Лабораторна робота 4

Інтерфейси та протоколи у Python

Виконала:
студентка групи МІТ-31
Півторак Каріна
Варіант -6

Мета: ознайомитися із поняттям інтерфейсів ооп та його призначення. Розглянути можливості Пайтон по створенню і використанню інтерфейсів

Завдання:

3. **ЗАВДАННЯ 1 (5 балів у разі повного і правильного виконання і захисту роботи). Написати власний протокол для основного батьківського класу по своєму варіанту з лабораторної роботи №2 та продемонструвати його використання при реалізації класів. Завдання реалізувати з використанням анотацій.**

Репозиторій: https://github.com/KariSpace/python-oop-labs.git

Хід виконання лабораторної роботи

Розглянемо створення протоколу для телефону, що може звонити. Для цього створюєтья клас, який наслідує Protocol із модуля typing. Протокол описує, що телефон може телефонувати(метод call())

```
@runtime_checkable
class SupportsPhone(Protocol):
    @abstractmethod
    def call(self) -> None:
        raise NotImplementedError
```

До класу-протоколу застосовано декоратор `@runtime_checkable.` Він дозволяє під час виконання програми перевіряти приналежність екземплярів до цього класу.

Реалізовуємо два класа:

```
class Phone(SupportsPhone):

    def __init__(self, company, battery):
        self.company = company
        self.battery = battery

    def call(self):
        phone_number = "0000000000"
        print(f'\nCalling to {phone_number}...')
```

```
class MobilePhone():

    def __init__(self, company, battery, screen):
        self.screen =  screen
        self.company = company
        self.battery = battery

    def call(self):
        phone_number = "0000000000"
        print(f'Calling to {phone_number} by facetime...')
```

Вони реалізують всі методи протоколу, але Phone наслідує SupportsPhone, а MobilePhone ні.

Анотації дозволяють перевірити тип даних у Пайтон, але інтерпретатор Пайтон, їх по-сути ігнорує, і вони використовуються лише IDE для перевірки типів даних

Створюємо функцію phone_all, що використовує анотацію `Iterable[SupportsPhone]) -> None`

```
def phone_all(phones: Iterable[SupportsPhone]) -> None:
    for t in phones:
        t.call()
```

Викликаємо функцію

```
phone_all([Phone("Samsung", 2500)])  # ok
phone_all([Phone("Samsung", 2500), MobilePhone("Samsung", 900, "16inch")])  # ok
```

Висновок: Був створений протокол для телефонів, що можуть телефонувати, за допомогою класу, що наслідує protocol із модуля typing