



Руслан АВСІЄВИЧ,
учитель інформатики ліцею №11 м.Хмельницького

Prompt-інженерія як інструмент розвитку алгоритмічного мислення

*при вивченні мови програмування Python
у закладах загальної середньої освіти*

Цифрова трансформація освіти досягла етапу, коли технології генеративного штучного інтелекту (ШІ) стали невід’ємною складовою професійного середовища. Найбільшого впливу ці зміни зазнали у сфері навчання програмуванню. Поява великих мовних моделей (LLM) – GPT-4, Gemini та Claude, здатних миттєво генерувати синтаксично коректний код на мові Python, ставить перед педагогічною наукою виклик: чи залишається актуальним традиційне вивчення синтаксису мови, якщо технічну роботу може виконати алгоритм?

Як зазначає у своїх працях Н. В. Морзе, сучасна освіта має зміщувати акцент із простого накопичення знань на розвиток стратегічних компетентностей, що дозволяють здобувачам освіти ефективно розв’язувати складні завдання в умовах невизначеності [4]. У фундаментальних методичних працях авторки закладено принципи свідомого оволодіння алгоритмізацією як базою логічного мислення. Проте в умовах масового поширення ШІ цей підхід потребує трансформації: від «механічного кодування» до високорівневого архітектурного проектування програмних рішень.

Актуальність статті зумовлена необхідністю адаптації концепції Нової української школи (НУШ) до вимог Державного стандарту базової середньої освіти, який

визначає цифрову компетентність як одну з ключових для сучасного випускника [1].

В роботі обґрунтовано тезу, що Prompt-інженерія – мистецтво формування структурованих запитів до ШІ, що є фактичною еволюцією псевдокоду. Вона вимагає не менш глибокого розуміння логіки, декомпозиції та абстракції, ніж класичне програмування, але звільняє когнітивні ресурси для архітектурного проектування програм.

У межах вивчення Python це створює унікальні можливості. Як підкреслює Ерік Маттес, програмування – це передусім процес перетворення концептуальних ідей на робочий інструмент [3]. Запропонований підхід дозволяє учню виступати в ролі «технічного лідера» (Senior Developer), який формулює стратегічне технічне завдання, тоді як ШІ виконує роль «виконавця». Це корелює з дослідженнями Е. Молліка, де ШІ розглядається як інструмент створення інтелектуальних систем навчання, що сприяють активному засвоєнню знань [6].

Трансформація ролі учня від «кодувальника» до «архітектора» вимагає переосмислення кожного етапу створення програми. Для наочного підтвердження цієї тези у таблиці 1 наведено порівняльний аналіз класичного та інноваційного підходів.

**Порівняння парадигм розробки програмного забезпечення
в освітньому процесі**

Етап розробки програмного продукту	Класична парадигма (традиційне алгоритмічне виконання)	Інноваційна парадигма (управління ШІ-асистентом)
1. Аналіз задачі	Побудова алгоритму (блок-схема)	Проектування стратегії запиту (Prompt)
2. Кодування	Репродуктивне написання коду	Делегована генерація синтаксису
3. Реалізація	Порядкове налагодження	Експертна верифікація цілісності
4. Валідація	Емпіричне тестування (IDLE)	Ітеративна оптимізація та рефакторинг

Промпт як сучасна форма алгоритму та теоретичний базис нової методики

У класичній парадигмі навчання інформатики алгоритм розглядається як жорстка послідовність команд. Проте з появою генеративного ШІ ми переходимо до декларативного програмування. Тепер учень

має не просто знати синтаксис мови Python, а вміти формалізувати задачу у вигляді структурованого запиту.

Можна стверджувати, що якісний промпт – це не довільний текст, а логічна конструкція, що має чіткі складники. Для наочності порівняємо два підходи до розв’язання задачі у таблиці 2.

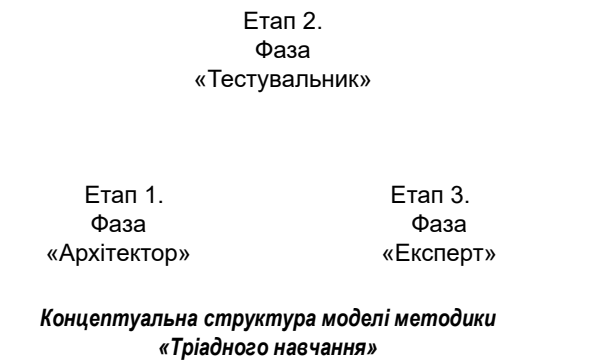
Порівняльна характеристика типів запитів до ШІ

Параметр порівняння	Побутовий запит (низька ефективність)	Алгоритмічний промпт (висока ефективність)
Ціль	«Напиши код для сортування списку»	«Реалізуй функцію сортування бульбашкою для списку цілих чисел»
Контекст	Відсутній	«Використовуй лише стандартні бібліотеки Python 3.10+»
Обмеження	Відсутні	«Додай обробку винятку, якщо вхідний список порожній»
Результат	Хаотичний код	Структурований скрипт із коментарями

Ця таблиця демонструє, що для написання ефективного промпту здобувач освіти повинен володіти знаннями про типи даних, алгоритмічні структури та обробку помилок. Таким чином, промпт стає «семантичним кодом», який ШІ лише трансліює у синтаксис Python.

Методика «Тріадного навчання»: концептуальна модель та практична реалізація

Для системного впровадження штучного інтелекту в освітній процес пропонується методика «Тріадного навчання» (рисунк 1). Її концептуальна основа полягає у зміщенні ролі учня від пасивного споживача контенту до активного управління (Tech Lead). Методика дозволяє вчителю контролювати процес мислення учня на кожному етапі, незважаючи на те, що технічну частину написання коду делеговано ШІ.



Методика візуалізується як замкнений цикл, що базується на послідовній зміні трьох ключових ролей учня.

Етап 1. Фаза «Архітектор» (проектування промпту)
На цій стадії учень проводить ментальну декомпозицію задачі. Замість написання синтаксису він фокусується на створенні високорівневої логічної структури: визначає роль ШІ, послідовність логічних кроків та формат вихідних даних.

Етап 2. Фаза «Тестувальник» (верифікація та дебагінг)
Після отримання коду від ШІ учень переходить до роботи у професійному середовищі розробки (IDE). Головна мета – ідентифікація технологічних артефактів ШІ (логічних помилок у синтаксично коректному коді) та перевірка алгоритму на різних наборах даних.

Етап 3. Фаза «Експерт» (Рефлексія та оптимізація)
Це стадія інтелектуального синтезу, де учень проводить ітеративний рефакторинг та оцінює якість продукту. Учень формулює фінальне стійке рішення, беручи на себе повну відповідальність за результат та його відповідність технічному завданню.

Практична реалізація методики через кейс-проект «Інтелектуальна система кліматичного контролю»

Практична апробація методики базується на вирішенні задачі з аналізу температурних показників, що дозволяє наочно продемонструвати перехід учня між трьома рольовими фазами.

Реалізація фази «Архітектор». На цьому етапі учень формалізує технічне завдання. Головна вимога, що промпт не повинен бути коротким запитом «напиши код», а має виступати детальною інструкцією, що демонструє розуміння алгоритмічних структур.

Приклад промпту учня:
«Дій як розробник на мові Python. Напиши програму, яка виконує такі завдання: отримує список температур temps, обчислює середню температуру, виключно через цикл for та класифікує показники за певними критеріями. Зокрема, значення менше 10° мають отримати статус «холодно», показники від 10° до 20° включно – «комфортно», а результати понад 20° – «жарко». У роботі використовуй лише базові оператори Python 3.x без залучення додаткових бібліотек».

Реалізація фази «Тестувальник». Учень отримує згенерований ШІ код і проводить його критичний аналіз у середовищі розробки. Штучний інтелект часто пропонує синтаксично правильні, але логічно вразливі рішення.

Особливістю цього етапу є зміна фокусу з написання коду на його аудит. Учень має усвідомити, що нейромережа орієнтована на статистичну ймовірність символів, а не на глибоке розуміння фізичного змісту задачі чи методичних обмежень навчального процесу. Використання ШІ як «генератора напівфабрикатів» дозволяє вчителю змістити акцент на розвиток навичок дебагінгу та верифікації, які є критичними для професійного розробника. Нижче наведено приклад коду (рисунк 2), отриманого за запитом, та результати його аналізу учнем у ролі тестувальника.

```
temps = [12, 15, 8, 11, 20, 24, 13]

average = sum(temps) / len(temps)

for t in temps:
    if t < 10:
        print("Холодно")
    elif t > 10 and t < 20:
        print("Комфортно")
    else:
        print("Жарко")
```

Згенерований ШІ код
(із прихованими технологічними артефактами)

Аналітична робота учня

Такий формат аудиту дозволяє трансформувати виявлені дефекти коду на цінні навчальні кейси, що сприяє глибокому розумінню логіки програмування. Учень ідентифікує недоліки програмного коду та фіксує їх у звіті за такими критеріями:

логічна помилка. Умова $t > 10$ and $t < 20$ не враховує граничні значення (10.0 та 20.0). Зокрема, при температурі рівно 10° програма помилково видасть «Жарко» через перехід до блоку else;

порушення технічного завдання. Використання вбудованої функції `sum()` замість обов'язкового циклу `for` суперечить навчальній меті заняття;

системна вразливість. Відсутність перевірки на порожній список температур, що за умови відсутності даних призведе до критичної помилки `ZeroDivisionError`.

Реалізація фази «Експерт». Учень самостійно вносить правки, створюючи фінальний продукт. Це демонструє здатність «навчити» ШІ працювати правильно або скоригувати його роботу вручну (рисунок 3).

```
temps = [10, 15, 8, 11, 20, 24, 10]

# 1. Захист від порожнього списку (валідність даних)
if not temps:
    print("Помилка: Датчик не передав дані. Список порожній.")
else:
    # 2. Розрахунок середнього значення через цикл for (згідно з ТЗ)
    total = 0
    for t in temps:
        total += t
    average = total / len(temps)
    print(f"Середня температура за період: {average:.1f}°C")
    # 3. виправлена логіка класифікації (враховано межі)
    for t in temps:
        if t < 10:
            status = "Холодно"
        elif 10 <= t <= 20: # Межі включено учнем власноруч
            status = "Комфортно"
        else:
            status = "Жарко"
        print(f"Показник {t}°C: {status}")
```

Фінальний (перевірений та оптимізований) код учня

Система оцінювання навчальних досягнень в умовах використання ШІ

Традиційний підхід до оцінювання в програмуванні («програма працює – 12 балів») втрачає сенс, оскільки технічну працездатність коду гарантує нейромережа. Розроблено критеріально-орієнтовану систему, де основна

вага оцінки припадає на аналітичну роботу учня.

Вчителю пропонується використовувати комплексну шкалу оцінювання (таблиця 3), яка базується на трьох функціональних компонентах. Такий підхід дозволяє об'єктивно виміряти не лише кінцевий результат (працездатність коду), а й інтелектуальний внесок учня на кожному етапі взаємодії з технологією штучного інтелекту.

Критерії та показники оцінювання навчальних досягнень учнів за тріадною моделлю

Компонент	Критерій оцінки	Вага в балах
Архітектурний (промпт)	Наявність у запиті визначеної ролі, контексту, чіткої послідовності кроків алгоритму та встановлених обмежень	3
Аналітичний (дебагінг)	Здатність ідентифікувати логічні помилки штучного інтелекту (зокрема некоректні межі в операторах if-elif) та самостійно їх виправити	5
Рефлексивний (захист)	Вміння аргументовано пояснити кожен рядок коду, обґрунтувати вибір структур даних та запропонувати шляхи подальшої оптимізації	4

Форми контролю.**«Метод сліпого рефакторингу»**

Як додатковий метод перевірки, вчитель може надати учню код, згенерований ШІ з навмисними помилками (наприклад, без обробки датчиків або з безкінечним циклом). Завдання учня: «вилікувати» код. Це унеможливорює плагіат і стимулює критичне мислення.

Висновки

Симбіоз логіки та синтаксичної експертності. Python трансформується в інструмент верифікації гіпотез, проте успішна Prompt-інженерія неможлива без фундаментальної мовної підготовки. Фаза «Експерта» вимагає від учня здатності самостійно вносити правки та обґрунтовувати вибір структур даних, тому вивчення синтаксису не скасовується, а переходить у площину експертного аудиту та рефакторингу.

Критичне мислення через аудит. Методика «Тріадного навчання» запобігає бездумному копіюванню. Ідентифікуючи «логічні невідповідності» ШІ та опрацьовуючи граничні випадки, учень стає «технічним лідером», який повністю відповідає за логічну та синтаксичну коректність продукту.

Методична та інклюзивна цінність. Оцінювання зміщується з працездатності коду на глибину когнітивних процесів та вміння аргументувати кожен рядок. Це забезпечує справжню інклюзію, наприклад, для учнів із дислексією ШІ нівелює бар'єр написання тексту, дозволяючи реалізувати інтелектуальний потенціал через логіку декомпозиції задач та аналіз готового коду.

Використані джерела

1. Державний стандарт базової середньої освіти: Постанова Кабінету Міністрів України від 30 вересня 2020 р. № 898. Законодавство України / Верховна Рада України. URL: <https://lnk.ua/0rGC27y8j> (дата звернення: 15.02.2026).

2. Доценко С. О. Штучний інтелект у загальній середній освіті: виклики та можливості. Наукові записки кафедри педагогіки. 2023. № 52. С. 45–53. URL: <http://pedagogy-journal.at.ua/> (дата звернення: 15.02.2026).

3. Маттес Е. Пришвидшений курс Python. Проєктне навчання: ігри, візуалізація даних, веб-додатки / пер. з англ. О. О. Кисіль. Харків: Вид-во «Ранок»: Фабула, 2023. 592 с.

4. Морзе Н. В. Методика навчання інформатики : посіб. для студ. пед. навч. закл.: у 3 ч. / за ред. М. І. Жалдака. Київ: Навчальна книга, 2004. Ч. 2 : Методика навчання інформаційних технологій. 287 с.

5. Mishra P. Practical Prompt Engineering: A Comprehensive Guide to Crafting Strategic and Effective Prompts. Berkeley, CA: Apress, 2023. 218 p.

6. Mollick E. R., Mollick L. Using AI to Implement Intelligent Tutoring Systems: Bridging the Gap Between Technology and Pedagogy. SSRN Electronic Journal. 2023. URL: <https://lnk.ua/IgUTY740j> (дата звернення: 15.02.2026).



Ірина КОВАЛЬЧУК,
вчителька інформатики гімназії №1 м.Славути

Компетентнісний підхід у 8 класі: навчаємо програмувати, приймаючи рішення

Алгоритми та програми — це одна з найскладніших тем шкільного курсу, і для восьмикласника програмування часто видається набором незрозумілих команд, поки він не побачить у коді відображення власного життя. Тема розгалужень — це ідеальний місток між синтаксисом Python та реальністю. У НУШ інформатика має не лише вчити писати код, а й готувати до життя в цифровому суспільстві.

Як зазначено в модельній програмі «Інформатика. 7–9 класи» для закладів загальної середньої освіти (автори Ривкінд Й. Я., Лисенко Т. І., Чернікова Л. А., Шакоцько В. В.), пріоритетом викладання є повага до особистості учня та врахування його інтересів при добірї об'єктів для навчання. Саме тому перехід від сухих консольних розрахунків до життєвих задач із графічним інтерфейсом є ключовим для підтримки пізнавального інтересу. Згідно з програмою,

вивчення інформатики має на меті формування ключових компетентностей, серед яких особливо важливі для 8 класу:

- Математична компетентність: логічне обґрунтування дій для розв'язування задач.
- Підприємливість і фінансова грамотність: через задачі на розрахунок вартості послуг чи товарів.
- Інноваційність: здатність створювати нові інформаційні продукти для розв'язування життєвих проблем.

У цій статті основний акцент робиться на тому, як перетворити звичайний урок інформатики у 8 класі на симуляцію реального бізнесу. Відійшовши від абстрактних «х та у», навчаю учнів створювати алгоритми для кур'єрських служб, мобільних операторів, розрахунку вартості покупок та підрахунку знижок у магазинах, обчислення суми відсотків у банку за депозит чи кредит.