МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Волинський національний університет імені Лесі Українки
Кафедра комп’ютерних наук та кібербезпеки

Глинчук Л. Я.

ПРОГРАМУВАННЯ: ЛАБОРАТОРНИЙ ПРАКТИКУМ ЗМІСТОВОГО МОДУЛЮ «ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ МОВОЮ PYTHON»

для студентів спеціальностей 122 Комп’ютерні науки та 125 Кібербезпека та захист інформації першого (бакалаврського) рівня

Луцьк 2024

УДК 004.43(072)

Рекомендовано до видання науково-методичною радою Волинського національного університету імені Лесі Українки (протокол № 3 від 22 листопада 2024 р.)

Рецензенти:

Пастернак В.В. – кандидат технічних наук, доцент кафедри загальної математики та методики навчання інформатики Волинського національного університету імені Лесі Українки;

Багнюк Н.В. – кандидат технічних наук, доцент кафедри комп'ютерної інженерії та кібербезпеки Луцького національного технічного університету.

Г 54 Програмування : лабораторний практикум змістового модулю «Об'єктно-орієнтоване програмування мовою Python». [Електронний ресурс] / укладач Л. Я. Глинчук; ВНУ ім. Лесі Українки. Електронні текстові данні (1 файл: 1 273 КБ). Луцьк : ВНУ ім. Лесі Українки, 2024. 73 с.

Практикум призначений для студентів спеціальностей 122 Комп'ютерні науки та 125 Кібербезпека та захист інформації першого (бакалаврського) рівня. Розробка містить: приклади виконання деяких практичних завдань мовою програмування Python; варіанти завдань для виконання здобувачами освіти, відповідно до кожної лекційної теми згідно вказаного модуля. Варіанти завдань підготовлені відповідно до тем силабусу ОК «Програмування» та відповідають теорії, що розглядається на відповідних лекційних заняттях. Практикум буде корисний усім, хто вивчає або викладає об'єктно-орієнтоване програмування мовою Python.

Методична розробка складена з урахуванням досвіду викладання програмування на факультеті інформаційних технологій і математики Волинського національного університету імені Лесі Українки.

ЗМІСТ

Лабораторна робота № 1. Оголошення класу, реалізація класу мовою Python. Конструктор, деструктор. Реалізація інкапсуляції мовою Python

Мета лабораторної роботи:

- ✓ Ознайомити здобувачів освіти з принципами оголошення та реалізації класів у мові Python.
- ✓ Навчити створювати та використовувати конструктори і деструктори для ініціалізації та завершення об'єктів.
- ✓ Засвоїти принципи інкапсуляції та механізми її реалізації в Python.
- ✓ Розвинути практичні навички створення та використання об'єктів, управління їх властивостями і поведінкою через методи класів.
- ✓ Поглибити розуміння основ об’єктно-орієнтованого програмування (ООП) у Python, формуючи уявлення про те, як працюють об’єкти і класи.

Завдання націлені на те, щоб здобувачі освіти закріпили основи ООП та вміли застосовувати ці концепції на практиці.

Розглянемо **приклад** виконання завдання 2 для класу «Гра»:

Розробіть клас «Гра» з приватними атрибутами для назви гри, жанру та рейтингу. Реалізуйте метод для оновлення рейтингу гри, який дозволяє змінювати рейтинг лише в межах певного діапазону (наприклад, від 0 до 10). Додайте також метод для відображення інформації про гру у форматі: «Назва: [назва], Жанр: [жанр], Рейтинг: [рейтинг]». Забезпечте доступ до атрибутів лише через методи класу.

Атрибути класу: `__title, __genre, __rating` (приватні атрибути, які зберігають дані про гру).

Методи: `update_rating(self, new_rating)` (дозволяє оновити рейтинг гри, перевіряючи, чи входить новий рейтинг у допустимий діапазон); `display_info(self)` (відображає інформацію про гру); `__private_method(self)` (приватний метод, який не повинен бути доступним ззовні).

Код програми буде мати вигляд:

```
class Game:
    def __init__(self, title, genre, rating):
        self.__title = title  # Приватний атрибут для назви гри
        self.__genre = genre  # Приватний атрибут для жанру гри
```

```
            self.__rating = rating # Приватний атрибут для
рейтингу гри (0-10)

        def update_rating(self, new_rating):
            # Метод для оновлення рейтингу, якщо новий рейтинг
у межах 0-10
            if 0 <= new_rating <= 10:
                self.__rating = new_rating
            else:
                print(«Рейтинг має бути в межах від 0 до 10.»)

        def display_info(self):
            # Метод для відображення інформації про гру
            print(f»Назва:        {self.__title},        Жанр:
{self.__genre}, Рейтинг: {self.__rating}»)

        def __private_method(self):
        # Приватний метод, який не повинен бути доступним ззовні
класу
            return f»Це приватний метод гри: {self.__title}»

    # Демонстрація роботи з класом
    # Створення екземпляра класу
    game1 = Game(«The Witcher 3», «RPG», 9)

    # Відображення інформації про гру
    game1.display_info()  # Назва: The Witcher 3, Жанр: RPG,
Рейтинг: 9

    # Оновлення рейтингу
    game1.update_rating(10)  # Рейтинг успішно оновлено
    game1.display_info()   # Назва: The Witcher 3, Жанр: RPG,
Рейтинг: 10

    # Порушення інкапсуляції: спроба змінити приватний атрибут
безпосередньо
    try:
```

```
        game1.__rating = 8  # Це не змінить приватний атрибут __rating
    except AttributeError as e:
        print(e)  # Виведе помилку, якщо спробувати отримати доступ до приватного атрибута

    # Перевірка на порушення інкапсуляції
    print(game1._Game__rating)    # Це обійде інкапсуляцію, отримаємо доступ до приватного атрибута
    # Звернення до приватного методу (порушення інкапсуляції)
    try:
        print(game1.__private_method())  # Виклик призведе до помилки
    except AttributeError as e:
        print(e)  # Виведе помилку, якщо спробувати викликати приватний метод

    # Правильний виклик приватного методу через його «іменовану» версію
    print(game1._Game__private_method())      # Це обійде інкапсуляцію і викличе приватний метод
```

В результаті створено екземпляр класу `Game` і використано методи для взаємодії з ним. Показано, як інкапсуляція захищає атрибути, а також демонструє порушення інкапсуляції через обхід механізму іменування `Python`.

Завдання на лабораторну роботу:

1. **Завдання А.**

   Змоделюйте роботу автомобільного сервісу.

- Напишіть клас з ім’ям CarService(). Метод __init__() класу CarService() повинен містити атрибут service_name та service_location. Створіть метод describe_service(), який виводить обидва атрибути, і метод open_service(), який виводить повідомлення про те, що сервіс відкритий. Створіть на основі класу екземпляр з ім’ям service. Виведіть атрибути окремо, потім викличте обидва методи.
- Створіть ще один екземпляр класу, викличте для нього метод describe_service().

- Додайте атрибут number_of_mechanics зі значенням за замовчуванням 0, що представляє кількість механіків в сервісі. Виведіть значення number_of_mechanics, а потім змініть number_of_mechanics і виведіть знову для екземпляру service.
- Додайте метод з ім’ям set_number_of_mechanics(), що дозволяє задати кількість механіків. Викличте метод з новим числом, знову виведіть значення. Додайте метод з ім’ям increment_number_of_mechanics(), який збільшує кількість механіків на задану величину. Викличте цей метод для екземпляру service.
- Збережіть код класу CarService() у модулі. Створіть окремий файл, що імпортує клас CarService(). Створіть екземпляр all_service і викличте один з методів CarService(), щоб перевірити, що команда import працює вірно.

Завдання Б.

Змоделюйте роботу магазину.

- Напишіть клас з ім’ям Shop(). Метод __init__() класу Shop() повинен містити два атрибути: shop_name і store_type. Створіть метод describe_shop(), який виводить два атрибути, і метод open_shop(), який виводить повідомлення про те, що онлайн-магазин відкритий. Створіть на основі класу екземпляр з ім’ям store. Виведіть два атрибути окремо, потім викличте обидва методи.
- Створіть ще один екземпляр класу, викличте для нього метод describe_shop().
- Додайте атрибут number_of_units зі значенням за замовчуванням 0; він представляє кількість видів товару у магазині. Виведіть значення number_of_units, а потім змініть number_of_units і виведіть знову для store.
- Додайте метод з ім’ям set_number_of_units(), що дозволяє задати кількість видів товару. Викличте метод з новим числом, знову виведіть значення. Додайте метод з ім’ям increment_number_of_units(), який збільшує кількість видів товару на задану величину. Викличте цей метод для store.
- Збережіть код класу Shop() у модулі. Створіть окремий файл, що імпортує клас Shop(). Створіть екземпляр all_shop і викличте один з методів Shop(), щоб перевірити, що команда import працює вірно.

Завдання В.

Змоделюйте роботу кафе.

- Напишіть клас з ім'ям Cafe(). Метод __init__() класу Cafe() повинен містити два атрибути: cafe_name і cuisine_type. Створіть метод describe_cafe(), який виводить два атрибути, і метод open_cafe(), який виводить повідомлення про те, що кафе відкрите. Створіть на основі класу екземпляр з ім'ям cafe. Виведіть два атрибути окремо, потім викличте обидва методи.
- Створіть ще один екземпляр класу, викличте для нього метод describe_cafe().
- Додайте атрибут number_of_tables зі значенням за замовчуванням 0; він представляє кількість столиків у кафе. Виведіть значення number_of_tables, а потім змініть number_of_tables і виведіть знову для cafe.
- Додайте метод з ім'ям set_number_of_tables(), що дозволяє задати кількість столиків. Викличте метод з новим числом, знову виведіть значення. Додайте метод з ім'ям increment_number_of_tables(), який збільшує кількість столиків на задану величину. Викличте цей метод для cafe.
- Збережіть код класу Cafe() у модулі. Створіть окремий файл, що імпортує клас Cafe(). Створіть екземпляр all_cafe і викличте один з методів Cafe(), щоб перевірити, що команда import працює вірно.

Завдання Г.

Змоделюйте роботу автомайстерні.

- Напишіть клас з ім'ям AutoRepairShop(). Метод __init__() класу AutoRepairShop() повинен містити два атрибути: shop_name і service_type. Створіть метод describe_auto(), який виводить два атрибути, і метод open_auto(), який виводить повідомлення про те, що автомайстерня відкрита. Створіть на основі класу екземпляр з ім'ям repair_auto. Виведіть два атрибути окремо, потім викличте обидва методи.
- Створіть ще один екземпляр класу, викличте для нього метод describe_auto().
- Додайте атрибут number_of_bays зі значенням за замовчуванням 0; він представляє кількість автомайстрів у майстерні. Виведіть значення number_of_bays, а потім змініть number_of_bays і виведіть знову для repair_auto.
- Додайте метод з ім'ям set_number_of_bays(), що дозволяє задати кількість автомайстрів. Викличте метод з новим числом, знову виведіть значення. Додайте метод з ім'ям increment_number_of_bays(), який

збільшує кількість автомайстрів на задану величину. Викличте цей метод для repair_auto.

- Збережіть код класу AutoRepairShop() у модулі. Створіть окремий файл, що імпортує клас AutoRepairShop(). Створіть екземпляр all_auto і викличте один з методів AutoRepairShop(), щоб перевірити, що команда import працює вірно.

2. Продемонструйте детальну роботу з класом використовуючи екземпляри та звернення до методів класу. Покажіть інкапсуляцію (має бути 2 підкреслення до назви) та її порушення (щоб інтерпритатор Python зреагував та щоб не зреагував). В завданнях вказано, який буде приватний атрибут, а який метод зробити приватним вирішувати ВАМ. Якщо потрібно дописати для роботи класу ще методи, то допишіть їх та поясність для чого вони:

1. Створіть клас «Банківський рахунок» з приватним атрибутом для балансу рахунку, і методами для зняття та поповнення балансу. Зробіть так, щоб змінити баланс можна було лише через методи класу.
2. Розробіть клас «Користувач» з приватними атрибутами для імені та пароля, і методами для авторизації. Забезпечте доступ до пароля тільки через методи класу.
3. Створіть клас «Автомобіль» із приватним атрибутом «пробіг», і методами для отримання та оновлення пробігу. Захистіть пробіг від прямого змінення ззовні.
4. Розробіть клас «Працівник» з приватними атрибутами для зарплати і років стажу, і методом для розрахунку премії. Зробіть атрибути «зарплати» та «стажу» недоступними для зміни безпосередньо.
5. Створіть клас «Електронний годинник» з приватними атрибутами для годин і хвилин, і методами для встановлення та зміни часу.
6. Розробіть клас «Кошик для покупок» з приватним атрибутом для списку товарів та методами для додавання, видалення і виведення товарів. Захистіть список товарів від прямого доступу.
7. Створіть клас «Бібліотека» з приватним атрибутом для списку книг та методами для видачі та повернення книг. Зробіть список книг недоступним для зміни ззовні.
8. Розробіть клас «Людина» з приватними атрибутами для віку та імені, і методами для зміни цих атрибутів.

9. Створіть клас «Банкомат» з приватним атрибутом для залишку грошей та методами для зняття та поповнення коштів. Захистіть атрибут «залишку грошей» від зміни.
10. Розробіть клас «Замовлення» з приватними атрибутами для товарів і кількості, і методами для розрахунку вартості та виведення інформації про замовлення.
11. Створіть клас «Студент» з приватними атрибутами для оцінок і методами для додавання та середнього обчислення оцінок. Захистіть атрибути «оцінок» від прямого доступу.
12. Розробіть клас «Пасажир» з приватними атрибутами для списку квитків і методами для додавання та видалення квитків. Захистіть список квитків від прямого доступу.
13. Створіть клас «Електронний гаманець» з приватним атрибутом для балансу і методами для додавання та зняття коштів. Захистіть баланс від прямого змінення.
14. Розробіть клас «Комп'ютер» з приватними атрибутами для обсягу пам'яті і швидкості процесора, і методами для отримання і оновлення цих характеристик. Зробіть атрибути «пам'яті» і «процесора» недоступними для зміни безпосередньо.
15. Створіть клас «Банк» з приватним атрибутом для списку клієнтів та методами для додавання та видалення клієнтів. Захистіть список клієнтів від прямого доступу.
16. Розробіть клас «Повітряний літак» з приватними атрибутами для максимальної швидкості та висоти польоту, і методами для управління цими параметрами.
17. Створіть клас «Контейнер» з приватними атрибутами для вмісту та об'єму, і методами для додавання та видалення речей з контейнера. Захистіть атрибути «вмісту» та «об'єму» від прямого доступу.
18. Розробіть клас «Команда футбольної гри» з приватним атрибутом для списку гравців і методами для додавання та видалення гравців. Захистіть список гравців від прямого доступу.
19. Створіть клас «Календар» з приватним атрибутом для списку подій та методами для додавання та видалення подій. Захистіть список подій від прямого доступу.

20. Розробіть клас «Ресторан» з приватним атрибутом для меню та методами для додавання та видалення страв. Захистіть список страв від прямого доступу.
21. Створіть клас «Квартира» з приватними атрибутами для кількості кімнат та площі і методами для зміни цих характеристик.
22. Розробіть клас «Відеокамера» з приватними атрибутами для моделі та роздільної здатності, і методами для зміни цих параметрів.
23. Створіть клас «Замок» з приватним атрибутом для комбінації та методами для зміни комбінації та відкривання замка.
24. Розробіть клас «Калькулятор» з приватним атрибутом для результату обчислення і методами для додавання, віднімання, множення та ділення.
25. Створіть клас «Магазин» з приватним атрибутом для списку товарів та методами для додавання та видалення товарів. Захистіть список товарів від прямого доступу.
26. Розробіть клас «Лікарня» з приватним атрибутом для списку пацієнтів та методами для додавання та видалення пацієнтів. Захистіть список пацієнтів від прямого доступу.
27. Створіть клас «Товариство» з приватними атрибутами для списку членів та бюджету і методами для додавання та видалення членів і оновлення бюджету.
28. Розробіть клас «Електронний документ» з приватними атрибутами для тексту та формату і методами для зміни тексту та формату.

Лабораторна робота № 2. Сетери, гетери, делетери. Статичні поля та методи. Графічна бібліотека turtle та її застосування. Реалізація класу на базі уже створеного класу: композиція

Мета лабораторної роботи:

- ✓ Ознайомити здобувачів освіти з концепціями інкапсуляції через використання сеттерів, геттерів та делетерів у класах.
- ✓ Засвоїти роботу зі статичними полями та методами, розуміючи їх специфіку та застосування.
- ✓ Навчитися використовувати графічну бібліотеку turtle для візуалізації алгоритмів.
- ✓ Поглибити розуміння об'єктно-орієнтованого підходу через реалізацію класу на основі композиції вже існуючого класу.
- ✓ Закріпити навички розробки власних класів, їх модифікації та взаємодії між ними.

Завдання спрямовані на формування практичних навичок роботи з об'єктами та їх взаємодією у Python.

Розглянемо **приклад** реалізації завдання 1.

Запрограмуйте клас та роботу з ним, сетер та гетер. Організуйте у класі статичний метод, тобто зробіть той метод у класі статичним, який можна, або напишіть до цього класу свій статичний метод. До класу: Trapezoid з властивостями довжини основ та висоти трапеції. Методи: set- та get- для основ і висоти, обчислення площі та периметра (враховуючи бічні сторони). Напишіть програму, яка демонструє роботу з цим класом.

Виділимо основне для реалізації програми:

- сеттери і геттери дадуть можливість змінювати та отримувати значення основ, висоти і бічних сторін;
- статичний метод `is_valid_trapezoid()` буде перевіряти, чи може існувати трапеція з такими сторонами;
- методи для обчислення площі та периметра будуть використовувати властивості трапеції та її сторін.

Код програми буде мати вигляд:

```
import math
class Trapezoid:
    def __init__(self, a=0, b=0, height=0, side1=0, side2=0):
        self._a = a       # Довжина першої основи
```

```
        self._b = b       # Довжина другої основи
        self._height = height   # Висота
        self._side1 = side1     # Перша бічна сторона
        self._side2 = side2     # Друга бічна сторона

    # Сеттери
    def set_base_a(self, a):
        self._a = a

    def set_base_b(self, b):
        self._b = b

    def set_height(self, height):
        self._height = height

    def set_side1(self, side1):
        self._side1 = side1

    def set_side2(self, side2):
        self._side2 = side2

    # Геттери
    def get_base_a(self):
        return self._a

    def get_base_b(self):
        return self._b

    def get_height(self):
        return self._height

    def get_side1(self):
        return self._side1

    def get_side2(self):
        return self._side2

    # Метод для обчислення площі
```

```
        def calculate_area(self):
            return (self._a + self._b) * self._height / 2

        # Метод для обчислення периметра
        def calculate_perimeter(self):
            return self._a + self._b + self._side1 + self._side2

        # Статичний метод для перевірки, чи може існувати трапеція з такими сторонами
        @staticmethod
        def is_valid_trapezoid(a, b, side1, side2):
            # Трапеція існує, якщо сума двох бічних сторін більша за різницю основ
            return abs(a - b) < side1 + side2

    # Демонстрація роботи класу
    trapezoid = Trapezoid()
    trapezoid.set_base_a(5)
    trapezoid.set_base_b(7)
    trapezoid.set_height(4)
    trapezoid.set_side1(3)
    trapezoid.set_side2(3)

    # Перевірка валідності трапеції
    if Trapezoid.is_valid_trapezoid(trapezoid.get_base_a(), trapezoid.get_base_b(), trapezoid.get_side1(), trapezoid.get_side2()):
        print(f»Площа трапеції: {trapezoid.calculate_area()}»)
        print(f»Периметр трапеції: {trapezoid.calculate_perimeter()}»)
    else:
        print(«Така трапеція не може існувати.»)
```

В результаті виконання програми отримаємо площу та периметр трапеції.

Інші завдання з лабораторної роботи не демонструються, оскільки приклади є в лекційному матеріалі.

Завдання на лабораторну роботу:

1. Запрограмуйте клас та роботу з ним, сетер та гетер. Організуйте у класі статичний метод, тобто зробіть той метод у класі статичним, який можна, або напишіть до цього класу свій статичний метод.

1. Опишіть клас Square, з властивістю довжина сторони квадрата. Методи: set- та get-, обчислення площі та периметра. Напишіть програму, яка демонструватиме роботу з цим класом.
2. Опишіть клас Circle, з властивістю радіус. Методи: set- та get-, обчислення площі круга та довжини кола. Напишіть програму, яка демонструватиме роботу з цим класом.
3. Опишіть клас Клієнт банку із властивостями ім'я, номер рахунку і баланс. Реалізуйте методи для встановлення та отримання цих властивостей (set- та get-), а також метод для зняття та поповнення балансу рахунку. Напишіть програму, яка демонструє роботу з цим класом, виконуючи операції з балансом.
4. Опишіть клас Equation, з властивостями a, b, c. Методи: set- та get-, розв’язування рівняння виду ax+b=c. Напишіть програму, яка демонструватиме роботу з цим класом.
5. Опишіть клас Time, з властивостями година, хвилина, секунда. Методи: set- та get-, збільшення значення часу на 1 секунду, на 1 хвилину, на 1 годину. Напишіть програму, яка демонструватиме роботу з цим класом.
6. Опишіть клас Rectangle, з властивостями довжина, ширина. Методи: set- та get-, обчислення периметру та площі. Напишіть програму, яка демонструватиме роботу з цим класом.
7. Опишіть клас Date з властивостями день, місяць, рік. Методи: set- та get-, наступний день, наступний місяць. Напишіть програму, яка демонструватиме роботу з цим класом.
8. Опишіть клас SquareEquation, з властивостями a, b, c. Методи: set- та get-, розв’язування рівняння виду ax2+bx+c=0. Напишіть програму, яка демонструватиме роботу з цим класом.
9. Опишіть клас Triangle, з властивостями 3 сторони трикутника. Методи: set- та get-, знаходження площі та периметра трикутника. Напишіть програму, яка демонструватиме роботу з цим класом.

10. Опишіть клас Calculation, з властивостями число_1, число_2. Методи: set- та get-, обчислення суми, різниці, добутку, частки цих двох чисел. Напишіть програму, яка демонструватиме роботу з цим класом.
11. Опишіть клас Book із властивостями назва, автор і рік видання. Реалізуйте методи для встановлення set- та отримання get- цих властивостей, а також методи: для пошуку книги за автором, для сортування книг в алфавітному порядку. Напишіть програму, яка демонструватиме роботу з цим класом.
12. Опишіть клас CalculationComp, з властивостями комплексне число_1, комплексне число_2. Методи: set- та get-, обчислення суми, різниці, добутку, частки цих двох чисел. Напишіть програму, яка демонструватиме роботу з цим класом.
13. Опишіть клас Employee (співробітник) з властивостями ім'я, прізвище, посада і заробітня плата. Реалізуйте методи для встановлення set- та отримання get- цих властивостей, а також методи: для розрахунку річного доходу співробітника, для сортування співробітників в алфавітному порядку. Напишіть програму, яка демонструє роботу з цим класом.
14. Опишіть клас Product із властивостями назва, ціна і кількість. Реалізуйте методи для встановлення та отримання цих властивостей (set- та get-), а також методи: для розрахунку вартості продукту (ціна * кількість), якого продукту найбільше. Напишіть програму, яка створює об'єкти класу Продукт та демонструє роботу з класом.
15. Опишіть клас School з властивостями назва, адреса і кількість учнів. Реалізуйте методи для встановлення та отримання цих властивостей (set- та get-), а також методи: для розрахунку середнього числа учнів на клас (кількість учнів / кількість класів), для виведення кількості різних класів, наприклад, 5 класів є 4 і т.д. Напишіть програму, яка використовує цей клас для роботи з даними про школу.
16. Опишіть клас PassengerPlane із властивостями модель, кількість місць і кількість пасажирів на борту. Реалізуйте методи для встановлення та отримання цих властивостей (set- та get-), а також методи для розрахунку вільних місць на борту та для сортування літаків за кількістю місць. Напишіть програму, яка використовує цей клас для обробки даних про пасажирський літак.

17. Опишіть клас Shop з властивостями назва, адреса і середній чек. Реалізуйте методи для встановлення та отримання цих властивостей (set- та get-), а також методи: для розрахунку річного доходу магазину (середній чек * 365) та визначення магазину у якого найбільший середній чек. Напишіть програму, яка використовує цей клас для роботи з даними про магазин.

18. Опишіть клас User з властивостями ім'я, вік і електронна пошта. Реалізуйте методи для встановлення та отримання цих властивостей (set- та get-). Додайте метод перевірка_повноліття, який перевіряє, чи користувач є повнолітнім (вік більше або дорівнює 18 рокам). Допишіть ще будь який свій метод. Напишіть програму, яка демонструє роботу з цим класом.

19. Опишіть клас Book з властивостями назва, автор і кількість сторінок. Реалізуйте методи для встановлення та отримання цих властивостей (set- та get-), а також метод визначення_типу, який визначає, чи книга є художньою чи науковою (якщо кількість сторінок менше або дорівнює 300, то це художня книга, інакше – наукова). Допишіть ще будь який свій метод. Напишіть програму, яка демонструє роботу з цим класом.

20. Опишіть клас Car з властивостями марка, модель і рік випуску. Реалізуйте методи для встановлення та отримання цих властивостей (set- та get-), а також методи: визначення_статусу, який визначає, чи автомобіль є новим, середнім або старим (якщо рік випуску відповідає поточному року, то це новий автомобіль, якщо відповідає минулому року – середній, інакше – старий) та виведи моделі одного року випуску. Напишіть програму, яка демонструє роботу з цим класом.

21. Опишіть клас BankAccount з властивостями номер рахунку і баланс. Реалізуйте методи для встановлення та отримання цих властивостей (set- та get-), а також методи: зняття_грошей та поповнення_рахунку, які дозволяють взаємодіяти з балансом рахунку. Напишіть програму, яка демонструє роботу з цим класом.

22. Опишіть клас Student з властивостями ім'я, прізвище і середній бал. Реалізуйте методи для встановлення та отримання цих властивостей (set- та get-), а також методи: визначення_стипендії, який визначає, чи студент має право на стипендію (якщо середній бал більше або

дорівнює 9.0, то має право) та знаходження студента з вказаним балом. Напишіть програму, яка демонструє роботу з цим класом.

23. Опишіть клас Product з властивостями назва, ціна і кількість на складі. Реалізуйте методи для встановлення та отримання цих властивостей (set- та get-), а також методи: визначення_вартості, який обчислює вартість продукту на складі (ціна * кількість) та знаходження ціни продукту за назвою. Напишіть програму, яка демонструє роботу з цим класом.

24. Опишіть клас MusicAlbum з властивостями назва, виконавець і кількість треків. Реалізуйте методи для встановлення та отримання цих властивостей (set- та get-), а також методи: визначення_типу_альбому, який визначає, чи альбом є студійним або концертним (якщо кількість треків більше 10, то це студійний альбом, інакше – концертний) та пошук альбому за виконавцем. Напишіть програму, яка демонструє роботу з цим класом.

25. Опишіть клас Order з властивостями номер, клієнт і сума. Реалізуйте методи для встановлення та отримання цих властивостей (set- та get-), а також методи: розрахунку_знижки, який визначає розмір знижки для замовлення в залежності від суми та визначення найдорожчого замовлення за день. Напишіть програму, яка демонструє роботу з цим класом.

26. Опишіть клас Teacher з властивостями прізвище, посада і зарплата. Реалізуйте методи для встановлення та отримання цих властивостей (set- та get-), а також методи: розрахунку_податку, який розраховує податок на зарплату вчителя в залежності від посади (10% для учителів, 15% для директорів) та сортування вчителів за прізвищем. Напишіть програму, яка демонструє роботу з цим класом.

27. Створіть клас AirTicket з властивостями номер рейсу, пункт відправлення і пункт призначення. Реалізуйте методи для встановлення та отримання цих властивостей (set- та get-), а також методи: розрахунку_відстані, який визначає відстань між пунктами відправлення та призначення (введіть відстань у кілометрах) та пошуку рейсу за пунктом призначення. Напишіть програму, яка демонструє роботу з цим класом.

28. Опишіть клас Task з властивостями назва, опис і термін виконання. Реалізуйте методи для встановлення та отримання цих властивостей (set- та get-), а також методи: перевірка_виконання, який перевіряє,

чи завдання виконане вчасно (чи термін виконання не минув) та пошук завдання за назвою. Напишіть програму, яка демонструє роботу з цим класом.

2. Створити клас, який будує ваше графічне зображення та відображає роботу з ним (зображення – ваша творчість (зірка, полігон і т.д.), прохання зображення зробити складніше, а не елементарне), використовуючи графічний модуль turtle та, як аналогію, клас, створений у лекції, де описана робота з колом. Продемонструвати детальну роботу з класом та створення декількох об'єктів, щоб в конструкторі можна було задавати координату початку побудови та колір.

3. Використайте композицію класів та продемонструйте детальну роботу з класом.

1. Створіть клас «Студент» (Student) з властивостями «ім'я» та «група». Створіть клас «Викладач» (Teacher) з властивостями «ім'я» та «предмет». Створіть клас «Курс» (Course), який складається з об'єктів «Студент» і «Викладач». Використовуйте композицію для пов'язання студентів і викладачів у складі курсу.
2. Розробіть клас «Автомобіль» (Car) з властивостями «марка» та «колір». Створіть клас «Двигун» (Engine) з властивістю «потужність». Створіть клас «Машина» (Vehicle), який складається з об'єктів «Автомобіль» і «Двигун». Використовуйте композицію для пов'язання автомобілів і двигунів у складі машин.
3. Створіть клас «Автор» (Author) з властивостями «ім'я» і «рік народження». Створіть клас «Книга» (Book) з властивостями «назва» і «автор», де «автор» є об'єктом класу «Автор». Створіть клас «Бібліотека» (Library), який складається з об'єктів «Книга». Використовуйте композицію для пов'язання книг і авторів у складі бібліотеки.
4. Розробіть клас «Користувач» (User) з властивостями «ім'я» і «адреса». Створіть клас «Замовлення» (Order) з властивостями «номер» і «користувач», де «користувач» є об'єктом класу «Користувач». Створіть клас «Магазин» (Shop), який складається з об'єктів «Замовлення». Використовуйте композицію для пов'язання замовлень і користувачів у складі магазину.

5. Створіть клас «Пасажир» (Passenger) з властивостями «ім'я» та «квиток». Створіть клас «Рейс» (Flight) з властивостями «номер» і «пасажир», де «пасажир» є об'єктом класу «Пасажир». Створіть клас «Авіакомпанія» (Airline), який складається з об'єктів «Рейс». Використовуйте композицію для пов'язання рейсів і пасажирів у складі авіакомпанії.
6. Розробіть клас «Ресторан» (Restaurant) з властивостями «назва» та «стіл». Створіть клас «Страва» (Dish) з властивостями «назва» і «ціна». Створіть клас «Замовлення» (Order), який складається з об'єктів «Страва» і «Стіл». Використовуйте композицію для пов'язання страв і столів у складі замовлення в ресторані.
7. Створіть клас «Фільм» (Movie) з властивостями «назва» і «режисер». Створіть клас «Актор» (Actor) з властивостями «ім'я» і «роль». Створіть клас «Кінотеатр» (Cinema), який складається з об'єктів «Фільм» і «Актор». Використовуйте композицію для пов'язання фільмів і акторів у складі кінотеатру.
8. Розробіть клас «Музичний альбом» (MusicAlbum) з властивостями «назва» і «виконавець». Створіть клас «Пісня» (Song) з властивостями «назва» і «тривалість». Створіть клас «Плейлист» (Playlist), який складається з об'єктів «Пісня» і «Музичний альбом». Використовуйте композицію для організації пісень і альбомів у складі плейлисту.
9. Створіть клас «Готель» (Hotel) з властивостями «назва» та «номер». Створіть клас «Клієнт» (Client) з властивостями «ім'я» і «номер кімнати». Створіть клас «Бронювання» (Reservation), який складається з об'єктів «Готель» і «Клієнт». Використовуйте композицію для пов'язання готелів і клієнтів у складі бронювання.
10. Розробіть клас «Комп'ютер» (Computer) з властивостями «модель» і «операційна система». Створіть клас «Монітор» (Monitor) з властивостями «розмір» і «роздільна здатність». Створіть клас «Робоча станція» (Workstation), який складається з об'єктів «Комп'ютер» і «Монітор». Використовуйте композицію для пов'язання комп'ютерів і моніторів у складі робочої станції.
11. Розробіть клас «Інгредієнт» (Ingredient) з властивостями «назва» і «кількість». Створіть клас «Рецепт» (Recipe) з властивостями «назва» і списком інгредієнтів, де «інгредієнт» є об'єктом класу «Інгредієнт». Створіть клас «Кулінарна книга» (Cookbook), який складається з

об'єктів «Рецепт». Використовуйте композицію для організації рецептів у кулінарній книзі.

12. Розробіть клас «Авіакомпанія» (Airline) з властивостями «назва» і «рейс». Створіть клас «Пасажир» (Passenger) з властивостями «ім'я» і «квиток», де «квиток» є об'єктом класу «Рейс». Використовуйте композицію для пов'язання пасажирів і рейсів у складі авіакомпанії.
13. Створіть клас «Книга» (Book) з властивостями «назва» та «автор». Створіть клас «Бібліотека» (Library) з властивостями «назва» і списком книг, де «книга» є об'єктом класу «Книга». Створіть клас «Відділ» (Department), який складається з об'єктів «Бібліотека» і «Книга». Використовуйте композицію для організації книг у відділах бібліотеки.
14. Розробіть клас «Магазин» (Shop) з властивостями «назва» та «товар». Створіть клас «Товар» (Product) з властивостями «назва» і «ціна», де «товар» є об'єктом класу «Магазин». Створіть клас «Замовлення» (Order), який складається з об'єктів «Магазин» і «Товар». Використовуйте композицію для організації товарів і магазинів у складі замовлень.
15. Створіть клас «Кафе» (Cafe) з властивостями «назва» і «стіл». Створіть клас «Страва» (Dish) з властивостями «назва» і «ціна». Створіть клас «Замовлення» (Order), який складається з об'єктів «Страва» і «Стіл». Використовуйте композицію для організації страв і столів у складі замовлень в кафе.
16. Розробіть клас «Пасажирський літак» (PassengerPlane) з властивостями «номер» і «кількість пасажирів». Створіть клас «Літаковий відділ» (AirplaneSection) з властивостями «клас» і «кількість місць», де «Літаковий відділ» є об'єктом класу «Пасажирський літак». Створіть клас «Авіакомпанія» (Airline), який складається з об'єктів «Пасажирський літак» і «Літаковий відділ». Використовуйте композицію для організації літаків і їх відділів у складі авіакомпанії.
17. Створіть клас «Клієнт» (Client) з властивостями «ім'я» і «номер акаунта». Створіть клас «Транзакція» (Transaction) з властивостями «сума» і «клієнт», де «клієнт» є об'єктом класу «Клієнт». Створіть клас «Банк» (Bank), який складається з об'єктів «Клієнт» і «Транзакція». Використовуйте композицію для пов'язання клієнтів і їх транзакцій у банку.

18. Розробіть клас «Університет» (University) з властивостями «назва» і «факультет». Створіть клас «Викладач» (Professor) з властивостями «прізвище» і «предмет», а також клас «Студент» (Student) з властивостями «ім'я» і «курс». Створіть клас «Курс» (Course), який складається з об'єктів «Викладач» і «Студент». Використовуйте композицію для пов'язання викладачів і студентів у складі курсів.

19. Створіть клас «Магазин» (Store) з властивостями «назва» і «товар». Створіть клас «Товар» (Product) з властивостями «назва» і «ціна», де «товар» є об'єктом класу «Магазин». Створіть клас «Замовлення» (Order), який складається з об'єктів «Магазин» і «Товар». Використовуйте композицію для організації замовлень товарів у магазині.

20. Розробіть клас «Робочий стіл» (Desk) з властивостями «розмір» і «комп'ютер». Створіть клас «Комп'ютер» (Computer) з властивостями «модель» і «операційна система». Створіть клас «Офіс» (Office), який складається з об'єктів «Робочий стіл» і «Комп'ютер». Використовуйте композицію для пов'язання комп'ютерів і робочих столів у складі офісу.

21. Створіть клас «Поштова скринька» (Mailbox) з властивостями «номер» і «лист». Створіть клас «Лист» (Letter) з властивостями «тема» і «вміст». Створіть клас «Користувач» (User), який складається з об'єктів «Поштова скринька» і «Лист». Використовуйте композицію для організації листів у поштовій скриньці користувача.

22. Розробіть клас «Ферма» (Farm) з властивостями «назва» і «тварини». Створіть клас «Тварина» (Animal) з властивостями «ім'я» і «вага», де «тварина» є об'єктом класу «Ферма». Створіть клас «Господарство» (Homestead), який складається з об'єктів «Ферма» і «Тварина». Використовуйте композицію для пов'язання ферм і їх тварин у складі господарства.

23. Створіть клас «Завдання» (Task) з властивостями «назва» і «виконавець». Створіть клас «Виконавець» (Executor) з властивостями «ім'я» і «прізвище». Створіть клас «Проект» (Project), який складається з об'єктів «Завдання» і «Виконавець». Використовуйте композицію для пов'язання виконавців і завдань у складі проектів.

24. Розробіть клас «Квартира» (Apartment) з властивостями «номер» і «житель». Створіть клас «Житель» (Tenant) з властивостями «ім'я» і

«прізвище». Створіть клас «Будинок» (House), який складається з об'єктів «Квартира» і «Житель». Використовуйте композицію для пов'язання квартир і жителів у складі будинку.

25. Створіть клас «Замовлення» (Order) з властивостями «номер» і «товар». Створіть клас «Товар» (Product) з властивостями «назва» і «ціна», де «товар» є об'єктом класу «Замовлення». Створіть клас «Магазин» (Store), який складається з об'єктів «Замовлення» і «Товар». Використовуйте композицію для організації товарів і замовлень у складі магазину.

26. Розробіть клас «Кімната» (Room) з властивостями «номер» і «меблі». Створіть клас «Мебель» (Furniture) з властивостями «тип» і «кількість», де «меблі» є об'єктом класу «Кімната». Створіть клас «Готель» (Hotel), який складається з об'єктів «Кімната» і «Мебель». Використовуйте композицію для пов'язання кімнат і меблів у складі готелю.

Лабораторна робота № 3. Реалізація класу на базі уже створеного класу: наслідування (звичайне наслідування, множинне наслідування)

Мета лабораторної роботи:

- ✓ Ознайомити здобувачів освіти з принципами наслідування у мові Python, зокрема із звичайним та множинним наслідуванням.
- ✓ Навчити створювати нові класи на базі існуючих, використовуючи механізми наслідування для повторного використання коду.
- ✓ Засвоїти роботу з ієрархіями класів та методами, які можна перевизначати в похідних класах.
- ✓ Розвинути розуміння ключових особливостей наслідування, таких як розширення функціональності базових класів, робота з методами і властивостями батьківських класів.
- ✓ Ознайомити студентів із викликом методів базових класів за допомогою функцій *super()* та інших механізмів Python.

Завдання спрямовані на закріплення практичних навичок роботи з наслідуванням, формуючи розуміння ієрархії класів.

Розглянемо **приклад** реалізації завдання 1 для: створіть клас MemoryStorage, що має обсяг (Гбайт). Створіть похідний від нього клас USBDrive, що додатково має параметри швидкості передачі даних (читає/пише) та тип підключення (USB 2.0, USB 3.0 тощо). Визначте необхідні конструктори, методи та деструктор. Розробіть програму, що демонструє роботу з класом.

Визначимо головне:

- ✓ клас `MemoryStorage` буде містити атрибут `capacity` для зберігання обсягу; метод `get_capacity()` буде повертати обсяг пам'яті; деструктор виводитиме повідомлення про знищення об'єкта.
- ✓ клас `USBDrive` – похідний клас, що буде містити додаткові параметри: швидкість читання, швидкість запису та тип підключення.

У першому випадку конструктор базового класу буде викликатися явно.

У другому випадку буде використовуватися метод `super()` для виклику конструктора базового класу.

Код програми буде мати вигляд:

```
class MemoryStorage:
    def __init__(self, capacity):
        self.capacity = capacity  # Обсяг в Гбайт

    def get_capacity(self):
```

```
            return self.capacity

        def __del__(self):
            print(«MemoryStorage destroyed.»)

    class USBDrive(MemoryStorage):
        def __init__(self, capacity, read_speed, write_speed,
connection_type):
            MemoryStorage.__init__(self, capacity)  # Виклик
конструктора базового класу
            #   або   super().__init__(capacity)   #   Виклик
конструктора базового класу через super()
            self.read_speed = read_speed  # Швидкість читання
            self.write_speed = write_speed  # Швидкість запису
            self.connection_type  =  connection_type   #  Тип
підключення

        def get_details(self):
            return (f»USBDrive: {self.capacity} GB, Read Speed:
{self.read_speed} MB/s, «
                    f»Write  Speed:  {self.write_speed}  MB/s,
Connection Type: {self.connection_type}»)

        def __del__(self):
            print(«USBDrive destroyed.»)

    # Демонстрація роботи класу без super()
    usb1 = USBDrive(64, 150, 100, «USB 3.0»)
    print(usb1.get_details())
    del usb1
```

Запустивши цей код, ви побачите, як реалізується звичайне наслідування та як змінюється виклик конструктора базового класу з використанням `super()`.

Завдання на лабораторну роботу:

1. Продемонструвати звичайне наслідування без методу super() та з використанням методу super().

1. Створити клас Rectangle (довжина, ширина), з методом обчислення його площі. Створити похідний від нього клас Box (довжина, ширина, висота) з методом обчислення об’єму. Всі дані для створення об’єктів задаються у програмі. Вивести на екран характеристики об’єктів, їх розміри, площу та об’єм. Розробити програму, що демонструватиме роботу з класом.
2. Створити клас DataStoradge, що має обсяг (Гбайт). Створити похідний від нього клас HardDrive, що додатково має параметри форматування (кількість циліндрів, доріжок, секторів) та марку. Визначити необхідні конструктори, методи, деструктор. Розробити програму, що демонструватиме роботу з класом.
3. Розробити клас «смартфон» з властивостями «модель», «пам'ять (Гб)», «розмір дисплея (дюйм)», «камера (Мп)», «операційна система». Створити похідний клас «ігровий смартфон», що має додатково параметри «процесор», «графічна карта», «оперативна пам'ять (Гб)». Реалізувати методи для встановлення та отримання значень властивостей, а також метод для виведення інформації про смартфон.
4. Створити клас «автомобіль» з властивостями «марка», «модель», «потужність двигуна (к.с.)», «розхід пального (л/100км)». Створити похідний клас «електромобіль», що має додатково параметр «потужність батареї (кВт·год)». Реалізувати методи для встановлення та отримання значень властивостей, а також метод для розрахунку пробігу на одній зарядці батареї.
5. Розробити клас «книга» з властивостями «назва», «автор», «кількість сторінок», «рік видання». Створити похідний клас «книжка з супергероєм», що має додатково параметри «супергерой» та «суперздібності». Реалізувати методи для встановлення та отримання значень властивостей, а також метод для виведення інформації про книгу з супергероєм.
6. Створити клас «фільм» з властивостями «назва», «рік випуску», «режисер», «жанр». Створити похідний клас «анімаційний фільм», що має додатково параметр «студія виробник». Реалізувати методи для встановлення та отримання значень властивостей, а також метод для виведення інформації про анімаційний фільм.
7. Розробити клас «студент» з властивостями «ім'я», «прізвище», «вік», «середній бал». Створити похідний клас «студент спеціаліст» з

додатковою властивістю «кількість наукових публікацій». Реалізувати методи для встановлення та отримання значень властивостей, а також метод для розрахунку середнього балу студента спеціаліста.

8. Розробити клас «продукт» з властивостями: «назва», «ціна», «тип», «вага». Створити похідний клас «фрукт» з додатковою властивістю «кількість вітамінів». Реалізувати методи для встановлення та отримання значень властивостей, а також метод для розрахунку кількості калорій у фрукті.
9. Розробити клас «туристичний маршрут» з властивостями: «назва», «країна», «тривалість», «складність». Створити похідний клас «екстремальний маршрут» з додатковою властивістю «кількість перешкод». Реалізувати методи для встановлення та отримання значень властивостей, а також метод для розрахунку вартості екстремального маршруту.
10. Створити клас Book, що має назву, автора, рік видання і кількість сторінок. Створити похідний від нього клас FictionBook, що додатково має жанр (наприклад, «фантастика», «роман», «детектив») і рейтинг (від 1 до 5). Визначити необхідні конструктори, методи, деструктор. Розробити програму, що демонструватиме роботу з класом.
11. Створити клас MobilePhone, що має модель, виробника, розмір екрану (у дюймах) і рік випуску. Створити похідний від нього клас Smartphone, що додатково має операційну систему (наприклад, «iOS», «Android», «Windows») і кількість камер (задня і фронтальна). Визначити необхідні конструктори, методи, деструктор. Розробити програму, що демонструватиме роботу з класом.
12. Створити клас Car, що має марку, модель, рік випуску і кількість дверей. Створити похідний від нього клас SportsCar, що додатково має максимальну швидкість (у км/год) і тип кузова (наприклад, «купе», «кабріолет», «спайдер»). Визначити необхідні конструктори, методи, деструктор. Розробити програму, що демонструватиме роботу з класом.
13. Створити клас Person, що має ім'я, вік, стать і зріст. Створити похідний від нього клас Student, що додатково має поле з номером залишку на картах і список предметів, які він вивчає. Визначити необхідні конструктори, методи, деструктор. Розробити програму, що демонструватиме роботу з класом.

14. Розробити клас «туристична фірма» з властивостями: «назва», «місто», «кількість років на ринку», «кількість задоволених клієнтів». Створити похідний клас «відділ екстремальних маршрутів» з додатковими властивостями: «назва маршруту», «кількість перешкод», «довжина маршруту». Реалізувати методи для встановлення та отримання значень властивостей, а також метод для розрахунку вартості екстремального маршруту.

2. Продемонструвати реалізацію множинного наслідування.

1. Описати класи Вид, Літаюче та ЛітаючаІстота. Клас ЛітаючаІстота є нащадком класів Вид та Літаюче. Клас Вид має поля «назва виду», «клас» та методи, створення та виведення виду. Клас Літаюче має поля «спосіб польоту», «кількість крил», «напрям польоту», та методи створення та виведення. Клас ЛітаючаІстота має власне поле «назва» та методи створення, виведення усіх характеристик літаючої істоти. З використанням цих класів скласти програму опису деякої літаючої істоти та вивести інформацію щодо неї.
2. Описати класи Жанр, Носій та Фільм. Клас Фільм є нащадком класів Жанр та Носій. Клас Жанр має поля «назва жанру», «характерні особливості» та методи, створення та виведення жанру. Клас Носій має поля «тип носія», «кольорова гама» та «тривалість зберігання» та методи створення та виведення. Клас Фільм має власні поля «назва», «автор сценарію», «режисер», «тривалість» та методи створення, виведення усіх характеристик фільму. З використанням цих класів скласти програму опису деякого фільму та вивести інформацію щодо нього.
3. Описати класи ХудожнійТвір, Техніка та Картина. Клас Картина є нащадком класів ХудожнійТвір та Техніка. Клас ХудожнійТвір має поля «назва твору», «рік написання», «автор», «жанр» та методи, створення, виведення художнього твору. Клас Техніка описує використану техніку живопису. Цей клас має поля «назва техніки» та «матеріал» та методи створення та виведення. Клас Картина має власні поля «ширина», «висота», «вартість» та методи створення та виведення усіх характеристик картини, а також методи читання та зміни вартості. З використанням цих класів скласти програму опису деякої картини та вивести інформацію щодо неї.

4. Описати класи Професія, РангДержслужби та ДержавнаПосада. Клас ДержавнаПосада є нащадком класів Професія та РангДержслужби. Клас Професія має поля «назва професії», «спеціальність за освітою» та методи, створення та виведення професії. Клас РангДержслужби має поля «номер рангу», «вимоги для отримання рангу» та методи створення та виведення. Клас ДержавнаПосада має власні поля «назва», «заробітна платня» та методи створення та виведення усіх характеристик державної посади. З використанням цих класів скласти програму опису деякої державної посади та вивести інформацію щодо неї.

5. Описати класи МузичнийТвір, Звукозапис та МузичнийЗапис. Клас МузичнийЗапис є нащадком класів МузичнийТвір та Звукозапис. Клас МузичнийТвір має поля «назва твору», «рік створення», «автор», «жанр» та методи, створення та виведення музичного твору. Клас Звукозапис має поля «частота», «стандарт запису» та «тривалість» та методи створення та виведення. Клас МузичнийЗапис має власні поля «носій», «розташування» та методи створення, виведення усіх характеристик музичного запису. З використанням цих класів скласти програму опису деякого музичного запису та вивести інформацію щодо нього.

6. Описати класи Текст, Шрифт та ТекстовийНадпис. Клас ТекстовийНадпис є нащадком класів Текст та Шрифт. Клас Текст має поля «зміст тексту», «мова» та методи, створеннята виведення тексту.Клас Шрифт має поля «назва шрифту», «кегль», «напівгрубий», «нахилений», «підкреслений» та методи створення та виведення. Клас ТекстовийНадпис має власне поле «колір» та методи створення та виведення усіх характеристик текстового надпису. З використанням цих класів скласти програму опису деякого текстового надпису та вивести інформації щодо нього.

7. Описати класи Музичний центр, Колонки та Блок живлення. Клас Музичний центр є нащадком класів Колонки та Блок живлення. Клас Колонки має поля «кількість», «максимальну потужність», «тип колонок» та методи, створення та виведення колонки. Клас Блок живлення має поля «потужність», «вихідний вольтаж» та методи створення та виведення. Клас Музичний центр має власні поля «бренд», «кількість USB-портів» та методи створення, виведення усіх характеристик музикального центру. З використанням цих класів

скласти програму опису деякого музикального центру та вивести інформацію щодо нього.

8. Описати класи Матеріал, ФункціональнеПристосування та Меблі. Клас Меблі є нащадком класів Матеріал та ФункціональнеПристосування. Клас Матеріал має поля «тип матеріалу», «назва матеріалу», «природність» та методи, створення та виведення матеріалу. Клас ФункціональнеПристосування має поля «назва пристосування», «вікові рекомендації» та методи створення та виведення. Клас Меблі має власні поля «тип», «назва» та методи створення, виведення усіх характеристик меблів. З використанням цих класів скласти програму опису деякого функціонального пристосування та вивести інформацію щодо нього.
9. Описати класи НастільнаЛампа, Блок живлення та Лампа. Клас НастільнаЛампа є нащадком класів Блок живлення та Лампа. Клас Блок живлення має поля «потужність», «вихідний вольтаж», «вихідний струм» та методи, створення та виведення блоку живлення. Клас Лампа має поля «тип цоколю», «потужність лампи» та методи створення та виведення. Клас НастільнаЛампа має власні поля «бренд», «розміри» та методи створення, виведення усіх характеристик настільної лампи. З використанням цих класів скласти програму опису деякої настільної лампи та вивести інформацію щодо неї.
10. Описати класи РозумнийГодинник, Акумулятор та Дисплей. Клас РозумнийГодинник є нащадком класів Акумулятор та Дисплей. Клас Акумулятор має поля «кількість mAh», «вихідний вольтаж», «робоча температура» та методи, створення та виведення акумулятора. Клас Дисплей має поля «розширення дисплею», «тип матриці» та методи створення та виведення. Клас РозумніЧаси має власні поля «бренд», «колір ланцюжка» та методи створення, виведення усіх характеристик розумних часів. З використанням цих класів скласти програму опису деякого розумного годинника та вивести інформацію щодо нього.
11. Описати класи Виконавець, Альбом та Жанр. Клас Альбом є нащадком класів Виконавець та Жанр. Клас Виконавець має поля «ім'я», «країна» та методи створення, виведення та зміни імені виконавця. Клас Жанр має поля «назва жанру» та методи створення, виведення та зміни назви жанру. Клас Альбом має поля «назва», «рік випуску», «виконавець» (об'єкт класу Виконавець), «жанр» (об'єкт

класу Жанр) та методи створення, редагування та виведення усіх характеристик альбому. З використанням цих класів скласти програму опису деякого виконавця та вивести інформацію щодо нього.

12. Описати класи Компанія, Продукт та Категорія. Клас Продукт є нащадком класів Компанія та Категорія. Клас Компанія має поля «назва», «рік заснування» та методи створення, виведення та зміни назви компанії. Клас Категорія має поля «назва категорії» та методи створення, виведення та зміни назви категорії. Клас Продукт має поля «назва», «ціна», «компанія» (об'єкт класу Компанія), «категорія» (об'єкт класу Категорія) та методи створення, редагування та виведення усіх характеристик продукту. З використанням цих класів скласти програму опису деякого продукту та вивести інформацію щодо нього.
13. Описати класи Країна, Місто та Континент. Клас Місто є нащадком класів Країна та Континент. Клас Країна має поля «назва країни», «столиця» та методи створення, виведення та зміни назви країни. Клас Континент має поля «назва континента» та методи створення, виведення та зміни назви континента. Клас Місто має поля «назва», «населення», «країна» (об'єкт класу Країна), «континент» (об'єкт класу Континент) та методи створення, редагування та виведення усіх характеристик міста. З використанням цих класів скласти програму опису деякого міста та вивести інформацію щодо нього.
14. Описати класи Продукт, Склад та Категорія. Клас Склад є нащадком класів Продукт та Категорія. Клас Продукт має поля «назва», «вартість», «кількість на складі» та методи створення, редагування та виведення усіх характеристик продукту. Клас Категорія має поля «назва категорії» та методи створення, виведення та зміни назви категорії. Клас Склад має поля «продукт» (об'єкт класу Продукт), «категорія» (об'єкт класу Категорія) та методи створення, редагування та виведення усіх характеристик складу. З використанням цих класів скласти програму опису деякого складу та вивести інформацію щодо нього.

Лабораторна робота № 4. Реалізація класу на базі уже створеного класу: наслідування (багаторівневе). Порядок виклику методів (MRO) в Python

Мета лабораторної роботи:

- ✓ Ознайомити здобувачів освіти з концепцією багаторівневого наслідування у мові Python, коли класи наслідуються через кілька рівнів ієрархії.
- ✓ Навчити студентів розуміти та реалізовувати багаторівневе наслідування, формуючи глибші ієрархії класів для більш складних програмних рішень.
- ✓ Пояснити механізм порядку виклику методів (MRO – Method Resolution Order) у Python, а також навчити користуватися ним для визначення черговості викликів методів у багаторівневих і множинних наслідуваннях.
- ✓ Закріпити навички створення ієрархій класів, що дозволяють ефективно повторно використовувати код і забезпечувати розширюваність функціоналу.

Завдання спрямовані на формування практичного розуміння багаторівневого наслідування та порядку виклику методів у складних системах класів.

Розглянемо **приклад** програмування завдання 3. Отже, маємо:

- створіть базовий клас Device, який має атрибут для зберігання назви пристрою та метод operate(), що повертає загальне повідомлення «Operating the device».
- створіть клас Smartphone, який наслідує від Device. Додайте метод call(), що повертає «Making a call».
- створіть клас Laptop, який також наслідує від Device. Додайте метод code(), що повертає «Writing code».
- створіть клас HybridDevice, який одночасно наслідує від класів Smartphone та Laptop. Реалізуйте метод operate(), який поєднує функції обох пристроїв.

Додаткові умови:

Створіть об'єкт класу HybridDevice і продемонструйте роботу всіх методів (operate(), call(), code()).

Створіть об'єкти класів Smartphone і Laptop, викличте відповідні методи.

Як бачимо, тут одночасно буде використано і множинне наслідування і багаторівневе. Хоча класи зовсім прості, потрібно уважно і правильно побудувати дану ієрархію.

Реалізація на Python буде мати вигляд:

```
# Базовий клас Device
```

```
    class Device:
        def __init__(self, name):
            self.name = name

        def operate(self):
            return f»Operating the {self.name}»

    # Клас Smartphone, який наслідує від Device
    class Smartphone(Device):
        def call(self):
            return «Making a call»

    # Клас Laptop, який також наслідує від Device
    class Laptop(Device):
        def code(self):
            return «Writing code»

    # Клас HybridDevice, який наслідує від класів Smartphone і
Laptop
    class HybridDevice(Smartphone, Laptop):
        def operate(self):
            return      f»{Smartphone.operate(self)}      and
{Laptop.code(self)}»

    # Створюємо об'єкти і демонструємо роботу методів
    hybrid_device = HybridDevice(«HybridDevice»)
    print(hybrid_device.operate())       #    Operating    the
HybridDevice and Writing code
    print(hybrid_device.call())     # Making a call
    print(hybrid_device.code())     # Writing code

    smartphone = Smartphone(«Smartphone»)
    print(smartphone.operate())     # Operating the Smartphone
    print(smartphone.call())        # Making a call

    laptop = Laptop(«Laptop»)
    print(laptop.operate())         # Operating the Laptop
    print(laptop.code())            # Writing code
```

Отримаємо результат:

```
Operating the HybridDevice and Writing code
Making a call
Writing code
Operating the Smartphone
Making a call
Operating the Laptop
Writing code
```

Це послідовний результат виконання всіх методів у програмі для кожного створеного об'єкта.

Завдання на лабораторну роботу:

1. Реалізувати багаторівневе наслідування, продемонструвати детальну роботу з класом. Створити багаторівневу структуру наслідування відповідно до Вашого варіанту.

1.

1. Створити базовий клас «Товар» (Product) з атрибутами «назва» (name), «ціна» (price) і методами «отримати назву» (get_name) і «отримати ціну» (get_price).

2. Створити похідний клас «Електроніка» (Electronics), що успадковує клас «Товар» і додає атрибут «гарантія» (warranty) і метод «отримати гарантію» (get_warranty).

3. Створити похідний клас «Мобільний телефон» (MobilePhone), що успадковує клас «Електроніка» і додає атрибути «оператор» (carrier) і «оперативна пам'ять» (ram) і методи «отримати оператора» (get_carrier) і «отримати оперативну пам'ять» (get_ram).

4. Створити об'єкт класу «Мобільний телефон» і викликати методи «отримати назву», «отримати ціну», «отримати гарантію», «отримати оператора» і «отримати оперативну пам'ять» для отримання відповідних значень.

2.

1. Створити базовий клас «Транспортний засіб» (Vehicle) з атрибутами «швидкість» (speed) і методами «отримати швидкість» (get_speed) і «розрахувати час подорожі» (calculate_travel_time), що розраховує час подорожі на вказану відстань.

2. Створити похідний клас «Автомобіль» (Car), що успадковує клас «Транспортний засіб» і додає атрибути «потужність двигуна» (engine_power) і «тип палива» (fuel_type) і метод «отримати потужність двигуна» (get_engine_power).
3. Створити похідний клас «Грузовик» (Truck), що успадковує клас «Автомобіль» і додає атрибут «вантажопідйомність» (carrying_capacity) і метод «отримати вантажопідйомність» (get_carrying_capacity).
4. Створити об'єкт класу «Грузовик» і викликати методи «отримати швидкість», «розрахувати час подорожі», «отримати потужність двигуна» і «отримати вантажопідйомність» для отримання відповідних значень.

3.

1. Створити базовий клас «Музичний інструмент» (MusicalInstrument) з атрибутом «тип інструменту» (instrument_type) та методами «отримати тип інструменту» (get_instrument_type) і «грати» (play), що виводить текст про звук інструменту.
2. Створити похідний клас «Струнний інструмент» (StringInstrument), що успадковує клас «Музичний інструмент» і додає атрибут «кількість струн» (number_of_strings) та метод «отримати кількість струн» (get_number_of_strings).
3. Створити похідний клас «Гітара» (Guitar), що успадковує клас «Струнний інструмент» і додає атрибут «тип гітари» (guitar_type) та метод «отримати тип гітари» (get_guitar_type).
4. Створити об'єкт класу «Гітара» і викликати методи «отримати тип інструменту», «отримати кількість струн» та «отримати тип гітари» для виведення відповідних даних.

4.

1. Створити базовий клас «Тварина» (Animal) з атрибутом «вид» (species) та методами «отримати вид» (get_species) і «видавати звук» (make_sound).
2. Створити похідний клас «Птах» (Bird), що успадковує клас «Тварина» і додає атрибут «можливість польоту» (can_fly) та метод «перевірити можливість польоту» (check_flight_ability).

3. Створити похідний клас «Орел» (Eagle), що успадковує клас «Птах» і додає атрибут «розмах крил» (wingspan) та метод «отримати розмах крил» (get_wingspan).
4. Створити об'єкт класу «Орел» і викликати методи «отримати вид», «перевірити можливість польоту» і «отримати розмах крил».
5.

1. Створити базовий клас «Електроніка» (Electronics) з атрибутом «споживана потужність» (power_usage) та методами «отримати потужність» (get_power_usage) і «увімкнути» (turn_on).
2. Створити похідний клас «Мобільний пристрій» (MobileDevice), що успадковує клас «Електроніка» і додає атрибут «мобільність» (mobility) та метод «перевірити мобільність» (check_mobility).
3. Створити похідний клас «Смартфон» (Smartphone), що успадковує клас «Мобільний пристрій» і додає атрибут «операційна система» (operating_system) та метод «отримати операційну систему» (get_operating_system).
4. Створити об'єкт класу «Смартфон» і викликати методи «отримати потужність», «перевірити мобільність» та «отримати операційну систему».

6.

1. Створити базовий клас «Рослина» (Plant) з атрибутом «тип» (plant_type) та методами «отримати тип» (get_plant_type) і «вирощувати» (grow).
2. Створити похідний клас «Дерево» (Tree), що успадковує клас «Рослина» і додає атрибут «висота» (height) та метод «отримати висоту» (get_height).
3. Створити похідний клас «Плодовите дерево» (FruitTree), що успадковує клас «Дерево» і додає атрибут «тип плодів» (fruit_type) та метод «отримати тип плодів» (get_fruit_type).
4. Створити об'єкт класу «Плодовите дерево» і викликати методи «отримати тип», «отримати висоту» та «отримати тип плодів».

7.

1. Створити базовий клас «Зброя» (Weapon) з атрибутом «калібр» (caliber) та методами «отримати калібр» (get_caliber) і «стріляти» (shoot).
2. Створити похідний клас «Вогнепальна зброя» (Firearm), що успадковує клас «Зброя» і додає атрибут «довжина ствола» (barrel_length) та метод «отримати довжину ствола» (get_barrel_length).
3. Створити похідний клас «Гвинтівка» (Rifle), що успадковує клас «Вогнепальна зброя» і додає атрибут «дальність стрільби» (range) та метод «отримати дальність стрільби» (get_range).
4. Створити об'єкт класу «Гвинтівка» і викликати методи «отримати калібр», «отримати довжину ствола» та «отримати дальність стрільби».

8.

1. Створити базовий клас «Робочий пристрій» (WorkDevice) з атрибутом «модель» (model) та методами «отримати модель» (get_model) і «виконувати роботу» (perform_work).
2. Створити похідний клас «Комп'ютер» (Computer), що успадковує клас «Робочий пристрій» і додає атрибут «оперативна пам'ять» (ram) та метод «отримати оперативну пам'ять» (get_ram).
3. Створити похідний клас «Ноутбук» (Laptop), що успадковує клас «Комп'ютер» і додає атрибут «розмір екрану» (screen_size) та метод «отримати розмір екрану» (get_screen_size).
4. Створити об'єкт класу «Ноутбук» і викликати методи «отримати модель», «отримати оперативну пам'ять» і «отримати розмір екрану».

9.

1. Створити базовий клас «Транспортний засіб» (Vehicle) з атрибутом «максимальна швидкість» (max_speed) та методами «отримати максимальну швидкість» (get_max_speed) і «рухатись» (move).
2. Створити похідний клас «Водний транспорт» (WaterVehicle), що успадковує клас «Транспортний засіб» і додає атрибут «водотоннажність» (displacement) та метод «отримати водотоннажність» (get_displacement).

3. Створити похідний клас «Корабель» (Ship), що успадковує клас «Водний транспорт» і додає атрибут «тип судна» (ship_type) та метод «отримати тип судна» (get_ship_type).
4. Створити об'єкт класу «Корабель» і викликати методи «отримати максимальну швидкість», «отримати водотоннажність» і «отримати тип судна».

10.

1. Створити базовий клас «Меблі» (Furniture) з атрибутом «матеріал» (material) та методами «отримати матеріал» (get_material) і «встановити» (place).
2. Створити похідний клас «Стілець» (Chair), що успадковує клас «Меблі» і додає атрибут «висота» (height) та метод «отримати висоту» (get_height).
3. Створити похідний клас «Офісний стілець» (OfficeChair), що успадковує клас «Стілець» і додає атрибут «наявність коліщаток» (has_wheels) та метод «перевірити наявність коліщаток» (check_wheels).
4. Створити об'єкт класу «Офісний стілець» і викликати методи «отримати матеріал», «отримати висоту» та «перевірити наявність коліщаток».

11.

1. Створити базовий клас «Книга» (Book) з атрибутом «кількість сторінок» (number_of_pages) та методами «отримати кількість сторінок» (get_number_of_pages) і «читати» (read).
2. Створити похідний клас «Наукова книга» (ScienceBook), що успадковує клас «Книга» і додає атрибут «галузь науки» (science_field) та метод «отримати галузь науки» (get_science_field).
3. Створити похідний клас «Фізична книга» (PhysicsBook), що успадковує клас «Наукова книга» і додає атрибут «тематика» (topic) та метод «отримати тематику» (get_topic).
4. Створити об'єкт класу «Фізична книга» і викликати методи «отримати кількість сторінок», «отримати галузь науки» і «отримати тематику».

12.

1. Створити базовий клас «Інструмент» (Tool) з атрибутом «призначення» (purpose) та методами «отримати призначення» (get_purpose) і «використовувати» (use).
2. Створити похідний клас «Ручний інструмент» (HandTool), що успадковує клас «Інструмент» і додає атрибут «розмір» (size) та метод «отримати розмір» (get_size).
3. Створити похідний клас «Молоток» (Hammer), що успадковує клас «Ручний інструмент» і додає атрибут «матеріал ручки» (handle_material) та метод «отримати матеріал ручки» (get_handle_material).
4. Створити об'єкт класу «Молоток» і викликати методи «отримати призначення», «отримати розмір» і «отримати матеріал ручки».

13.

1. Створити базовий клас «Їжа» (Food) з атрибутом «калорійність» (calories) та методами «отримати калорійність» (get_calories) і «їсти» (eat).
2. Створити похідний клас «Фрукт» (Fruit), що успадковує клас «Їжа» і додає атрибут «колір» (color) та метод «отримати колір» (get_color).
3. Створити похідний клас «Яблуко» (Apple), що успадковує клас «Фрукт» і додає атрибут «сорт» (variety) та метод «отримати сорт» (get_variety).
4. Створити об'єкт класу «Яблуко» і викликати методи «отримати калорійність», «отримати колір» і «отримати сорт».

14.

1. Створити базовий клас «Напій» (Beverage) з атрибутом «об'єм» (volume) та методами «отримати об'єм» (get_volume) і «пити» (drink).
2. Створити похідний клас «Газований напій» (Soda), що успадковує клас «Напій» і додає атрибут «вміст цукру» (sugar_content) та метод «отримати вміст цукру» (get_sugar_content).
3. Створити похідний клас «Кола» (Cola), що успадковує клас «Газований напій» і додає атрибут «бренд» (brand) та метод «отримати бренд» (get_brand).

4. Створити об'єкт класу «Кола» і викликати методи «отримати об'єм», «отримати вміст цукру» та «отримати бренд».

2. Застосувати порядок виклику методів (MRO) в Python. Для цього: реалізувати власну "гібридну" структуру класів, тобто, щоб було множинне і багаторівневе наслідування. Далі, для того аби навчитися розуміти як відбувається пошук методів написати самому, вручну, так як ви розумієте, послідовність пошуку методів у класах, а тоді використати для перевірки виведення послідовності пошуку методів спеціальний атрибут із іменем __mro__. Продемонструвати виведення кортежів для декількох класів, звірити ваші результати. Обов'язково побудувати рисунок та прописати до нього аналогічний код до лекційних прикладів.

3. Реалізувати систему класів, у якому буде багаторівневе наслідування і множинне наслідування, з відповідними властивостями та методами, які ілюструють ці принципи. Продемонструвати детальну роботу з класом. За бажанням можна створити ВЛАСНУ систему класів як у прикладах або виконати дане завдання.

1.

Вимоги до виконання:

- Створіть базовий клас Animal, який буде мати атрибут для зберігання імені тварини та метод sound(), що повертає загальний звук тварини.
- Створіть клас Mammal, який наслідує від Animal. Додайте до нього властивість для зберігання кількості кінцівок (leg_count) та метод walk(), що виводить інформацію про те, як ссавець пересувається.
- Створіть окремий клас CanFly, який має метод fly(), що повертає рядок «I can fly!».
- Створіть клас Bat, який одночасно наслідується від класів Mammal і CanFly, демонструючи множинне наслідування. У класі Bat реалізуйте метод sound(), який повертає специфічний для кажана звук.
- У класі Bat використайте конструктор базових класів для ініціалізації відповідних атрибутів.

Додаткові умови:

- Створіть об'єкт класу Bat і викличте всі доступні йому методи (sound(), walk(), fly()).
- Створіть об'єкт класу Mammal і продемонструйте роботу методу walk() та спадкованого методу sound().

2.

Вимоги до виконання:

- Створіть базовий клас Vehicle, який має атрибут для зберігання кількості коліс та метод move(), що виводить загальне повідомлення «Vehicle is moving».
- Створіть клас Car, який наслідує від Vehicle. Додайте до нього атрибут для зберігання типу пального і метод fuel_type(), який виводить тип пального.
- Створіть клас Electric, який має метод charge(), що повертає «Battery charging».
- Створіть клас ElectricCar, який наслідує від класів Car і Electric, демонструючи множинне наслідування. Додайте метод fuel_type(), що замінює батьківський, та повертає «Electric power».
- В класі ElectricCar ініціалізуйте відповідні атрибути базових класів.

Додаткові умови:

- Створіть об'єкт класу ElectricCar і продемонструйте роботу методів move(), fuel_type(), charge().
- Створіть об'єкт класу Car і покажіть роботу методів move() та fuel_type().

3.

Вимоги до виконання:

- Створіть базовий клас Employee, який має атрибут для зберігання імені та метод work(), що повертає загальне повідомлення «Working».
- Створіть клас Manager, який наслідує від Employee. Додайте метод manage(), що повертає «Managing team».
- Створіть клас Programmer, який також наслідує від Employee. Додайте метод code(), що повертає «Writing code».

- Створіть клас TechLead, який одночасно наслідує від класів Manager та Programmer. Реалізуйте метод work(), який поєднує роботу обох ролей.

Додаткові умови:

- Створіть об'єкт класу TechLead і продемонструйте роботу всіх методів (work(), manage(), code()).
- Створіть об'єкти класів Manager і Programmer, викличте відповідні методи.

4.

Вимоги до виконання:

- Створіть базовий клас Food, який має атрибут для назви продукту та метод eat(), що виводить «Eating food».
- Створіть клас Fruit, який наслідує від Food, додайте метод taste(), що повертає «Tastes sweet».
- Створіть клас Vegetable, який наслідує від Food, і додайте метод nutrients(), що повертає «Rich in vitamins».
- Створіть клас Tomato, який одночасно наслідує від класів Fruit і Vegetable. Реалізуйте метод taste(), що повертає «Sweet and savory».

Додаткові умови:

- Створіть об'єкт класу Tomato і викличте методи eat(), taste(), nutrients().
- Створіть об'єкти класів Fruit і Vegetable, викличте відповідні методи.

Лабораторна робота № 5. Реалізація поліморфізму мовою Python. Віртуальний метод. Зв'язки між класами: асоціація, агрегація, композиція, спадкування, залежність, реалізація

Мета лабораторної роботи:

- ✓ Навчитися створювати та використовувати поліморфні методи, що дозволяють викликати функції різних класів через спільний інтерфейс.
- ✓ Зрозуміти механізм реалізації віртуальних методів у Python, які дають можливість перевизначати методи в похідних класах.
- ✓ Дослідити зв'язки між класами: навчитися описувати взаємодію об'єктів без створення жорсткої залежності між ними (асоціація), опанувати роботу з об'єктами, де один клас виступає як частина іншого, але може існувати незалежно (агрегація), зрозуміти, як один об'єкт повністю залежить від іншого, і створювати класи, що керують життєвим циклом внутрішніх об'єктів (композиція).
- ✓ Засвоїти принципи успадкування, які дозволяють використовувати методи й атрибути базового класу в похідних класах, оптимізуючи код.
- ✓ Зрозуміти, як один клас може використовувати інший для виконання своїх функцій, що впливає на архітектуру програми.

Завдання спрямовані на формування практичного розуміння поліморфізму, віртуального методу та зв’язків між класами.

Розглянемо **приклад** виконання завдання 2. Отже, продемонструйте використання поліморфізму у Python без наслідування та детальну роботу з класами та об'єктами.

Класи:

Shufr1

Метод encrypt(message) – шифрує повідомлення методом заміни, замінюючи кожну літеру на деяку двійкову послідовність з 8 символів.

Метод decrypt(message) – дешифрує повідомлення, зашифроване методом encrypt (шифрування є симетричним).

Shufr2

Метод encrypt(message) – шифрує повідомлення за допомогою методу перестановки: переставляє дві сусідні літери.

Метод decrypt(message) – дешифрує повідомлення, зашифроване методом encrypt ((шифрування є симетричним).

Класи Shufr1 та Shufr2 мають однакові методи encrypt та decrypt, але реалізують їх по-різному, тому код на Python буде мати вигляд:

```
    class Shufr1:
        #  Метод  шифрування  заміною  літер  на  двійкові
послідовності
        def encrypt(self, message):
            binary_message = ''.join(format(ord(char), '08b')
for char in message)
            return binary_message

        # Метод дешифрування заміною двійкових послідовностей
на літери
        def decrypt(self, message):
            chars  =  [message[i:i  +  8]  for  i  in  range(0,
len(message), 8)]
            decrypted_message = ''.join(chr(int(char, 2)) for
char in chars)
            return decrypted_message

    class Shufr2:
        # Метод шифрування перестановкою сусідніх літер
        def encrypt(self, message):
            message = list(message)
            for i in range(0, len(message) - 1, 2):
                message[i], message[i + 1] = message[i + 1],
message[i]
            return ''.join(message)

        # Метод дешифрування перестановкою назад
        def decrypt(self, message):
            message = list(message)
            for i in range(0, len(message) - 1, 2):
                message[i], message[i + 1] = message[i + 1],
message[i]
            return ''.join(message)

    # Поліморфізм: робота з різними класами через однаковий
інтерфейс
    def process_message(shufr, message):
        encrypted = shufr.encrypt(message)
```

```
        decrypted = shufr.decrypt(encrypted)
        return encrypted, decrypted

    # Тестування
    message = "hello"
    shufr1 = Shufr1()
    shufr2 = Shufr2()

    print("Shufr1:")
    encrypted1, decrypted1 = process_message(shufr1, message)
    print(f"Encrypted: {encrypted1}")
    print(f"Decrypted: {decrypted1}")

    print("\nShufr2:")
    encrypted2, decrypted2 = process_message(shufr2, message)
    print(f"Encrypted: {encrypted2}")
    print(f"Decrypted: {decrypted2}")
```

Shufr1: Шифрує повідомлення, замінюючи кожну літеру на двійковий код із 8 символів. Після цього дешифрує, перетворюючи двійковий код назад у літери. Shufr2: Шифрує повідомлення методом перестановки сусідніх літер. Дешифрує, повторюючи ту ж операцію.

Виведення:

```
Shufr1:
Encrypted: 01101000011001010110110001101100011011111
Decrypted: hello

Shufr2:
Encrypted: ehll o
Decrypted: hello
```

Таким чином, обидва класи мають однакові методи, але реалізують їх різними способами, що ілюструє поліморфізм без наслідування.

Розберемо детальніше шифрування у класі Shufr1, маємо рядок коду:

```
binary_message = ''.join(format(ord(char), '08b') for char in message)
```

Цей рядок шифрує повідомлення, замінюючи кожну літеру на двійкову послідовність із 8 бітів. Розглянемо його частинами:

- **ord(char)**: функція ord() перетворює символ у його числове представлення (ASCII-код). Наприклад, для символу 'h' вона повертає число 104.

- **format(ord(char), '08b')**: функція format() перетворює число в двійкову (бінарну) форму. Параметр '08b' означає, що результат має бути 8-бітним (з нулями на початку, якщо потрібно). Наприклад, для числа 104 це буде '01101000'.
- **for char in message**: це цикл, що перебирає всі символи у рядку message.
- **''.join(...)**: функція join() об'єднує всі елементи в один рядок без пробілів. Тобто, всі двійкові послідовності для кожного символу будуть з'єднані в один рядок.

  Наприклад, для повідомлення 'hello':

  h → '01101000'

  e → '01100101'

  l → '01101100'

  l → '01101100'

  o → '01101111'

  Результат:

  0110100001100101011011000110110001101111

  А тепер дешифрування у класі Shufr1:

```
chars = [message[i:i + 8] for i in range(0, len(message), 8)]
```

Цей рядок розбиває двійкове повідомлення на послідовності по 8 символів (біт), щоб потім їх можна було перетворити назад у символи:

- **range(0, len(message), 8)**: створює послідовність індексів з кроком 8, тобто 0, 8, 16 і так далі до кінця рядка. Це потрібно, щоб виділяти 8-бітні блоки.
  - **message[i:i + 8]**: для кожного індексу i з послідовності отримує 8-символьний фрагмент рядка. Наприклад: message[0:8] → '01101000', message[8:16] → '01100101', і так далі.

  В результаті отримаємо список, де кожен елемент — це 8-бітний блок:

  ['01101000', '01100101', '01101100', '01101100', '01101111']

  Наступний рядок:

```
decrypted_message = ''.join(chr(int(char, 2)) for char in chars)
```

Цей рядок перетворює двійкові послідовності назад у символи та об'єднує їх у повідомлення:

- **int(char, 2)**: перетворює двійковий рядок char в ціле число. Параметр 2 вказує на те, що це двійковий код. Наприклад, '01101000' перетвориться на число 104.
- **chr(int(char, 2))**: функція chr() перетворює отримане число назад у символ ASCII. Наприклад, для числа 104 це буде символ 'h'.

- **for char in chars**: перебирає всі 8-бітні блоки зі списку chars.
- **''.join(...)**: об'єднує всі символи в один рядок.

Наприклад, для вхідних даних:
['01101000', '01100101', '01101100', '01101100', '01101111']
отримаємо:
'hello'
Інший метод простіший, по коду все повинно бути зрозуміло.

Завдання на лабораторну роботу:

1. Продемонструйте використання поліморфізму у Python при наслідуванні у класах згідно Вашого варіанту:

1. Типи акаунтів у банку

Базовий клас: BankAccount

- Методи:
  deposit(amount) – додає кошти на рахунок.
  withdraw(amount) – знімає кошти з рахунку.

Класи-наслідники:

- SavingsAccount
  deposit(amount) – додає бонусні відсотки до депозиту.
  withdraw(amount) – обмежує зняття коштів раз на місяць.
- CurrentAccount
  deposit(amount) – просто додає кошти до поточного балансу.
  withdraw(amount) – дозволяє знімати кошти без обмежень.
- PremiumAccount
  deposit(amount) – додає більше бонусів при внесенні великих сум.
  withdraw(amount) – дозволяє знімати кошти без обмежень та з преміальними умовами, наприклад, без комісії.

2. Види транспорту

Базовий клас: Vehicle

- Методи:
  start() – запускає транспортний засіб.
  stop() – зупиняє транспортний засіб.

Класи-наслідники:

- Car
  start() – заводить двигун і відтворює звук старту.
  stop() – вимикає двигун і паркує авто.
- Bike
  start() – перевіряє шини та відтворює звук дзвінка.

stop() – просто зупиняє рух.

- ElectricCar
  start() – перевіряє рівень заряду батареї і запускає двигун.
  stop() – вимикає двигун та зберігає енергію.

3. Робота з документами

Базовий клас: Document

- Методи:
  open() – відкриває документ для редагування.
  save() – зберігає зміни в документі.

Класи-наслідники:

- WordDocument
  open() – відкриває текстовий файл у редакторі.
  save() – зберігає текст, форматування та стилі.
- Spreadsheet
  open() – відкриває електронну таблицю.
  save() – зберігає дані та формули.
- Presentation
  open() – відкриває презентацію зі слайдами.
  save() – зберігає слайди, текст і анімації.

4. Типи повідомлень

Базовий клас: Message

- Методи:
  send() – надсилає повідомлення.
  schedule() – планує відправку повідомлення.

Класи-наслідники:

- EmailMessage
  send() – перевіряє наявність вкладень і відправляє лист.
  schedule() – планує відправку на конкретний час.
- SMSMessage
  send() – перевіряє, чи не перевищено ліміт символів, і надсилає SMS.
  schedule() – планує відправку з урахуванням тимчасових зон.
- PushNotification
  send() – відправляє сповіщення на мобільний пристрій.
  schedule() – надсилає сповіщення через систему планування.

5. Види працівників

Базовий клас: Employee

- Методи:

  work() – виконує роботу працівника.

  calculate_salary() – обчислює зарплату працівника.

Класи-наслідники:

- Manager

  work() – координує команди і керує проектами.

  calculate_salary() – розраховує зарплату на основі керівних обов'язків і бонусів.

- Developer

  work() – пише код і розробляє функціонал програмного забезпечення.

  calculate_salary() – розраховує зарплату на основі кількості написаного коду і завершених задач.

- Designer

  work() – створює макети, графіку та візуальні елементи.

  calculate_salary() – розраховує зарплату на основі завершених проектів і рівня креативності.

6. Форми на площині

Базовий клас: Shape

- Методи:

  area() – обчислює площу фігури.

  perimeter() – обчислює периметр фігури.

Класи-наслідники:

- Circle

  area() – обчислює площу за формулою $\pi * r^2$.

  perimeter() – обчислює довжину кола за формулою $2 * \pi * r$.

- Rectangle

  area() – обчислює площу за формулою ширина * висота.

  perimeter() – обчислює периметр за формулою 2 * (ширина + висота).

- Triangle

  area() – обчислює площу за формулою ½ * основа * висота.

  perimeter() – обчислює периметр як суму всіх сторін.

7. Різні види магазинів

Базовий клас: Store

- Методи:

  sell_item() – продає товар клієнту.

  refund_item() – оформляє повернення товару.

Класи-наслідники:

- GroceryStore
  sell_item() – перевіряє термін придатності продукту перед продажем.
  refund_item() – перевіряє, чи не зіпсований товар для повернення.
- ElectronicsStore
  sell_item() – перевіряє гарантію та стан товару перед продажем.
  refund_item() – перевіряє гарантійний період для повернення.
- ClothingStore
  sell_item() – пропонує додаткові аксесуари під час продажу.
  refund_item() – перевіряє стан повернутої речі (відсутність пошкоджень).

8. Моделі літаків

Базовий клас: Airplane

- Методи:
  take_off() – підйом в повітря.
  land() – посадка літака.

Класи-наслідники:

- PassengerPlane
  take_off() – перевіряє наявність пасажирів та дозвіл на зліт.
  land() – знижує швидкість і здійснює м'яку посадку.
- CargoPlane
  take_off() – перевіряє вантаж і вагу перед зльотом.
  land() – забезпечує безпечне транспортування вантажу під час посадки.
- FighterJet
  take_off() – активує прискорювачі та злітає з великою швидкістю.
  land() – використовує гальмівні парашути для швидкої зупинки.

9. Спортивні тренування

Базовий клас: Training

- Методи:
  start_training() – початок тренування.
  end_training() – завершення тренування.

Класи-наслідники:

- FootballTraining
  start_training() – розминка, тактичні вправи, гра в міні-футбол.
  end_training() – заключна розтяжка та короткий підсумок тренера.
- BasketballTraining

start_training() – вправи з кидків, командна робота, захисні стратегії.

end_training() – вправи на кидки з лінії штрафного кидка.

- TennisTraining

start_training() – вправи на техніку ударів, подачі, стратегія одиночної гри.

end_training() – завершальні вправи на витривалість та техніку.

10. Типи перевезень

Базовий клас: Transport

- Методи:

load() – завантаження товару.

unload() – розвантаження товару.

Класи-наслідники:

- Truck

load() – перевіряє відповідність габаритів товару розмірам вантажівки.

unload() – швидке розвантаження та підготовка до нової партії товарів.

- Ship

load() – перевіряє масу та баланс вантажу для запобігання перевантаження.

unload() – забезпечує поступове розвантаження для збереження стабільності корабля.

- Train

load() – перевіряє кількість вагонів та їхню здатність перевозити вантаж.

unload() – розвантажує товари на різних станціях, відповідно до розкладу.

11. Система замовлення їжі

Базовий клас: FoodOrder

- Методи:

place_order() – оформлення замовлення.

cancel_order() – скасування замовлення.

Класи-наслідники:

- PizzaOrder

place_order() – вибір інгредієнтів для піци, розмір та додаткові соуси.

cancel_order() – перевіряє, чи не пішло замовлення в процес готування, перед скасуванням.

- SushiOrder

  place_order() – вибір різновидів суші та кількість ролів.

  cancel_order() – скасовує замовлення, якщо його ще не почали готувати.
- BurgerOrder

  place_order() – вибір булки, м'яса та додаткових інгредієнтів.

  cancel_order() – скасовує замовлення, якщо воно ще не розпочате.

12. Типи медичних послуг

Базовий клас: MedicalService

- Методи:

  perform_service() – виконання медичної послуги.

  charge_patient() – виставлення рахунку пацієнту.

Класи-наслідники:

- Surgery

  perform_service() – проводить хірургічну операцію із залученням команди лікарів.

  charge_patient() – виставляє рахунок за проведену операцію та використовувані матеріали.
- Checkup

  perform_service() – проводить базове обстеження пацієнта.

  charge_patient() – виставляє рахунок за консультацію та обстеження.
- PhysicalTherapy

  perform_service() – проводить сеанс фізичної терапії (масаж, вправи).

  charge_patient() – виставляє рахунок за кожен сеанс фізіотерапії.

13. Типи файлів

Базовий клас: File

- Методи:

  open() – відкриває файл.

  close() – закриває файл.

Класи-наслідники:

- TextFile

  open() – відкриває текстовий файл для читання або редагування.

  close() – зберігає та закриває текстовий файл.
- ImageFile

  open() – відкриває зображення у графічному редакторі.

  close() – зберігає зміни і закриває зображення.
- AudioFile

open() – відкриває аудіофайл для прослуховування або редагування.

close() – зупиняє відтворення та закриває файл.

14. Типи будівель

Базовий клас: Building

- Методи:

calculate_area() – обчислює загальну площу будівлі.

describe() – описує особливості будівлі.

Класи-наслідники:

- House

calculate_area() – обчислює площу житлової зони та додаткових приміщень (гараж, підвал).

describe() – описує кількість кімнат, наявність двору, тераси тощо.

- Office

calculate_area() – обчислює площу офісного простору, кімнат для нарад та загальних зон.

describe() – описує кількість робочих місць, конференц-залів і зручностей для співробітників.

- Mall

calculate_area() – обчислює площу торговельних зон, фудкортів та зон розваг.

describe() – описує кількість магазинів, зон для відпочинку і парковок.

2. Продемонструйте використання поліморфізму у Python без наслідування та детальну роботу з класами та об'єктами.

1. Фінансові транзакції

Класи:

- Loan

Метод calculate_interest(amount, rate, years) – обчислює загальну суму відсотків по кредиту за формулою amount * rate * years.

Метод total_payment(amount, rate, years) – обчислює загальну суму для виплати, включаючи відсотки.

- Mortgage

Метод calculate_interest(amount, rate, years) – обчислює щомісячні виплати за іпотекою з використанням складних відсотків.

Метод total_payment(amount, rate, years) – обчислює загальну вартість виплат, включаючи відсотки і страхування.

- Investment

Метод calculate_interest(amount, rate, years) – обчислює очікуваний прибуток від інвестицій з використанням складних відсотків.

Метод total_payment(amount, rate, years) – обчислює загальну суму, яку інвестор отримає після закінчення періоду.

2. Прості фігури

Класи:

- DiagonalLineDrawer

Метод draw(t, length) – малює діагональну лінію з вказаною довжиною length, використовуючи об'єкт черепахи t.

- EllipseDrawer

Метод draw(t, width, height) – малює еліпс (приблизно), використовуючи об'єкт черепахи t, ширину width і висоту height.

- PolygonDrawer

Метод draw(t, sides, size) – малює правильний багатокутник з sides сторонами, використовуючи об'єкт черепахи t і розмір size.

3. Системи керування транспортом

Класи:

- TaxiService

Метод calculate_fare(distance, time) – обчислює вартість поїздки на таксі залежно від відстані та часу.

Метод calculate_wait_time(traffic_level) – обчислює очікуваний час поїздки залежно від рівня заторів на дорогах.

- RideSharing

Метод calculate_fare(distance, time) – обчислює вартість поїздки з урахуванням спільного маршруту та можливих пасажирів.

Метод calculate_wait_time(traffic_level) – обчислює час очікування водія на замовлення залежно від заторів.

- BusService

Метод calculate_fare(distance) – обчислює вартість квитка на автобус залежно від відстані.

Метод calculate_wait_time(schedule) – обчислює час до прибуття автобуса залежно від розкладу.

4. Математичні операції

Класи:

- Matrix

Метод multiply(matrix1, matrix2) – виконує множення двох матриць.

Метод transpose(matrix) – обчислює транспоновану матрицю.

- Vector

Метод multiply(vector1, vector2) – виконує скалярне або векторне множення векторів.

Метод magnitude(vector) – обчислює довжину вектора.

- Polynomial

Метод multiply(poly1, poly2) – виконує множення двох поліномів.

Метод differentiate(poly) – обчислює похідну полінома.

5. Фінансові розрахунки

Класи:

- Salary

Метод calculate_annual_salary(monthly_salary) – обчислює річну зарплату, виходячи з місячного доходу.

Метод calculate_tax(annual_salary) – обчислює податки на основі річного доходу.

- Contractor

Метод calculate_annual_salary(hourly_rate, hours_per_week) – обчислює річний дохід на основі погодинної ставки і кількості годин на тиждень.

Метод calculate_tax(annual_salary) – обчислює податки на основі річного доходу.

- Freelancer

Метод calculate_annual_salary(project_rates) – обчислює річний дохід на основі виконаних проектів та їх ставок.

Метод calculate_tax(annual_salary) – обчислює податки з річного доходу з урахуванням витрат на виконання проектів.

6. Моделювання руху

Класи:

- Projectile

Метод calculate_trajectory(speed, angle) – обчислює траєкторію руху об'єкта, який кинувся під певним кутом і швидкістю.

Метод calculate_time_of_flight(speed, angle) – обчислює час польоту об'єкта.

- Car

Метод calculate_trajectory(speed, acceleration) – обчислює траєкторію автомобіля під впливом постійного прискорення.

Метод calculate_time_to_stop(speed, deceleration) – обчислює час до повної зупинки автомобіля при гальмуванні.

- Boat

Метод calculate_trajectory(current_speed, wind_speed) – обчислює траєкторію руху човна, враховуючи швидкість течії та вітру.

Метод calculate_time_to_destination(distance, speed) – обчислює час до прибуття в пункт призначення.

7. Криптографія (прості методи шифрування)

Класи:

- ROT13Cipher

Метод encrypt(message) – шифрує повідомлення за допомогою шифру ROT13, зміщуючи кожну літеру на 13 позицій в алфавіті.

Метод decrypt(message) – дешифрує повідомлення, зашифроване шифром ROT13 (оскільки ROT13 є симетричним, той самий метод можна використовувати і для шифрування, і для дешифрування).

- VigenereCipher

Метод encrypt(message, keyword) – шифрує повідомлення за допомогою шифру Віженера з використанням ключового слова.

Метод decrypt(message, keyword) – дешифрує повідомлення, зашифроване шифром Віженера.

8. Числові ряди

Класи:

- ArithmeticSeries

Метод sum(first_term, common_difference, num_terms) – обчислює суму арифметичної прогресії.

Метод nth_term(first_term, common_difference, term_number) – знаходить n-й член арифметичної прогресії.

- GeometricSeries

Метод sum(first_term, common_ratio, num_terms) – обчислює суму геометричної прогресії.

Метод nth_term(first_term, common_ratio, term_number) – знаходить n-й член геометричної прогресії.

- FibonacciSeries

Метод sum(num_terms) – обчислює суму перших n членів ряду Фібоначчі.

Метод nth_term(term_number) – знаходить n-й член ряду Фібоначчі.

9. Статистичні операції

Класи:

- MeanCalculator

Метод calculate(data) – обчислює середнє арифметичне для набору даних.

Метод weighted_mean(data, weights) – обчислює зважене середнє для набору даних.

- MedianCalculator

Метод calculate(data) – знаходить медіану для набору даних.

Метод grouped_median(data) – обчислює медіану для згрупованих даних.

- ModeCalculator

Метод calculate(data) – знаходить моду для набору даних.

Метод frequency_table(data) – створює таблицю частот для набору даних.

10. Прості візерунки

Класи:

- GridDrawer

Метод draw(t, rows, columns, cell_size) – малює сітку з вказаною кількістю рядків rows, стовпців columns і розміром клітинки cell_size, використовуючи об'єкт черепахи t.

- CrossDrawer

Метод draw(t, size) – малює хрест з вказаним розміром size, використовуючи об'єкт черепахи t.

- PlusDrawer

Метод draw(t, size) – малює знак плюс з вказаним розміром size, використовуючи об'єкт черепахи t.

11. Аналіз часу виконання алгоритмів

Класи:

- BubbleSortAnalysis

Метод sort(array) – виконує сортування масиву методом бульбашки.

Метод time_complexity(n) – обчислює час виконання для методу сортування бульбашкою.

- MergeSortAnalysis

Метод sort(array) – виконує сортування масиву методом злиття.

Метод time_complexity(n) – обчислює час виконання для методу сортування злиттям.

- QuickSortAnalysis

Метод sort(array) – виконує сортування масиву методом швидкого сортування.

Метод time_complexity(n) – обчислює час виконання для методу швидкого сортування.

12. Криптографія (прості методи шифрування)

Класи:

- CaesarCipher

Метод encrypt(message, shift) – шифрує повідомлення за допомогою шифру Цезаря, зміщуючи кожну літеру на задану кількість позицій.

Метод decrypt(message, shift) – дешифрує повідомлення, яке було зашифроване шифром Цезаря шляхом зміщення літер назад на ту ж кількість позицій.

- AtbashCipher

Метод encrypt(message) – шифрує повідомлення за допомогою шифру Атбаша, замінюючи кожну літеру на її «дзеркальну» літеру (A -> Z, B -> Y, тощо).

Метод decrypt(message) – дешифрує повідомлення, зашифроване шифром Атбаша, виконуючи ту ж саму операцію заміни літер у зворотному порядку.

13. Гармонійний ряд

Класи:

- HarmonicSeries

Метод sum(num_terms) – обчислює суму перших n членів гармонійного ряду (1 + 1/2 + 1/3 + ... + 1/n).

Метод nth_term(term_number) – знаходить n-й член гармонійного ряду (1/n).

- ReciprocalSeries

Метод sum(num_terms) – обчислює суму ряду зворотних значень для перших n членів (1 + 1/a + 1/b + ...).

Метод nth_term(term_number) – знаходить n-й член ряду, де кожен член є зворотним значенням певного числа (1/a, 1/b, ...).

- ExponentialSeries

Метод sum(num_terms, base) – обчислює суму експоненційного ряду з базою base, тобто 1 + base + base² + ... + base^(n-1).

Метод nth_term(term_number, base) – знаходить n-й член експоненційного ряду з базою base (base^(n-1)).

14. Фігури

Класи:

- TriangleDrawer

  Метод draw(t, size) – малює рівносторонній трикутник з використанням об'єкта черепахи t і вказаного розміру size.

- FivePointStarDrawer

  Метод draw(t, size) – малює п'ятикутну зірку за допомогою черепахи t і розміру променя size.

- CircleDrawer

  Метод draw(t, size) – малює трапецію з використанням об'єкта черепахи t і 2-ма сторонами size.

3. Продемонструвати розуміння зв'язків між класами на вибір: або агрегація, або залежність, або реалізація. Аналогічно прикладів у лекції продемонструвати зв'язки між 2-ма класами. Самому придумати класи (можна взяти з попередніх завдань чи лабораторних робіт) та застосувати зв'язок. Обґрунтувати чому саме такий зв'язок між такими класами.

Лабораторна робота № 6. Спеціальні поля та методи. Особливості перевантаження операторів

Мета лабораторної роботи:

- ✓ Ознайомити здобувачів освіти з поняттям спеціальних полів та методів у класах.
- ✓ Ознайомити з особливостями перевантаження операторів у мовах програмування.
- ✓ Навчитися реалізовувати спеціальні методи для роботи з об'єктами.
- ✓ Навчитися перевантажувати оператори для поліпшення їхньої функціональності.

Завдання спрямовані на формування практичного розуміння використання спеціальних полів і методів у класах, а також особливостей перевантаження операторів для створення більш зручних та ефективних об'єктів у програмуванні.

Розглянемо **приклад** реалізації завдання 1. Продемонструвати роботу з використанням спеціальних полів та методів. Крім того, що виконати завдання по списку, то ще і для кожного завдання вивести спеціальні поля для класу і для екземпляру. Продемонструвати детальну роботу з класом. Створіть клас «Архів документів». Реалізуйте методи для звертання до документів за індексом (__getitem__), видалення документів за індексом (__delitem__) та перевірки наявності документу в архіві (__contains__).

```
class Archive:
    """Клас для зберігання документів в архіві"""
    num_instances = 0  # Лічильник кількості екземплярів класу

    def __init__(self, documents):
        self.documents = documents
        Archive.num_instances += 1  # Збільшуємо кількість екземплярів
        self.change_log = []  # Спеціальне поле для історії змін

    # Метод для отримання документа за індексом
    def __getitem__(self, index):
        return self.documents[index]
```

```
        # Метод для видалення документа за індексом
        def __delitem__(self, index):
            removed_doc = self.documents.pop(index)
            self.change_log.append(f"Deleted document at index
{index}: {removed_doc}")

        # Метод для перевірки наявності документа в архіві
        def __contains__(self, item):
            return item in self.documents

        # Метод для виведення інформації про архів
        def __str__(self):
            return   f"Archive   with   {len(self.documents)}
documents. Instance number: {Archive.num_instances}"

    # Демонстрація роботи класу
    if __name__ == "__main__":
        # Створюємо архів з кількома документами
        archive    =    Archive(["Document1",    "Document2",
"Document3"])

        # Використання спеціальних полів класу і екземпляра
        print("Class       name       (назва       класу):",
archive.__class__.__name__)
        print("Module            name            (модуль):",
archive.__class__.__module__)
        print("Class dictionary (словник атрибутів класу):",
archive.__class__.__dict__)
        print("Instance   dictionary   (словник   атрибутів
екземпляра):", archive.__dict__)
        print("Docstring       (документація       класу):",
archive.__class__.__doc__)

        # Отримання документа за індексом
        print("\nGetting document at index 1:")
        print(archive[1])  # Document2
```

```
# Перевірка наявності документа в архіві
print("\nChecking if 'Document2' is in archive:")
print('Document2' in archive)  # True

# Видалення документа за індексом
print("\nDeleting document at index 1:")
del archive[1]
print(archive)  # Архів тепер містить 2 документи

# Перевірка наявності документа після видалення
print("\nChecking if 'Document2' is still in archive:")
print('Document2' in archive)  # False

# Виведення журналу змін
print("\nChange log:")
print(archive.change_log)
```

Приклад виведення:

```
Class name (назва класу): Archive
Module name (модуль): __main__
Class dictionary (словник атрибутів класу): {'__module__': '__main__', '__doc__': 'Клас для зберігання документів в архіві', 'num_instances': 1, '__init__': <function Archive.__init__ at 0x7f8d0b60b5e0>, '__getitem__': <function Archive.__getitem__ at 0x7f8d0b60b700>, '__delitem__': <function Archive.__delitem__ at 0x7f8d0b60b790>, '__contains__': <function Archive.__contains__ at 0x7f8d0b60b820>, '__str__': <function Archive.__str__ at 0x7f8d0b60b8b0>}
Instance dictionary (словник атрибутів екземпляра): {'documents': ['Document1', 'Document2', 'Document3'], 'change_log': []}
Docstring (документація класу): Клас для зберігання документів в архіві

Getting document at index 1:
Document2

Checking if 'Document2' is in archive:
```

```
True

Deleting document at index 1:
Archive with 2 documents. Instance number: 1

Checking if 'Document2' is still in archive:
False

Change log:
['Deleted document at index 1: Document2']
```

Завдання на лабораторну роботу:

1. Продемонструвати роботу з використанням спеціальних полів та методів. Крім того, що виконати завдання по списку, то ще і для кожного завдання вивести спеціальні поля для класу і для екземпляру. Продемонструвати детальну роботу з класом.

1. Створіть клас «Фігура». Реалізуйте методи для перевірки, чи міститься точка всередині фігури (__contains__), перетворення фігури на рядок для відображення її властивостей (__str__) та отримання її абсолютної величини як площі або об'єму (__abs__).
2. Створити клас «Рядок», який має методи: (__contains__), який повертає True, якщо певна підстрока міститься у рядку, (__getitem__) дозволяє отримати символ за індексом, (__len__) повертає довжину рядка.
3. Реалізувати клас «Дата», який має такі методи: (__str__) повертає рядок у форматі «день:місяць:рік», (__getitem__): дозволяє отримати день, місяць або рік за індексом (0 – день, 1 – місяць, 2 – рік), (__call__) дозволяє викликати об'єкт дати для перевірки, чи є ця дата високосним роком (повертає True для високосного року, інакше – False).
4. Створити клас «Ряд Фібоначчі», який містить два атрибути – перше та друге число послідовності Фібоначчі. Реалізуйте такі методи: (__int__) конвертує об'єкт у ціле число, обчислюючи суму перших *n* чисел послідовності Фібоначчі, (__getitem__) дозволяє отримати *n*-е число послідовності за індексом, (__len__) повертає кількість елементів у послідовності до заданого числа *n*.
5. Реалізувати клас «Час» та методи: (__str__) повертає рядок у форматі «години:хвилини:секунди», (__getitem__) дозволяє отримати значення години, хвилини або секунди за індексом (0 — години, 1 — хвилини, 2

— секунди), (__int__) конвертує об'єкт у ціле число, що представляє загальну кількість секунд.

6. Створіть клас «Матрична операція». Реалізуйте методи для звертання до елементів матриці за індексами (__getitem__), зміни елементів матриці (__setitem__) та перевірки чи є булеве значення у матриці (__bool__).
7. Створіть клас «Часова точка». Реалізуйте методи для перетворення часу в рядок (__str__), отримання абсолютного значення часу (наприклад, секунд від початку доби) (__abs__), перевірки, чи входить певний час в інтервал (__contains__).
8. Створіть клас «Архів документів». Реалізуйте методи для звертання до документів за індексом (__getitem__), видалення документів за індексом (__delitem__) та перевірки наявності документу в архіві (__contains__).
9. Створіть клас «Множина символів». Реалізуйте методи для видалення символу за індексом (__delitem__), перевірки наявності символу в множині (__contains__) та отримання її довжини (__len__).
10. Створіть клас «Паліндромний рядок». Реалізуйте метод для перевірки, чи є рядок паліндромом (__bool__), обчислення довжини рядка (__len__) та інверсії його символів (__reversed__).
11. Створіть клас «Числовий стек». Реалізуйте метод для перетворення об'єкта в дійсне число (__float__), обчислення абсолютного значення елементів стека (__abs__) та перевірки, чи міститься певний елемент в стеці (__contains__).
12. Створіть клас, який представляє 3D-вектор. Реалізуйте метод для перетворення об'єкта в ціле число (__int__), обчислення абсолютного значення вектора (__abs__) та отримання рядкового представлення вектора (__str__).
13. Створіть клас для двозначних чисел. Реалізуйте метод для перетворення об'єкта в дійсне число (__float__), для отримання комплексної форми числа з двох цифр числа (__complex__), метод для інверсії порядку цифр у числах (__reversed__).
14. Створіть клас «Студент», який має такі методи: (__str__) повертає рядок у форматі «Ім'я: Прізвище», (__reversed__) повертає значення студента у зворотному порядку (прізвище, ім'я), (__getitem__) дозволяє отримати літеру за індексом у записі «Ім'я: Прізвище».

2. Виконати перевантаження операторів.

1. Розробити клас Time, який представляє час. Перевантажити операції = =, !=, >, < для порівняння часу.
2. Створити клас Calendar. Перевантажити операцію = для присвоювання дат, +=, -= для додавання та віднімання заданої кількості днів.
3. Розробити клас Matrix, який представляє матриці. Перевантажити операції +, -, * для додавання, віднімання та множення матриць.
4. Створити клас CorrectFraction, що є відношенням двох цілих чисел (дріб, вводяться 2 числа). Перевантажити операції = =, !=, >=, <=, для порівняння дробів.
5. Розробити клас PolynomPow3 (степінь 3). Коефіцієнти представити у вигляді поля-масиву класу. Перевантажити операції + додавання двох поліномів, - віднімання двох поліномів, * множення полінома на число.
6. Розробити клас Book, який представляє книги. Перевантажити операції = =, !=, >, < для порівняння книг за назвою, автором, роком видання, кількістю сторінок.
7. Створити клас Calk для роботи зі словами і виконати перевантаження операторів +=, -=, *=, /=. Тобто продумати, що будете розуміти під цими операторами для слів.
8. Розробити клас Figure, який представляє геометричні фігури. Перевантажити операції = =, !=, >, < для порівняння фігур за площею, периметром та кількістю сторін.
9. Створити клас Vector для роботи з векторами і виконати перевантаження операторів +=, - =, *=, <.
10. Розробити клас Time, який представляє час. Перевантажити операції +, -, * для дій з часом.
11. Створити клас Calendar. Перевантажити відповідно, операції = =, !=, >=, <=, для порівняння дат.
12. Створіть клас для банківського рахунку. Перевантажити оператори += (для поповнення рахунку), -= (для зняття грошей), == (для перевірки, чи два рахунки мають однаковий баланс) та > (для порівняння балансів).
13. Створіть клас, що представляє колір у форматі RGB. Перевантажити оператори + (для змішування кольорів), - (для обчислення різниці між

кольорами), * (для затемнення кольору) та == (для порівняння кольорів).

14. Створіть клас, що представляє раціональне число (дріб). Перевантажити оператори: + (для додавання двох раціональних чисел), - (для віднімання одного раціонального числа від іншого), * (для множення раціональних чисел), == (для перевірки рівності двох раціональних чисел).

Лабораторна робота № 7. Абстрактні класи. Поняття про метапрограмування. Декоратори

Мета лабораторної роботи:

- ✓ Сформувати у здобувачів освіти практичні навички роботи з абстрактними класами.
- ✓ Сформувати практичні навички роботи з метапрограмуванням у Python, зокрема через використання та програмування декораторів.
- ✓ Навчитися створювати декоратори для функцій, реалізовувати вкладені декоратори, а також використовувати та реалізовувати декоратори для класів.
- ✓ Навчитися динамічно змінювати поведінку програм, що дозволить більш гнучко керувати логікою програмного коду та розширювати функціональні можливості функцій і класів.

Завдання спрямовані на формування практичного розуміння використання абстрактних класів, принципів метапрограмування, застосування та програмування декораторів.

Розглянемо **приклад** реалізації завдання 2. Тобто реалізуємо декоратор, який обчислює кількість подвоєних літер у тексті, а також функцію, яку він обгортає. У цьому прикладі функція підраховує загальну кількість літер у тексті, а декоратор додає функціональність для підрахунку саме подвоєних літер.

```
# Декоратор, який рахує кількість подвоєних літер у тексті
def count_double_letters(func):
    def decorator(text):
        # Викликаємо оригінальну функцію
        result = func(text)

        # Рахуємо кількість подвоєних літер у тексті
        double_letter_count = 0
        for i in range(len(text) - 1):
            if text[i] == text[i + 1]:
                double_letter_count += 1

        print(f"Кількість подвоєних літер у тексті: {double_letter_count}")
        return result
```

```
        return decorator

    # Функція, яка підраховує загальну кількість літер у тексті
    @count_double_letters
    def count_letters(text):
        letter_count = 0 # Лічильник для кількості літер
         # Проходимо по кожному символу у тексті
          for char in text:
             # Перевіряємо, чи є символ літерою
             if char.isalpha():
                 letter_count += 1 # Якщо так, додаємо до
лічильника
        print(f"Загальна кількість літер у тексті:
{letter_count}")
        return letter_count

    # Використання
    text = "Гарний день для подвоєних літер, ггг!"
    count_letters(text)
```

Результат:

```
Загальна кількість літер у тексті: 29
Кількість подвоєних літер у тексті: 3
```

Декоратор `count_double_letters` обгортає функцію `count_letters` і перед її виконанням аналізує текст на наявність подвоєних літер. Рахує кількість літер, які йдуть підряд і однакові. Після цього викликає основну функцію, яка працює з текстом (у нашому випадку рахує загальну кількість літер).

Завдання на лабораторну роботу:

1. Продемонструйте створення абстрактного класу (мінімум на 3 методи). А далі реалізуйте похідні 2 класи. Продемонструйте просту роботу з класами та реалізуйте виникнення помилок, про які говориться в лекції (а потім приберіть ті помилки). Класи – власна ваша розробка, але щоб відповідали правилам створення абстрактних класів та звичайних класів.

2. Продемонструвати використання декоратора для функції. Функцію створюєте відповідно до завдання самі, але щоб підходила для обгортки заданим декоратором.

1. Напишіть декоратор, який перевіряє аргументи, передані до функції, і гарантує, що вони відповідають певним умовам. Перевірити чи всі аргументи – це числа. Якщо ні, то виведіть повідомлення про це і попросіть ввести правильно.
2. Напишіть декоратор для автоматичного переведення вихідних даних функції у двійкову та шістнадцяткову систему числення, а потім переводить ці дані у рядок та виводить.
3. Напишіть декоратор, який бере вхідні аргументи для виконання функції з файлу та фіксує час, коли це було зроблено.
4. Напишіть декоратор, який обмежує доступ до функції за допомогою пароля.
5. Напишіть декоратор, який перевіряє аргументи, передані до функції, і гарантує, що вони відповідають певним умовам. Перевірити чи всі аргументи – це додатні числа, якщо ні, то зробити з них додатні.
6. Напишіть декоратор, який виводить дату та рахує час виконання функції.
7. Напишіть декоратор для перевірки випадку, який може призвести до помилки (ділення на 0) у функції: якщо ділення можливе то виводиться результат обчислення, якщо помилка – то виводиться відповідне повідомлення.
8. Напишіть декоратор, який записує всі вхідні аргументи та результати виконання функції у файл та виводить про це відповідне повідомлення.
9. Напишіть декоратор для перевірки випадку, чи входять вхідні аргументи для функції у потрібний проміжок: якщо все добре, то виводиться результат, якщо ні – то виводиться відповідне повідомлення, де пропонується ввести інші вхідні аргументи.
10. Напишіть декоратор, який змінює результат функції, змінюючи всі букви у відповідному рядку на великі.
11. Створіть декоратор, який збільшує всі аргументи функції на певне значення перед виконанням функції і виводить про це повідомлення.
12. Розробіть декоратор, який вимірює кількість використаної оперативної пам'яті під час виконання функції.
13. Створіть декоратор, який перевіряє, чи введене користувачем число є парним, і в разі необхідності змінює його на парне. Воно ж і подається на вхід у функцію потім.

14. Створіть декоратор, який обчислює кількість від’ємних, додатних та нульових чисел в результаті виконання функції.

3. Продемонструйте використання декоратора для класу. Увага! Клас для декоратора створіть самостійно. Дослідити необхідні інші модулі (вбудовані бібліотеки чи функції) для виконання завдання, якщо такі потрібні.

1. Створіть декоратор, який перетворює всі результати методів у двійкову та шістнадцяткову систему числення.
2. Створіть декоратор для класу, який ловить винятки, що виникають під час виконання методів, і виводить їх повідомлення.
3. Створіть декоратор для класу, який змінює поведінку декількох методів, додаючи певний функціонал до них.
4. Розробіть декоратор для класу, який фіксує кількість викликів кожного методу класу та виводить результат після закінчення роботи програми.
5. Створіть декоратор для класу, який змінює результати викликів методів, змінюючи їх значення.
6. Створіть декоратор для класу, який записує у файл інформацію про створення та знищення об'єктів класу.
7. Розробіть декоратор для класу, який вимірює кількість використаної оперативної пам'яті під час виконання методів.
8. Створіть декоратор для класу, який змінює результати методів, змінюючи їх формат виведення.
9. Створіть декоратор, який знаходить максимальне та мінімальне значення, повернуті методами класу.
10. Створіть декоратор, який автоматично зберігає результати виконання методів класу у файл.
11. Розробіть декоратор, який друкує назву та параметри методу класу перед кожним його викликом.
12. Створіть декоратор для класу, який зберігає історію викликів методів і дозволяє отримати статистику про їх виконання.
13. Розробіть декоратор, який фіксує час та дату, коли виконувався кожен метод класу.

14. Створіть декоратор для класу, який змінює результат виклику деякого методу на його обернене значення.

4. Додаткове завдання. Продемонструйте використання вкладених декораторів для функції. Оберіть з першого завдання 2 чи більше декоратори, але таких, які б підійшли для обгортки однієї функції АБО придумайте та реалізуйте власні 2 декоратори (але не просто вивести щось на екран). Напишіть під них функцію (функція має бути не така як у завданні 1 та декоратори інші). Показати роботу з декораторами в одному напрямі та у зворотному.

СПИСОК СИКОРИСТАНИХ ДЖЕРЕЛ

1. Креневич А.П. Python у прикладах і задачах. Частина 2. Об’єктно-орієнтоване програмування. Навчальний посібник. К.: ВПЦ "Київський Університет", 2020. 152 с.
2. Замуруєва О. В., Кримусь А. С., Ольхова Н. В. Об’єктно-орієнтоване програмування в Python : курс лекцій. Луцьк : Вежа-Друк, 2018. 64 с.
3. Онищенко В.В., Довженко Т.П. Спеціалізовані мови програмування. 1-е вид. К.: Державний університет телекомунікацій, 2019. 146с.
4. The Python Standard Library. *Python documentation*. URL: https://docs.python.org/3/library/index.html.
5. Підручник з Python/W3Schools.com. *W3Schools Online Web Tutorials*. URL: https://www.w3schools.com/python/default.asp.
6. Python Tutorials – Real Python. *Python Tutorials – Real Python*. URL: https://realpython.com/.
7. Олександр Мізюк. Практикум з програмування мовою Python. URL: https://pythonexercises.rozh2sch.org.ua/.

УДК
Г 54

Електронне мережне навчальне видання

Л. Я. Глинчук

ПРОГРАМУВАННЯ: ЛАБОРАТОРНИЙ ПРАКТИКУМ ЗМІСТОВОГО МОДУЛЮ «ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ МОВОЮ PYTHON»

для студентів спеціальностей 122 Комп'ютерні науки та 125 Кібербезпека та захист інформації першого (бакалаврського) рівня

Друкується в авторській редакції