М І Ф

Математика, Інформатика, Фізика: Наука та Освіта, Том 2, № 1 (2025), с. 169–175.
Mathematics, Informatics, Physics: Science and Education, Volume 2, No. 1 (2025), pp. 169–175.
Journal homepage: https://intranet.vspu.edu.ua/miph

УДК 004.5:378.147

Об'єктно-орієнтований підхід до створення графічного інтерфейсу користувача в Python із використанням модуля Tkinter

Галина Ковтонюк[1], Сергій Бак[2], Ярослав Крупський[3]

[1]Вінницький державний педагогічний університет імені Михайла Коцюбинського,
кафедра математики та інформатики, м. Вінниця, Україна
kovtonyukgm@vspu.edu.ua
https://orcid.org/0000-0002-3352-0358

[2]Вінницький державний педагогічний університет імені Михайла Коцюбинського,
кафедра математики та інформатики, м. Вінниця, Україна
sergiy.bak@vspu.edu.ua
https://orcid.org/0000-0003-1508-2144

[3]Вінницький державний педагогічний університет імені Михайла Коцюбинського,
кафедра математики та інформатики, м. Вінниця, Україна
krupskyi.ya@vspu.edu.ua
https://orcid.org/0000-0001-6324-2697

Анотація. У статті проаналізовано застосування об’єктно-орієнтованого підходу у процесі створення програм з графічним інтерфейсом користувача мовою Python із використанням стандартного модуля Tkinter. Об’єктом дослідження виступають підходи до розробки GUI-додатків у контексті навчання програмуванню, а також їх методичне значення для професійної підготовки майбутніх учителів математики. На основі порівняльного аналізу програмної реалізації простого калькулятора у процедурному стилі та з використанням об’єктно-орієнтованого підходу продемонстровано переваги останнього з точки зору структурованості коду, зменшення дублювання та спрощення подальшого розширення функціональності програмного продукту. У статті акцентується увага на ключових принципах об’єктно-орієнтованого програмування, таких як інкапсуляція, спадкування та поліморфізм, і їхній реалізації у Python. Підкреслено, що використання класів дозволяє ізолювати логіку GUI-компонентів, поліпшує підтримку коду та полегшує масштабування додатків. Окремий акцент зроблено на освітньому аспекті дослідження: створення GUI-додатків на базі об’єктно-орієнтованого програмування сприяє розвитку алгоритмічного, логічного та об’єктного мислення у студентів. Зазначено, що така діяльність гармонійно вписується у навчальні програми підготовки майбутніх учителів математики, зокрема в рамках дисциплін інформатичного спрямування.

Ключові слова: об’єктно-орієнтоване програмування, Python, графічний інтерфейс користувача, Tkinter.

e-ISSN 3041-1955 DOI: https://doi.org/10.31652/3041-1955-2025-02-01-05

1. Вступ

Сьогодні однією із найпопулярніших мов програмування є мова Python. В статті [1] проведено кількісний та якісний аналіз зростання популярності мови Python, використовуючи статистичні дані з GitHub, Google Trends, Stack Overflow. Автор вказує, що Python стає мовою першого вибору як в академічних колах, так і в комерційних проєктах. Основними причинами цього визнано простоту синтаксису, багату бібліотечну підтримку, кросплатформеність та активну спільноту розробників.

Крім того, Python є мовою програмування, яка активно підтримує об’єктно-орієнтованого програмування (ООП) та надає зручні механізми для створення класів, об’єктів та взаємодії між ними. ООП у Python реалізовано у гнучкій формі, що робить його зручним для моделювання та масштабованих проєктів. Об’єктно-орієнтоване програмування є одним із найбільш розповсюджених підходів у сучасному програмуванні. Воно дозволяє створювати програми, що легко масштабуються, розширюються та підтримуються. У статті [2] детально проаналізовано ООП-механізми, реалізовані в Python. Автори здійснюють концептуальне розмежування між класичним ООП (як у Java чи C++) та більш гнучким підходом Python. Акцентовано увагу на інкапсуляції, спадкуванні, поліморфізмі, а також на унікальних особливостях Python — зокрема, можливості динамічно змінювати структуру класу в процесі виконання.

Автори статті [3] дослідили застосування найпопулярніших бібліотек Python у різних доменах. Зокрема, висвітлено переваги бібліотек NumPy, Pandas, Matplotlib, TensorFlow, Flask. Встановлено, що Python має чітко структуровану екосистему для наукових обчислень, аналізу даних, машинного навчання та веб-розробки.

Одна з важливих областей застосування ООП у Python — створення графічного інтерфейсу користувача (GUI). Tkinter є стандартним модулем Python для створення GUI-програм, і його використання спрощує розробку застосунків. Інтерфейсні рішення на основі Tkinter залишаються актуальними в навчальному контексті.

Джон Ґрейсон у своїй праці "Python and Tkinter Programming" ([4]) заклав фундаментальні засади побудови графічного інтерфейсу засобами бібліотеки Tkinter. Попри дату публікації, видання зберігає актуальність для початкового рівня розробки GUI-програм. Автор пропонує покроковий розбір принципів обробки подій, управління віджетами та структури вікон.

Зауважимо, що зараз є багато різноманітних посібників вітчизняних авторів, в яких доступно і достатньо повно викладені основи програмування мовою Python (див., наприклад, [5, 6]).

2. Постановка проблеми

В статті [7] наголошується на важливості інформатичних компетентностей майбутніх учителів фізико-математичних дисциплін, зокрема навичок з програмування. Цю тему розвинено в статті [8], де розглядаються можливості створення GUI під час вивчення програмування майбутніми вчителями математики. Акцентується увага на методичних підходах до використання Python у навчанні, підкреслюючи його дидактичну ефективність. Автори демонструють, як створення графічного інтерфейсу сприяє формуванню алгоритмічного та логічного мислення у студентів.

Метою даної статті є демонстрація ефективності застосування ООП у програмах з графічним інтерфейсом користувача, що використовують модуль Tkinter, а також порівняння підходів із використанням та без використання класів в контексті вивчення програмування.

3. Основні результати

Нагадаємо, що «графічний інтерфейс користувача (англ. Graphical User Interface або GUI) – тип інтерфейсу, який дозволяє користувачеві взаємодіяти з комп'ютерною програмою або електронним пристроєм через графічні зображення та візуальні вказівки, на відміну від текстових інтерфейсів» ([8]).

У Python для створення графічних інтерфейсів часто використовують бібліотеку Tkinter. Можна реалізувати графічний інтерфейс як у процедурному стилі, так і за допомогою об'єктно-орієнтованого підходу. Розглянемо обидва підходи на прикладі створення простого калькулятора.

1. Створення графічного інтерфейсу користувача без використання ООП (процедурний стиль)

Створити програму з графічним інтерфейсом користувача у Python за допомогою модуля Tkinter можна найпростішим способом – написавши код у процедурному стилі, тобто без використання класів. У цьому підході потрібно створити головне вікно, додати віджети (кнопки, мітки тощо) та керувати ними за допомогою окремих функцій або безпосередньо в основному потоці виконання. Це підходить для невеликих програм, але стає важким для підтримки та масштабування, коли програма зростає.

Розглянемо приклад простої програми без використання класів (рис. 1).

```
import tkinter as tk

def on_button_click(button_text):
    current_text = result_var.get()

    if button_text == "=":
        try:
            result = eval(current_text)
            result_var.set(result)
        except Exception as e:
            result_var.set("Помилка")
    elif button_text == "C":
        result_var.set("")  # Очистити екран
    else:
        result_var.set(current_text + button_text)

# Створення головного вікна Tkinter
root = tk.Tk()
root.title("Калькулятор")
root.geometry("300x400")

result_var = tk.StringVar()

result_label = tk.Label(root, textvariable=result_var, height=2, font=("Arial", 20))
result_label.pack()

# Створення кнопок для калькулятора
button_frame = tk.Frame(root)
button_frame.pack()

buttons = [
    ('7', '8', '9', '/'),
    ('4', '5', '6', '*'),
    ('1', '2', '3', '-'),
    ('0', '.', '=', '+'),
    ('C', '(', ')', '%')  # Новий рядок кнопок для додаткових функцій
]

for row in buttons:
    row_frame = tk.Frame(button_frame)
    row_frame.pack(fill="x")
    for button_text in row:
        button = tk.Button(row_frame, text=button_text, width=5, height=2, font=("Arial", 14),
                           command=lambda text=button_text: on_button_click(text))
        button.pack(side="left")

root.mainloop()
```

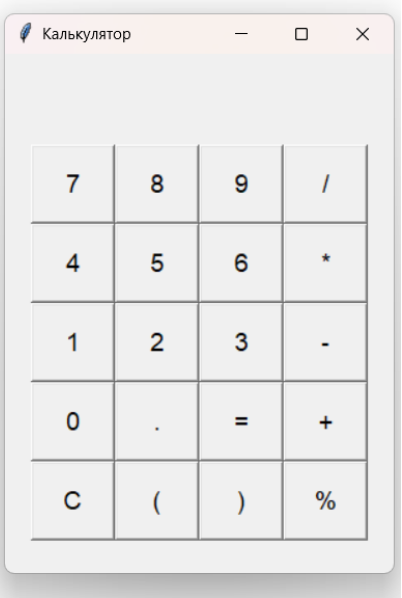

Рис. 1. Створення базового GUI з Tkinter без ООП

У програмі без ООП створюється головне вікно (root), додаються віджети, такі як мітка (Label), кнопки (Button). Кнопка викликає подію, яка змінює текст мітки.

Недоліки цього підходу полягають у тому, що всі елементи створюються в глобальному просторі, що може призвести до конфліктів змінних, якщо програма стане більш складною. Якщо потрібно додавати нові елементи або змінювати інтерфейс, код стає важким для підтримки та розширення. Крім того, для створення схожих елементів потрібно повторювати код, що збільшує кількість дубльованих фрагментів і ускладнює внесення змін. Відсутність інкапсуляції робить програму менш гнучкою, оскільки всі

частини програми можуть взаємодіяти одна з одною без чітких меж, що збільшує ризик помилок.

Для подолання цих проблем можна застосувати об'єктно-орієнтований підхід, що дозволяє структурувати код, ізолювати елементи інтерфейсу в окремі класи та забезпечити кращу підтримку та масштабованість.

Розглянемо приклад програми з використанням об'єктно-орієнтованого підходу (ООП) для створення графічного інтерфейсу з використанням модуля Tkinter.

2. Створення програми з використанням ООП (об'єктно-орієнтований підхід)

У об'єктно-орієнтованому підході всі елементи інтерфейсу та їх взаємодія з користувачем інкапсульовані в окремі класи. Це дозволяє створювати більш структуровані і масштабовані програми.

Тепер розглянемо приклад простої програми з використанням класів (рис. 2).

```python
import tkinter as tk
class CalculatorButton:
    def __init__(self, parent, text, command):
        self.button = tk.Button(parent, text=text, width=5, height=2, font=("Arial", 14), command=command)
        self.button.pack(side="left")
class CalculatorApp:
    def __init__(self, root):
        self.root = root
        self.result_var = tk.StringVar()
        self.result_label = tk.Label(root, textvariable=self.result_var, height=2, font=("Arial", 20))
        self.result_label.pack()

        self.button_frame = tk.Frame(root)
        self.button_frame.pack()

        self.buttons = [
            ('7', '8', '9', '/'),
            ('4', '5', '6', '*'),
            ('1', '2', '3', '-'),
            ('0', '.', '=', '+'),
            ('C', '(', ')', '%')  # Новий рядок кнопок для додаткових функцій
        ]

        self.create_buttons()
    def create_buttons(self):
        for row in self.buttons:
            row_frame = tk.Frame(self.button_frame)
            row_frame.pack(fill="x")
            for button_text in row:
                CalculatorButton(row_frame, button_text, lambda text=button_text: self.on_button_click(text))

    def on_button_click(self, button_text):
        current_text = self.result_var.get()

        if button_text == "=":
            try:
                result = eval(current_text)
                self.result_var.set(result)
            except Exception as e:
                self.result_var.set("Помилка")
        elif button_text == "C":
            self.result_var.set("")  # Очистити екран
        else:
            self.result_var.set(current_text + button_text)
# Створення головного вікна Tkinter
root = tk.Tk()
root.title("Калькулятор")
```

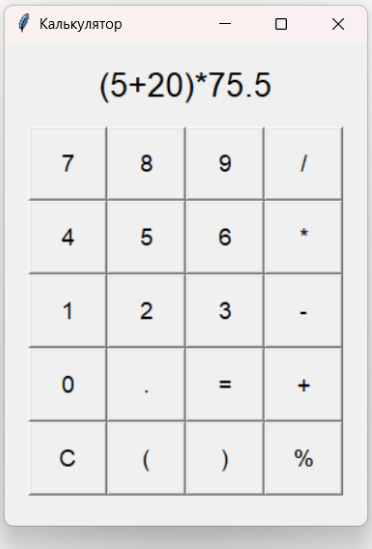

Рис. 2. Створення базового GUI з Tkinter з використанням ООП

У програмі з використанням об'єктно-орієнтованого підходу всі елементи інтерфейсу, такі як мітка (Label), кнопки (Button), а також їх взаємодія з користувачем, інкапсульовані в окремому класі. Це дозволяє створити чітку структуру коду, де кожен клас відповідає за конкретний елемент інтерфейсу або групу елементів. Так, створення об'єкта класу забезпечує більшу гнучкість та масштабованість програми, оскільки можна безперешкодно додавати нові елементи або змінювати вже існуючі без впливу на інші частини програми.

У наведеному прикладі створюється клас CalculatorApp, який інкапсулює всю логіку програми та взаємодію з користувачем. Така організація коду дозволяє зберігати всю інформацію про інтерфейс і його елементи в одному місці, що полегшує подальші зміни та підтримку програми.

У методі __init__ цього класу створюються всі елементи інтерфейсу, такі як кнопки та мітки. Цей метод забезпечує ініціалізацію основних компонентів програми та їх зв’язок з функціональними частинами.

Метод create_buttons використовується для створення кнопок калькулятора. Це дозволяє уникнути дублювання коду при додаванні нових кнопок або зміні їхнього

розташування. Завдяки методу створення кнопок у межах класу, додавання нових елементів інтерфейсу стає значно простішим.

Логіка обробки натискання кнопок залишається в методі on_button_click, але тепер вона працює з даними класу, що дозволяє забезпечити кращу організацію коду і уникнути конфліктів змінних.

Завдяки об'єктно-орієнтованому підходу, додавання нових кнопок стає простим і зручним.

Приклад програми з використанням ООП дозволяє значно зменшити кількість повторюваного коду, оскільки клас можна використовувати для створення різних інтерфейсів із подібною структурою. Завдяки цьому код стає більш чистим і легким для підтримки. ООП також дозволяє зберігати логіку програми та графічний інтерфейс у межах одного класу, що робить програму легшою для розуміння та модифікації.

У порівнянні з процедурним підходом, об'єктно-орієнтований підхід надає можливість інкапсуляції функцій і даних, що призводить до кращої організації коду і зменшує ризик виникнення помилок. Крім того, для великої кількості інтерфейсів, які можуть мати спільні елементи, ООП дозволяє використовувати успадкування та поліморфізм, що знову ж таки робить програму більш гнучкою та зручною для розширення.

Таким чином, використання ООП в Tkinter значно покращує структуру програми, робить її більш організованою та легшою для розширення і підтримки.

Зауважимо, що зазначена тематика включена до чинних навчальних програм з інформатики та програмування для спеціальностей 111 Математика (освітньо-професійна програма «Комп’ютерна математика») та 014.04 Середня освіта (Математика) (освітньо-професійна програма «Середня освіта. Математика, інформатика»), схвалених Вченою радою Вінницького державного педагогічного університету імені Михайла Коцюбинського.

Висновки. Таким чином, використання об’єктно-орієнтованого підходу у створенні програм з графічним інтерфейсом з використанням модуля Tkinter забезпечує вищу читабельність коду завдяки його структурованості, на відміну від процедурного підходу, де код швидко ускладнюється зі збільшенням функціональності. У контексті повторного використання коду ООП значно спрощує додавання нових графічних вікон, тоді як у процедурному варіанті це вимагає суттєвого дублювання та модифікації існуючих фрагментів коду. При цьому розширюваність програмного продукту також істотно підвищується, оскільки цей підхід дозволяє легко доповнювати функціональність без радикального переписування програми, чого не можна сказати про процедурний підхід. Однак з огляду на логіку формування програмістських компетентностей, оволодіння об’єктно-орієнтованим програмуванням передбачає попереднє вивчення процедурного підходу як необхідної методологічної основи. Подальші дослідження можуть бути зосереджені на вивченні особливостей створення складних GUI-додатків із використанням ООП та розширених бібліотек, таких як PyQt або Kivy.

Конфлікт інтересів і етика. Автори заявляють, що не мають конфліктів інтересів. Автори також заявляють про повне дотримання всіх правил етики журнальних досліджень, а саме щодо анонімності участі людей та/або згоди на публікацію.

Подяки. Автори заявляють про відсутність спеціального фінансування цієї роботи.

Список використаних джерел

1. Srinath K. R. Python – The Fastest Growing Programming Language. *International Research Journal of Engineering and Technology*. 2017. Volume 4, Issue 12. P. 354–357.

2. Nzerue-Kenneth P. E., Onu F. U., Denis A. U., Igwe J. S., Ogbu N. H. Detailed Study of the Object-Oriented Programming (OOP) Features in Python. *British Journal of Computer, Networking and Information Technology*. 2023. Volume 6, № 1. P. 83–93. DOI: https://doi.org/10.52589/BJCNIT-FACSOJAO
3. Saabith A. L. S., Vinothraj T., Fareez M. M. M. Popular Python libraries and their application domains. *International Journal of Advance Engineering and Research Development*. 2020. Volume 7, Issue 11. P. 18–26.
4. Grayson J. E. Python and Tkinter Programming. Shelter Island: Manning Publications Co., 2000. 660 p.
5. Креневич А. П. Python у прикладах і задачах. Ч. 2. Об'єктно-орієнтоване програмування: навч. посіб. Київ: ВПЦ «Київський університет», 2020. 152 с.
6. Руденко В. Д., Жугастров О. О. Основи алгоритмізації і програмування мовою Python. Харків: Вид-во «Ранок», 2019. 192 с.
7. Ковтонюк Г. М. До питання формування інформатичної компетентності майбутніх учителів фізико-математичних дисциплін. *Нова педагогічна думка*. 2017. Том 91, № 3. С. 49–51.
8. Бак С. М., Ковтонюк Г. М. Особливості створення графічного інтерфейсу користувача під час вивчення програмування мовою Python майбутніми вчителями математики. *Сучасні інформаційні технології та інноваційні методики навчання у підготовці фахівців: методологія, теорія, досвід, проблеми*: збірник наукових праць. Вінниця: ТОВ «Друк плюс», 2021. Вип. 60. С. 143–157. DOI: https://doi.org/10.31652/2412-1142-2021-60-143-157

UDC 004.5:378.147

Object-oriented approach to creating graphical user interface in Python using the Tkinter module

Halyna Kovtoniuk, Serhii Bak, Yaroslav Krupskyi

Abstract. The article analyzes the application of the object-oriented approach in the development of programs with a graphical user interface using the Python programming language and its standard Tkinter module. The focus of the research is on approaches to GUI application development in the context of teaching programming, as well as their methodological significance for the professional training of future mathematics teachers. Based on a comparative analysis of the implementation of a simple calculator in procedural style and using the object-oriented approach, the advantages of the latter are demonstrated in terms of code structuring, reduction of redundancy, and simplification of further functionality extension. The article highlights the key principles of object-oriented programming such as encapsulation, inheritance, and polymorphism, and their implementation in Python. It is emphasized that the use of classes allows isolating the logic of GUI components, improves code maintainability, and facilitates application scalability. Particular attention is given to the educational aspect of the study: the development of GUI applications based on object-oriented programming fosters the development of algorithmic, logical, and object-oriented thinking in students. It is noted that such activities are well aligned with the curricula for training future mathematics teachers, particularly within informatics-related disciplines.

Keywords: object-oriented programming, Python, graphical user interface, Tkinter.

References

1. Srinath, K. R. (2017). *Python – The Fastest Growing Programming Language*, International Research Journal of Engineering and Technology, **4** (12), 354–357.
2. Nzerue-Kenneth, P. E., Onu, F. U., Denis, A. U., Igwe, J. S., Ogbu, N. H. (2023), *Detailed Study of the Object-Oriented Programming (OOP) Features in Python*, British Journal of Computer, Networking and Information Technology, **6** (1), 83–93. https://doi.org/10.52589/BJCNIT-FACSOJAO
3. Saabith, A. L. S., Vinothraj, T., Fareez, M. M. M. (2020). *Popular Python libraries and their application domains*, International Journal of Advance Engineering and Research Development, **7** (11), 18–26.
4. Grayson, J. E. (2000). *Python and Tkinter Programming*, Manning Publications Co., Shelter Island.
5. Krenevych, A. P. (2020). *Python in examples and problems*, Part 2, Object-oriented programming: a textbook, Kyiv University, Kyiv. [in Ukrainian]
6. Rudenko, V. D., Zhugastrov, O. O. (2019). *Fundamentals of algorithmization and programming in Python*, Publ. "Ranok", Kharkiv. [in Ukrainian]
7. Kovtoniuk, H. M. (2017). *On the issue of forming informatics competence of future teachers of physics and mathematics disciplines*, Nova Pedagogichna Dumka, **91** (3), 49–51. [in Ukrainian]

8. Bak, S. M., Kovtoniuk, H. M. (2021). *Features of creating a graphical user interface when studying Python programming by future mathematics teachers*, Modern information technologies and innovative teaching methods in the training of specialists: methodology, theory, experience, problems: collection of scientific papers, LLC "Druk Plus", Vinnytsia, **60**, 143–157. [in Ukrainian]. https://doi.org/10.31652/2412-1142-2021-60-143-157

Про авторів / About the authors

Галина Ковтонюк, кандидат педагогічних наук, доцент, кафедра математики та інформатики, Вінницький державний педагогічний університет імені Михайла Коцюбинського, вул. Острозького, 32, м. Вінниця, 21001, Україна;

Halyna Kovtoniuk, Candidate of Science in Pedagogy, Associate Professor, Department of Mathematics and Informatics, Vinnytsia Mykhailo Kotsiubynskyi State Pedagogical University, 32 Ostrozkyi Str., Vinnytsia 21001, Ukraine;

Сергій Бак, доктор фізико-математичних наук, професор, кафедра математики та інформатики, Вінницький державний педагогічний університет імені Михайла Коцюбинського, вул. Острозького, 32, м. Вінниця, 21001, Україна;

Serhii Bak, Doctor of Science in Physics and Mathematics, Professor, Department of Mathematics and Informatics, Vinnytsia Mykhailo Kotsiubynskyi State Pedagogical University, 32 Ostrozkyi Str., Vinnytsia 21001, Ukraine;

Ярослав Крупський, кандидат педагогічних наук, доцент, кафедра математики та інформатики, Вінницький державний педагогічний університет імені Михайла Коцюбинського, вул. Острозького, 32, м. Вінниця, 21001, Україна;

Yaroslav Krupskyi, Candidate of Science in Pedagogy, Associate Professor, Department of Mathematics and Informatics, Vinnytsia Mykhailo Kotsiubynskyi State Pedagogical University, 32 Ostrozkyi Str., Vinnytsia 21001, Ukraine.

Отримано / Received 19.03.2025
Прийнято до друку / Accepted 18.04.2025
Опубліковано / Published 21.05.2025