МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,
СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Факультет «Информационные технологии»

Кафедра «Математическая кибернетика и информационные технологии»

Дисциплина: «Введение в информационные технологии»

Лабораторная работа №6

Работа с классами (часть 2)

Выполнил:

Студент группы БВТ2402

Чимитов Намжил

Москва

2024

Цель работы

Получить практический опыт работы с ООП в Python, освоить использование инкапсуляции и наследования.

Задачи

1. **Защита данных пользователя.**

   - Создать класс *UserAccount*, который представляет аккаунт пользователя с атрибутами: имя пользователя (*username*), электронная почта (*email*) и приватный атрибут пароль (*password*).
   - Использовать конструктор *__init__* для инициализации этих атрибутов.
   - Реализовать метод *set_password(new_password)*, который позволяет безопасно изменить пароль аккаунта.
   - Реализовать метод *check_password(password)* для проверки, соответствует ли введённый пароль текущему паролю аккаунта и возвращает *True* или *False*.
   - Создать объект класса *UserAccount*, попробовать изменить пароль и проверить его с помощью методов *set_password* и *check_password*.

2. **Полиморфизм и наследование.**

   - Определить базовый класс *Vehicle* с атрибутами: *make* (марка) и *model* (модель), а также методом *get_info()*, который возвращает информацию о транспортном средстве.
   - Создать класс *Car*, наследующий от *Vehicle*, и добавьте в него атрибут *fuel_type* (тип топлива). Переопределить метод *get_info()* таким образом, чтобы он включал информацию о типе топлива.

Ход работы

Задача 1. Создан класс *UserAccount* с требуемыми атрибутами, инициализированными через конструктор *__init__*. Реализован метод *set_password(new_password)* для безопасного изменения пароля аккаунта. Реализован метод *check_password(password)* для проверки пароля. Создан объект класса *UserAccount*, протестированы методы класса.

```
class UserAccount:
    def __init__(self, username, email, password):
        self.username = username
        self.email = email
        self.__password = password

    def set_password(self, new_password):
        self.__password = new_password
        return 'Пароль изменён :)'

    def check_password(self, password):
        if password == self.__password:
            return True
        return False


user = UserAccount('nmzhlk', 'nmzhlk@gmail.com', 'nmzhlk123')
print(f'Пользователь `{user.username}` с почтой `{user.email}`')

print(user.check_password('nmzhlk123'))

print(user.set_password('nmzhlk456'))
print(user.check_password('nmzhlk123'))
print(user.check_password('nmzhlk456'))
```

Задача 2. Определен базовый класс *Vehicle* с требуемыми атрибутами и методом *get_info()*, который возвращает информацию о ТС. Создан класс *Car*, наследующий от *Vehicle* с новым атрибутом *fuel_type*. Переопределен метод *get_info()*.

```
class Vehicle:
    def __init__(self):
        self.make = ''
        self.model = ''

    def get_info(self):
        return f'Это транспортное средство марки {self.make}, модель:
{self.model}'


class Car(Vehicle):
    def __init__(self):
        super().__init__()
        self.fuel_type = ''

    def get_info(self):
        return (f'Это транспортное средство марки {self.make},
модель: {self.model}.\n'
                f'Работает на топливе: {self.fuel_type}.')
```

```
ts = Vehicle()
print(ts.make, ts.model, '← Тут пусто')
ts.make, ts.model = 'КамАЗ', '6282 / Это электробус.'
print(ts.get_info())

car = Car()
print(car.make, car.model, car.fuel_type, '← Тут пусто')
car.make, car.model, car.fuel_type = 'Лада', 'Гранта', 'Бензин'
print(car.get_info())

# Выше показана очень плохая практика! Изменять атрибуты объекта
класса напрямую не стоит никогда.
```

Вывод

В результате выполнения лабораторной работы был получен практический опыт работы с ООП в Python, освоено использование инкапсуляции и наследования.