**2023/11/01 00:00|Лекция. Практика. Часть 2.**

## Практика. Часть 2.

**Продолжаем работать над нашей системой регистрации.**  
На предыдущем уроке мы сделали 2 класса(рис.1), первый из которых — это Database, наша база данных. Она содержит в себе словарик с данными, куда будут прилетать имя пользователя и пароль, также, соответственно, метод для добавления данных в этот словарик. Второй класс — это пользователь User, он непосредственно отвечает за создание пользователя, ввод имени, пароля и подтверждения пароля.

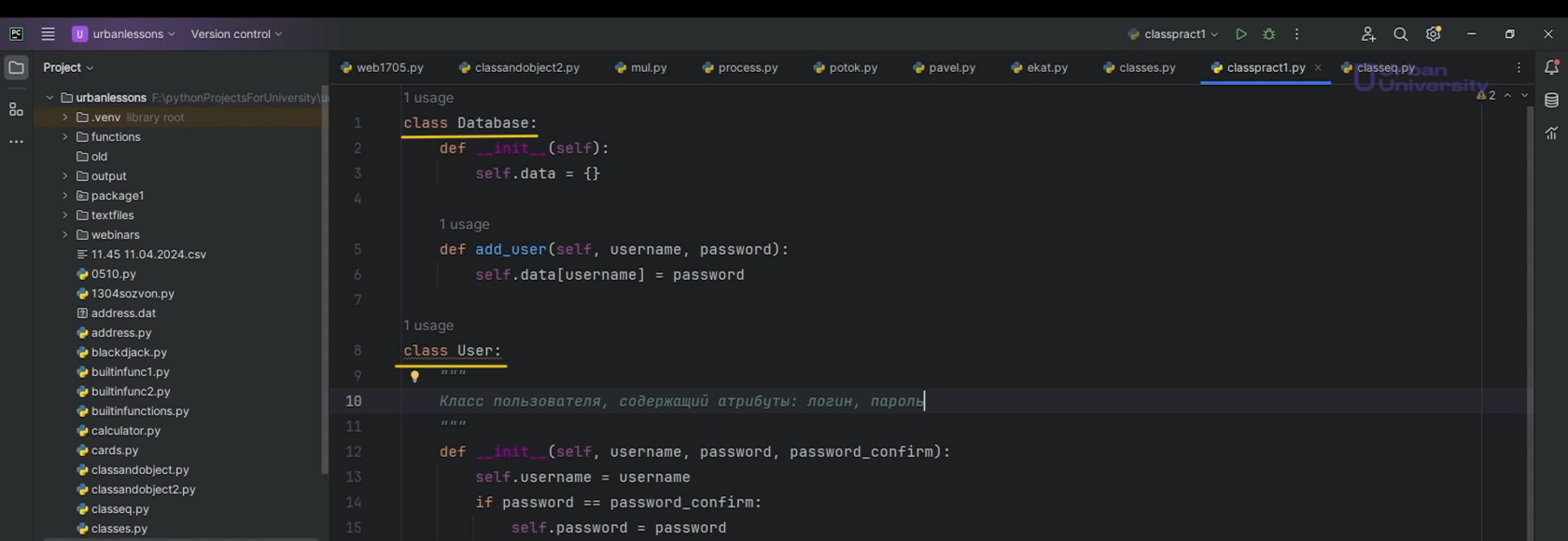


Рис.1

Остановились на том, что добавили в наш словарик введённое имя пользователя и пароль. Давайте посмотрим, как эти данные будут выглядеть. Для этого обратимся к объекту класса Database и достанем из него атрибут data. Запустим нашу программу, создадим какого-нибудь пользователя. Видим словарь, где ключом будет являться — логин пользователя, а значением — пароль, который будет соответствовать этому логину(рис.2).

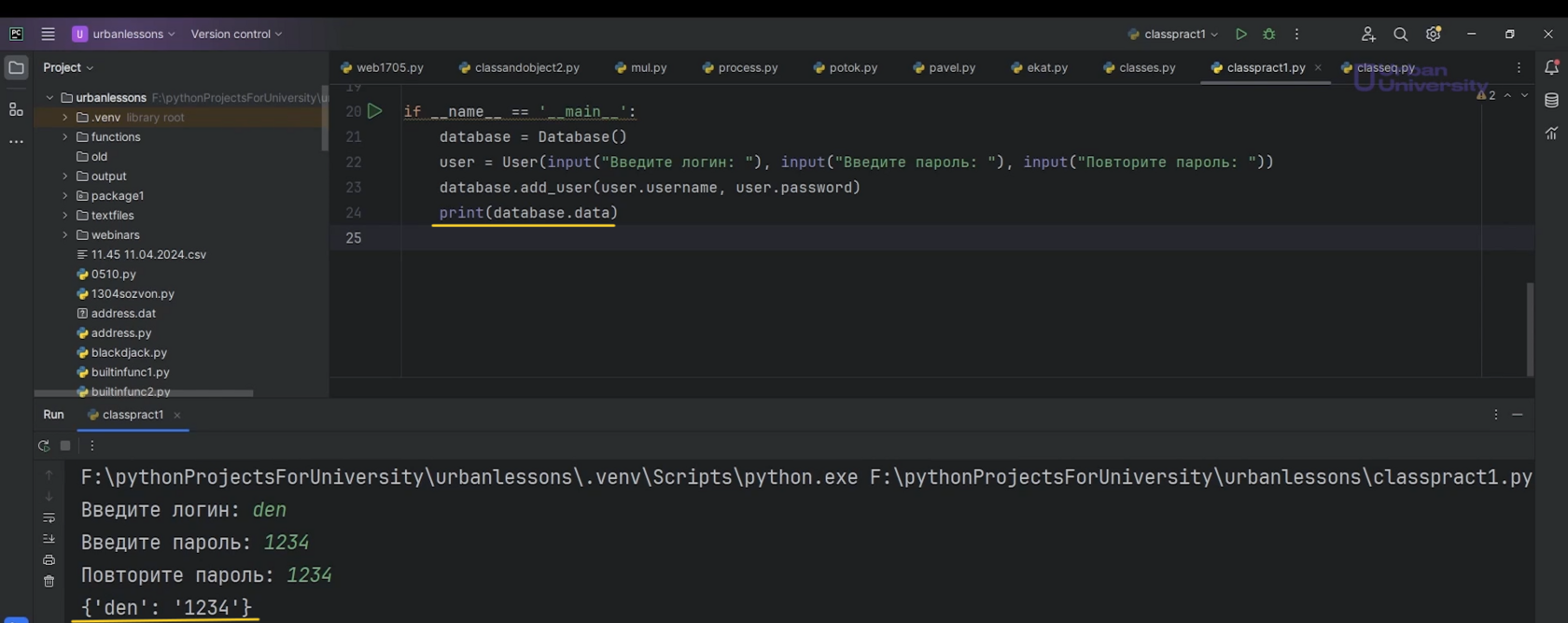


Рис.2

Однако обратите внимание, что сейчас атрибут password у пользователя создаётся только в том случае, если введённые пароли совпадают. Но бывает такое, что люди ошибаются при вводе пароля и у нас это не предусмотрено. К сожалению, сейчас мы не владеем информацией о том, как перехватывать ошибки, но в будущем обязательно этому научимся, и нам все равно надо с этим что-то делать. Приходится придумывать обходные пути. **Как это можно реализовать?**

Если введённые пароли не будут совпадать, то будем получать ошибку, потому что у нас не будет создаваться атрибут password. Иначе говоря, если условие self.password = password не выполнится, у User не будет пароля, соответственно, при вызове метода add\_user() будем получать ошибку.

Давайте проверим эту догадку. В результате действительно видим ошибку(Рис.3).

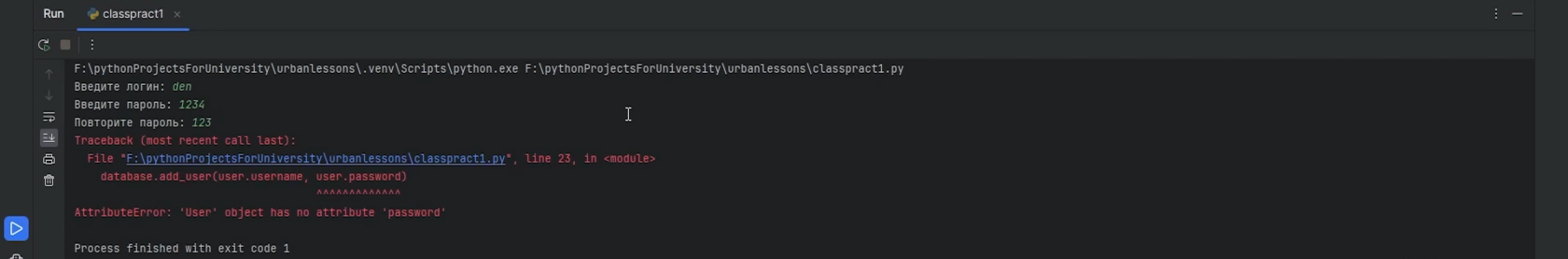


Рис.3

Можно добавить блок else. В случае, если введённые пароли не будут совпадать, можем выполнить какие-то строчки кода. Однако создавать заглушку в виде пароля, даже пустую строку, это будет не совсем правильно. Значит, нам нужно как-то предотвратить переход программы на строчку, где выполняется действие по добавлению пользователя.

Конструктор пользователя пока оставим в покое. Если посмотрим на строчку, где вызываются 3 функции input(), вместо параметров, которые запрашивают ввод данных с клавиатуры. По сути, можно сравнить введённый пароль и повтор пароля. Если они не совпадают, следует организовать какой-нибудь выход из программы, но сейчас этот ввод не хранится в памяти компьютера в какой-либо переменной. Это нужно исправить.

На данном этапе познакомимся с таким интересным оператором, имеющим достаточно смешное название **“Моржовый оператор”**. Название произошло из-за внешней схожести с моржом и называется он walrus (“морж”).

Создадим переменную password и поставим двоеточие и знак равно (**:=**)(рис.4). Соединение этих знаков и похоже на моржа.



Рис.4

**Почему так, а не обычный знак равно?**Дело в том-то, что внутри таких конструкций невозможно применить знак равно, так как это будет считаться параметром. Нам необходимо, чтобы в данной строчке, в текущем контексте, создавалась переменная password, и ей присваивалось значение ввода. То есть данный оператор можно применять внутри функции.

Например, внутри функции print(). Если вы попробуете создать переменную и присвоить ей значение обычным способом в функции print(), получите ошибку(рис.5). Password внутри функции воспринимается как какой-то параметр. Однако при использовании моржового оператора, здесь создастся переменная и произойдёт к ней обращение(рис.6). Теперь можем обратиться к нашему паролю и к повторному вводу этого пароля. Если наш пароль не совпадает с его повтором, будем просто завершать нашу программу. Для того чтобы завершить программу, будем вызывать функцию exit()(рис.7).

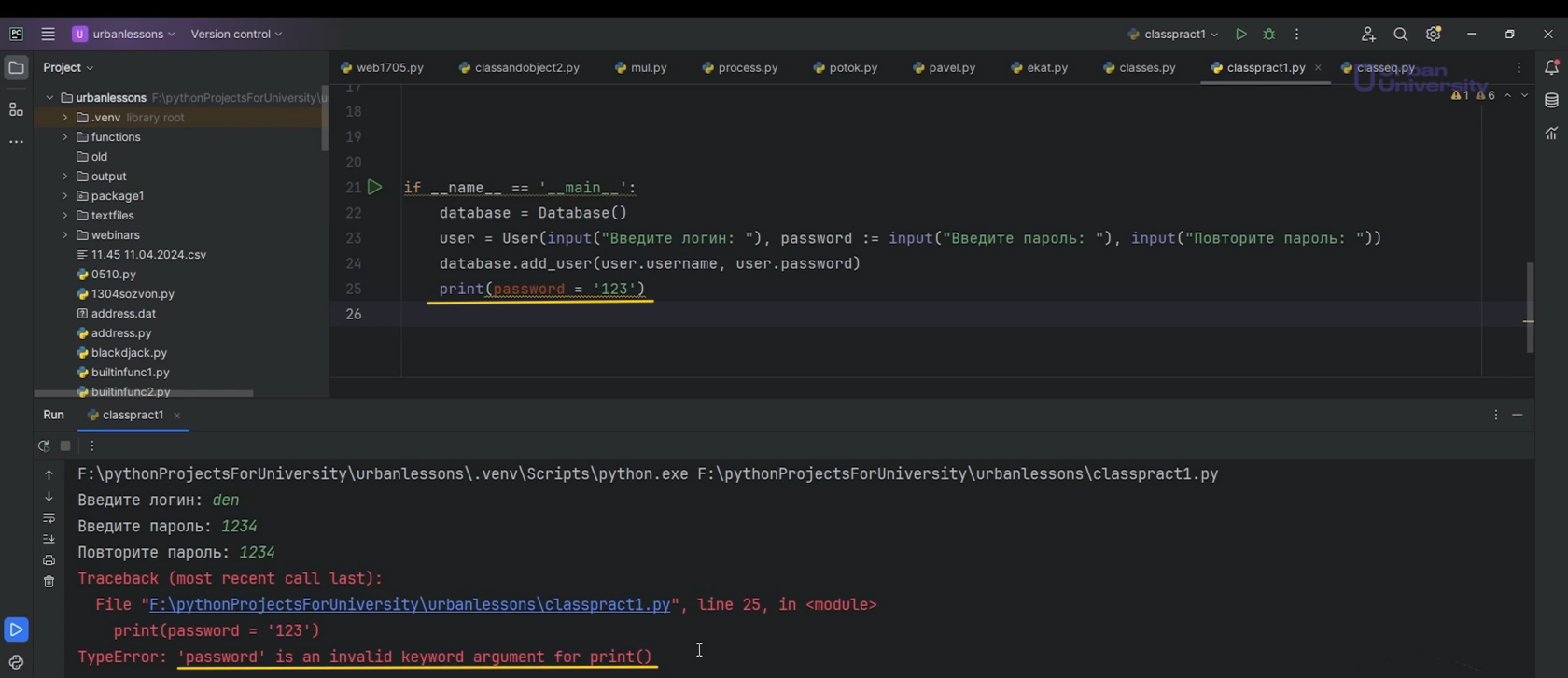


Рис.5

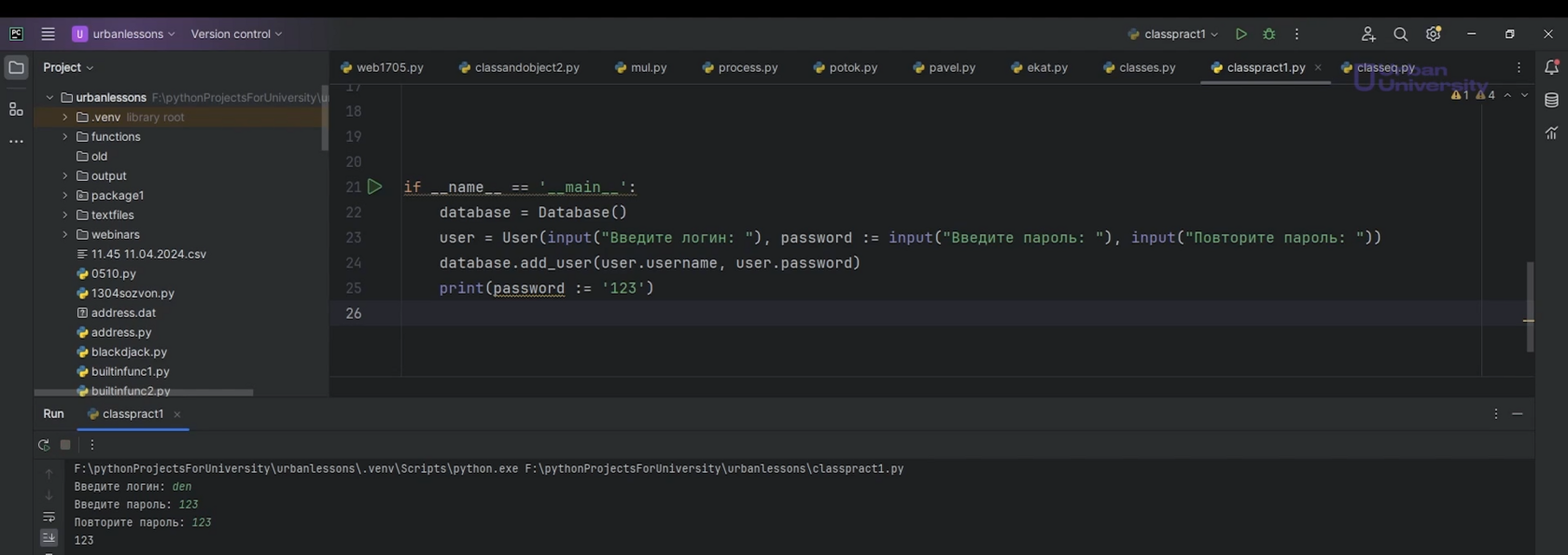


Рис.6

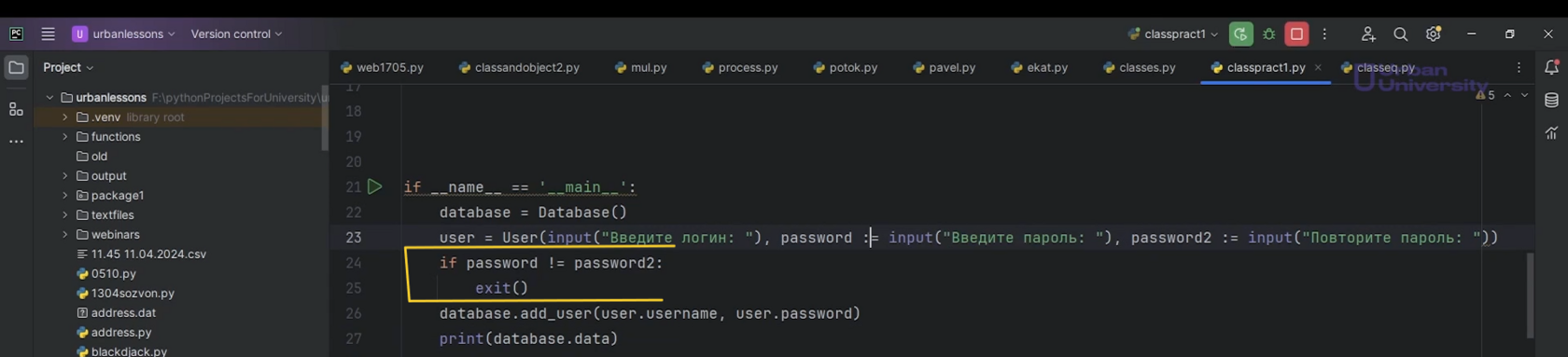


Рис.7

**Запустим нашу программу и проверим.**Все работает(рис.8). Теперь, если введённые пароли не совпадают, у нас предусмотрен выход из программы. В будущем это немного подправим, потому что у пользователей, скорее всего, должен быть выбор: логин (вход) или регистрация. То есть, если человек заходит впервые, то ему нужно зарегистрироваться, если человек заходит во второй раз, то у него уже есть учётная запись, ему регистрироваться незачем, он хочет просто войти.

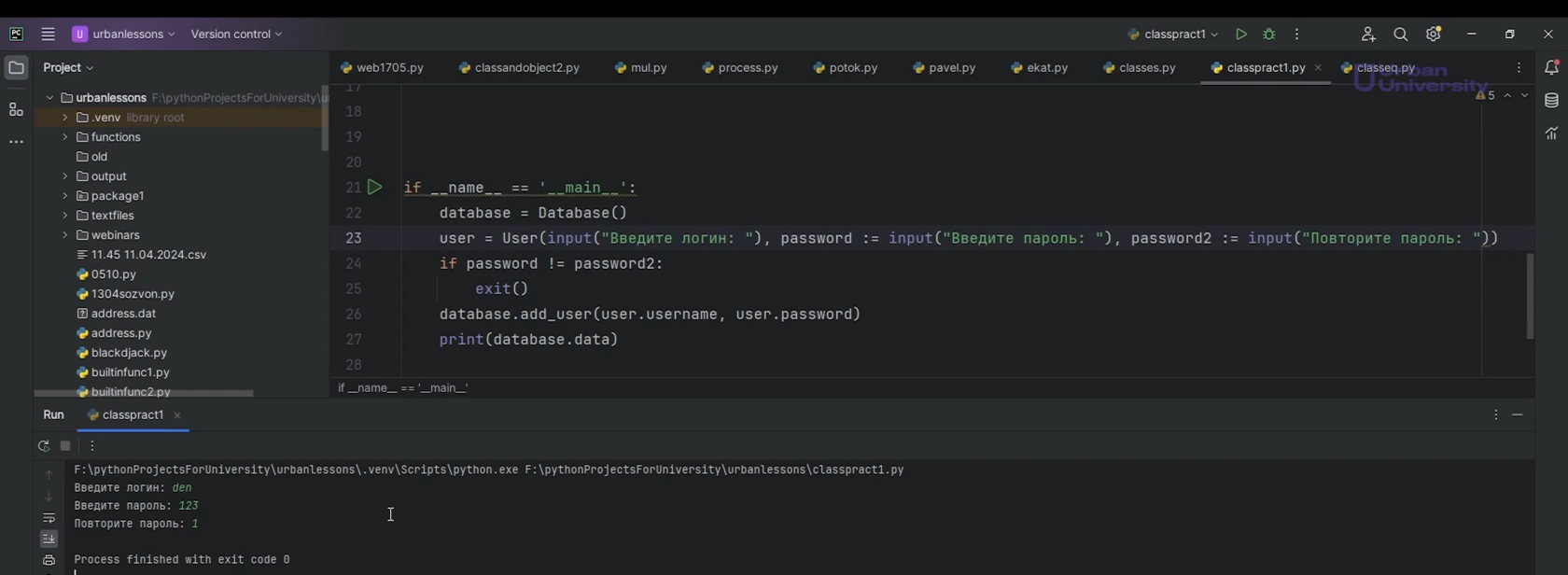


Рис.8

Сейчас мы не работаем ни с каким хранилищем данных. В будущем изучим, как работать с файлами. Можно будет реализовать сохранение данных с помощью файлов. Если захотите, вы сможете эти доработки внести в этот проект. Позднее научимся работать с базами данных. Когда у нас будут базы данных, тогда можно будет здесь сделать более правильную, более удобную и гибкую систему хранения информации, повторного обращения.

Сейчас у нас вся эта система регистрации держится в рамках одной сессии. То есть, когда запускаем программу, у нас появляется возможность зарегистрироваться. Однако при повторном запуске снова работаем с пустыми данными, но работаем с тем, что имеем, с тем, что пока знаем.

**Давайте доводить эту регистрацию до ума.** Выход предусмотрели, добавление пользователя есть. Дальше нужно сделать выбор пользователю и, возможно, некое зацикливание программы. Можно поступить простым образом: запустить бесконечный цикл, который будет создавать пользователей. На что стоит обратить внимание? Каждый раз нам создавать новую базу данных не надо, поэтому базу данных мы с вами вынесем за пределы цикла while(рис.9).

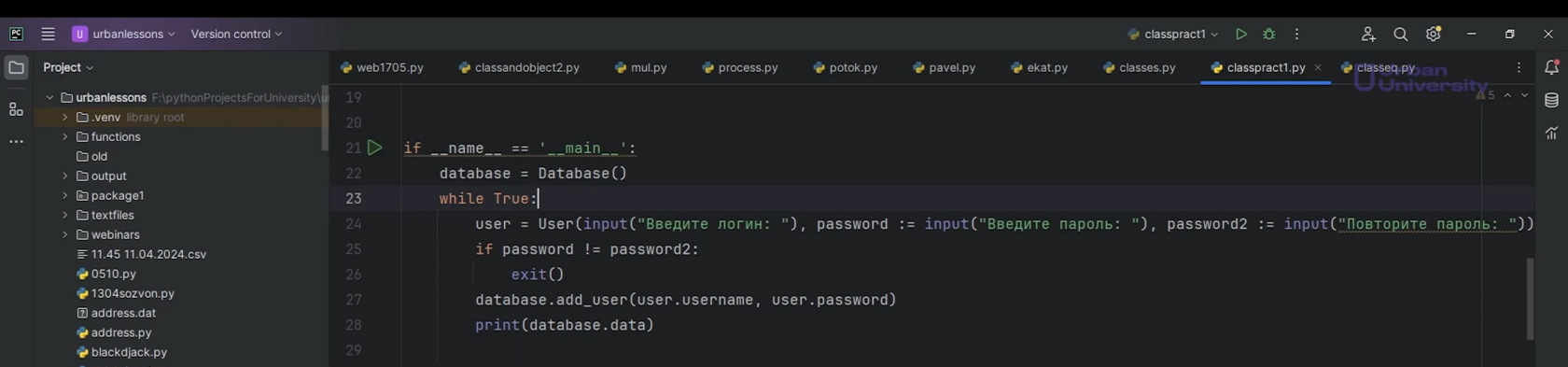


Рис.9

Цикл while (“пока”) позволяет выполнить одну и ту же последовательность действий, пока проверяемое условие истинно. Условие записывается до тела цикла и проверяется до выполнения тела цикла. Как правило, цикл while используется, когда невозможно определить точное значение количества проходов исполнения цикла.

Далее предоставим пользователю выбор действия. Для того чтобы это сделать просто добавим блок с условием, но перед этим попросим у пользователя выбрать какое-либо действие. Это можно сделать с помощью ввода с клавиатуры. Создадим какую-нибудь переменную choice(рис.10). Она будет запрашивать у пользователя ввод, где его поприветствуем “Приветствую! Выберите действие:”. Здесь воспользуемся символом переноса на новую строку — backslash (\) n, чтобы у нас был красивый вывод. Напишем первый выбор — “Вход”. Также перенесём на новую строку второй выбор “Регистрация”. Закончим этим же символом переноса на новую строку, чтобы было красиво.

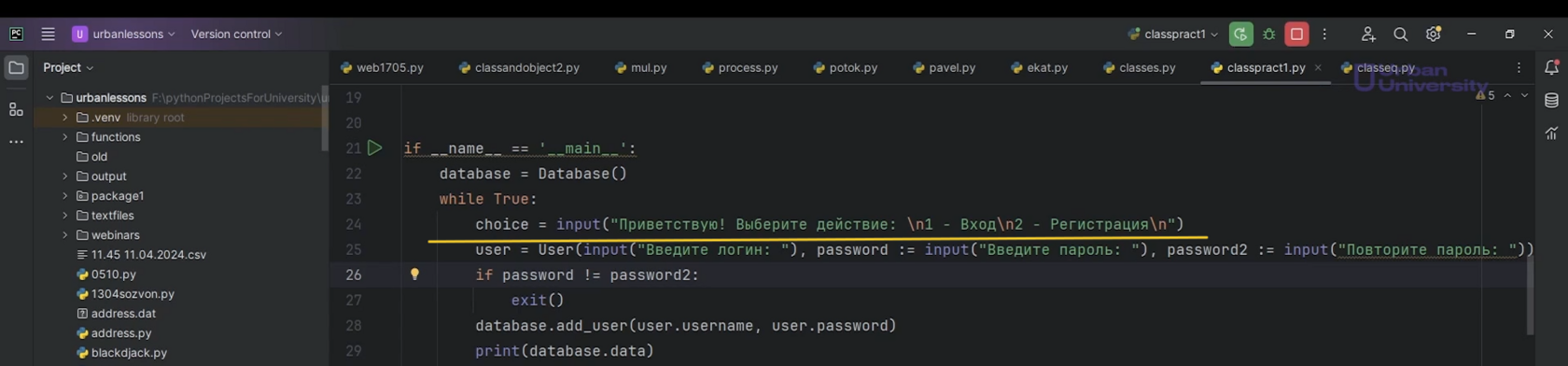


Рис.10

Посмотрим, как это будет выглядеть(рис.11):

“Выберите действие:  
1 - Вход  
2 - Регистрация”.

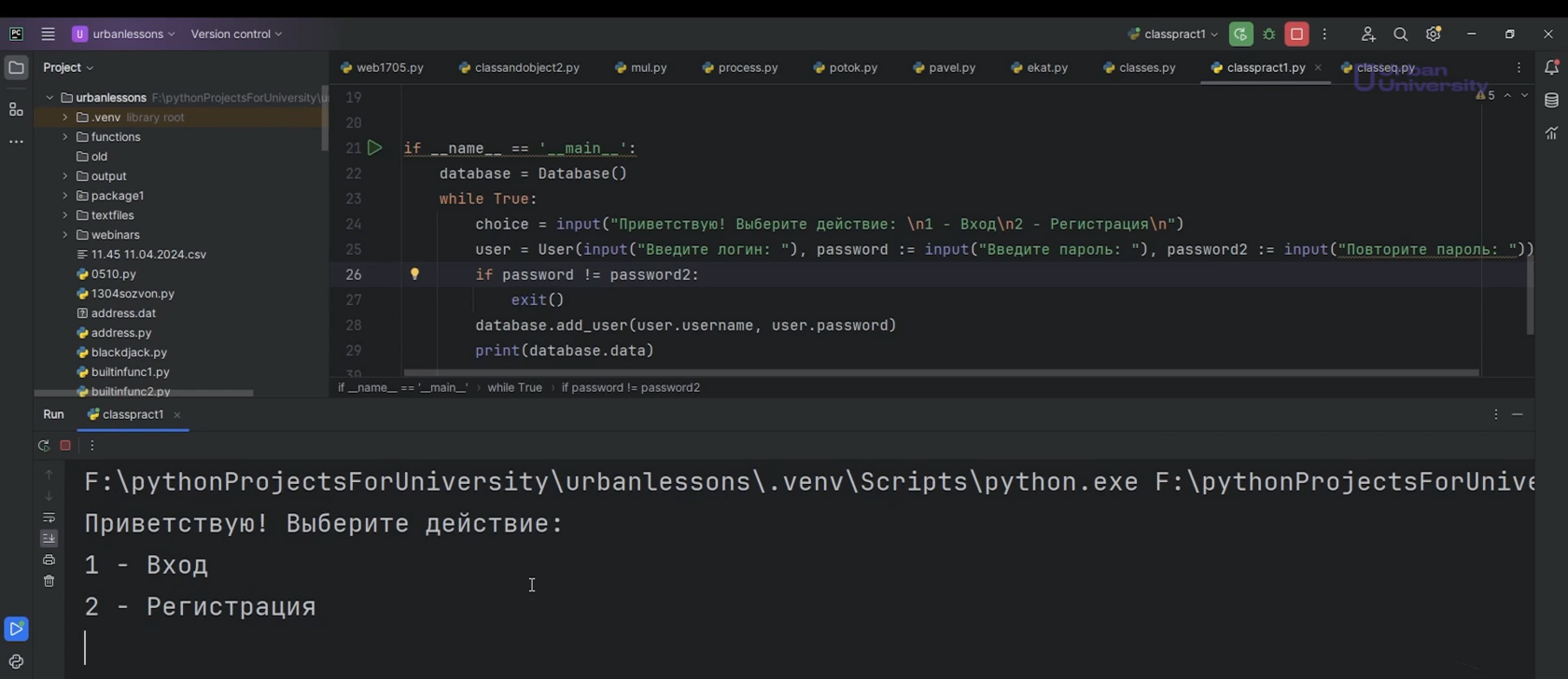


Рис.11

Пользователь будет выбирать действие, то есть вводить 1 или 2 и в зависимости от этого выбора у нас будет происходить дальнейшее определение в программе: будет выполняться либо создание нового пользователя, либо же будем брать данные из нашей базы и сверять корректность введённых данных для того, чтобы сказать, что пользователь вошёл либо же, что он ошибся.

Все эти тонкости сделаем с вами уже в следующем занятии. **На данном этапе рекомендуем потренироваться.** Как вариант для развития данного проекта — это возможность проверки пароля, некой валидации, то есть проверки на соответствие. Сейчас наиболее распространённая система проверки паролей, это как минимум, наличие хотя бы 8 символов, наличие 1 заглавной буквы и хотя бы 1 цифры в пароле. Так как вы уже умеете работать со строками, попробуйте добавить сюда эту проверку. То есть берите пароль, проверяйте, что хотя бы 1 символ в нём заглавный и хотя бы 1 символ является цифрой. Если эти условия соблюдаются, то можно продолжать регистрацию, если нет, можно выводить какое-нибудь предупреждение и просить пользователя заново выбрать действие.