**2023/09/29 00:00|Лекция. Стиль кода часть II. Цикл While**

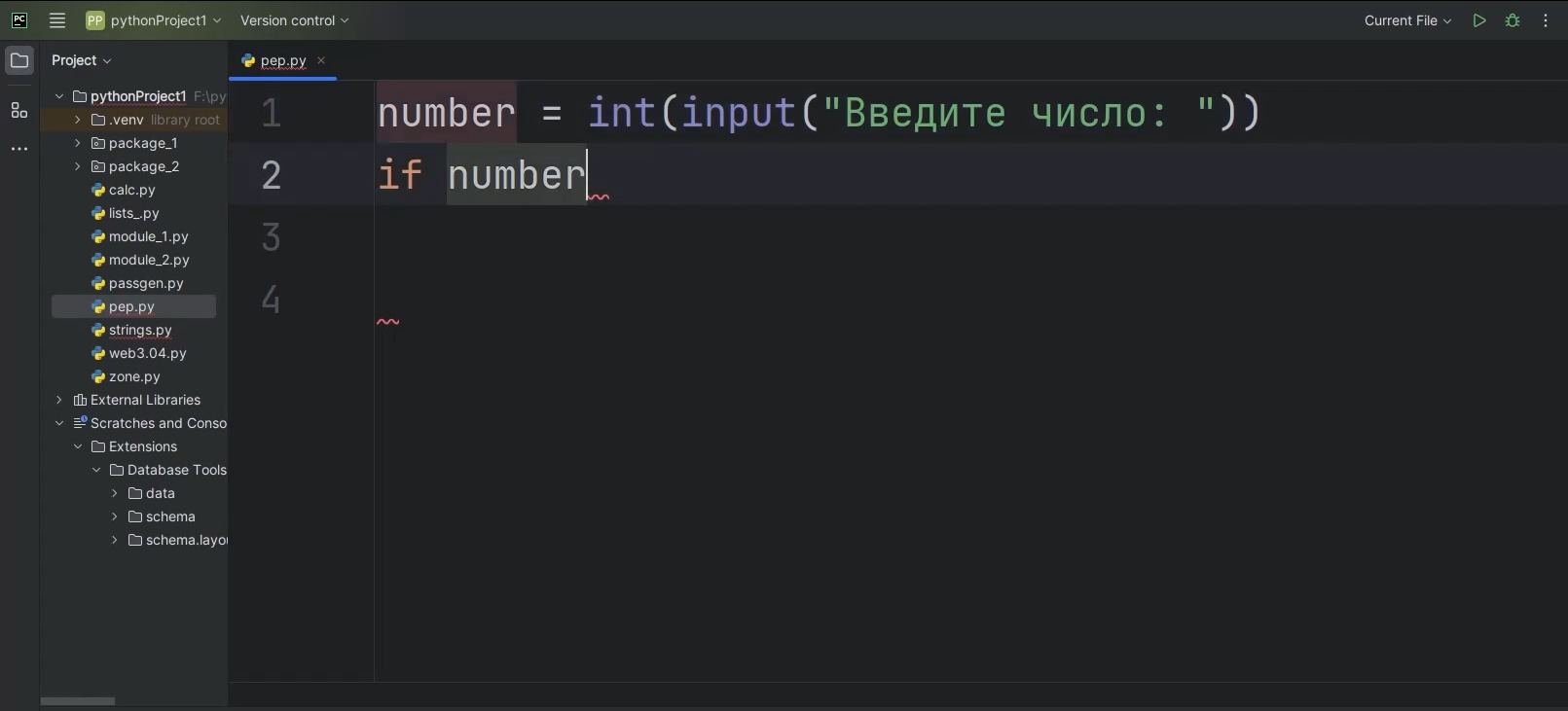
## Стиль кода часть II. Цикл While

Сегодня мы продолжаем тему стиля кода, вместе с тем знакомясь с циклом while. Давайте скорее начнем!

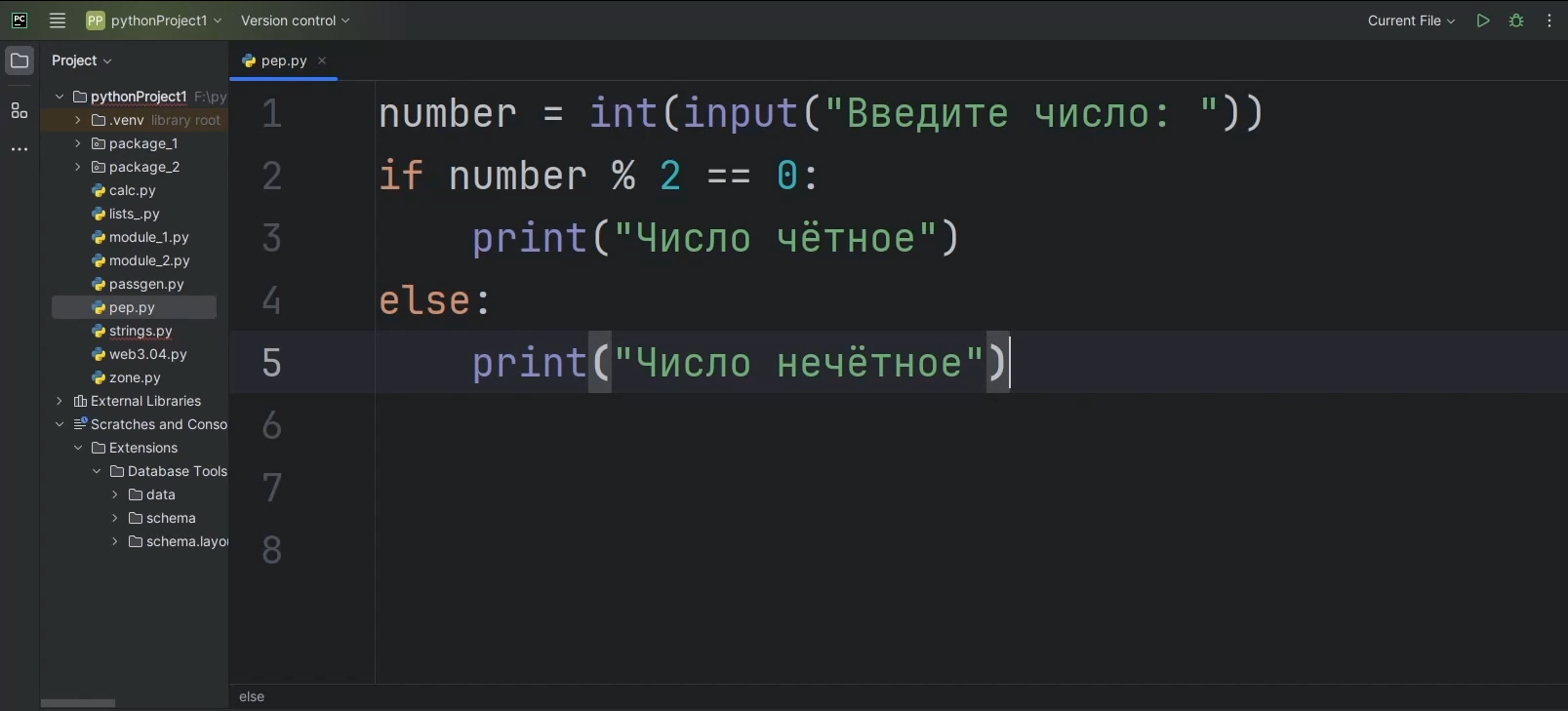
### Зачем нужны циклы?

**Циклы – это команды, которые повторяют другие команды**. Это очень полезная вещь, которая часто используется на практике.

Например, у нас есть код, где нам нужно ввести какое-либо число (Рис.1). Программа должна будет проверить четное число или нечетное. То есть если число делится на 2 без остатка, то мы выведем, что это четное число. А при любом другом раскладе выведем, что число нечетное (Рис.2).

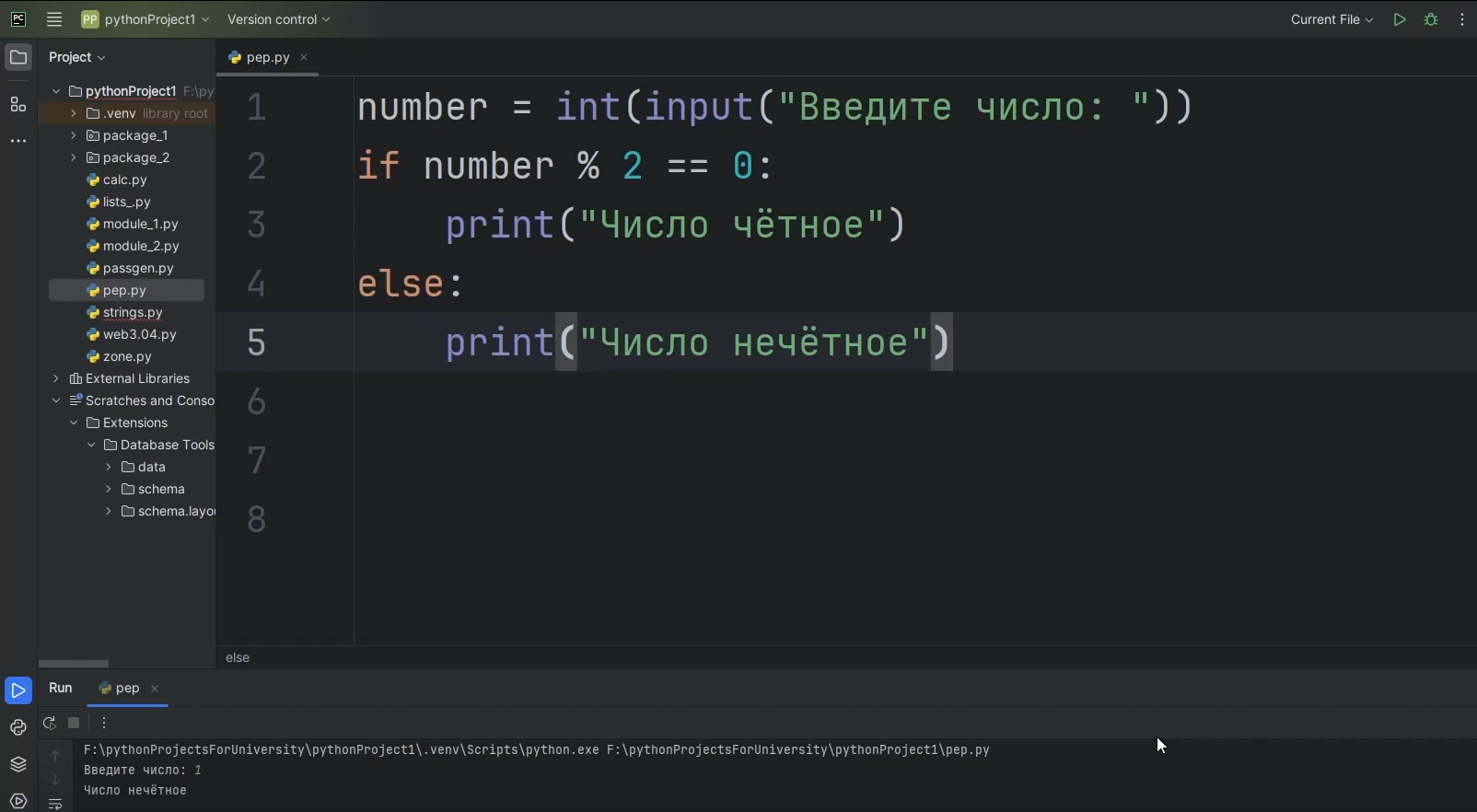


(Рис.1)

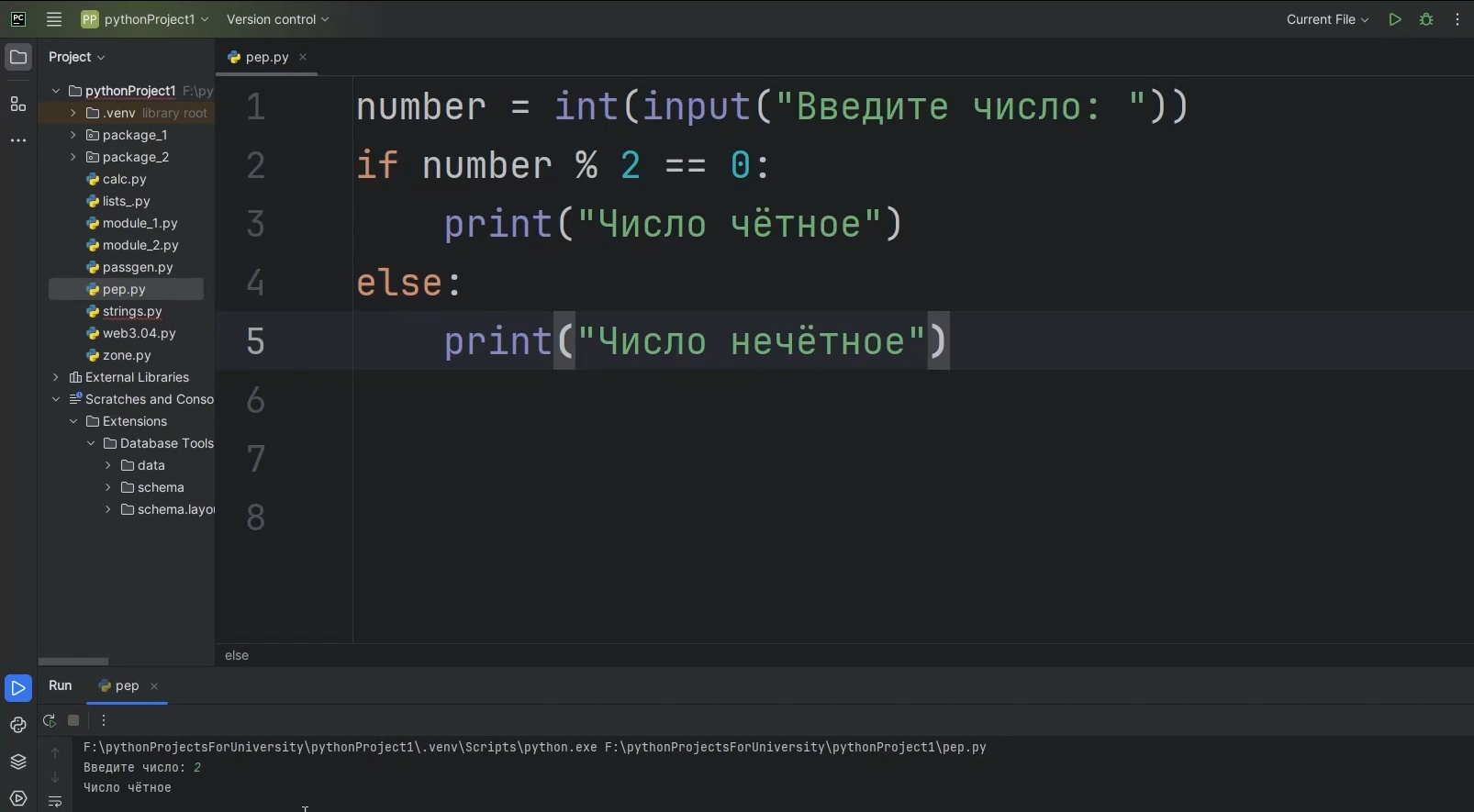


(Рис.2)

Если мы запустим код и введем число 1, то увидим, что число нечетное (Рис.3). А если запустим еще раз и введем число 2, то число уже будет четным (Рис.4).

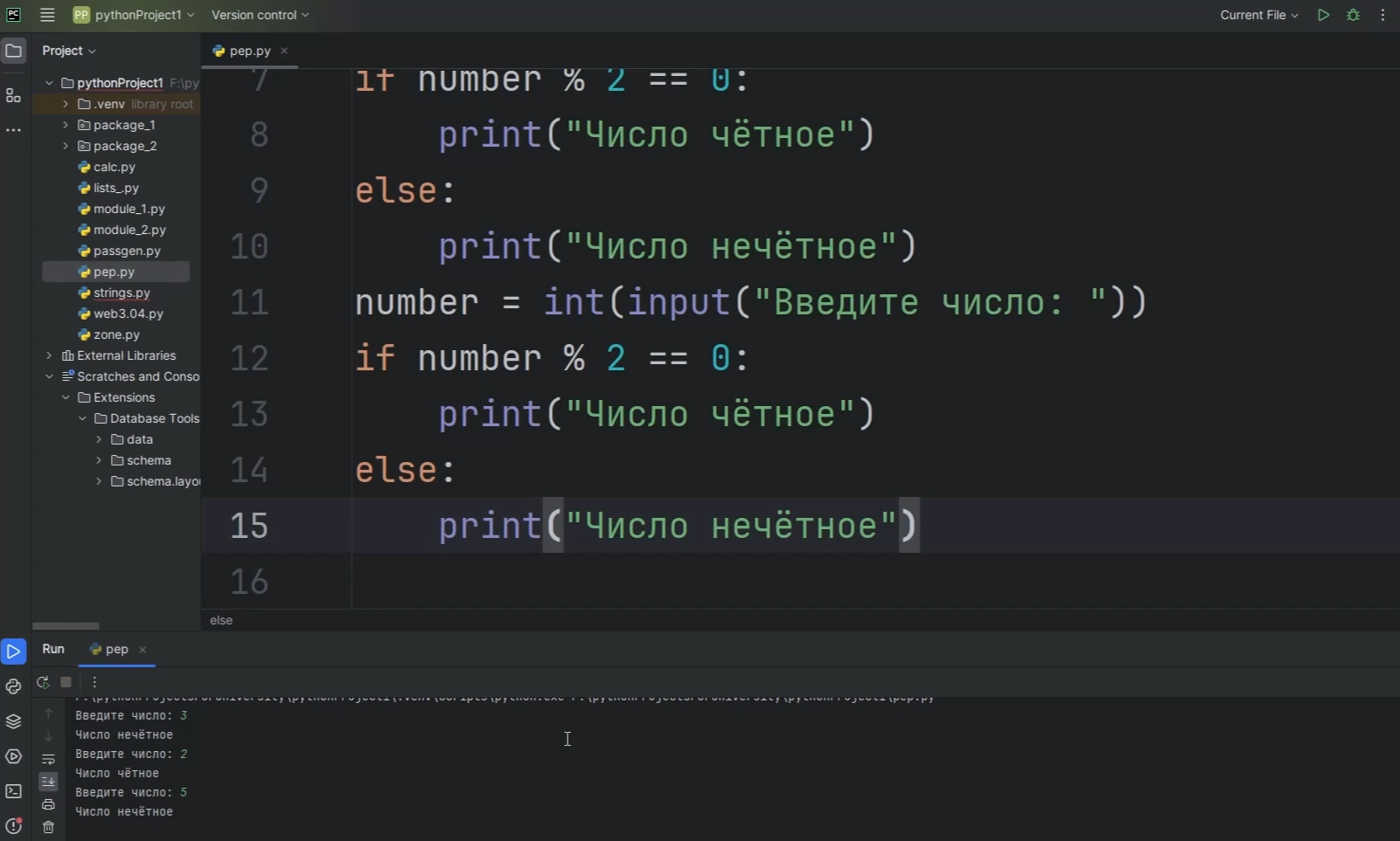


(Рис.3)



(Рис.4)

Но что, если нам нужно таким образом проверить много чисел, и так, чтобы это происходило постоянно? Конечно, мы можем банально скопировать код и вставить несколько раз (Рис.5). При запуске он будет запрашивать у нас число столько раз, сколько мы скопировали код.



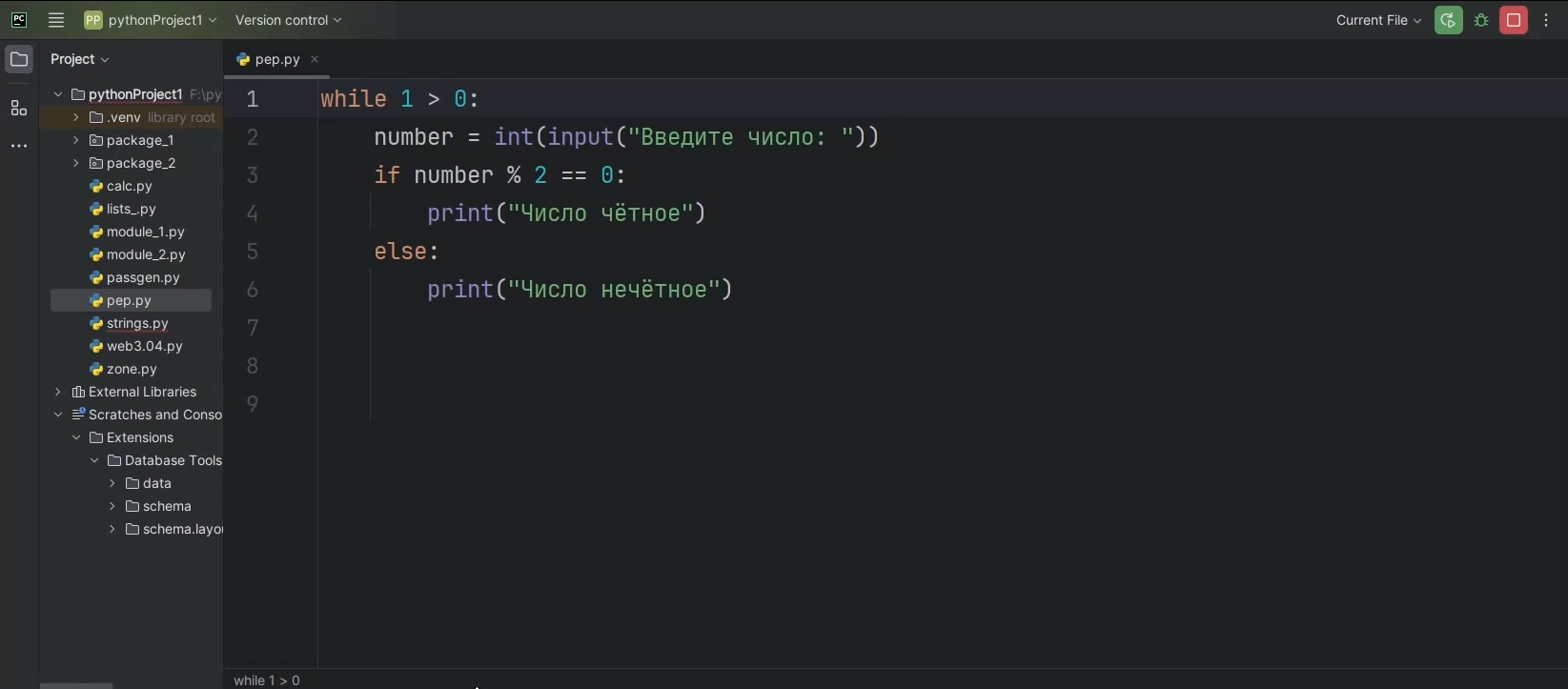
(Рис.5)

Но такой код будет занимать слишком много строк. К тому же существует более правильный способ решения нашей ситуации. И это циклы. Всего их существует два.

### Как работает цикл while?

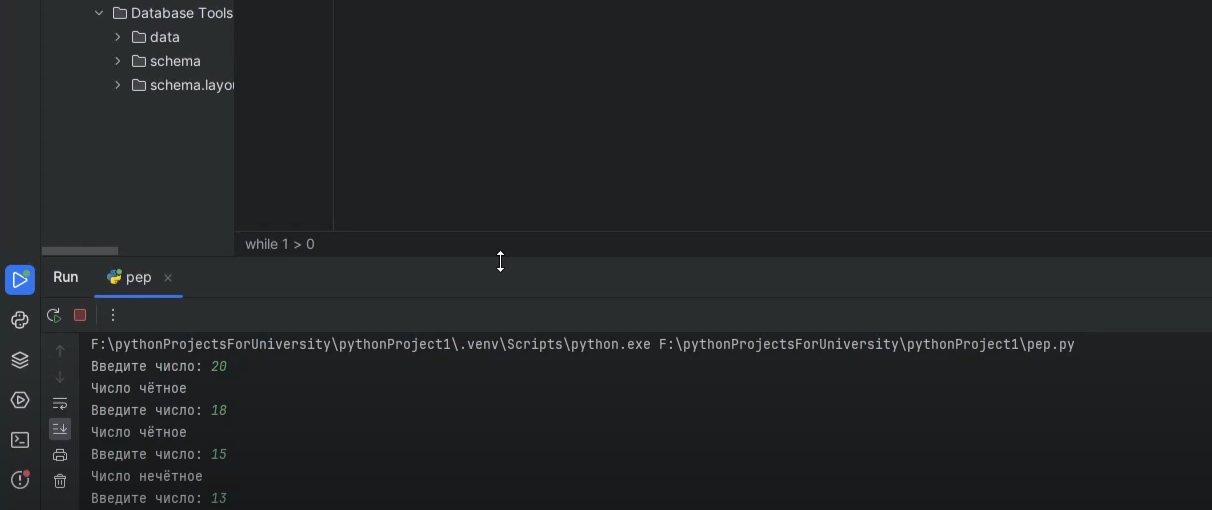
Сегодня речь пойдет о первом из них, цикле **while**. Принцип его работы схож с условием.

Он работает таким образом. Мы пишем ключевое слово «while», после которого пишем условие, например «1 > 0» (Рис.6). Не забываем, что после двоеточия весь код, который мы будем писать после, должен быть с отступом.



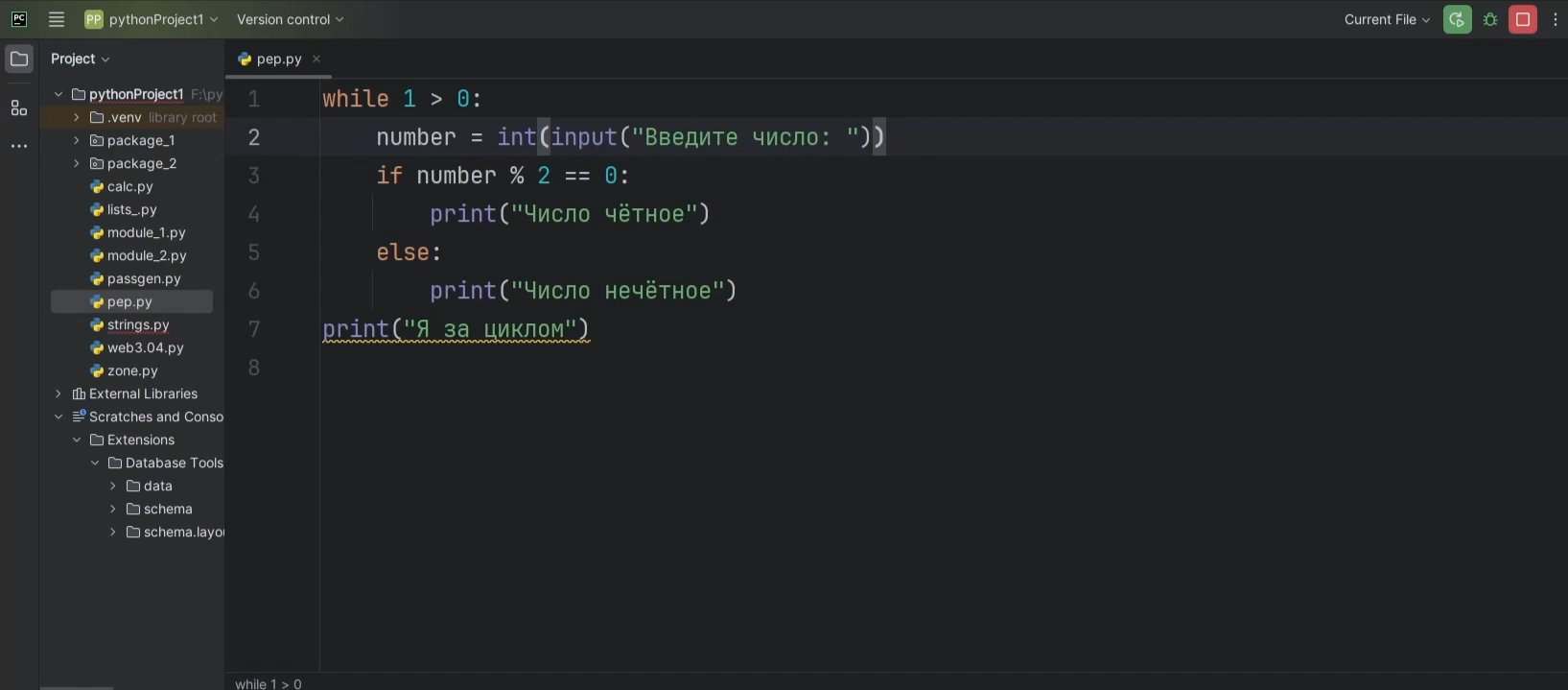
(Рис.6)

Если мы запустим код, то увидим, что нам снова и снова выходит просьба ввести число (Рис.7). Код, написанный с отступом, так и будет повторяться бесконечное количество раз.

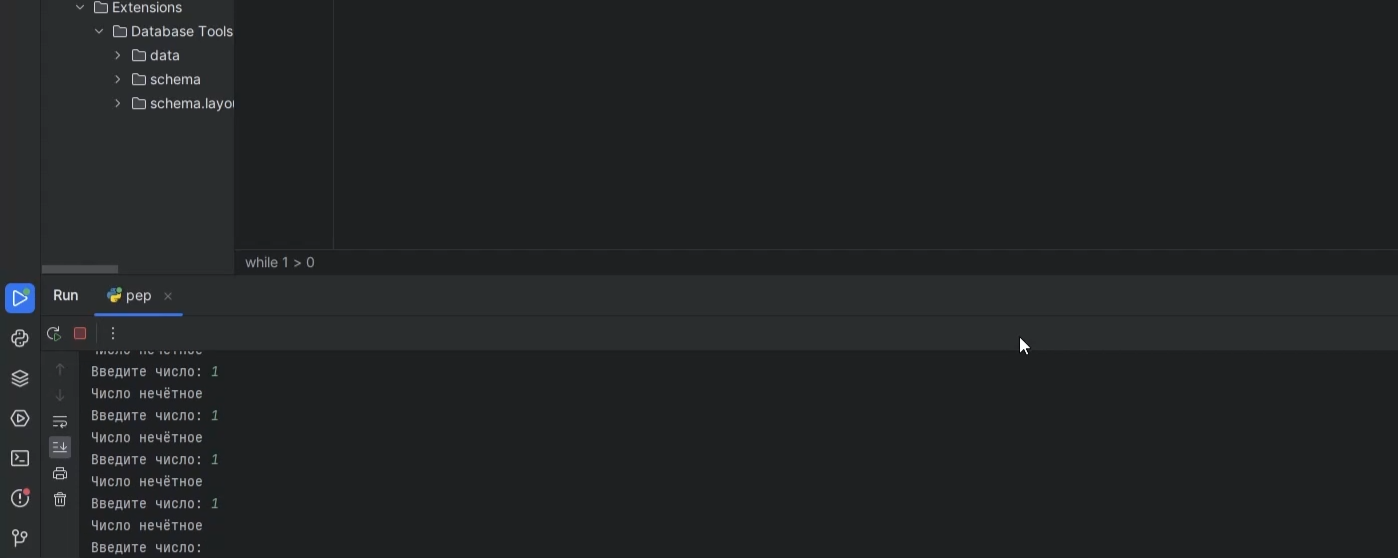


(Рис.7)

Но если мы напишем следующую строчку кода без отступа (Рис.8) и снова запустим код, то при любой попытке ввести число мы не увидим эту строку (Рис.9).



(Рис.8)



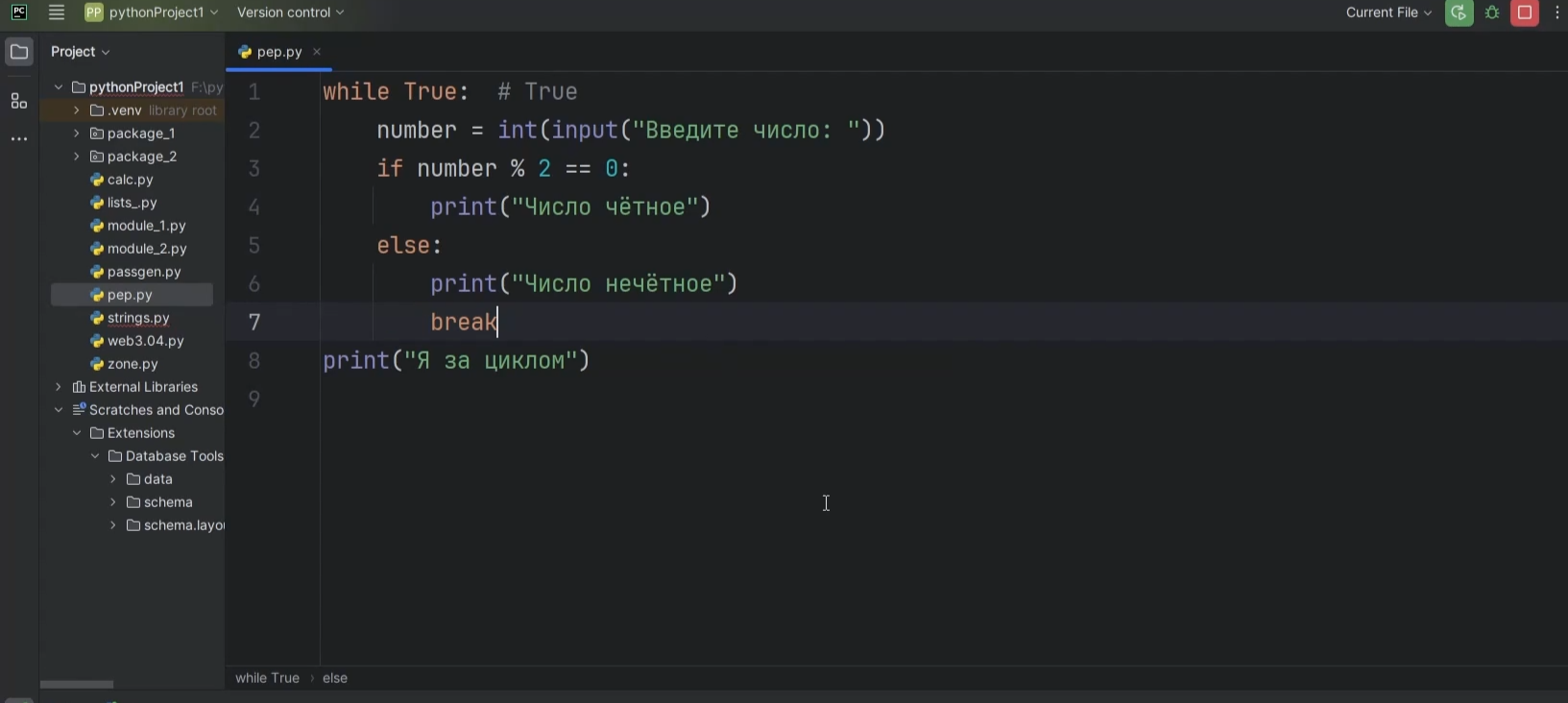
(Рис.9)

Это происходит из-за того, что программа до нее не доходит, она застопорилась внутри цикла, потому что наше условие «1 > 0» истинно и всегда будет выполняться. Из-за этого получается, что наш цикл бесконечный.

Как и говорилось, цикл while работает как условие if, но в отличие от него, while будет повторяться пока условие остается правдивым.

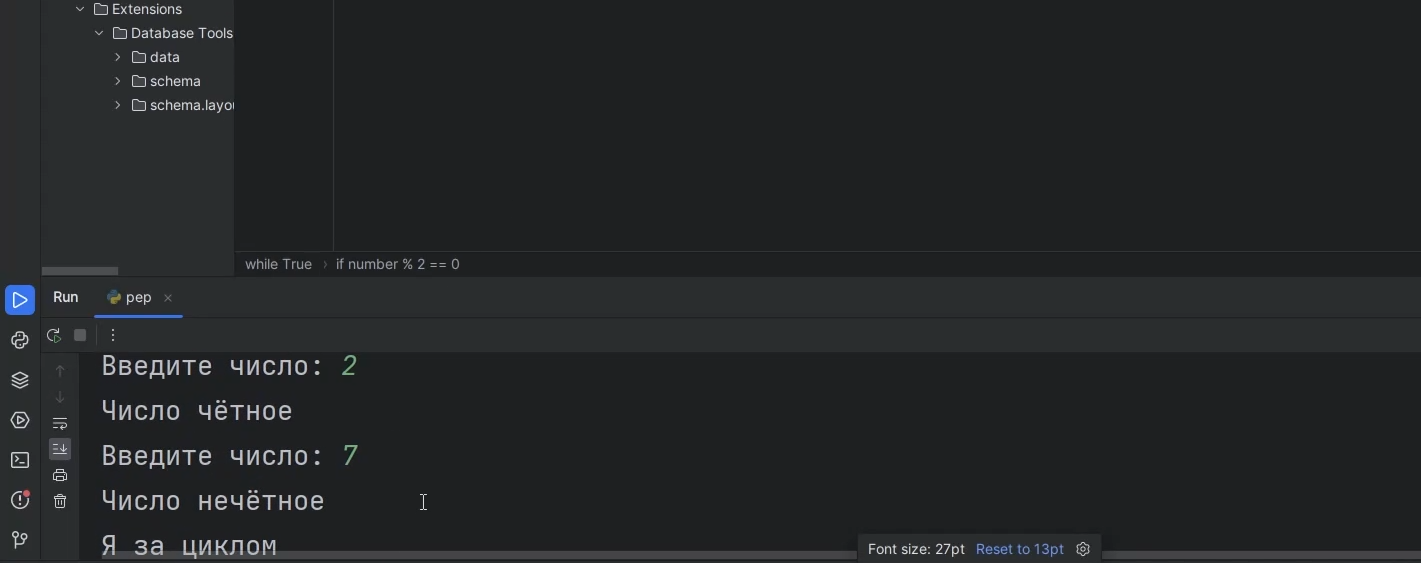
### Операторы break и continue

Как же тогда выйти из этого бесконечного цикла. К примеру, если число будет нечетным, мы можем использовать оператор **break**, который прервет выполнение цикла (Рис.10).



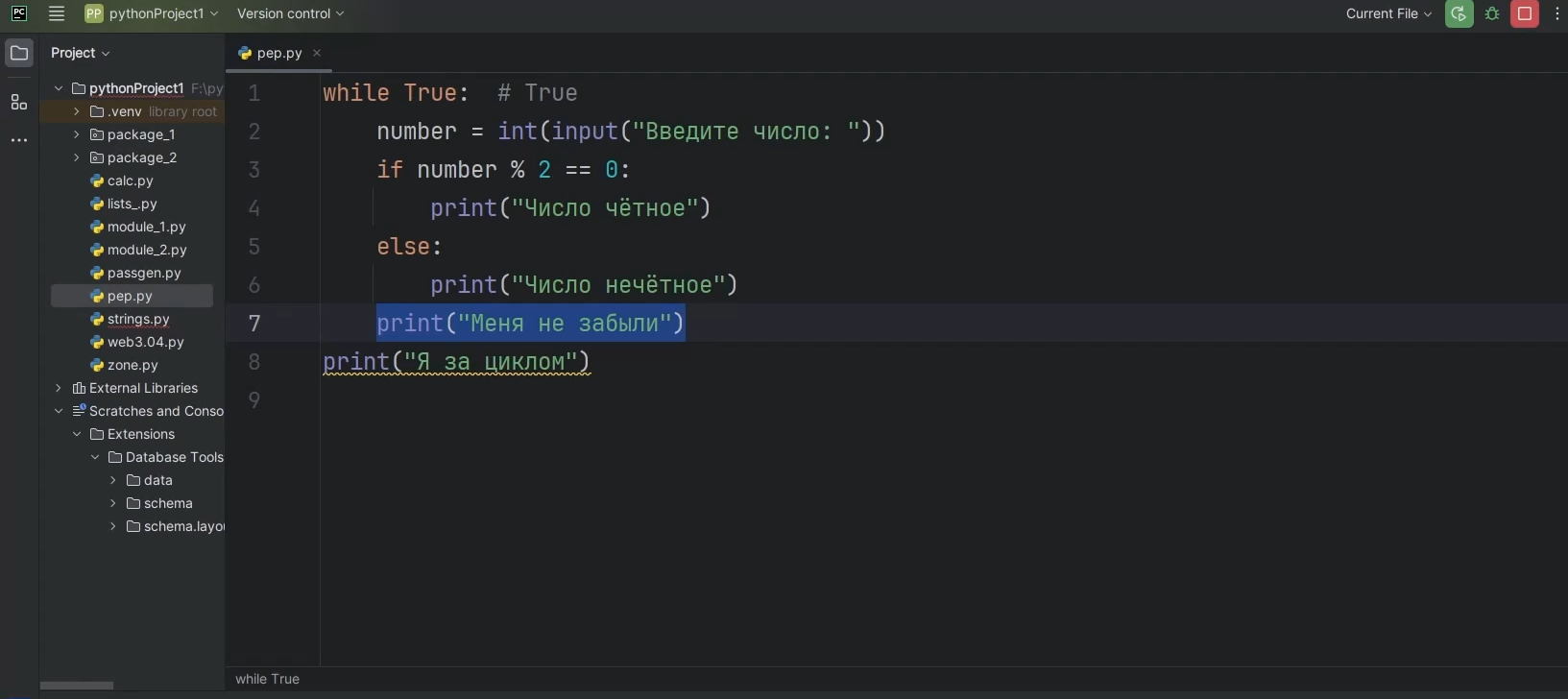
(Рис.10)

Если мы введем четное число, например 2, то цикл так и продолжит беспрепятственно выполняться, но стоит нам ввести нечетное, как он тут же завершится и мы увидим наш конечный код без отступа (Рис.11).



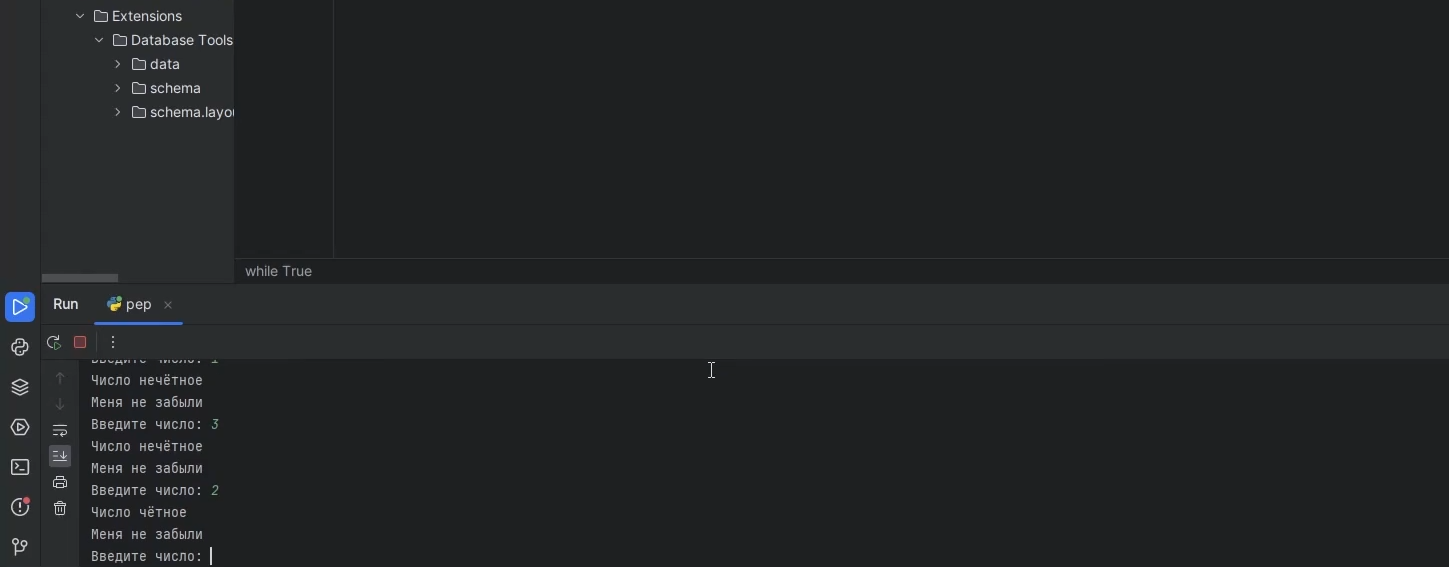
(Рис.11)

Рассмотрим еще одну полезную вещь в циклах. Напишем команду print c отступом в четыре пробела от начала строки, как и команды if и else (Рис.12). Она будет срабатывать после условий.



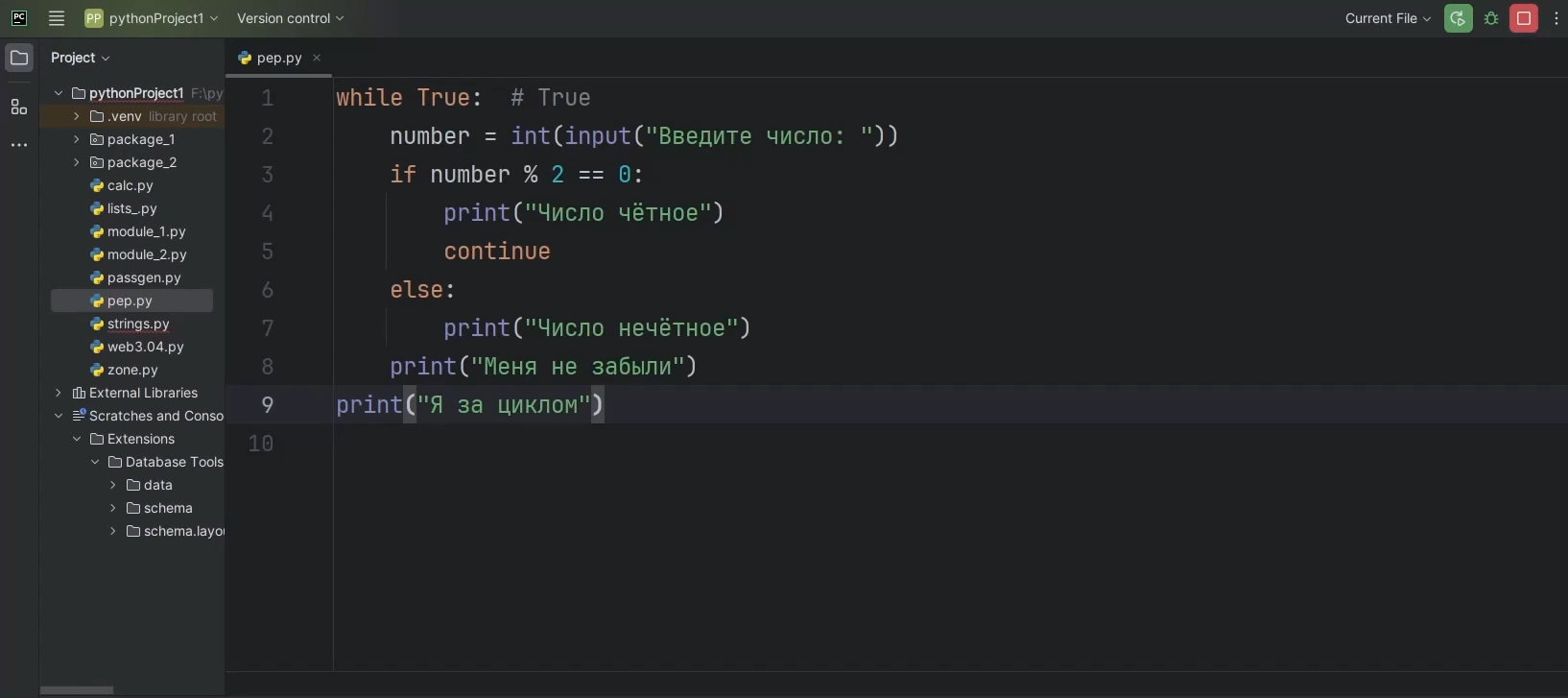
(Рис.12)

Теперь каждый раз, как мы будем вводить число, неважно какое, компьютер будет выводить «Меня не забыли», то есть нашу команду print (Рис.13).



(Рис.13)

Однако, если мы поставим после одного из условий оператор **continue**, например после условия с четным числом (Рис.14), то после запуска программы, если число будет четным, строчку «Меня не забыли» мы уже не увидим (Рис.15).



(Рис.14)



(Рис.15)

Этот код не сработал, потому что оператор continue пропускает выполнение всех оставшихся команды и переходит к следующему повторению цикла.

Операторы break и continue применяются, как к циклу while, так и к циклу for, о котором мы поговорим на следующем занятии.