Домашня робота №53 ООП. Поліформізм. Наслідування.

Завдання 1

Визначте три класи: Printer, Lamp і Car. Кожен з них має тільки один метод does(). Він повертає значення 'print' (для Printer), 'glow' (для Lamp) або 'ride' (для Car). Далі визначте клас Robot з методом __init__, який містить по одному екземпляру кожного з цих класів як власні атрибути. Визначте метод do_it() для класу Robot, який виводить на екран усі дії, що роблять його компоненти.

P.S. Тут наслідування не робити, це завдання на поліморфізм щодо методу does() для різних типів об'єктів.

Завдання 2

Створіть ієрархію класів для представлення осіб з різними характеристиками, використовуючи клас `Person` як батьківський клас.

1. Створіть клас `Person` з атрибутами `name` і `age`. Реалізуйте метод __init__. Додайте також метод `introduce()`, який виводить ім'я та вік особи.

2. Створіть клас `Student`, який успадковує клас `Person`. Додайте атрибут ` id` при ініціалізації об'єкта (перевизначте метод __init__, в якому буде викликатись за допомогою super батьківський __init__). Додайте метод `study(subject)`, який виводить повідомлення, що студент {self.name}-{self.id} навчається предмету subject.

3. Створіть клас `Teacher`, який успадковує клас `Person`. Додайте атрибут `subject` та метод ` teach(s)`, який виводить повідомлення, що «викладач {self.name} навчає {self.subject} студента {s.name}», де s – це деякий екземпляр Student. У методі teach зробіть перевірку чи s є дійсно об`єктом з класу Student, якщо ні, то вивести відповідне повідомлення.

4. Створіть клас `Employee`, який успадковує клас `Person`. Додайте атрибути salary, specialty та метод `work()`, який виводить повідомлення про роботу співробітника та його зарплату.

5. Створіть по одному об'єкту кожного з класів `Student`, `Teacher`, `Employee` і викличте методи, що відповідають їхній ролі. Протестуйте роботу методів, що задаються у них

7. Перевизначте метод ` introduce()` у класах `Student`, `Teacher`, `Employee`, щоб вони давали всю інформацію про особу та назву класу, до якого вона відноситься. Протестуйте роботу.

Завдання 3. Creating a consultation appointment system

Suppose you are running a hospital and building a consultation appointment system. You want to be able to schedule appointments for various types of patients.

In this exercise, you will start with the previously defined `Adult` and `Baby` classes and create `OrganizedAdult` and `OrganizedBaby` classes by inheriting from a second parent class, `Calendar`:

1. Import the `datetime` module:

```
import datetime
```

2. Start with the `Baby` and `Adult` classes, as defined previously:

```
class Person:
    def __init__(self, first_name, last_name):
        self.first_name = first_name
        self.last_name = last_name
class Baby(Person):
    def speak(self):
        print('Blah blah blah')
class Adult(Person):
    def speak(self):
        print('Hello, my name is', self.first_name)
```

3. Create a `Calendar` class that you can use to help the adults and babies become more organized:

```
class Calendar:
    def book_appointment(self, date):
        print('Booking appointment for date', date)
```

4. Create `OrganizedBaby` and `OrganizedAdult` classes that inherit from multiple parent classes:

```
class OrganizedAdult(Adult, Calendar):
    pass
class OrganizedBaby(Baby, Calendar):
    pass
```

5. Create some objects from your new classes and test their methods:

```
andres = OrganizedAdult('Andres', 'Gomez')
boris = OrganizedBaby('Boris', 'Bumblebutton')
andres.speak()
boris.speak()
boris.book_appointment(datetime.date(2018,1,1))
```

6. Suppose you wanted to warn a user when they try to book an appointment with a baby. You could override the `book_appointment` method, using the `super()` method, to run the `book_appointment` method on the `Calendar` class:

```
class OrganizedBaby(Baby, Calendar):
    def book_appointment(self, date):
        print('Note that you are booking an appointment
with a baby.')
        super().book_appointment(date)
```

7. Now, test whether it works:

```
boris = OrganizedBaby('Boris', 'Bumblebutton')
boris.book_appointment(datetime.date(2018,1,1))
```

The output is as follows:

```
Note that you are booking an appointment with a baby.
Booking appointment for date 2018-01-01
```

The output is as follows:

```
Hello, my name is Andres
Blah blah blah
Booking appointment for date 2018-01-01
```