

Министерство цифрового развития и массовых коммуникаций
Российской Федерации

Ордена Трудового Красного Знамени федеральное государственное
бюджетное образовательное учреждение
высшего образования
«Московский технический университет связи и информатики»
(МТУСИ)

Кафедра Информационных технологий

Отчет по лабораторной работе № 6
по дисциплине «Введение в информационные технологии»
на тему: «Работа с классами»

Выполнил: студент группы БВТ2402
Левая Валерия Валерьевна

Москва 2024

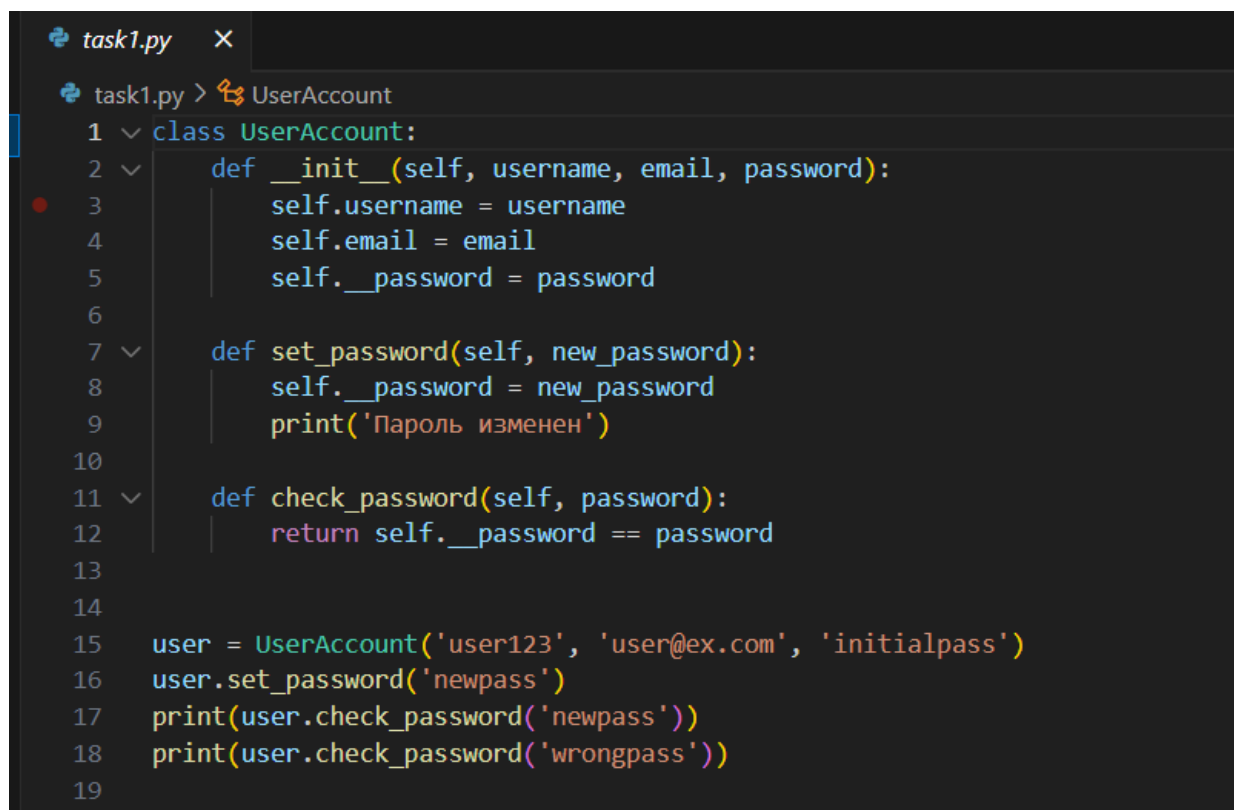
Цель работы: Получить практический опыт работы с ООП в Python.

использование инкапсуляции, наследования.

Ход работы:

Задание 1: Защита данных пользователя.

1. Создадим класс **UserAccount**, который представляет аккаунт пользователя с атрибутами: имя пользователя (**username**), электронная почта (**email**) и приватный атрибут пароль (**password**).
2. Используем конструктор **__init__** для инициализации этих атрибутов.
3. Реализуем метод **set_password(new_password)**, который позволяет безопасно изменить пароль аккаунта.
4. Реализуем метод **check_password(password)**, который проверяет, соответствует ли введенный пароль текущему паролю аккаунта и возвращает **True** или **False**.
5. Создадим объект класса **UserAccount**, изменим пароль и проверить его с помощью методов **set_password** и **check_password**.



```
task1.py  X
task1.py > UserAccount
1 class UserAccount:
2     def __init__(self, username, email, password):
3         self.username = username
4         self.email = email
5         self.__password = password
6
7     def set_password(self, new_password):
8         self.__password = new_password
9         print('Пароль изменен')
10
11    def check_password(self, password):
12        return self.__password == password
13
14
15    user = UserAccount('user123', 'user@ex.com', 'initialpass')
16    user.set_password('newpass')
17    print(user.check_password('newpass'))
18    print(user.check_password('wrongpass'))
19
```

Результат:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS C:\Users\levaa\OneDrive\Рабочий стол\ввикт\лаб6> & C:/Users/levaa/OneDrive/Рабочий стол\ввикт\лаб6/task1.py
Пароль изменен
True
False
○ PS C:\Users\levaa\OneDrive\Рабочий стол\ввикт\лаб6>
```

Задание 2: Полиморфизм и наследование.

1. Определим базовый класс **Vehicle** с атрибутами: **make** (марка) и **model** (модель), а также методом **get_info()**, который возвращает информацию о транспортном средстве.
2. Создадим класс **Car**, наследующий от **Vehicle**, и добавим в него атрибут **fuel_type** (тип топлива). Переопределим метод **get_info()** таким образом, чтобы он включал информацию о типе топлива.

```
task2.py x
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS C:\Users\levaa\OneDrive\Рабочий стол\ввикт\лаб6> & C:/Users/levaa/OneDrive/Рабочий стол\ввикт\лаб6/task2.py
Марка: Hyundai, Модель: Solaris, Тип топлива: Бензин
○ PS C:\Users\levaa\OneDrive\Рабочий стол\ввикт\лаб6>

10 class Car(Vehicle):
11     def __init__(self, make, model, fuel_type):
12         super().__init__(make, model)
13         self.fuel_type = fuel_type
14
15     def get_info(self):
16         return f"{super().get_info()}, Тип топлива: {self.fuel_type}"
17
18
19 car = Car('Hyundai', 'Solaris', 'Бензин')
20 print(car.get_info())
```

Результат:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS C:\Users\levaa\OneDrive\Рабочий стол\ввикт\лаб6> & C:/Users/levaa/OneDrive/Рабочий стол\ввикт\лаб6/task2.py
Марка: Hyundai, Модель: Solaris, Тип топлива: Бензин
○ PS C:\Users\levaa\OneDrive\Рабочий стол\ввикт\лаб6>
```