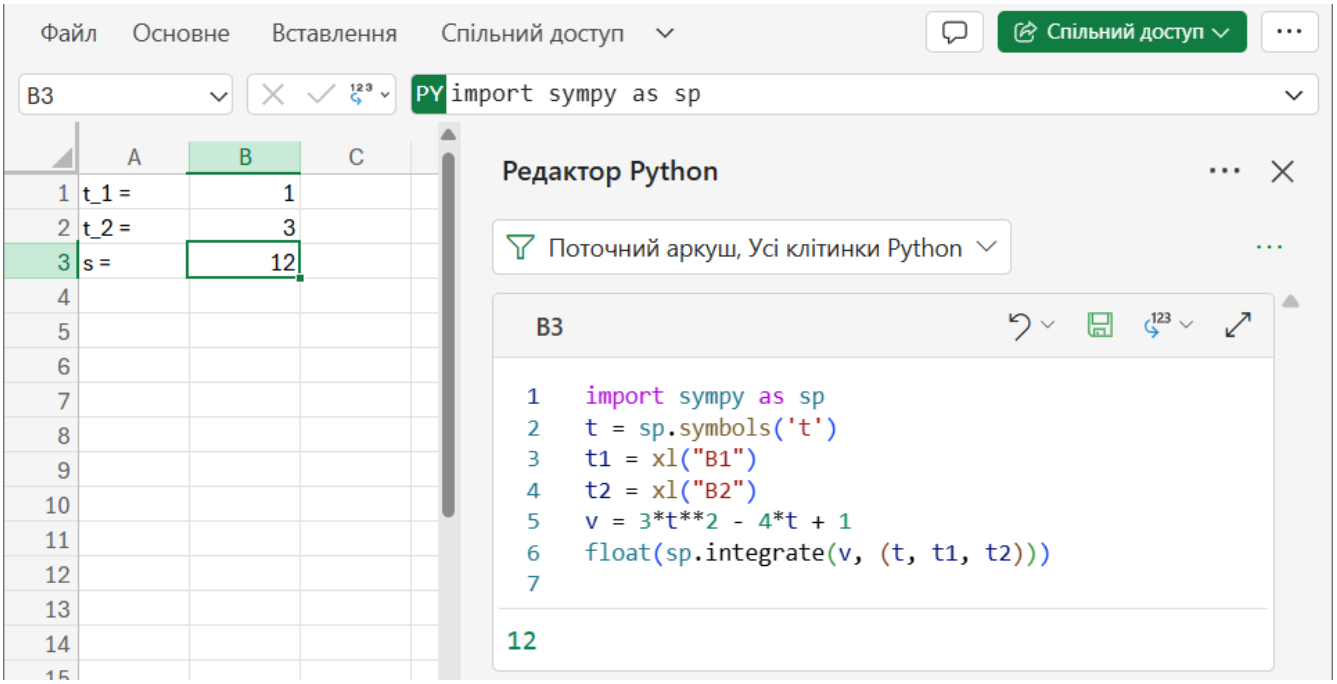

Рис. 2.7. Знаходження шляху матеріальної точки.

II спосіб. Використаємо наближене обчислення визначеного інтегралу методом трапецій. Використаємо бібліотеку numpy для (2.11).

Лістинг 2.8. Наближене обчислення визначеного інтегралу методом трапецій.

```
import numpy as np
def v(t):
    return 3*t**2 - 4*t + 1
t = np.linspace(xl("B1"), xl("B2"), 1000)
s = np.trapz(v(t), t)
s
```

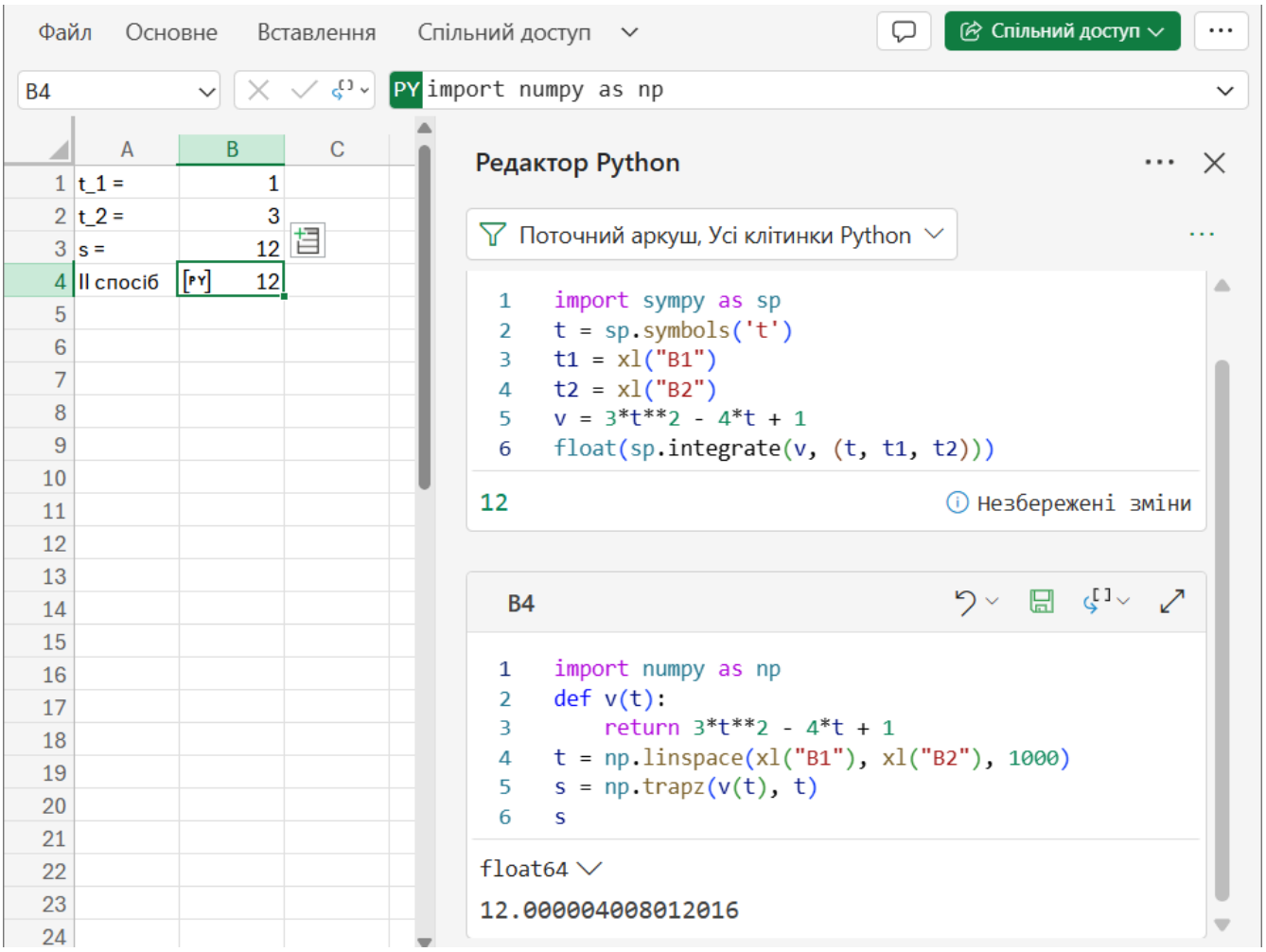

Рис. 2.8. Знаходження шляху матеріальної точки 2-ма способами.

Приведені приклади демонструють, як за допомогою Python можна знаходити миттєву швидкість як похідну від закону руху або обчислювати шлях як визначений інтеграл від швидкості. Це забезпечує зв’язок між аналітичним і чисельним підходами до розв’язування задач з кінематики.

Отже, використання Python у лабораторних роботах з фізики сприяє формуванню в учнів предметної, інформаційно-цифрової та дослідницької компетентностей, а також реалізує принципи STEM-освіти через інтеграцію фізики та інформатики. Такий підхід забезпечує наочність, підвищує мотивацію до навчання та готує учнів до використання сучасних інструментів опрацювання даних у подальшій освітній і професійній діяльності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Basics of STEM education: Навчальний посібник / О. С. Грицюк, О. Б. Кобильська, Т. А. Григорова, Т. С. Бриль. Кременчук: Видавничий відділ КрНУ, 2023. 151 с. ISBN 978-617-7304-13-4
2. Research on Jupyter/cloud notebooks as teaching tools. Duda et al. (2021) and related studies on cloud-based notebooks for interactive coding education.
3. Semenykhina O.V., Rudenko Y.O. Проблеми навчання програмувати учнів старших класів та шляхи їх подолання, ITLT, вип. 66, вип. 4, с. 54-64, Вер 2018. doi: https://doi.org/10.33407/itlt.v66i4.2149.
4. Ботузова Ю.В. Методика навчання математики з Python на прикладі теми «Числові послідовності». *Наукові записки*. Вип. 2(4). 2024. https://doi.org/10.32782/cusu-pmtp-2024-2-3
5. Державний стандарт базової середньої освіти. URL: https://mon.gov.ua/osvita-2/zagalna-serednya-osvita/nova-ukrainska-shkola-2/derzhavniy-standart-bazovoi-serednoi-osviti
6. Інформатика для 10-11 класів (профільне навчання). [Електронний ресурс]. URL: https://mon.gov.ua/storage/app/media/zagalna%20serednya/programy-10-11-klas/2018-2019/01/10-11-profilniy-riven.docx
7. Інформатика. Навчальна програма вибірково-обов’язкового предмету для учнів 10-11 класів загальноосвітніх навчальних закладів (рівень стандарту). [Електронний ресурс]. URL: https://mon.gov.ua/storage/app/media/zagalna%20serednya/programy-10-11-klas/2018-2019/informatikastandart-10-11.docx
8. Кобильник Т.П., Когут У.П., Жидик В.Б. Методичні аспекти вивчення основ алгоритмізації і програмування мовою Python у шкільному курсі інформатики у старших класах. *Фізико-математична освіта*, 2021. Вип. 5(31). С. 36–44.

9. Кобильник Т.П., Сікора О.В., Жидик В.Б., Шаран О.В. Python як засіб навчання основ алгоритмізації у закладах загальної середньої освіти. Інформаційні технології і засоби навчання, 2022, Том 89, №3. https://10.33407/itlt.v89i3.4896
10. Костюченко А.О. Основи програмування мовою Python: навчальний посібник. Чернігів: ФОП Баликіна С.М., 2020. 180 с.
11. Креневич А.П. Python у прикладах і задачах. Частина 2. Об'єктно-орієнтоване програмування. Навчальний посібник. К.: ВПЦ "Київський Університет", 2020. 152 с.
12. Маловічко О., Конюхов С. Застосування спеціалізованого педагогічного програмного комплексу у процесі вивчення програмування у восьмому класі. *Ukrainian Journal of Educational Studies and Information Technology*, 5 (4), с. 38-55, 2017. doi: https://doi.org/10.32919/uesit.2017.04.04
13. Мерзляк А.Г., Полонський В.Б., Якір М.С. Алгебра: підруч. для 9 кл. Харків: Гімназія, 2017. 272 с.
14. Міцкан Л., Вербицька Л., Базурін В. Порівняльний аналіз мов Python і Free Pascal як перших мов програмування для учнів 8 класу. *Актуальні питання природничо-математичної освіти*, № 2 (10), с. 130-139, 2017.
15. Основи програмування. Python. Частина 1 [Електронний ресурс]: підручник для студ. спеціальності 122 "Комп'ютерні науки", спеціалізації "Інформаційні технології в біології та медицині" / А. В. Яковенко ; КПІ ім. Ігоря Сікорського. Електронні текстові данні (1 файл: 1,59 Мбайт). К.: КПІ ім. Ігоря Сікорського, 2018. 195 с.
16. Палагін В. В. Основи Python та програмування електронних систем : [навч. посіб.] [Електронний ресурс] / В. В. Палагін, О. А. Палагіна, О. С. Зорін ; М-во освіти і науки України, Черкас. держ. технол. ун-т. Черкаси : ЧДТУ, 2024. 216 с.

17. Погромська Г., Махровська Н. Синергія програмування та природничо-математичних дисциплін у контексті STEM-освіти. *Теорія, методика і практика навчання*. № 3 (106) 2025. https://doi.org/10.54662/veresen.3.2025.04
18. Руденко В.Д., Жугастров В. Д. Основи алгоритмізації і програмування. Харків, Україна: Вид-во «Ранок», 2019.
19. Руденко В.Д., Речич В.Д., Потієнко В.Д. Інформатика (профільний рівень): підруч. для 10 кл. закл. загал. серед. освіти. Харків, Україна: Вид-во «Ранок», 2018.
20. Яковлєва О., Пенко О. Формування міжпредметних зв’язків на прикладі бінарного уроку з математики та інформатики на тему «Центральна та осьова симетрія у координатах». *Освіта. Інноватика. Практика*. Том 10, №6, 2022. https://10.31110/2616-650X-vol10i6-008