Звіт

З лабораторної роботи №4

Студента групи МІТ-31

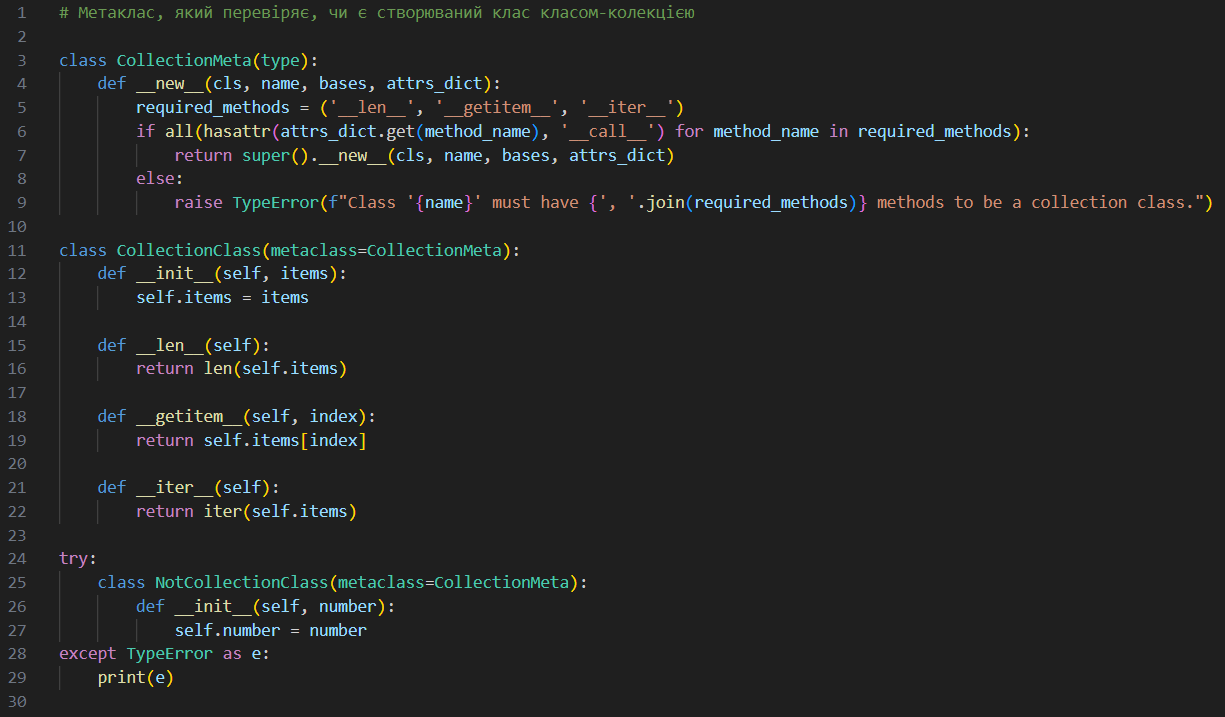
Добровольського Арсенія Михайловича

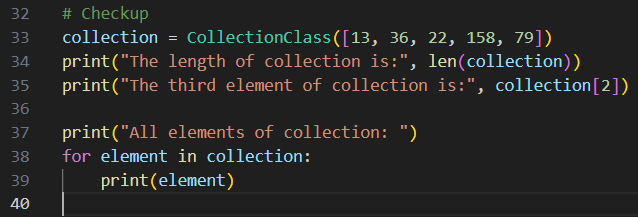
Варіант №9

**Тема роботи:** Метакласи та метапрограмування

**Мета роботи:** розглянути концепцію метапрограмування у Python. Ознайомитися з поняттям метакласу у Python, навчитися створювати та використовувати класи з наслідуванням метакласів.

Завдання 1

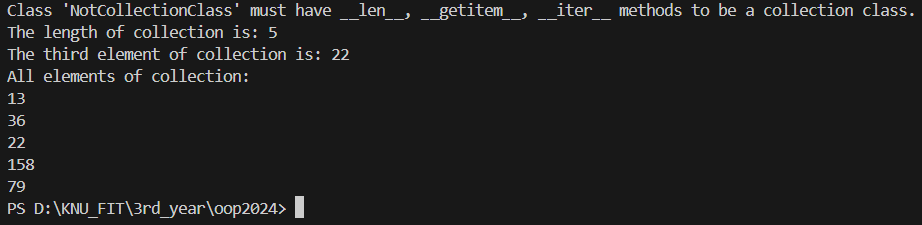




В даному коді реалізовано метаклас **CollectionMeta**, який перевіряє, чи має створюваний клас необхідні методи для того, щоб вважатися класом-колекцією (а саме методи **\_\_len\_\_**, **\_\_getitem\_\_** та **\_\_iter\_\_**). У методі **\_\_new\_\_** перевіряється наявність кожного з цих методів в словнику атрибутів створюваного класу. Якщо всі три методи присутні, метаклас дозволяє створити клас. В іншому випадку генерується помилка **TypeError**, яка сповіщає про нестачу необхідних методів у класі.

Далі створюються класи **CollectionClass** і **NotCollectionClass**. Метакласом обидвох цих класів виступає CollectionMeta. Проте CollectionClass реалізує всі необхідні методи, характерні класу-колекції, водночас NotCollectionClass, навпаки, не має всіх обов'язкових методів, що призводить до виникнення помилки **TypeError** та її обробки за допомогою конструкції try-except.

Результат виконання програми має вигляд:



Завдання 2

Метапрограмування має багато застосувань, зокрема:

* Створення компілятора
* Створення додатків
* Аналіз коду
* Загальне проєктування компонентів
* Обслуговування/еволюція/конфігурація програмного забезпечення
* Завчасна оптимізація
* Шаблони проєктування
* Розв’язування звичайних диференціальних рівнянь (ODE)
* Такі IDE, як Visual Studio Code або Pycharm, позначають потенційні помилки без необхідності запускати код. Обидві платформи пропонують рефакторинг базового рівня, перетворюючи блок коду в більш читабельний синтаксис.
* Так зване «відображення», при якому програмне забезпечення є власною метапрограмою. Іншими словами, код сприймає себе як дані та переписує себе відповідно до своїх потреб.

Прикладами метапрограм є компілятори, транспілятори, асемблери та інтерпретатори. Вони приймають програми в одній формі та перетворюють їх на машинний код, байт-код або навіть вихідний код іншою мовою. Lex і Yacc є метапрограмами. Іншими прикладами є програми, які читають метадані бази даних і виводять код EJB або XML. Динамічні або інтерпретовані мови зазвичай мають функцію eval(), яка дозволяє виконувати код, наданий у вигляді рядків. Це теж метапрограмування.

Показовим прикладом також є модульна бібліотека графіків Hone, написана мовою Julia. Точки даних, осі та легенди компонуються як рядки та виконуються для створення графіків.

Таким чином, можна мати метапрограми практично для будь-чого, починаючи з адаптації програмного забезпечення залежно від робочого навантаження до зміни інфраструктури за потреби.

**Висновок:** в ході виконання даної лабораторної роботи я детально ознайомився з поняттями метакласу та метапрограмування у Python, а також реалізував власний метаклас, який був використаний для перевірки того, чи є створюваний клас класом-колекцією.