**2023/09/27 00:00|Лекция. Цели и задачи. Поток выполнения программы.**

## Цели и задачи. Поток выполнения программы. Как интерпретатор показывает переменные.

На сегодняшнем занятии мы с вами поговорим про то, как программа выполняется.

Вы уже дошли до 2 модуля, то есть уже разобрались с различными структурами данных, познакомились с числами, строками, множествами, словарями и т.д. Но этот модуль будет посвящен уже другим конструкциям. В этом модуле мы с вами познакомимся с циклами, познакомимся с условиями и немножко коснёмся функции. Прежде чем мы приступим к изучению всех этих интересных вещей, посмотрим на то, **как у нас программа выполняется.**

Многие из вас уже, скорее всего, успели заметить то, что программа выполняется как бы построчно, начиная с 1 строчки, заканчивая самой последней, то есть когда мы запускаем программу, у нас выполняется сначала 1 строчка кода, затем 2, 3, 4, 5, 6, 7 и так далее. Например, если сейчас запустить программу(рис.1), то мы увидим, что через некоторое время выводится строка print(‘Фух, 4 секунды прошло’)(рис.2). Мы специально подключили команду sleep, с этим мы тоже познакомимся, с тем, как подключать что-то к своей программе, которая позволяет подождать 4 секунды. Сделали это для того, чтобы вы могли увидеть то, что здесь программа представляет собой один, скажем так, порядок действий. У нас не выполняется команда print(‘Фух, 4 секунды прошло’) до тех пор, пока не выполнится команда, которая стояла перед ней - sleep(4), то есть сейчас сработал print(‘Я тут’)(рис.1). После ожидания сработал print(‘Фух, 4 секунды прошло’)(рис.2).

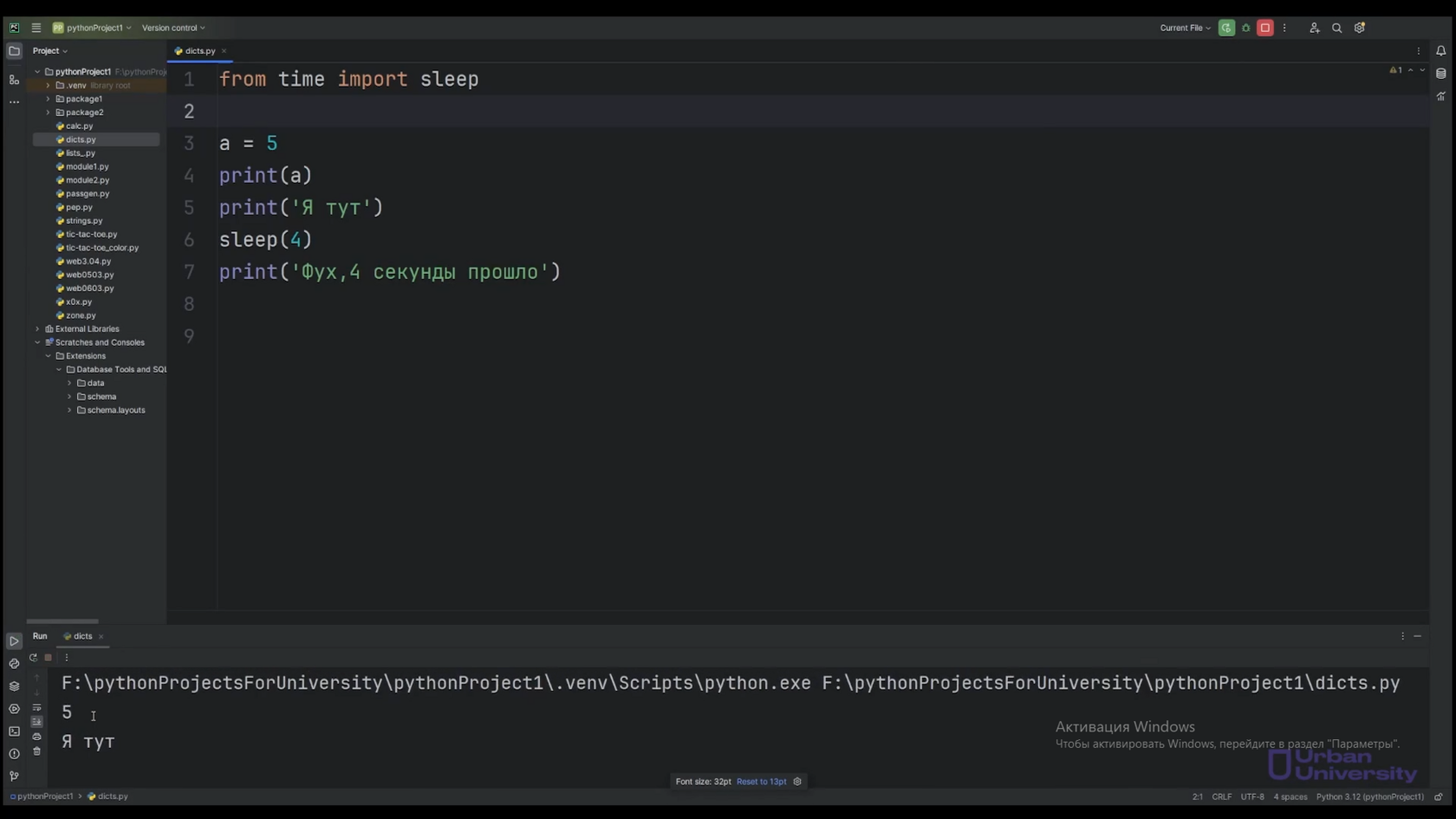


Рис.1

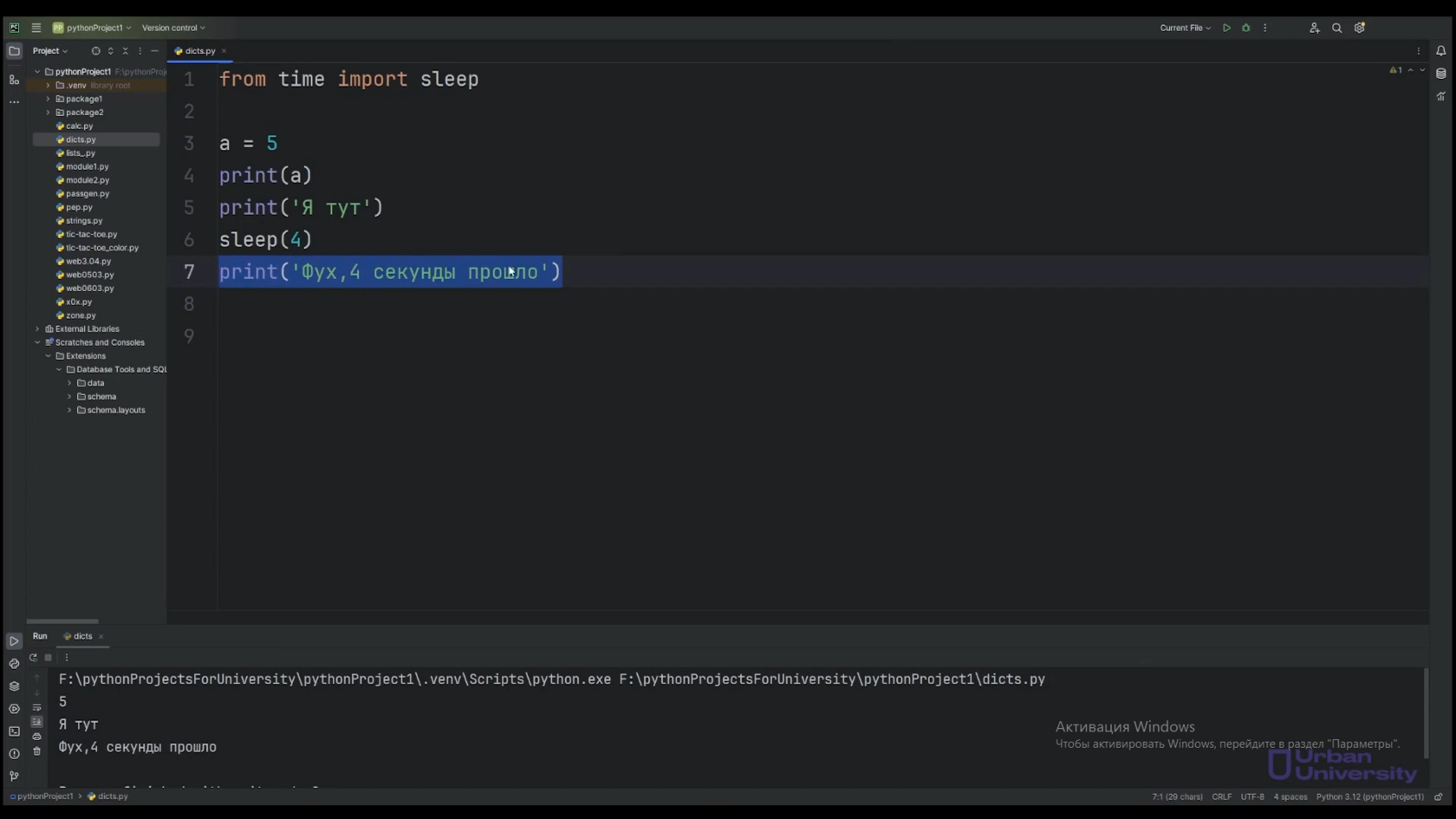


Рис.2

У нашей программы сейчас всего один поток выполнения, то есть у нас все действия выполняются друг за другом. Она не может выполнять несколько команд одновременно. В будущем мы сами будем разбирать, как можно сделать несколько потоков, несколько процессов и т.д., но пока мы работаем с такими однопоточными программами.

У нас есть такой инструмент**debug**, который находится прямо рядом с кнопкой Run(рис.3) и многие, наверное, видели вот эти точки(рис.3), может ставили случайно, думали, а зачем они нужны? Это точки остановки.

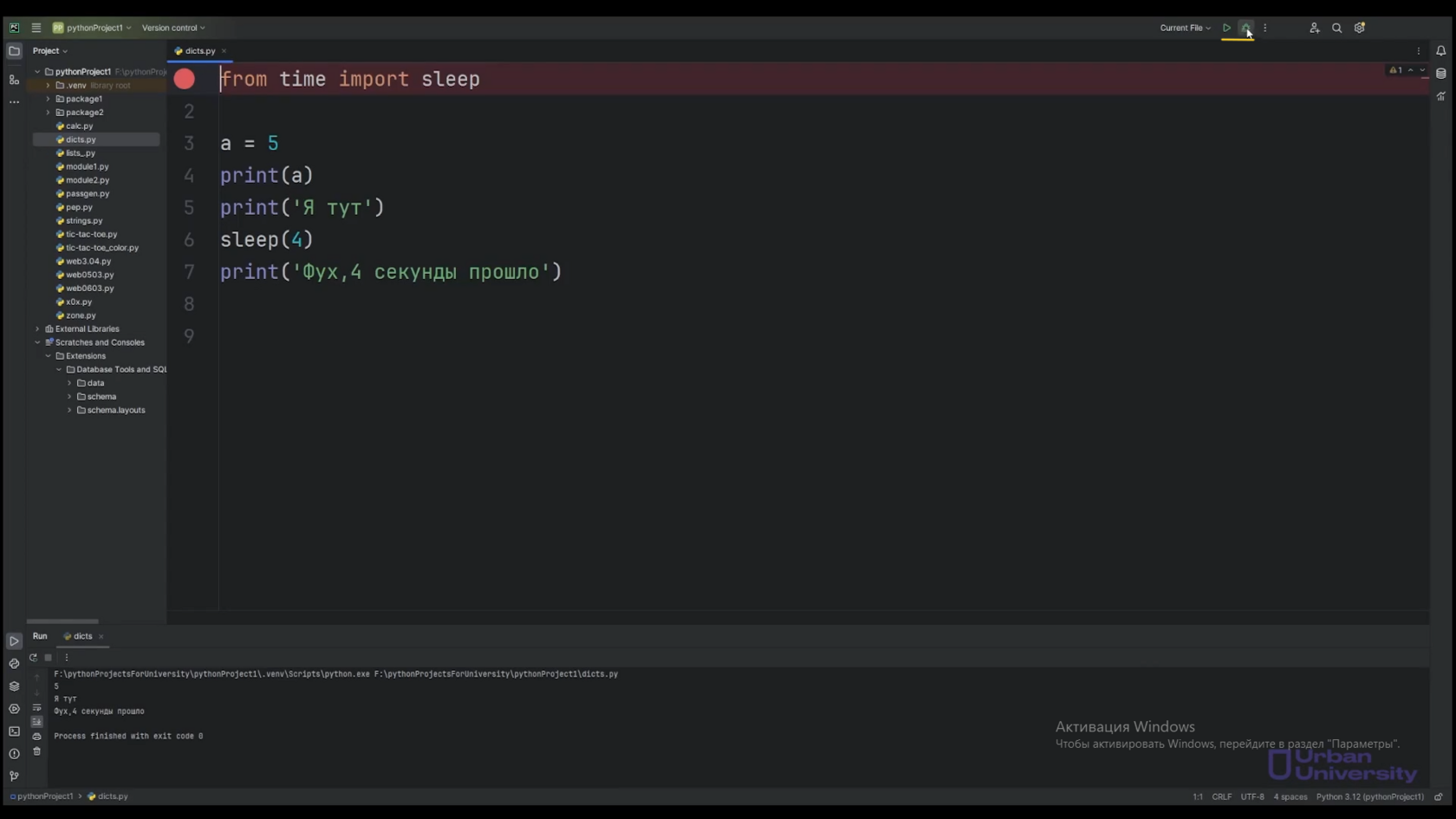


Рис.3

Если мы запускаем режим отладки, он останавливается на этой самой точке. Тут также можно увидеть, что у нас есть поток выполнения программы, он у нас здесь один. Запущен , dicts.py,также можно посмотреть на переменные, которые тут существуют(рис.4).

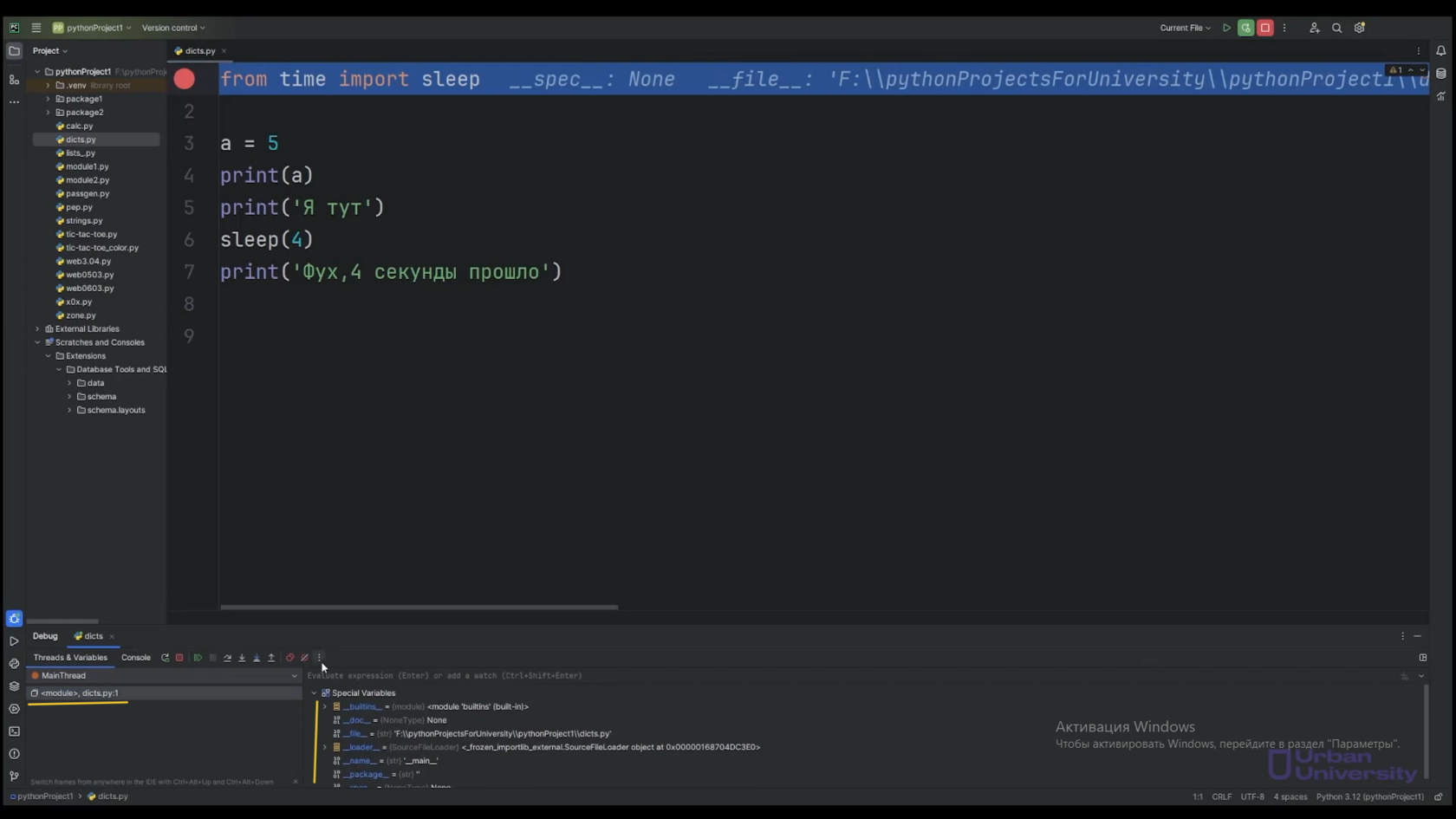


Рис.4

В консоли здесь пока ничего не происходит. Здесь есть кнопки, которые могут выполнять какие-то действия. Нас интересует кнопка step over(рис.5). При нажатии на нее выполнение перескакивает на следующую строчку, на следующий шаг программы.

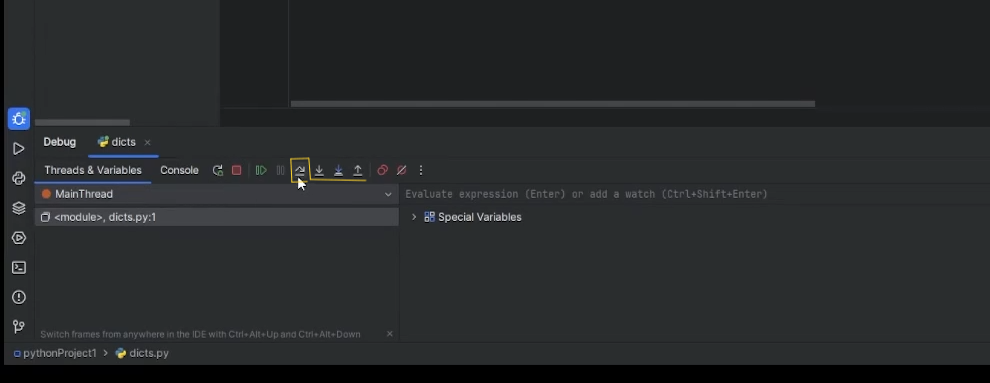


Рис.5

Что тут происходит? Мы перескочили на создание переменной, а на 1 строчке появилось значение sleep, и мы видим его в окошке режима отладки(рис.6). Это значение sleep(рис.7) мы взяли из другого, скажем так, файла и подключил к своему.

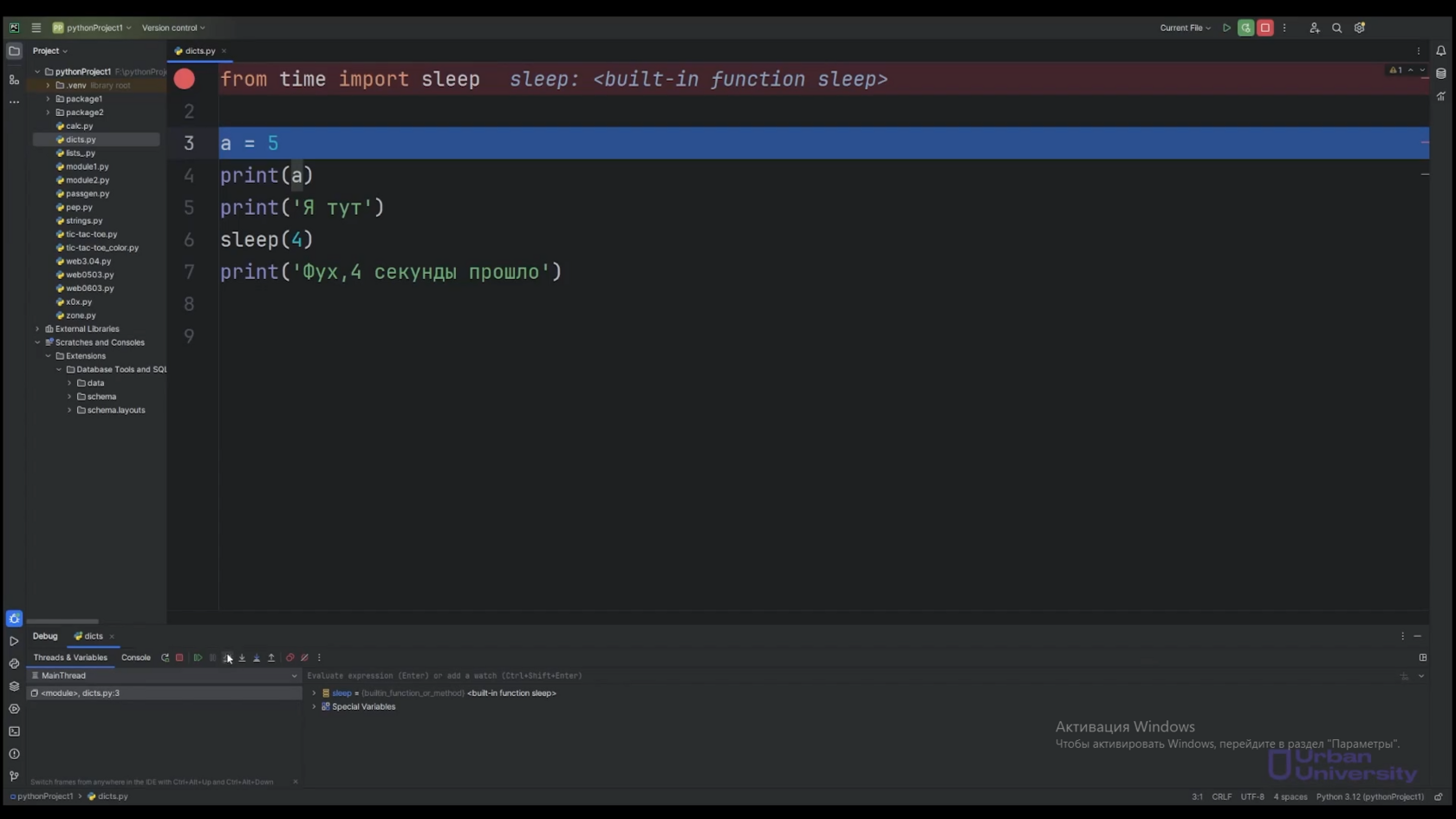


Рис.6

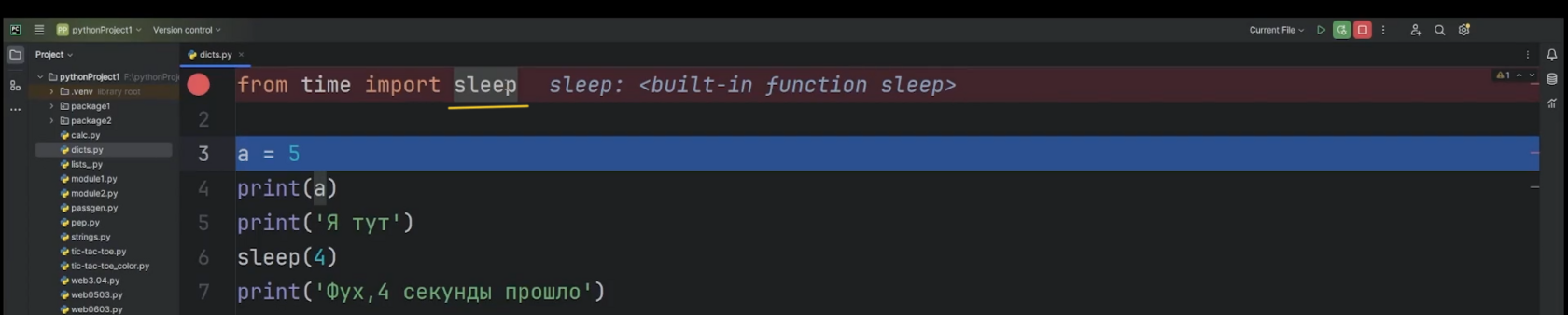


Рис.7

После того, как у нас создалась переменная “а”, которая является целым числом, мы можем увидеть ее в списке режима отладки(рис.8), то есть она создалась, а затем ее вывели. Смотрим на консоль и видим результат. Там у нас сработала строчка print(a), она была обработана.

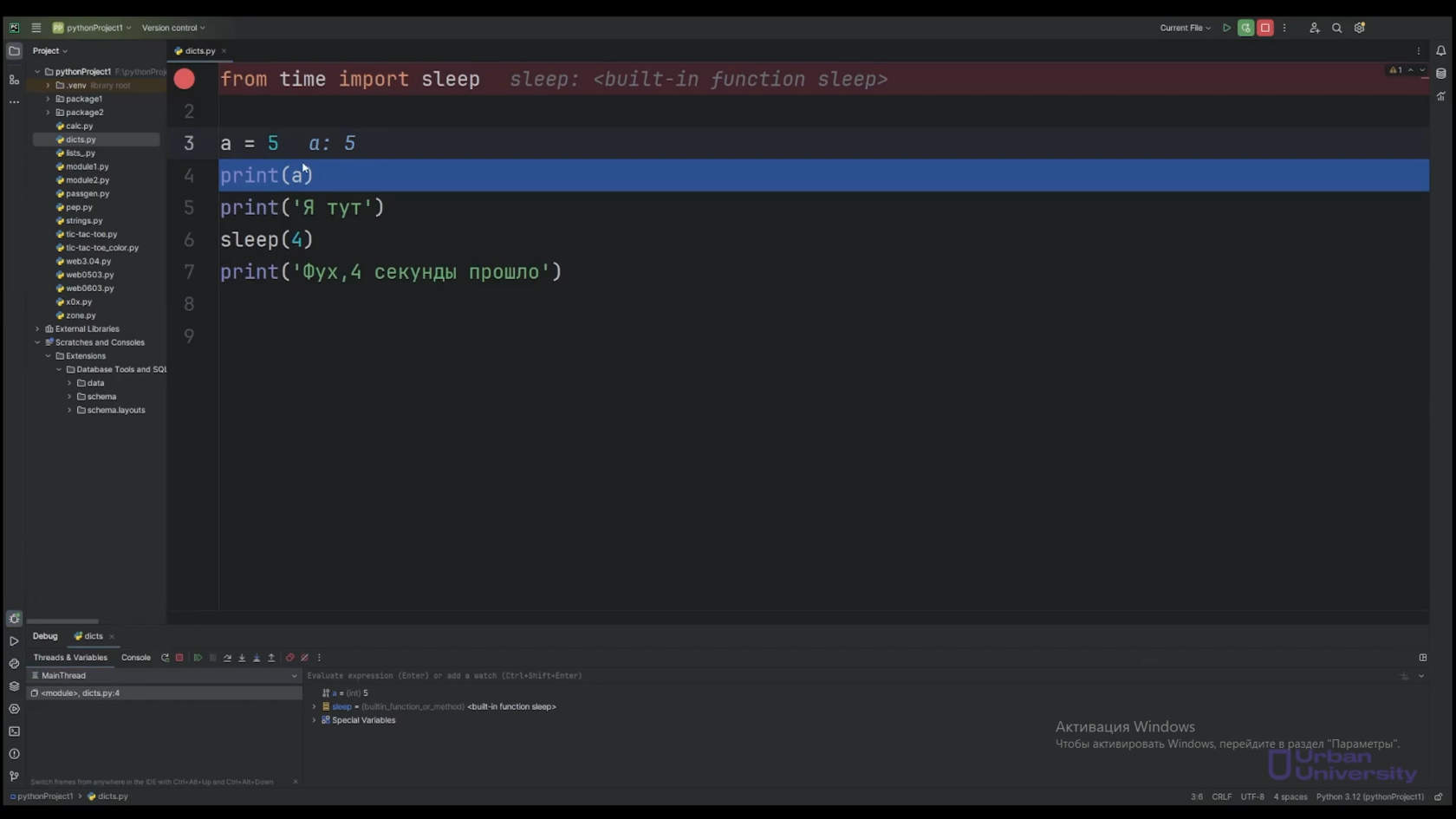


Рис.8

Далее, возвращаясь к консоли, видим “Я тут”(рис.9), видим, что никаких переменных уже не создавалось(рис.10).

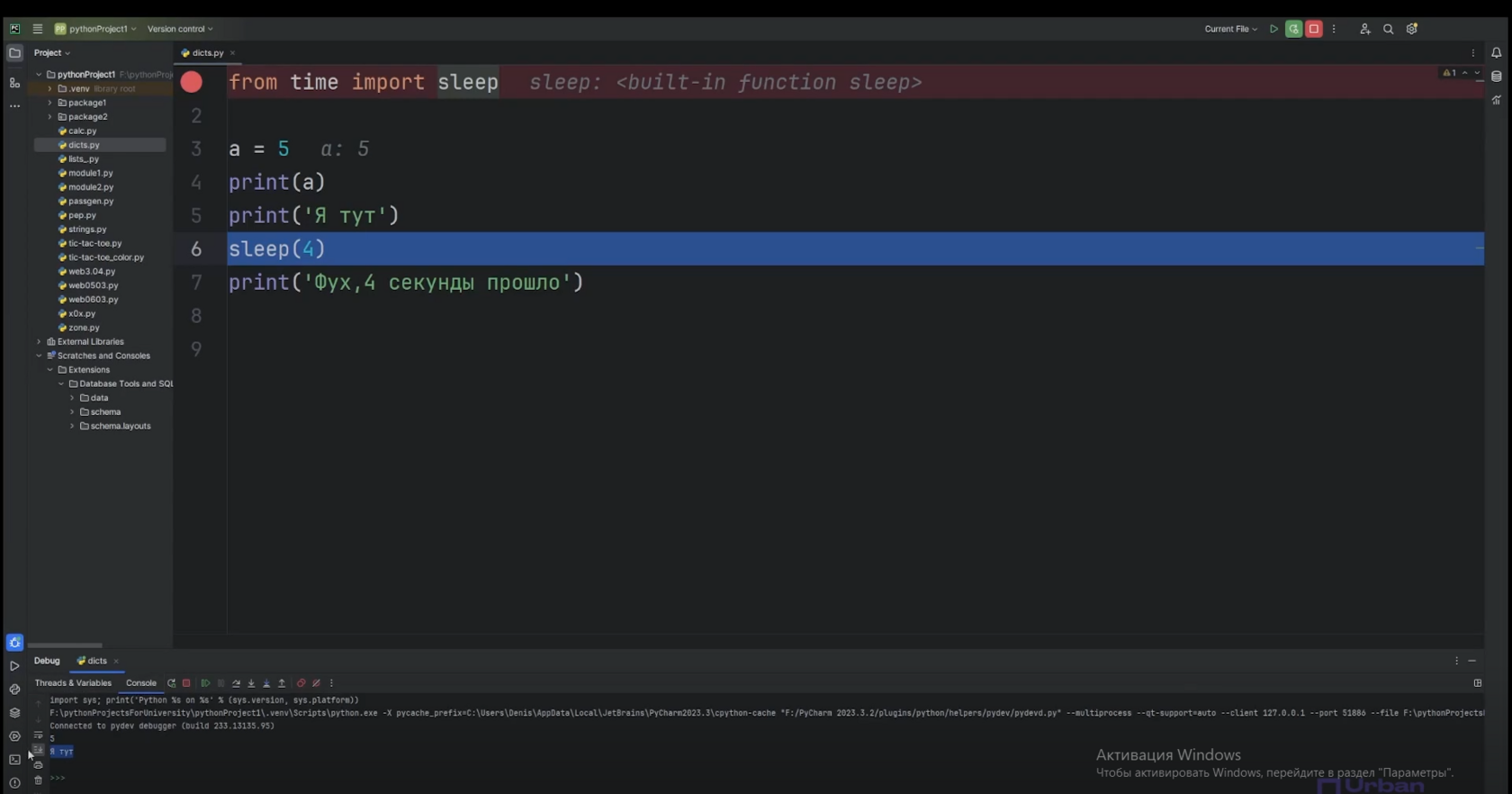


Рис.9

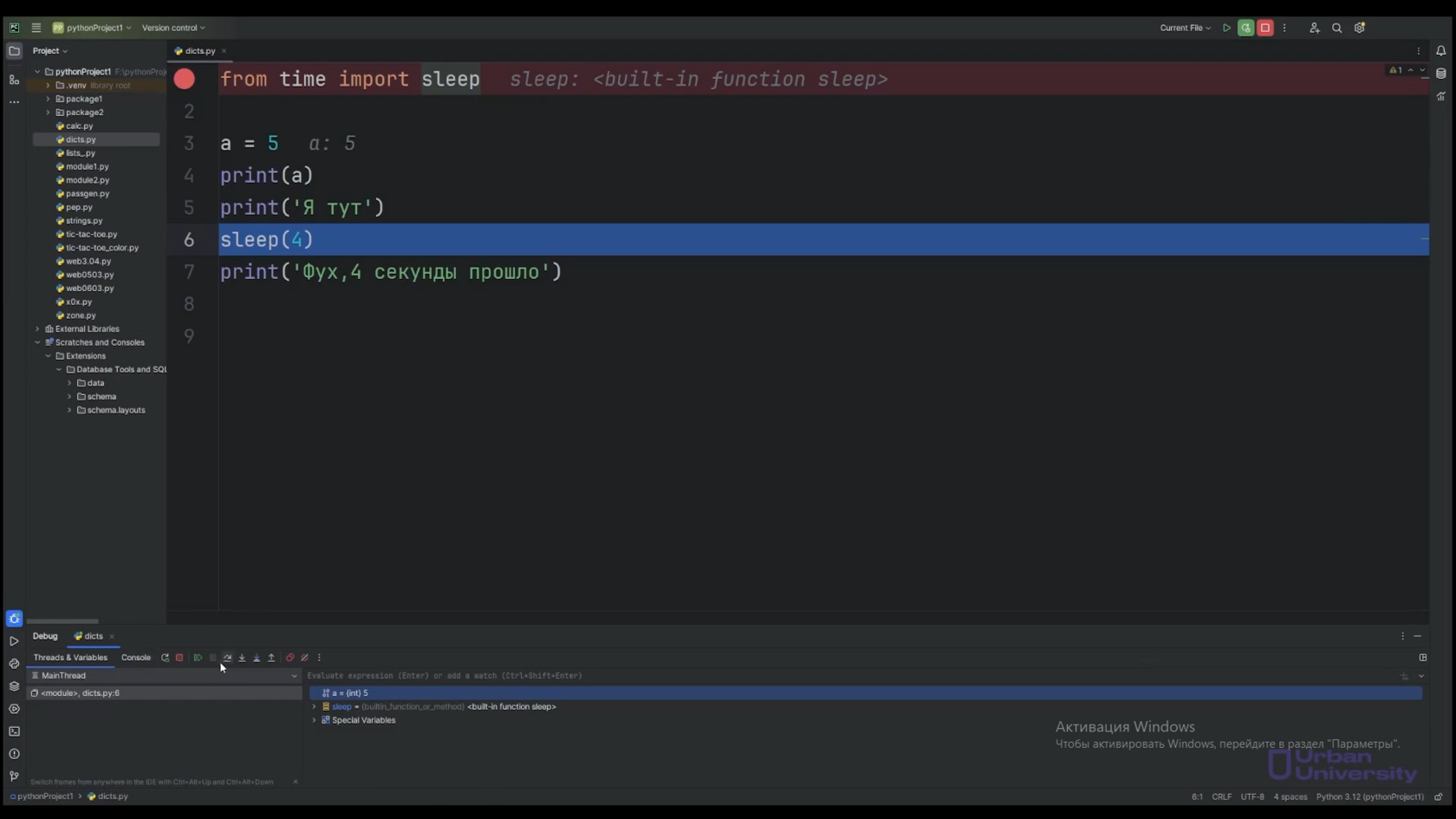


Рис.10

Затем прошло ожидание 4 секунды. У нас debug, так скажем, ожил и тут пока ничего не появилось, но после этого ожидания обрабатывается команда print(‘Фух, 4 секунды прошло’) и видим новый вывод, что прошло 4 секунды(рис.11).



Рис.11

Таким образом, мы могли наблюдать то, что у нас каждое действие в нашей программе выполняется одно за другим, то есть сначала выполняется действие, которое было ранее объявлено, потом уже выполняется следующее. Этот инструмент debug позволяет вам отследить логику выполнения вашей программы и при изучении циклов, при изучении условий, это может сильно помочь разобраться в том, как это работает. Поэтому активно стараемся использовать инструмент отладки, когда мы пишем большие программы, в которых, скажем так, практически не видно, как у нас идёт сама программа, то есть ход мысли программы.