Лабораторна Робота 9 – Класи: спадкування.

Мета роботи – вивчити та засвоїти навички роботи з спадкуванням класів у Python. Ознайомитися з поняттями доповнення методу, перевизначення методу, ієрархія класів та набути практичні навички їх застосування.

1.1 Спадкування

Спадкування - важлива складова об'єктно-орієнтованого програмування. Так чи інакше ми вже стикалися з ним, адже об'єкти успадковують атрибути своїх класів. Однак зазвичай під спадкуванням в ООП розуміється наявність класів і підкласів. Також їх називають супер-або надкласу і класами, а також батьківськими і дочірніми класами.

Суть спадкування схожа з успадкуванням об'єктами від класів. Дочірні класи успадковують атрибути батьківських, а також можуть перевизначати атрибути і додавати свої.

1.2 Просте успадкування методів батьківського класу

Як приклад розглянемо розробку класу столів і його двох підкласів - кухонних і письмових столів. Всі столи, незалежно від свого типу, мають довжину, ширину і висоту. Нехай для письмових столів важлива площа поверхні, а для кухонних - кількість посадкових місць. Загальна винесемо в клас, приватна - в підкласи.

Зв'язок між батьківським і дочірнім класом встановлюється через дочірній: батьківські класи перераховуються в дужках після його імені.

```
class Table:
    def __init__(self, l, w, h):
        self.lenght = l
        self.width = w
        self.height = h
class KitchenTable(Table):
    def setPlaces(self, p):
        self.places = p
class DeskTable(Table):
    def square(self):
        return self.width * self.length
t1 = KitchenTable()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-4-75572ae70330> in <module>
     10     def square(self):
     11         return self.width * self.length
---> 12 t1 = KitchenTable()

TypeError: __init__() missing 3 required positional arguments: 'l', 'w', and 'h'
```

В даному випадку класи KitchenTable і DeskTable не мають своїх власних конструкторів, тому успадковують його від батьківського класу. При створенні примірників цих столів, передавати аргументи для __init __() обов'язково, інакше виникне помилка:

[27]:
```
t1 = KitchenTable(2, 2, 0.7)
t2 = DeskTable(1.5, 0.8, 0.75)
t3 = KitchenTable(1, 1.2, 0.8)
print(t2.square())
```

```
1.2000000000000002
```

Безсумнівно можна створювати столи і від батьківського класу Table. Однак він не буде, згідно деяким родинними зв'язками, мати доступ до методів setPlaces () і square (). Точно також як об'єкт класу KitchenTable не має доступу до одноосібних атрибутам сестринського класу DeskTable.

У цьому сенсі термінологія "батьківський і дочірній клас" не зовсім вірна. Спадкування в ООП - це скоріше аналог систематизації та класифікації на зразок тієї, що є в живій природі. Всі ссавці мають четирехкамерное серце, але тільки носороги - ріг.

1.3 Повне перевизначення методу надкласу

Що якщо в підкласі нам не підходить код методу його надкласу. Припустимо, ми вводимо ще один клас столів, який є дочірнім по відношенню до DeskTable. Нехай це будуть комп'ютерні столи, при обчисленні робочої поверхні яких треба забирати задану величину. Має сенс внести в цей новий підклас його власний метод square ():

```
[ ]: class ComputerTable(DeskTable):
         def square(self, e):
             return self.width * self.length - e
```

При створенні об'єкта типу ComputerTable як і раніше потрібно вказувати параметри, так як інтерпретатор в пошуках конструктора піде по дереву успадкування спочатку в батька, а потім в прабатька і знайде там метод __init __ (). Однак коли буде викликатися метод square (), то оскільки він буде виявлений в самому ComputerTable, то метод square () з DeskTable залишиться невидимим, тобто для об'єктів класу ComputerTable він виявиться перевизначеним.

1.4 Доповнення (розширення) методу

Якщо подивитися на обчислення площі, то частина коду надкласу дублюється в підкласі. Цього можна уникнути, якщо викликати батьківський метод, а потім доповнити його:

```
class ComputerTable(DeskTable):
    def square(self, e):
        return DeskTable.square(self) - e
```

Тут викликається метод іншого класу, а потім доповнюється своїми висловлюваннями. В даному випадку відніманням.

Розглянемо ще один приклад. Припустимо, в класі KitchenTable нам не потрібен метод, поле places має встановлюватися при створенні об'єкта в конструкторі. У класі можна створити конструктор з чистого аркуша, ніж перевизначити батьківський:

[29]:
```
class KitchenTable(Table):
    def __init__(self, l, w, h, p):
        self.length = l
        self.width = w
        self.height = h
        self.places = p
```

Однак це не кращий спосіб, якщо дублюється майже весь конструктор надкласу. Простіше викликати батьківський конструктор, після чого доповнити своїм кодом:

[30]:
```
class KitchenTable(Table):
    def __init__(self, l, w, h, p):
        Table.__init__(self, l, w, h)
        self.places = p
```

Тепер при створенні об'єкта типу KitchenTable треба вказувати в конструкторі чотири аргументи. Три з них будуть передані вище по сходах спадкування, а четвертий оприбутковано на місці.

2. Варіанти завдання

У таблиці 1 задано сімейство об'єктів, що мають деяку схожість (загальні ознаки). Необхідно виділити найбільш загальні риси об'єктів, на основі яких скласти базовий клас. На основі базового класу розробити ієрархію класів-нащадків, кінцевими гілками в якій будуть задані об'єкти. Ієрархія класів повинна бути мінімум трирівнева, тобто між базовим класом і класами, що описують задані об'єкти, повинен бути хоча б один проміжний рівень ієрархії. Продемонструвати роботу з об'єктами, заданими за таблицею 1. У звіті привести діаграму класів.

Таблиця 1 – Варіанти завдань

№	*Завдання*
1.	собака, ведмідь, корова, акула, орел, кит, шпак, людина, тигр
2.	помідор, малина, полуниця, банан, картопля, виноград, яблуко, ківі
3.	абітурієнт, студент, першокурсник, бакалавр, спеціаліст, магістр
4.	кухня, спальня, аудиторія, кабінет, ванна кімната, спортивний зал, гараж
5.	море, річка, океан, озеро, водосховище, ставок, затока, канал, калюжа
6.	пістолет, автомат, танк, граната, динаміт, ніж, револьвер
7.	меч, спис, шпага, рапіра, ніж, кинджал, сюрікени
8.	чоботи, кросівки, туфлі, босоніжки, валянки, черевики, сандалі, тапки
9.	пиріг, шашлик, розсольник, млинці, гуляш, рагу, плов, борщ, солянка
10.	чай, компот, вино, квас, коньяк, пепсі-кола, пиво, сік, кава
11.	автомобіль, тролейбус, автобус, маршрутка, трамвай, метро, електричка, мотоцикл, велосипед
12.	процесор, монітор, клавіатура, миша, жорсткий диск, принтер, сканер, оперативна пам'ять, веб-камера
13.	рука, голова, серце, око, вухо, легкі, ребро, печінка, мозок
14.	рядовий, генерал, майор, лейтенант, капітан, сержант, єфрейтор, полковник, прапорщик
15.	батько, мати, син, дочка, дядько, тітка, племінник, брат, сестра
16.	флейта, гітара, саксофон, арфа, віолончель, скрипка

3. Приклад

Варіант -16.

Файл програми

[1]:
```
class mus_tool:                             #клас музичні інстументи
    def __init__(self):
        self.producer = " "                 # виробник інструменту
        self.cost = " "                     # вартість інструменту
    def add(self):                          # метод додавання
        self.producer = input("введіть назву компанії виробника :")
        self.cost = int(input("введіть вартість :"))

class stringed_tool (mus_tool):             # клас струнні, наслідування класу музичні інструмекнти
    def __init__(self):
        self.stringed_quantity = 0          # кількість струн:
    def add(self):
        self.stringed_quantity = int(input("введіть кількість струн:"))
        mus_tool.add(self)

class plucked (stringed_tool):              # клас щіпкові, наслідування класу струнні
    def __init__(self):
        self.harmony_quantity = 0           # кількість ладів:
    def add(self):
        self.harmony_quantity = int(input("введіть кількість ладів:")) # кількість ладів:
        stringed_tool.add(self)

class guitar (plucked):                     # клас гітара, наслідування класу щіпкові
    def __init__(self):
        self.model = " "                    # кількість модель гітари.
    def add(self):
        self.model = input("введіть назву моделі гітари:")
        plucked.add(self)
    def info(self):
        print(self.model, self.harmony_quantity, self.stringed_quantity, self.cost, self.producer, sep = "\n")

a = guitar()
a.add()
a.info()
```

```
введіть назву моделі гітари: Tx-56_kf
введіть кількість ладів: 12
введіть кількість струн: 6
введіть назву компанії виробника : Guitar boy
введіть вартість : 150
Tx-56_kf
12
6
150
Guitar boy
```

4. Контрольні запитання

1. У яких випадках необхідно вибирати успадкування?
2. У чому можуть виражатися основні помилки при спадкуванні?
3. Що таке вкладені класи? Наведіть приклад використання.
4. Опишіть механізми розширення методу, та принципи їх використання.
5. Що таке перевизначення методу надкласу? Наведіть приклад використання.