МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ПРИКАРПАТСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАСИЛЯ СТЕФАНИКА

Мазуренко В.В., Дмитришин М.І., Василишин П.Б.

ОБ'ЄКТНО–ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ з PYTHON

ЛАБОРАТОРНИЙ ПРАКТИКУМ

Івано-Франківськ — 2023

УДК 004.43
ББК 32.973-018.1
М13

Рекомендовано Вченою радою факультету математики та інформатики Прикарпатського національного університету імені Василя Стефаника (протокол № 10 від 21.11.2023 р.).

Рецензенти:

Олександр МАХНЕЙ, кандидат фізико-математичних наук, доцент (Прикарпатський національний університет імені Василя Стефаника, Івано-Франківськ);

Роман ДМИТРИШИН, доктор фізико-математичних наук, професор (Прикарпатський національний університет імені Василя Стефаника, Івано-Франківськ).

М13 Мазуренко В.В., Дмитришин М.І., Василишин П.Б. Об'єктно–орієнтоване програмування з Python: Лабораторний практикум. – Ів.-Фр.: Голіней, 2023. – 48 с.

У лабораторному практикумі наведено методичні рекомендації до виконання лабораторних робіт з об'єктно-орієнтованого програмування мовою Python.

Для студентів спеціальності «прикладна математика». Практикум буде корисним також для студентів галузей знань «математика і статистика» та «інформаційні технології».

ЗМІСТ

ПЕРЕДМОВА

В об'єктно-орієнтованому стилі програмування (ООП) будь-яку програму розглядають як сукупність «об'єктів», які взаємодіють між собою і кожен з яких є екземпляром якогось «класу». Така парадигма програмування є ефективною при розробці складних програмних проєктів. Програми, написані в об'єктно-орієнтованому стилі, мають більшу модульність і гнучкість, оптимізованість коду і читабельність, захищеність даних і адаптованість до змін.

Лабораторний практикум охоплює важливі аспекти ООП, починаючи від базових понять, таких як класи, об'єкти та атрибути, і завершуючи високорівневими поняттями, такими як абстрактні класи, інтерфейси та шаблони проєктування. Здобувачі освіти мають можливість поглибити свої навики зі створення функцій і методів, використання спеціальних полів і магічних методів, а також розуміння ключових концепцій ООП: абстракції, інкапсуляції, успадкування і поліморфізму. Вивчення шаблонів ООП на основі абстрактних класів та інтерфейсів стимулює творчий підхід до розробки програмного забезпечення та розвиває навики реалізації архітектурно складних програм.

Практикум зорієнтований на використання мови програмування Python. Це високорівнева, інтерпретована мова програмування, яка відзначається простотою синтаксису і читабельністю коду. Її універсальність у поєднанні з численними бібліотеками робить Python ідеальним вибором для широкого спектру завдань — від автоматизації задач і наукових обчислень, аж до інтелектуального аналізу даних і веб-розробки. Водночас варто відзначити, що завдання лабораторного практикуму є інваріантними стосовно мови програмування і за потреби можуть бути використані для вивчення ключових концепцій і практик ООП іншими мовами.

Практикум комбінує індивідуальні лабораторні роботи (для кожної з яких вказана тема, короткий перелік необхідних для виконання роботи теоретичних знань, 15 варіантів завдань та вимоги до оформлення звіту) разом з лабораторними роботами для виконання у команді з двох-чотирьох осіб, що сприяють, зокрема, формуванню м'яких навиків у здобувачів освіти.

.

Лабораторна робота 1

СТВОРЕННЯ ФУНКЦІЙ

I. Мета лабораторної роботи

формування навиків створення і використання функцій мовою Python

II. Для виконання лабораторної роботи необхідно знати

- що таке підпрограми (процедури і функції), основні етапи опису функцій
- як повернути кілька значень однією функцією
- що таке формальні і фактичні аргументи функції і як відбувається їх передача у функцію
- що таке позиційне і ключове співставлення формальних і фактичних аргументів функції
- як задавати типові (за замовчуванням) значення аргументів
- як описуються функції з довільною кількістю аргументів
- яка різниця між локальними і глобальними змінними у тілі функції і які області їх видимості
- що таке анонімні (lambda-) і рекурсивні функції та функції-генератори
- призначення і опис декораторів для функцій

РЕКОМЕНДОВАНА ЛІТЕРАТУРА: [**2**, п. 3.1-9], [**3**, п. 10.1-10], [**5**, п. 6.6]
ДОДАТКОВА ЛІТЕРАТУРА: [**8, 9**]

III. Завдання на лабораторну роботу

Написати програми для виконання наступних завдань (***номер варіанту є порядковим номером студента в електронному журналі академгрупи***).

Варіант 1.

1. Для введеного натурального числа n обчисліть добуток перших n чисел Фібоначчі. У програмі описати і використати функцію `phibonacci` (k), яка повертає k-те число Фібоначчі.
2. Послідовність натуральних чисел завершується введенням 0 (який не належить до послідовності). Описати і застосувати рекурсивну функцію для

Лабораторна робота 2

КЛАСИ, ОБ'ЄКТИ, АТРИБУТИ

I. Мета лабораторної роботи

формування навиків створення класів, об'єктів (екземплярів) класу та методів мовою Python

II. Для виконання лабораторної роботи необхідно знати

- що таке клас і об'єкт (екземпляр) класу, шаблон класу
- як ініціалізувати (створити) об'єкт з допомогою конструктора
- як деініціалізувати (знищити) об'єкт з допомогою деструктора
- що таке атрибути (поля і методи) об'єкта і класу
- чим метод відрізняється від функції і для чого призначений вказівник self
- спосіб виклику атрибутів
- як створюються поверхневі і повні копії об'єктів

РЕКОМЕНДОВАНА ЛІТЕРАТУРА: [**2**, п. 6], [**3**, п. 14.1-3], [**4**, п. 1.1], [**5**, п. 10.1-3,6]
ДОДАТКОВА ЛІТЕРАТУРА: [**8, 9**]

III. Завдання на лабораторну роботу

Написати програми для виконання наступних завдань (***номер варіанту є порядковим номером студента в електронному журналі академгрупи***).

Структурою-парою називають клас з двома полями, які зазвичай мають імена `first` і `second`.

У кожному варіанті потрібно:

- реалізувати представлення даних структурою-парою з такими методами:
  – конструктор `__init__` має контролювати значення аргументів на коректність (для перевірки приналежності об'єкта певному класу даних можна використати функцію `isinstance(Object, Class)`, яка повертає логічне значення `True`, в разі якщо `Object` **in** `Class`, або `False`, якщо `Object` **not in** `Class`; для перевірки істинності тверджень можна використовувати інструкцію `assert Condition, Message`, яка генерує повідомлення

Лабораторна робота 3

КОНЦЕПЦІЯ ІНКАПСУЛЯЦІЇ

I. Мета лабораторної роботи

формування навиків застосування концепції інкапсуляції в об'єктно-орієнтованому програмуванні мовою Python

II. Для виконання лабораторної роботи необхідно знати

- що таке інкапсуляція та інтерфейс класу
- як оголошуються публічні і приватні атрибути і методи класу
- механізми доступу до приватних атрибутів і методів
- що таке статичні (класові) атрибути і методи і як їх викликати
- що таке UML-діаграма класів
- які існують типи зв'язків між класами

РЕКОМЕНДОВАНА ЛІТЕРАТУРА: [**3**, п. 14.2,6], [**4**, п. 1.2-5], [**5**, п. 10.2.3]
ДОДАТКОВА ЛІТЕРАТУРА: [**8, 9**]

III. Завдання на лабораторну роботу

Працюючи у парах (***відповідний варіант виконує пара студентів згідно з таблицею нижче***), написати об'єктно-орієнтовану програму для виконання відповідного завдання.

Таблиця 1 Варіанти завдань

Варіант	**Номер студента**	
1	1	5
2	3	6
3	4	8
4	7	9
5	11	2
6	12	10
7	13	14

Лабораторна робота 4

КОНЦЕПЦІЯ УСПАДКУВАННЯ

I. Мета лабораторної роботи

формування навиків застосування концепції успадкування (наслідування) в об'єктно-орієнтованому програмуванні мовою Python

II. Для виконання лабораторної роботи необхідно знати

- що таке успадкування та ієрархія класів
- шаблон опису підкласу (дочірній клас / клас-нащадок) на основі базового класу (батьківський клас / клас-предок)
- механізми заміщення (перевизначення) методів
- особливості роботи з методом–конструктором базового класу у підкласі і призначення функції super()
- способи виклику оригінального і заміщеного методів
- що таке суперклас object
- як зображається успадкування на UML-діаграмах класів
- що таке множинне успадкування і які проблеми з ним пов'язані

РЕКОМЕНДОВАНА ЛІТЕРАТУРА: [**2**, п. 7.1], [**3**, п. 14.4], [**4**, п. 2.1,3-4], [**5**, п. 10.4,5]
ДОДАТКОВА ЛІТЕРАТУРА: [**8, 9**]

III. Завдання на лабораторну роботу

Працюючи у трійках (***відповідний варіант виконують троє студентів згідно з таблицею нижче***), написати об'єктно-орієнтовану програму для виконання відповідного завдання.

Таблиця 1 Варіанти завдань

Варіант	**Номер студента**		
1	1	11	6
2	4	9	13
3	7	3	5
4	12	2	8
5	10	14	15

Лабораторна робота 5

КОНЦЕПЦІЯ ПОЛІМОРФІЗМУ

I. Мета лабораторної роботи

формування навиків застосування концепції поліморфізму в об'єктно-орієнтованому програмуванні мовою Python

II. Для виконання лабораторної роботи необхідно знати

- що таке поліморфізм
- що таке динамічне зв'язування об'єктів і методів (віртуальність)
- суть способу реалізації поліморфізму з допомогою віртуальних методів

РЕКОМЕНДОВАНА ЛІТЕРАТУРА: [**3**, п. 14.5], [**4**, п. 2.2]
ДОДАТКОВА ЛІТЕРАТУРА: [**8, 9**]

III. Завдання на лабораторну роботу

Працюючи у четвірках ***(відповідний варіант виконують четверо студентів згідно з таблицею нижче)***, написати об'єктно-орієнтовану програму для виконання відповідного завдання.

Таблиця 1 Варіанти завдань

Варіант	**Номер студента**			
1	6	12	5	13
2	7	4	9	11
3	3	2	8	1
4	10	14	15	16

У кожному варіанті потрібно:

- побудувати UML-діаграму класів, на якій слід відобразити атрибути (поля і методи) класів та ієрархію класів: приватність і публічність атрибутів встановлювати самостійно; спільні для класів-нащадків поля і методи слід описувати у базовому класі; методи, реалізація яких залежить від властивостей класу-нащадка, слід описувати у всіх класах-нащадках, а в базовому класі оголосити ці методи як (чисті) віртуальні методи;

Лабораторна робота 6

СПЕЦІАЛЬНІ МЕТОДИ І ПОЛЯ

I. Мета лабораторної роботи

формування навиків роботи зі спеціальними (магічними) методами і полями в об'єктно-орієнтованому програмуванні мовою Python

II. Для виконання лабораторної роботи необхідно знати

- основне призначення спеціальних полів і методів
- які їх ознаки і в чому проявляється "магічність" спеціальних методів
- спеціальні поля для доступу до інформації про клас
- призначення магічних методів `__call__` і `__abs__`
- магічні методи перетворення типів
- магічні методи класів–колекцій
- магічні методи для перевантаження арифметичних операторів
- магічні методи для перевантаження операторів порівняння
- магічні методи для перевантаження операторів присвоєння
- магічні методи для перевантаження унарних операторів

РЕКОМЕНДОВАНА ЛІТЕРАТУРА: [**2**, п. 7.2], [**3**, п. 14.8], [**4**, п. 3.1-3]
ДОДАТКОВА ЛІТЕРАТУРА: [**8, 9**]

III. Завдання на лабораторну роботу

Працюючи у четвірках (***номер варіанту виконує четвірка студентів відповідно до таблиці нижче***), написати об'єктно-орієнтовану програму для виконання відповідного завдання.

Таблиця 1 Варіанти завдань

Варіант	**Номер студента**			
1	6	12	5	13
2	7	4	9	11
3	3	2	8	1
4	10	14	15	16

Лабораторна робота 7

АБСТРАКТНІ КЛАСИ І ШАБЛОНИ

I. Мета лабораторної роботи

формування навиків застосування абстрактних класів та інтерфейсів для реалізації шаблонів об'єктно-орієнтованого програмування мовою Python

II. Для виконання лабораторної роботи необхідно знати

- що таке абстрактний клас і як його створити
- яким чином оголошуються абстрактні методи
- для чого призначені шаблони (патерни) об'єктно–орієнтованого програмування
- UML-діаграму і логіку функціонування шаблону Visitor (Відвідувач)
- що таке класи-інтерфейси і чим вони відрізняються від абстрактних класів
- способи створення інтерфейсів
- UML-діаграму і логіку функціонування шаблону Observer (Спостерігач)
- для чого призначені класи-домішки

РЕКОМЕНДОВАНА ЛІТЕРАТУРА: [**1**], [**4**, п. 5.1-3], [**6**]
ДОДАТКОВА ЛІТЕРАТУРА: [**7**–**9**]

III. Завдання на лабораторну роботу

Описати вказаний у таблиці нижче (***номер варіанту є порядковим номером студента в електронному журналі академгрупи***) шаблон (патерн) об'єктно-орієнтованого проєктування за такою схемою:

- ***Тип шаблону***: *породжуючий*, *структурний*, *поведінковий*
- ***Суть (призначення)***
- ***Проблема (мотивація)***
- ***Вирішення (застосовність)***
- ***Аналогія з життя***
- ***Структура (UML-діаграма)***
- ***Схема реалізації на Python***
- ***Переваги і недоліки***
- ***Споріднені шаблони***

Таблиця 1 Варіанти завдань

Варіант	**Шаблон (укр.)**	**Шаблон (англ.)**	**Складність**
1	Одинак	Singleton	●○○
2	Посередник	Mediator, Controller	●●○
3	Фасад	Facade	●○○
4	Будівельник	Builder	●●○
5	Шаблонний метод	Template Method	●○○
6	Декоратор	Decorator, Wrapper	●●○
7	Стратегія	Strategy	●○○
8	Фабричний метод	Factory Method	●○○
9	Команда	Command, Action	●●○
10	Ланцюг обов'язків	Chain of Responsibility	●●○
11	Перехідник	Adapter	●○○
12	Компонувальник	Composite	●●○
13	Стан	State	●○○
14	Ітератор	Iterator	●●○
15	Прототип	Prototype	●○○
16	Абстрактна фабрика	Abstract Factory	●●○
17	Спостерігач	Observer	●●○
18	Знімок	Memento	●●●
19	Міст	Bridge	●●●
20	Відвідувач	Visitor	●●●

IV. Звіт по лабораторній роботі

Зміст звіту:

1. Титульний аркуш зі стандартними атрибутами
2. Код реалізації шаблону з коментарями
3. Результат реалізації шаблону
4. Висновки

Лабораторна робота 8

ООП У ПРЕДМЕТНІЙ ОБЛАСТІ

I. Мета лабораторної роботи

формування навиків проектування класів для вирішення проблем певної предметної області з допомогою об'єктно-орієнтованого програмування мовою Python

II. Для виконання лабораторної роботи необхідно знати

- теоретичні основи об'єктно-орієнтованого програмування

РЕКОМЕНДОВАНА ЛІТЕРАТУРА: [**1**–**6**]
ДОДАТКОВА ЛІТЕРАТУРА: [**7**–**9**]

III. Завдання на лабораторну роботу

Працюючи у парах (***відповідний варіант виконує пара студентів згідно з таблицею нижче***), написати об'єктно-орієнтовану програму для виконання відповідного завдання.

Таблиця 1 Варіанти завдань

Варіант	**Номер студента**	
1	1	13
2	11	2
3	3	8
4	7	4
5	5	9
6	12	6
7	10	14
8	15	16
9	17	18
10	19	20

Варіант 1. Предметна область: **Вокзал**.

Використовуючи UML-діаграми, спроєктуйте об'єктно-орієнтовану модель роботи вокзалу. На основі принципів ООП реалізуйте класи, які представлятимуть різні об'єкти та їх взаємодію на вокзалі.

1. «Station»:
   - поля: список потягів, які прибули або відправляються з вокзалу.
   - методи: конструктор, приєднання потяга до вокзалу, відображення розкладу, прибуття та відправлення потягів.
2. «Train»:
   - поля: номер і тип потяга, станції відправлення і призначення.
   - методи: конструктор, додавання пасажирів до потяга, час відправлення потяга, відображення інформації про потяг.
3. «Schedule»:
   - поля: дні тижня, години прибуття і відправлення потягів.
   - методи: конструктор, відображення розкладу.
4. «Passenger»:
   - поля: ім'я, прізвище, номер студентського/учнівського квитка.
   - методи: конструктор, відображення інформації про пасажира.
5. «Ticket»:
   - поля: номер і тип квитка, вартість квитка.
   - методи: конструктор, відображення інформації про квиток.

Протестуйте функціональність класів. Створіть об'єкти цих класів і продемонструйте їх взаємодію на прикладі реєстрації пасажира, придбання квитка, відправлення потяга, виведення інформації про станцію тощо. Розгляньте можливість додаткого функціоналу, як наприклад, пошук потягів за місцем призначенням, виведення списку пасажирів на потязі тощо.

Варіант 2. Предметна область: **Банк**.

Використовуючи UML-діаграми, спроєктуйте об'єктно-орієнтовану модель роботи банку. На основі принципів ООП реалізуйте класи, які представлятимуть різні об'єкти та їх взаємодію у банку.

1. «Bank»:
   - поля: список клієнтів банку та їх рахунків.
   - методи: конструктор, відкриття нового рахунку для клієнта, виведення інформації про всі рахунки в банку.
2. «Client»:
   - поля: ім'я, прізвище, номер рахунку, адреса.
   - методи: конструктор, відображення інформації про клієнта.

СПИСОК ЛІТЕРАТУРИ

[1] Будай А. Дизайн-патерни — просто як двері. – Ел. вид., 2012. – Режим доступу: https://sites.google.com/site/designpatternseasy

[2] Васильєв О.М. Програмування мовою Python. – Тернопіль: Навчальна книга — Богдан, 2019.

[3] Висоцька В. А., Оборська О. В. Python: алгоритмізація та програмування: навчальний посібник. – Львів: Новий Світ — 2000, 2021.

[4] Креневич А.П. Python у прикладах і задачах. Частина 2. Об'єктно-орієнтоване програмування: Навчальний посібник. – К.: ВПЦ "Київський університет", 2020.

[5] Мізюк О. Путівник мовою програмування Python. – Ел. вид., 2020. – Режим доступу: https://pythonguide.rozh2sch.org.ua

[6] Швець О. Занурення в патерни проектування. – Ел. вид., 2021. – Режим доступу: https://refactoring.guru/uk/design-patterns/book

[7] Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns. Elements of Reusable Object-Oriented Software. – Addison-Wesley Professional, 1994.

[8] PEP 8 — Style Guide for Python Code. – Access mode: https://peps.python.org/pep-0008/#source-file-encoding

[9] The Python Tutorial. – Access mode: https://docs.python.org/3/tutorial/index.html