Лабораторна робота №2

Звіт

З дисципліни “Об'єктно-орієнтоване програмування ”

на тему: “Наслідування у пайтон. Створення об’єктів з агрегацією та композицією ”.

Студента 3 курсу: Групи МІТ-31 Демиденко Андрій

**Київ - 2024р.**

**Комплексний звіт**

**Мета:** ознайомитися з поняттям наслідування в об'єктно-орієнтованому

програмуванні, навчитися створювати класи та об'єкти з наслідуванням у

Пайтон. Навчитися використовувати такі зв'язки між об'єктами в ООП, як

агрегація та композиція

**Завдання для самостійного опрацювання варіант**

**Варіант №7**

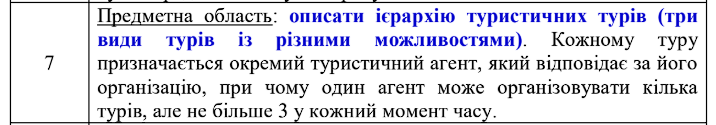


Рис. 2.1 – завдання для варіанту

**Завдання №1**

Для початку визначимо сутності предметної області та класи, тим самим склавши план зв’язків між класами (рис. 2.2). Клас **Tour** буде батьківським та матиме в собі опис найпростішого узагальненого туру для відпочинку. Класи **ExtremeTour, RelaxTour, CulturalTour** будуть нащадками та матимуть особливості, якими і доповнюватимуть основні властивості батьківського класу. Також створимо клас **TourAgent** він і буде прикладом застосування агрегації, та працюватиме з об’єктами класів Tour та його дочірніх.

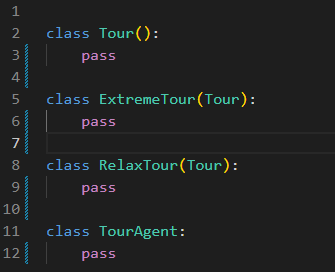


Рис. 2.2 – Предметна область та класи

Реалізація **TourAgent** буде у вигляді агрегації через відсутність жорсткої залежності існування туру та агента, за відсутності агентів, тури можуть існувати незалежно, так як і можуть снувати агенти без турів.

Після визначення умовного плану можна приступити до наповнення класів. Для класу **Tour** будуть додані наступні властивості:

Метод **\_\_init\_\_** — ініціалізує основні властивості та додає об’єкт в список (рис. 2.3).

Метод **\_\_str\_\_** — відповідає за вивід інформації про тур (рис. 2.4).

Метод **expensive\_check** — призначений для перевірки відношення ціни до кількості осіб та часу туру (рис. 2.5).

Метод **get\_all\_tours** — метод класу призначений для виводу всіх створених об’єктів (рис. 2.6).

Метод **ensure\_list** — статичний метод для обгортання значення в список, він буде використовуватись класами нащадками (рис. 2.7).

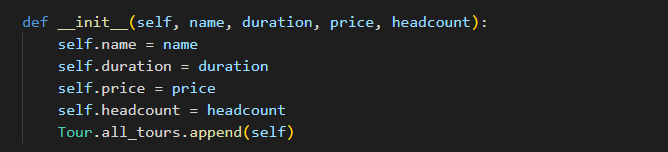


Рис. 2.3 – Метод **\_\_init\_\_**

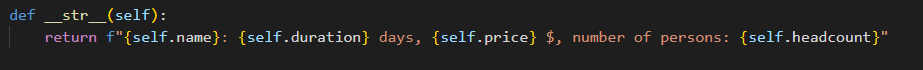


Рис. 2.4 – Метод **\_\_str\_\_**

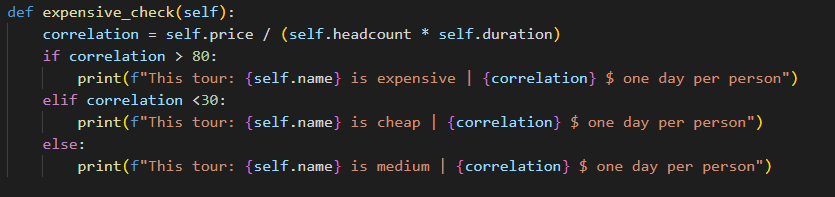


Рис. 2.5 – Метод **expensive\_check**

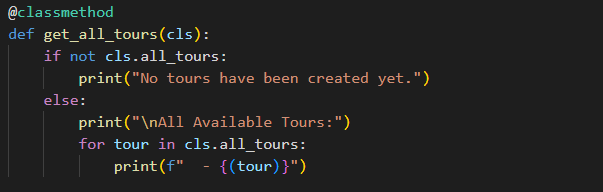


Рис. 2.6 – Метод **get\_all\_tours**

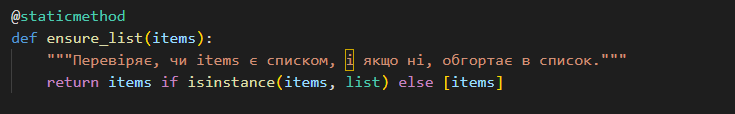


Рис. 2.7 – Метод **ensure\_list**

Перейдімо до класів нащадків, оскільки вони є досить одноманітними пропоную розглянути їх більш абстрактно і не зупинятись на роботі окремих методів, а вирізнити найбільш явні прояви наслідування (рис. 2.8 - рис. 2.10).

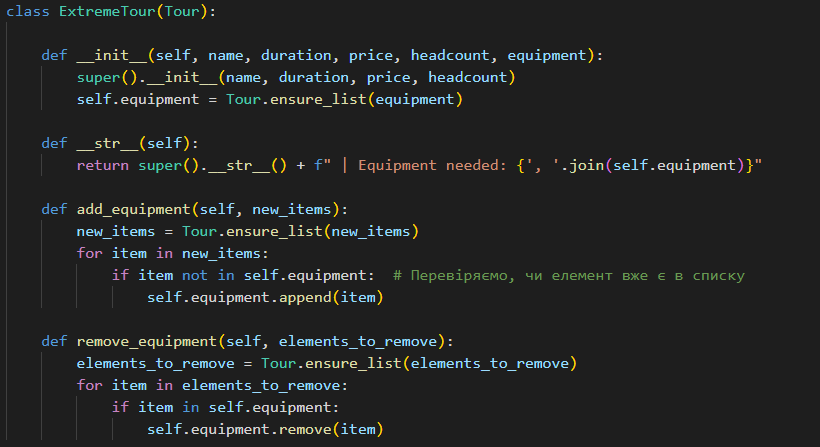


Рис. 2.8 – Клас **ExtremeTour**

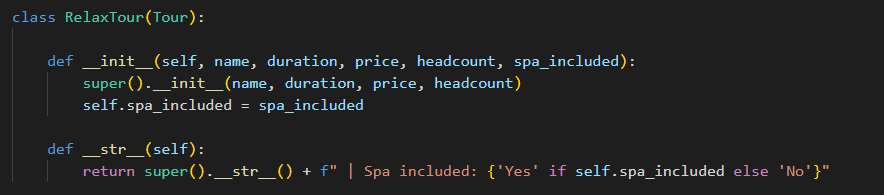


Рис. 2.9 – Клас **RelaxTour**

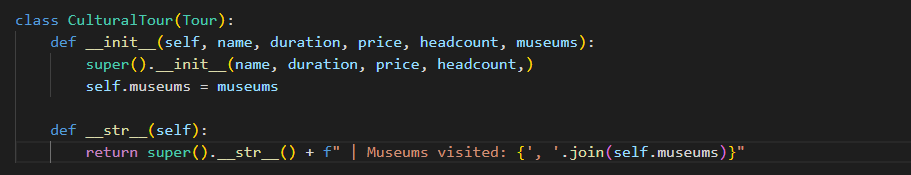


Рис. 2.10 – Клас **CulturalTour**

З основних особливостей наслідування це конструкція ініціалізації, яка наслідуєтся з супер класу і доповнюється елементами—особливостями класу нащадка (рис. 2.11). Також перевизначаєтся магічний метод **\_\_str\_\_** з доповненим виводом інформації (рис. 2.12). Також в класі **ExtremeTour** додані особливі методи класу, яких немає в батьківському класі (рис. 2.13).

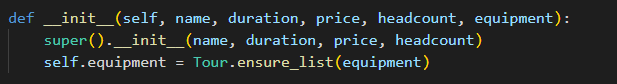
****

Рис. 2.11 – Метод \_\_**init**\_\_ класу **RelaxTour**

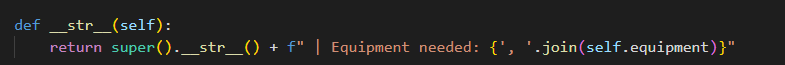


Рис. 2.12 – Метод **\_\_str\_\_** класу **RelaxTour**

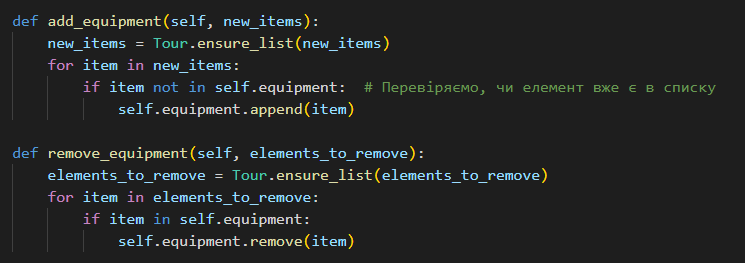


Рис. 2.13 – Додаткові метод класу **RelaxTour**

Зараз перейдімо до класу **TourAgent,**  він і є яскравим прикладом застосування агрегаці, яка визначається через принцип роботи з об’єктами. Розгляньмо методи детальніше:

Метод **\_\_init\_\_** — ініціалізує основні властивості та додає об’єкт в список (рис. 2.14).

Метод **\_\_str\_\_** — відповідає за вивід інформації про агента (рис. 2.15).

Метод **add\_tour** –- відповідає за додаванням туру агенту у разі, коли кількість турів агента менше ніж 3 (рис. 2.16).

Метод **remove\_tour** –- відповідає за видалення туру агента у разі, коли кількість турів агента більша ніж 0 (рис. 2.17).

Метод **get\_all\_tours** — метод класу призначений для виводу всіх турів кожного агента (рис. 2.18).

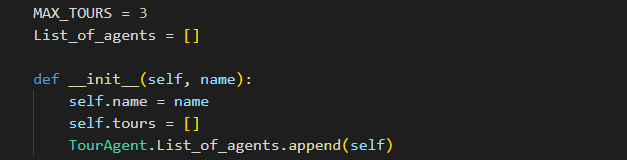


Рис. 2.14 – Метод **\_\_init\_\_**

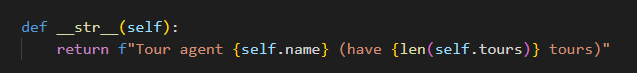


Рис. 2.15 – Метод **\_\_str\_\_**

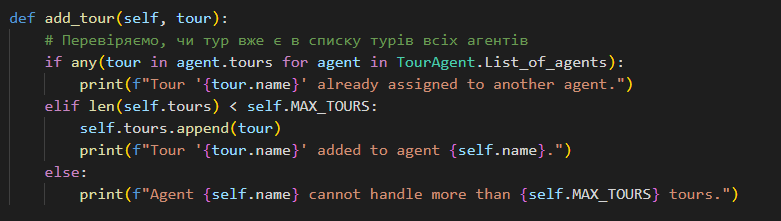


Рис. 2.16 – Метод **add\_tour**

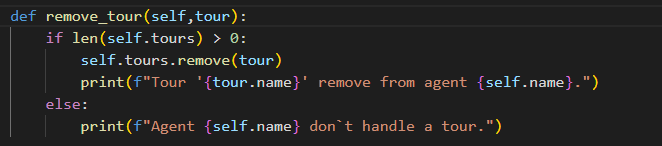


Рис. 2.17 – Метод **remove\_tour**

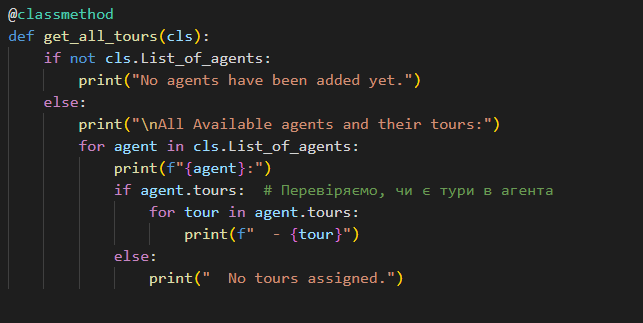


Рис. 2.18 – Метод **get\_all\_tours**

Тепер можна в основному тілі програми викликати методи класів, та перевірити працездатність на виході (рис. 2.19).

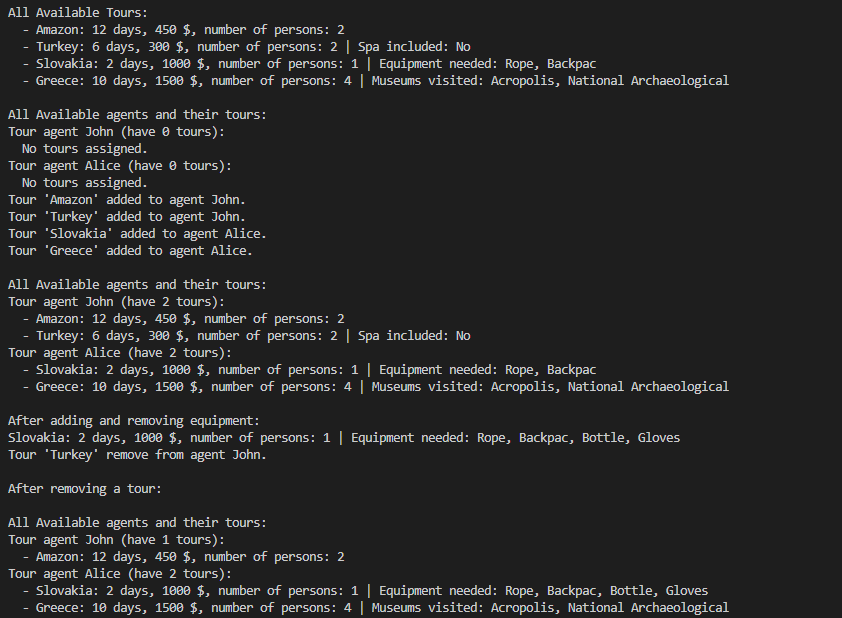


Рис. 2.19 – вивід у консоль

**Висновок:** Виконано завдання з використанням принципів об'єктно-орієнтованого програмування на прикладі класів турів і агентів. Python виявився зручним для реалізації ООП завдяки простоті, гнучкості та легкості роботи з об'єктами. Принципи наслідування, агрегації та композиції були ефективно застосовані. Python дозволяє швидко створювати та керувати об'єктами, що робить його ідеальним вибором для таких завдань.

[Git Reposytory](https://github.com/AndrewDem0/OOP-Learning)