Реферат

Тема: Інкапсуляція

Інкапсуляція — один з трьох основних механізмів [об'єктно-орієнтованого програмування](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F" \o "Об'єктно-орієнтоване програмування). Йдеться про те, що об’єкт вміщує не тільки дані, але і правила їх обробки, оформлені в вигляді виконуваних фрагментів ([методів](https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)" \o "Метод (програмування))). А також про те, що доступ до стану об'єкта напряму заборонено, і ззовні з ним можна взаємодіяти виключно через заданий [інтерфейс](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_(%D0%BE%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)" \o "Інтерфейс (об'єктно-орієнтоване програмування)) (відкриті поля та методи), що дозволяє знизити [зв'язність](https://uk.wikipedia.org/wiki/%D0%97%D0%B2%27%D1%8F%D0%B7%D0%BD%D1%96%D1%81%D1%82%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)" \o "Зв'язність (програмування)). Таким чином контролюються звернення до полів класів та їхня правильна ініціалізація, усуваються можливі помилки пов'язані з неправильним викликом методу. Оскільки користувачі працюють лише через відкриті елементи класів, то розробники класу можуть як-завгодно змінювати всі закриті елементи і навіть перейменовувати та видаляти їх, не турбуючись, що десь хтось їх використовує у своїх програмах.

За умовчанням атрибути в класах загальнодоступні, а це означає, що з будь-якого місця програми ми можемо отримати атрибут об'єкта і змінити його.

Інкапсуляція — обмеження доступу до компонентів (методів і змінних). Інкапсуляція робить деякі компонент доступними тільки всередині класу.

Інкапсуляція в Python працює лише на рівні угоди між програмістами про те, які атрибути загальнодоступні, а які — внутрішні.

Одиночне підкреслення на початку імені атрибуту говорить про те, що змінна або метод не призначений для використання поза методами класу, однак атрибут доступний за цим ім'ям.

Щодо інкапсуляції безпосередньо в мові програмування Python приховати атрибути класу можна не тільки зробивши їх приватними або закритими та обмеживши доступ до них через спеціальні методи, які ще називаються властивостями.

def get\_age(self):

return self.\_\_age

Вище ми розглянули, як створювати властивості. Але Python має ще один - більш елегантний спосіб визначення властивостей. Цей спосіб передбачає використання анотацій, які передуються символом @.

Для створення властивості-гетера над властивістю ставиться інструкція @property.

Для створення властивості-сеттера над властивістю встановлюється інструкція ім'я\_властивості\_гетера.setter.

Перепишео клас Person з використанням анотацій:

class Person:

    def \_\_init\_\_(self, name):

        self.\_\_name = name      # устанавливаем имя

        self.\_\_age = 1          # устанавливаем возраст

    def set\_age(self, age):

        if age in range(1, 100):

            self.\_\_age = age

        else:

            print("Недопустимый возраст")

    def get\_age(self):

        return self.\_\_age

    def get\_name(self):

        return self.\_\_name

    def display\_info(self):

        print("Имя:", self.\_\_name, "\tВозраст:", self.\_\_age)

tom = Person("Tom")

tom.display\_info()          # Имя: Tom  Возраст: 1

tom.set\_age(-3486)          # Недопустимый возраст

tom.set\_age(25)

tom.display\_info()          # Имя: Tom  Возраст: 25

Подвійне підкреслення на початку імені атрибута дає великий захист: атрибут стає недоступним для цього імені.

1. Що таке інкапсуляція?
2. Що вирішує інкапсуляція?
3. Чи можна використати інкапсуляцію не знаючи, що це?
4. Де використовується інкапсуляція?
5. Як створити приватні поля?
6. Чи можна отримати приватні поля з класу?
7. Чи можна отримати приватні поля з об’єкту?
8. Де використати інкапсуляцію?
9. Як використовується два подвійних підкреслення?
10. Як приховати поля від запису?
11. Що таке сеттер та геттер?
12. Чому важливо використовувати сеттери та геттери?
13. Чому сеттери та геттери пов’язані з інкапсуляцією?
14. Коли варто використовувати сеттери та геттери?
15. Чи можна приховати усі властивості у класі?
16. Чи можна на створити клас без інкапсуляції?
17. Як використовується два подвійних підкреслення?
18. Чи можна отримати приватні поля з об’єкту?
19. Чи можна використати приватні властивості у класі, якому вони належать?
20. Чи можна використати публічні методи у класі, де вони створені?
21. Що таке @property?
22. Де використовується @property?
23. Навіщо використовують символ @?
24. Що можна створити за допомогою @?
25. Що можна створити за допомогою @property?
26. Чи можна створити конструктор у базовому класі?

Джерела:

<https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D1%96%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)>

https://metanit.com/python/tutorial/7.2.php