# **Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования «Московский технический университет связи и информатики»**.

# Лабораторная работа №6:

**‘’Работа с классами ч.2’’**

Выполнил: Безматерных Иван Алексеевич БВТ2402

Москва 2024

**Цель работы**: получить практический опыт работы с ООП в Python. использование инкапсуляции, наследования.

**Задание 1: защиты данных пользователя.**

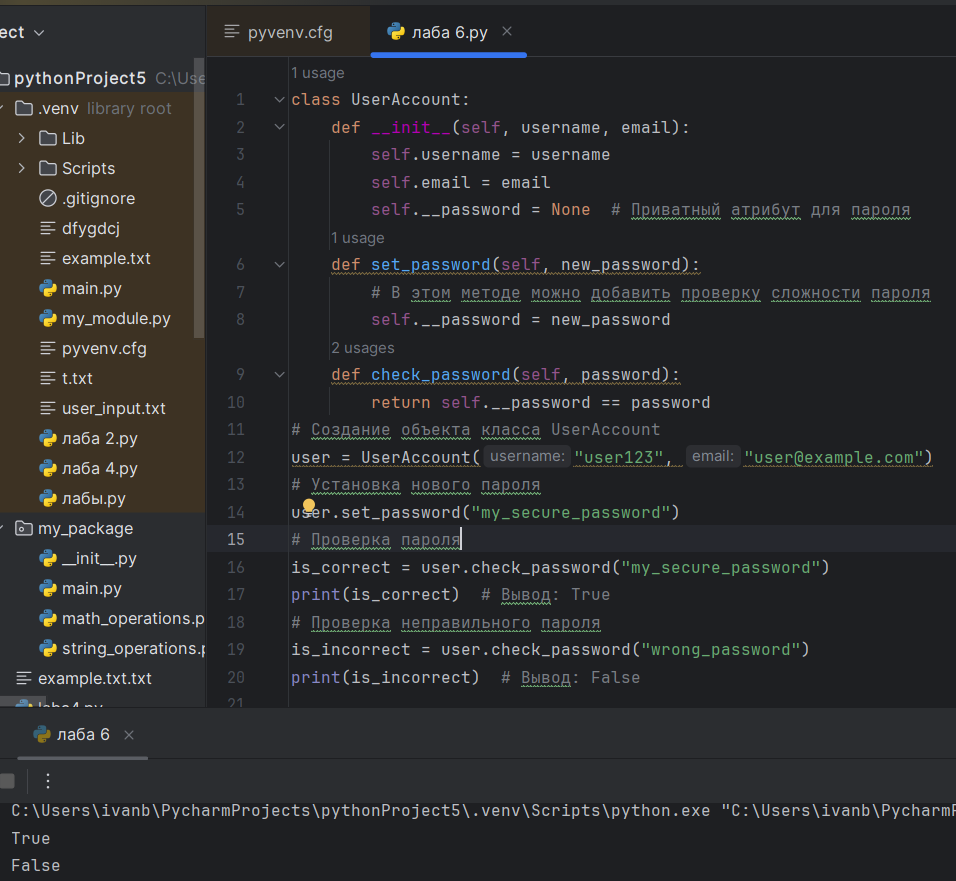
1. Создайте класс **UserAccount**, который представляет аккаунт пользователя с атрибутами: имя пользователя (**username**), электронная почта (**email**) и приватный атрибут пароль (**password**).
2. Используйте конструктор **\_\_init\_\_** для инициализации этих атрибутов.
3. Реализуйте метод **set\_password**(**new\_password**), который позволяет безопасно изменить пароль аккаунта.
4. Реализуйте метод **check\_password**(**password**), который проверяет, соответствует ли введённый пароль текущему паролю аккаунта и возвращает **True** или **False**.
5. Создайте объект класса **UserAccount**, попробуйте изменить пароль и проверить его с помощью методов **set\_password** и **check\_password**.

**Код программы:**

class UserAccount:  
 def \_\_init\_\_(self, username, email):  
 self.username = username  
 self.email = email  
 self.\_\_password = None # Приватный атрибут для пароля  
 def set\_password(self, new\_password):  
 # В этом методе можно добавить проверку сложности пароля  
 self.\_\_password = new\_password  
 def check\_password(self, password):  
 return self.\_\_password == password  
# Создание объекта класса UserAccount  
user = UserAccount("user123", "user@example.com")  
# Установка нового пароля  
user.set\_password("my\_secure\_password")  
# Проверка пароля  
is\_correct = user.check\_password("my\_secure\_password")  
print(is\_correct) # Вывод: True  
# Проверка неправильного пароля  
is\_incorrect = user.check\_password("wrong\_password")  
print(is\_incorrect) # Вывод: False

**Ход работы:**

Создал класс UserAccount с атрибутами username, email и приватным атрибутом password.  
Метод set\_password позволяет установить новый пароль.  
Метод check\_password проверяет, совпадает ли введённый пароль с текущим паролем и возвращаеTrue или False.   
Объект user демонстрирует, как использовать данные методы.



**Задание 2: полиморфизм и наследование.**

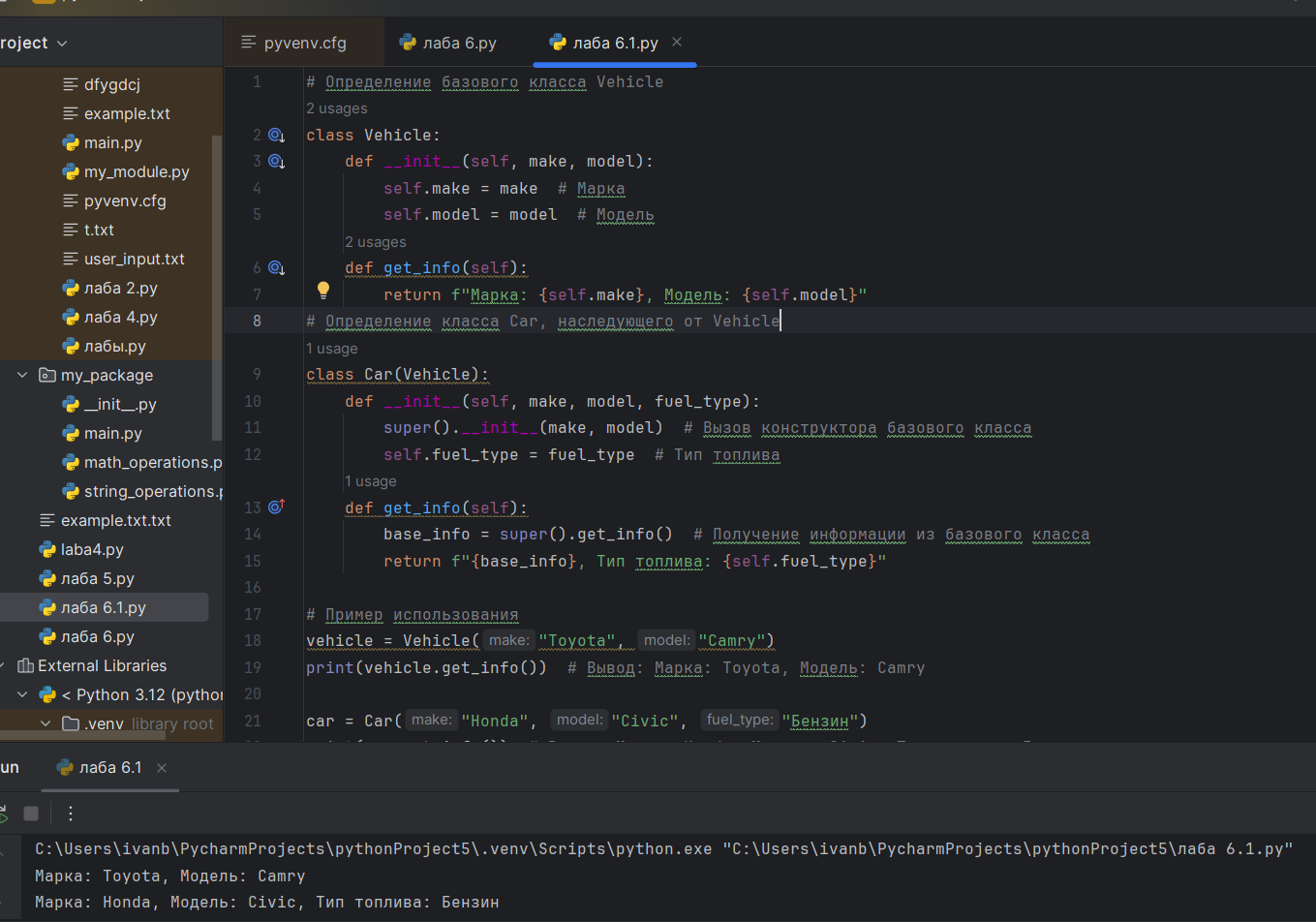
1. Определите базовый класс **Vehicle** с атрибутами: **make** (марка) и **model** (модель), а также методом **get\_info**(), который возвращает информацию о транспортном средстве.
2. Создайте класс **Car**, наследующий от **Vehicle**, и добавьте в него атрибут **fuel\_type** (тип топлива). Переопределите метод **get\_info**() таким образом, чтобы он включал информацию о типе топлива.

**Код программы:**

# Определение базового класса Vehicle  
class Vehicle:  
 def \_\_init\_\_(self, make, model):  
 self.make = make # Марка  
 self.model = model # Модель  
 def get\_info(self):  
 return f"Марка: {self.make}, Модель: {self.model}"  
# Определение класса Car, наследующего от Vehicle  
class Car(Vehicle):  
 def \_\_init\_\_(self, make, model, fuel\_type):  
 super().\_\_init\_\_(make, model) # Вызов конструктора базового класса  
 self.fuel\_type = fuel\_type # Тип топлива  
 def get\_info(self):  
 base\_info = super().get\_info() # Получение информации из базового класса  
 return f"{base\_info}, Тип топлива: {self.fuel\_type}"  
  
# Пример использования  
vehicle = Vehicle("Toyota", "Camry")  
print(vehicle.get\_info()) # Вывод: Марка: Toyota, Модель: Camry  
  
car = Car("Honda", "Civic", "Бензин")  
print(car.get\_info()) # Вывод: Марка: Honda, Модель: Civic, Тип топлива: Бензин

**Ход работы:**

1. Класс Vehicle имеет атрибуты make и model, а также метод get\_info(), который возвращает строку с информацией о транспортном средстве.  
2. Класс Car наследует от Vehicle и добавляет атрибут fuel\_type. Метод get\_info() переопределён для включения информации о типе топлива, используя метод родительского класса для получения базовой информации.



**Вывод:** я получил практический опыт работы с ООП в Python. использование инкапсуляции, наследования.