

УДК 378.147:004.42

DOI <https://doi.org/10.32782/2412-9208-2026-3-101-110>

FORMATION OF STUDENTS' TRANSFERABLE PROGRAMMING SKILL

ФОРМУВАННЯ ТРАНСФЕРНИХ НАВИЧОК ПРОГРАМУВАННЯ СТУДЕНТІВ

Mariia SEMANKIV,

Candidate of Technical Sciences,
Associate Professor at the
Department of Computer Science
and Information Systems,
Vasyl Stefanyk Carpathian National
University
57, Shevchenko Str., Ivano-
Frankivsk, 76018, Ukraine

mariia.semankiv@pnu.edu.ua
<https://orcid.org/0000-0002-1314-8923>

Марія СЕМАНЬКІВ,

кандидат технічних наук, доцент
кафедри комп'ютерних наук
та інформаційних систем,
Карпатський національний
університет імені Василя
Стефаника
вул. Шевченка, 57 м. Івано-
Франківськ, 76018, Україна

Ivan YAREMIY,

Doctor of Physical and Mathematical
Sciences, Professor, Professor at the
Department of Applied Physics and
Materials Science,
Vasyl Stefanyk Carpathian National
University
57, Shevchenko Str., Ivano-
Frankivsk, 76018, Ukraine

yaremiy@pnu.edu.ua
<https://orcid.org/0000-0002-8549-1173>

Іван ЯРЕМІЙ,

доктор фізико-математичних наук,
професор,
професор кафедри прикладної
фізики і матеріалознавства,
Карпатський національний
університет імені Василя
Стефаника
вул. Шевченка, 57, м. Івано-
Франківськ 76018, Україна

ABSTRACT

The article is devoted to the study of the problems associated with the development of students' transferable programming skills when transitioning between programming languages, particularly from statically typed to dynamically typed languages. Particular attention is paid to the influence of prior programming experience on the acquisition of Python, the manifestations of positive and negative transfer, and the difficulties involved in adapting problem-solving approaches to the syntactic, typing, and conceptual characteristics of a new language.

The aim of the study is to identify the specific features of developing students' transferable programming skills when transitioning between programming languages and to substantiate approaches to overcoming negative transfer. The transfer of algorithmic and programming knowledge between statically and dynamically typed languages is analyzed, the main difficulties students encounter when adapting to a new language are identified, and manifestations of the mechanical transfer of previously acquired programming approaches, as well as the insufficient use of Python-specific features, are examined.

The analysis demonstrates that effective transition between programming languages cannot be reduced to learning new syntax and mechanically reproducing previously known algorithms. It has been established that previously formed programming models can both accelerate the learning process and create obstacles to mastering the specific characteristics of a new language. In particular, students often retain ways of thinking and implementing solutions that are characteristic of statically typed languages, which is manifested in the excessive use of loops, insufficient application of built-in functions and high-level constructs, and failure to use idiomatic Python features. Based on these findings, it is concluded that the development of effective transferable programming skills should involve not the mechanical transfer of programming knowledge, but its adaptation to the conceptual, typing, syntactic, and idiomatic characteristics of the new language. Such an approach contributes to the development of flexible programming thinking and enhances students' readiness to independently acquire new programming languages and technologies.

Key words: *programming language, transferable skills, Python.*

Вступ. Важливим аспектом професійної підготовки здобувачів освіти вищих навчальних закладів є здатність адаптуватися до використання різних мов і парадигм програмування в процесі навчання. Опановуючи нові дисципліни та програмні технології, студенти можуть переходити як від статично типізованих мов до динамічно типізованих, так і у зворотному напрямку, що зумовлює необхідність адаптації вже сформованих програмістських знань до нових мовних середовищ. Такий перехід передбачає зміну не лише синтаксичних конструкцій, а й підходів до типізації, виконання програм, організації програмного коду, роботи з пам'яттю, використання бібліотек та програмних середовищ. Наприклад, перехід від Python до C++, Java або C# потребує адаптації до статичної типізації, компіляції та більш формалізованих підходів до організації програмного коду, тоді як перехід у зворотному напрямку вимагає переосмислення ролі типів, способів роботи зі змінними та використання можливостей динамічної типізації [6; 3]. Незалежно від напрямку переходу, студенти стикаються з необхідністю відокремлювати фундаментальні алгоритмічні та програмістські знання від особливостей конкретної мови. Тому важливого значення набувають трансферні навички програмування, які забезпечують перенесення набутих знань і способів розв'язання задач між різними мовами з одночасною їх адаптацією до синтаксичних, семантичних, типізаційних та парадигмальних особливостей нового середовища. Формування таких навичок є важливою складовою професійної

компетентності майбутнього фахівця, оскільки забезпечує здатність ефективно опановувати нові мови, технології та програмні інструменти протягом професійної діяльності [5].

Метою дослідження є визначення особливостей формування трансферних навичок програмування студентів під час переходу між мовами програмування та обґрунтування педагогічних підходів до подолання негативного трансферу попередніх знань.

Методи та методики дослідження. Для дослідження особливостей формування трансферних навичок програмування студентів під час переходу від статично типізованих мов до Python використано комплекс теоретичних та емпіричних методів, зокрема аналіз, систематизацію й узагальнення наукових і навчально-методичних джерел, порівняльний аналіз особливостей статичної та динамічної типізації, педагогічне спостереження, аналіз студентських програмних робіт [3; 4]. Емпіричне дослідження спрямовано на виявлення труднощів, що виникають у студентів під час перенесення сформованих раніше алгоритмічних і програмістських знань на Python. Для оцінювання рівня сформованості трансферних навичок проаналізовано здатність студентів переносити алгоритмічні знання, адаптувати способи розв'язання задач до особливостей нової мови, використовувати її ідіоматичні конструкції та самостійно обирати доцільні засоби реалізації програмних рішень [2].

Результати та дискусії. Актуальним напрямом удосконалення професійної підготовки фахівців є дослідження та врахування проблем, з якими студенти стикаються під час переходу від однієї мови програмування до іншої. Такий перехід передбачає адаптацію сформованих алгоритмічних моделей, способів мислення, підходів до організації програмного коду та практик використання програмних засобів до нових мовних і парадигмальних умов [5]. Неврахування зазначених труднощів може призводити до виникнення негативного трансферу, механічного перенесення конструкцій і підходів попередньої мови, зниження ефективності навчання та недостатнього використання можливостей нової мови програмування. Тому систематичне виявлення, обговорення та педагогічне врахування таких проблем є важливою передумовою підвищення результативності навчання програмування. Особливого значення набуває розроблення методичних підходів, спрямованих не лише на подолання мовних і синтаксичних бар'єрів, а й на формування трансферних навичок програмування, що забезпечують усвідомлене перенесення фундаментальних знань між мовами та одночасне врахування їхніх концептуальних, типізаційних і парадигмальних особливостей [3]. Такий підхід сприяє формуванню у студентів гнучкого програмістського мислення та здатності самостійно адаптуватися до

нових мов і технологій, що є важливою складовою їхньої професійної компетентності.

Перехід студентів від статично типізованих мов програмування до динамічно типізованих, зокрема до Python, характеризується зміною не лише мовних засобів, а й сформованих підходів до проектування та реалізації програм. Розглянемо декілька основних аспектів, з якими стикаються студенти та на які доцільно звернути особливу увагу викладачам:

- *відсутність необхідності явно оголошувати тип змінної* в Python часто сприймається як суттєве спрощення, оскільки зменшується обсяг службових конструкцій, а процес написання програм стає швидшим і гнучкішим. Однак таке початкове спрощення поступово супроводжується необхідністю переосмислення ролі типів даних у програмі. Студентам, які звикли до статичної типізації, може бути складно адаптуватися до ситуації, коли тип визначається під час виконання, а помилки, пов'язані з несумісністю типів, можуть бути виявлені не на етапі компіляції, а лише під час виконання відповідної ділянки коду. Викладачу під час навчального процесу слід зосередитись на формуванні в студентів нового підходу до контролю типів, розумінні поведінки об'єктів та відповідальності програміста за коректність операцій над ними. Таким чином, початкова перевага динамічної типізації у вигляді спрощення та скорочення програмного коду поступово трансформується у новий навчальний виклик – необхідність усвідомленого контролю типів і передбачення їхньої поведінки під час виконання програми [2];

- *вивчення особливостей синтаксису та стандартів оформлення програмного коду в Python*. Важливим є перехід від звичних правил форматування, характерних для статично типізованих мов, до принципів оформлення коду, прийнятих у Python. Особливого значення набуває використання відступів для структурування програмних блоків, дотримання єдиного стилю іменування змінних та функцій, організація коду відповідно до принципів PEP8, а також використання характерних для Python конструкцій і підходів до написання програм. На цьому етапі викладач повинен наголосити, що відсутність необхідності явно оголошувати типи змінних не означає відсутності вимог до структури та якості програмного коду, що читабельність, послідовність форматування та дотримання усталених стилістичних правил є важливими складовими професійного програмування [7]. Таким чином, адаптація до Python передбачає не лише засвоєння нового синтаксису, а й формування нової культури написання та оформлення програмного коду, що відрізняється від практик, сформованих під час роботи зі статично типізованими мовами;

– на подальшому етапі навчання студенти стикаються з проблемою **перенесення сформованих способів розв’язання задач зі статично типізованої мови програмування на синтаксис Python**. Зокрема, алгоритм розв’язання задачі спочатку формується у свідомості з використанням конструкцій і програмістських моделей, характерних для вже вивченої статично типізованої мови, після чого здійснюється фактичне «перекладання» цього алгоритму на синтаксис Python. Такий підхід часто призводить до створення програмного коду, який формально є коректним, але не використовує переваг та ідіоматичних можливостей Python. Студент у цьому випадку фактично продовжує мислити категоріями попередньої мови, сприймаючи Python переважно як інший синтаксичний засіб реалізації вже сформованих алгоритмів. Це може ускладнювати засвоєння характерних для Python конструкцій, зокрема спискових включень, ітераторів, генераторів, функцій вищого порядку та інших засобів, що сприяють більш компактному й виразному представленню алгоритмів. Тому одним із важливих завдань навчання є поступовий перехід від синтаксичного перекладу програмного коду до безпосереднього формування алгоритмічних рішень з урахуванням концептуальних та ідіоматичних особливостей Python.

Наприклад, програмний код розв’язання задачі знаходження суми цифр натурального числа двома мовами програмування показує, що алгоритмічна ідея зберігається, але спосіб її реалізації принципово змінюється відповідно до можливостей Python:

На C++:

```
#include <iostream>
using namespace std;
int main() {
    int n;
    cin >> n;
    int sum = 0;
    while (n > 0) {
        sum += n % 10;
        n /= 10;
    }
    cout << sum << endl;
    return 0;
}
```

На Python:

```
n = input()
result = sum(map(int, n))
print(result)
```

Використання синтаксичного перекладу програмного коду призводить до недостатнього використання можливостей та специфічних особливостей Python, оскільки студенти фактично відтворюють у ньому підходи, сформовані під час роботи зі статично типізованими

мовами програмування. У результаті значна частина переваг Python – лаконічність синтаксису, динамічна типізація, високорівневі конструкції, вбудовані засоби роботи з колекціями, функціональні можливості та розвинена екосистема бібліотек) залишається недостатньо реалізованою у практичній діяльності студентів. Використання Python переважно як «синтаксичної заміни» іншої мови фактично нівелює значну частину його переваг, збільшує обсяг програмного коду та може формувати хибне уявлення про доцільність і ефективність використання мови. Тому важливим завданням навчання є формування в студентів не лише здатності реалізовувати вже відомі алгоритми засобами Python, а й уміння мислити категоріями самої мови, використовуючи її специфічні конструкції, абстракції та ідіоматичні підходи до розв'язання задач. Викладач повинен спрямовувати навчальний процес на формування здатності студентів виокремлювати загальні алгоритмічні принципи, критично переосмислювати попередній програмувальний досвід та адаптувати способи розв'язання задач відповідно до концептуальних, синтаксичних й ідіоматичних особливостей нової мови програмування. Ефективний трансфер передбачає не механічне відтворення раніше засвоєних програмних конструкцій, а вміння відокремлювати універсальні принципи програмування від специфічних особливостей конкретної мови та обирати способи розв'язання задач, найбільш природні для нового програмного середовища.

– Python має власну **філософію програмування**, яка визначає характерні підходи до проектування та реалізації програмних рішень. Філософія Python ґрунтується на принципах простоти, читабельності, ясності та лаконічності програмного коду, що відображено, зокрема, у принципах The Zen of Python. Тому ефективне опанування Python передбачає не механічне засвоєння його синтаксису або перенесення алгоритмів з інших мов, а поступове формування специфічного способу програмістського мислення. Студент має навчитися не лише реалізовувати певний алгоритм мовними засобами Python, а й визначати, який спосіб його реалізації є природним та ідіоматичним саме для цієї мови. У цьому контексті перехід від статично типізованих мов до Python є не просто зміною синтаксису, а зміною підходів до побудови програмного коду, що потребує поступового усвідомлення філософії, принципів та характерних програмістських практик Python.

– однією з проблем під час опанування Python студентами, які мають попередній досвід програмування іншими мовами, є **недостатнє використання вбудованих функцій та високорівневих засобів мови**. Звикнувши до реалізації алгоритмів за допомогою базових циклічних і умовних конструкцій, студенти часто продовжують застосовувати аналогічні підходи в Python, не використовуючи значний набір вбудова-

них функцій, призначених для спрощення та підвищення виразності програмного коду [1]. У результаті програмні рішення можуть бути довгими, менш виразними та складнішими для аналізу, хоча аналогічні операції можуть бути реалізовані за допомогою стандартних засобів мови значно компактніше. Така ситуація свідчить про те, що формальне засвоєння синтаксису Python не завжди супроводжується опануванням його концептуальних та ідіоматичних можливостей. Тому важливим завданням навчання є формування у студентів здатності не лише відтворювати відомі алгоритмічні конструкції, а й визначати доцільність використання вбудованих функцій та обирати найбільш природний для Python спосіб розв'язання поставленої задачі.

Під час переходу від однієї мови програмування до іншої студент переносить сформовані раніше алгоритмічні знання, однак разом із ними може переносити й звичні способи реалізації програмних рішень. Трансфер відбувається на рівні загального алгоритму, але недостатньо адаптується до специфіки нової мови. Ефективні трансферні навички передбачають здатність не лише перенести наявне знання на нове програмне середовище, а й переосмислити спосіб його практичної реалізації відповідно до можливостей та принципів нової мови програмування. Для формування трансферних навичок програмування викладачеві доцільно конструювати навчальні завдання з Python таким чином, щоб їх розв'язання передбачало використання широкого спектра вбудованих функцій, структур даних, операторів та мовних конструкцій. Такі завдання мають спонукати студентів не лише відтворювати відомі алгоритми, а й здійснювати самостійний вибір засобів мови відповідно до поставленої задачі.

Висновки. Під час формування трансферних навичок програмування викладачу необхідно враховувати не лише рівень сформованих у студентів алгоритмічних знань, а й особливості їхнього попереднього мовного та програмістського досвіду. Особливу увагу слід приділяти розмежуванню універсальних програмістських концепцій і засобів їх реалізації, специфічних для конкретної мови програмування. Навчальні завдання доцільно будувати таким чином, щоб студент спочатку визначав алгоритмічну сутність задачі, а потім обирав спосіб її реалізації з урахуванням особливостей нової мови. Важливим є також виявлення та корекція негативного трансферу, коли попередній досвід стає перешкодою для засвоєння нових конструкцій і підходів. Зокрема, під час переходу до Python необхідно звертати увагу на механічне перенесення циклічних конструкцій, способів роботи зі змінними та типами даних, характерних для статично типізованих мов, а також на недостатнє використання вбудованих функцій, колекцій, генераторів, спискових включень та інших високорівневих засобів Python. Доцільним є

порівняння декількох реалізацій однієї задачі різними мовами з подальшим обговоренням їхніх переваг, недоліків та відповідності принципам конкретної мови. Таким чином, викладач має спрямовувати студента від механічного перенесення програмного коду до усвідомленого перенесення програмістських концепцій, формуючи здатність адаптувати наявні знання, критично переосмислювати попередній досвід і самостійно обирати найбільш доцільні засоби реалізації задач у новому мовному середовищі.

Література

1. Дегтярьова Н., Петренко С., Удовиченко О. Робота з графічними віджетами при вивченні мови програмування Python в закладах загальної середньої освіти. *Освіта. Інноватика. Практика*. 2023. Т. 11, № 4. С. 26–34. DOI: 10.31110/2616-650X-vol11i4-004.
2. Ткачук С., Мельник О. Методика вивчення семантики мов програмування студентами закладів професійної освіти. *Наукові записки Тернопільського національного педагогічного університету імені Володимира Гнатюка*. Серія: педагогіка. 2023. № 1(1). С. 177–184. DOI: 10.25128/2415-3605.23.1.23.
3. Denny P., Becker B. A., Bosch N., Prather J., Reeves B., Whalley J. Novice Reflections During the Transition to a New Programming Language. In: *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education (SIGCSE 2022)*. New York : Association for Computing Machinery, 2022. P. 948–954. DOI: 10.1145/3478431.3499314.
4. Gomes J. P., Monteiro R., Moreira A. S., Mendonça V., Pinho A., Cunha C. R. Teaching introductory programming in top universities: A global study of languages, paradigms, assessment, and AI. *Journal of Computer Languages*. 2026. Vol. 86. Article 101384. DOI: 10.1016/j.col.2025.101384.
5. Tshukudu E., Cutts Q., Foster M. E. Evaluating a Pedagogy for Improving Conceptual Transfer and Understanding in a Second Programming Language Learning Context. In: *21st Koli Calling International Conference on Computing Education Research*. 2021. P. 1–10. DOI: 10.1145/3488042.3488050.
6. Tshukudu E., Cutts Q. Understanding Conceptual Transfer for Students Learning New Programming Languages. In: *Proceedings of the 2020 ACM Conference on International Computing Education Research (ICER '20)*. New York : Association for Computing Machinery, 2020. P. 227–237. DOI: 10.1145/3372782.3406270.
7. Treitjak D. Python programming language as a tool in scientific and pedagogical activities. *Information Technologies and Learning Tools*. 2025. Vol. 107, No. 3. P. 168–181. DOI: 10.33407/itlt.v107i3.6007.

References

1. Dehtiarova, N., Petrenko, S., & Udovychenko, O. (2023). Robota z hrafichnymy vidzhetaamy pry vyvchenni movy prohramuvannia Python v zakladakh zahalnoi serednoi osvity [Working with graphic widgets when learning the Python programming language in general secondary education institutions]. *Osvita. Innovatyka. Praktyka*, 11(4), 26–34. <https://doi.org/10.31110/2616-650X-vol11i4-004> [in Ukrainian].
2. Tkachuk, S., & Melnyk, O. (2023). Metodyka vyvchennia semantyky mov prohramuvannia studentamy zakladiv profesiinoi osvity [Methodology for teaching programming language semantics to students of vocational education institutions]. *Naukovi zapysky Ternopilskoho natsionalnoho pedahohichnoho universytetu imeni Volodymyra Hnati-*

uka. Seria: Pedahohika, 1(1), 177–184. <https://doi.org/10.25128/2415-3605.23.1.23> [in Ukrainian].

3. Denny, P., Becker, B. A., Bosch, N., Prather, J., Reeves, B., & Whalley, J. (2022). Novice reflections during the transition to a new programming language. In Proceedings of the 53rd ACM Technical Symposium on Computer Science Education (SIGCSE 2022) (pp. 948–954). Association for Computing Machinery. <https://doi.org/10.1145/3478431.3499314> [in English].

4. Gomes, J. P., Monteiro, R., Moreira, A. S., Mendonça, V., Pinho, A., & Cunha, C. R. (2026). Teaching introductory programming in top universities: A global study of languages, paradigms, assessment, and AI. *Journal of Computer Languages*, 86, 101384. <https://doi.org/10.1016/j.cola.2025.101384> [in English].

5. Tshukudu, E., Cutts, Q., & Foster, M. E. (2021). Evaluating a pedagogy for improving conceptual transfer and understanding in a second programming language learning context. In 21st Koli Calling International Conference on Computing Education Research (pp. 1–10). Association for Computing Machinery. <https://doi.org/10.1145/3488042.3488050> [in English].

6. Tshukudu, E., & Cutts, Q. (2020). Understanding conceptual transfer for students learning new programming languages. In Proceedings of the 2020 ACM Conference on International Computing Education Research (ICER '20) (pp. 227–237). Association for Computing Machinery. <https://doi.org/10.1145/3372782.3406270> [in English].

7. Treitjak, D. (2025). Python programming language as a tool in scientific and pedagogical activities. *Information Technologies and Learning Tools*, 107(3), 168–181. <https://doi.org/10.33407/itt.v107i3.6007> [in English].

АНОТАЦІЯ

Статтю присвячено дослідженню проблем формування трансферних навичок програмування студентів під час переходу між мовами програмування, зокрема від статично типізованих до динамічно типізованих мов. Особливу увагу приділено впливу попереднього програмістського досвіду на засвоєння Python, проявам позитивного та негативного трансферу, а також труднощам адаптації способів розв'язання задач до синтаксичних, типізаційних і концептуальних особливостей нової мови.

Метою дослідження є визначення особливостей формування трансферних навичок програмування студентів під час переходу між мовами програмування та обґрунтування підходів до подолання негативного трансферу. Проаналізовано перенесення алгоритмічних і програмістських знань між статично та динамічно типізованими мовами, визначено основні труднощі адаптації студентів і прояви механічного перенесення попередніх способів програмування, а також недостатнє використання специфічних можливостей Python.

Проведений аналіз показав, що ефективний перехід між мовами програмування не може зводитися до засвоєння нового синтаксису та механічного відтворення вже відомих алгоритмів. Встановлено, що сформовані раніше програмістські моделі можуть як прискорювати процес навчання, так і створювати перешкоди для опанування специфіки нової мови. Зокрема, студенти часто зберігають способи мислення та реалізації задач, характерні для статично типізованих мов, що проявляється у недостатньому застосуванні вбудованих функцій і високорівневих конструкцій та ігноруванні ідіоматичних засобів Python. Зроблено висновок, що формування якісних трансферних навичок має передбачати не механічне перенесення програмістських знань, а їх адаптацію до концептуальних, типізаційних, синтаксичних та ідіоматичних особливостей нової мови. Такий підхід сприяє фор-

муванню гнучкого програмістського мислення та підвищує готовність студентів до подальшого самостійного опанування нових мов і технологій.

Ключові слова: мова програмування, трансферні навички, Python.

Дата першого надходження статті до видання: 02.09.2026

Дата прийняття статті до друку після рецензування: 21.09.2026

Дата публікації (оприлюднення) статті: 27.10.2026



Стаття поширюється на умовах ліцензії відкритого доступу (CC BY 4.0)