**2023/10/07 00:00|Лекция. Параметры по умолчанию внутри функции**

## Параметры по умолчанию внутри функции

На это уроке вновь поговорим о функциях. Но мы рассмотрим некоторые дополнительные возможности.

Например, представим такую ситуацию. Мы пишем большую программу. У нас есть какая-то функция, но, пробегаясь по коду программы, мы видим, что каждый вызов функции у нас имеет один и тот же результат. Также, допустим, у нас функция принимающая, и в эту принимающую функцию в момент вызова мы передаём одни и те же значения. Например, функция, которая выводит нам сумму переданных в неё параметров(a+b). Где-то в программе мы её вызываем и видим, что такой вызов у нас встречается несколько раз. То есть при запуске мы видим один и тот же результат(рис.1). Для таких случаев у нас есть возможность задавать параметры по умолчанию, то есть значения, которые будут храниться внутри этих параметров, если мы ничего в неё не передадим.

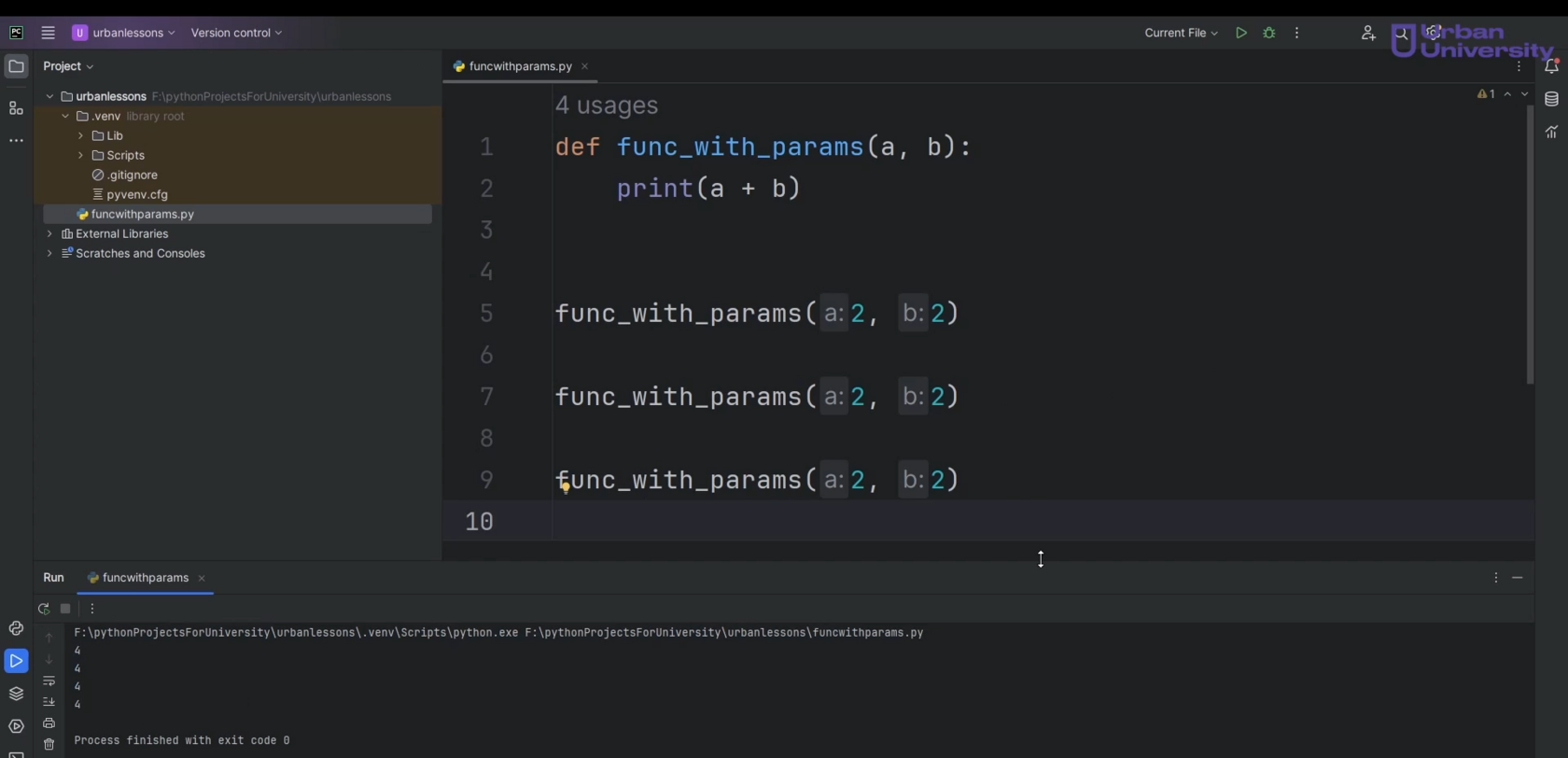


Рис.1

Давайте проверим. Чтобы задать значение по умолчанию, нам нужно в голове функции при определении, после имени параметра поставить оператор присвоения, то есть знак равно и указать значение, которое этот параметр будет принимать по умолчанию. Сделаем, что “а=2” и “b=2”, соответственно нам это даёт возможность вызывать функцию, при этом не передавая в неё никаких значений.

Запускаем и здесь видим 4, 4, 4, 4, то есть мы получили тот же самый результат(рис.2). При этом мы ничего в эту функцию не передавали.

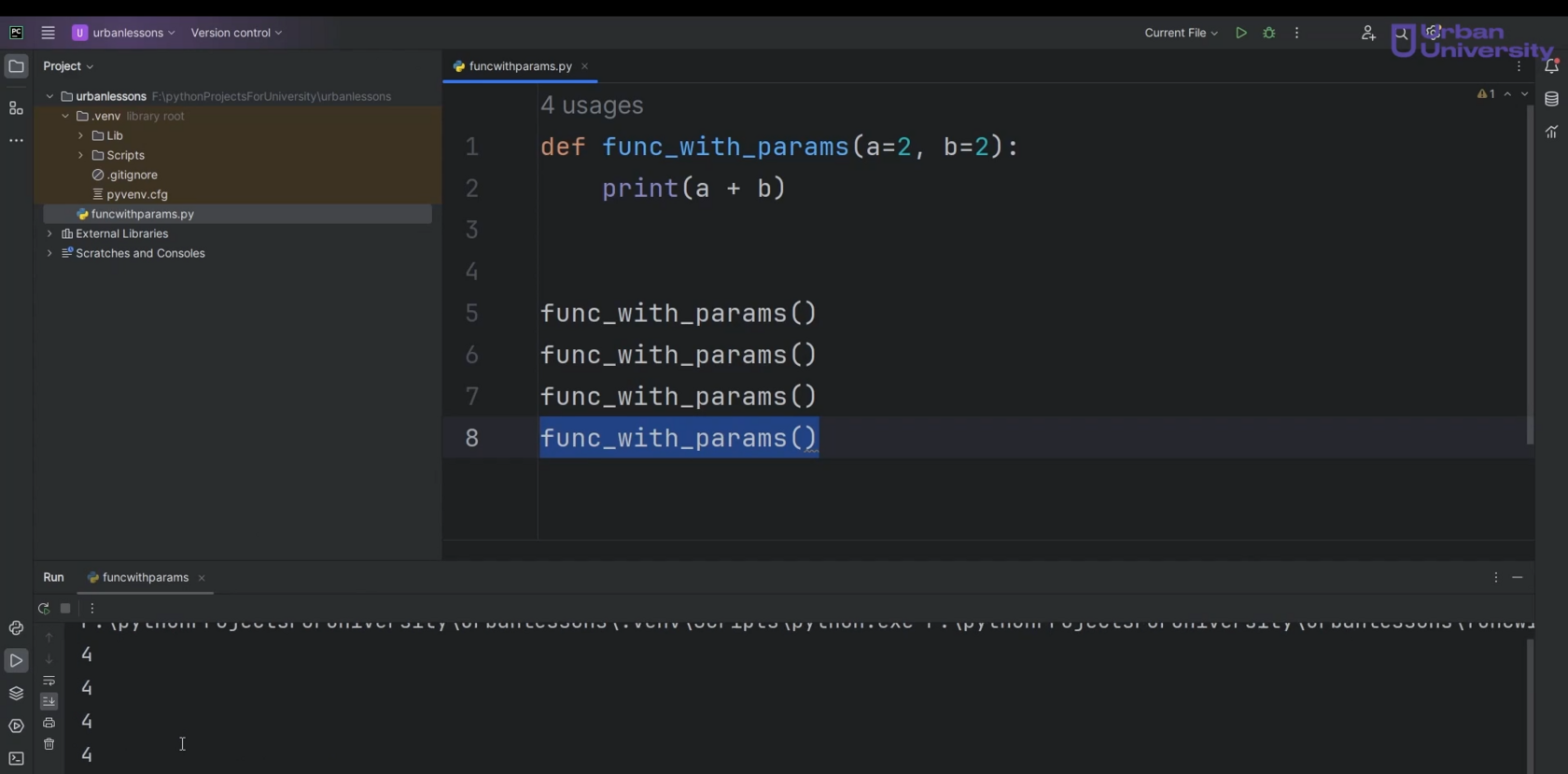


Рис.2

Но если мы передадим сюда свои значения и запустим, то функция уже будет работать не с теми значениями, которые были определены изначально, а с теми, которые в момент вызова мы ей передали(рис.3).

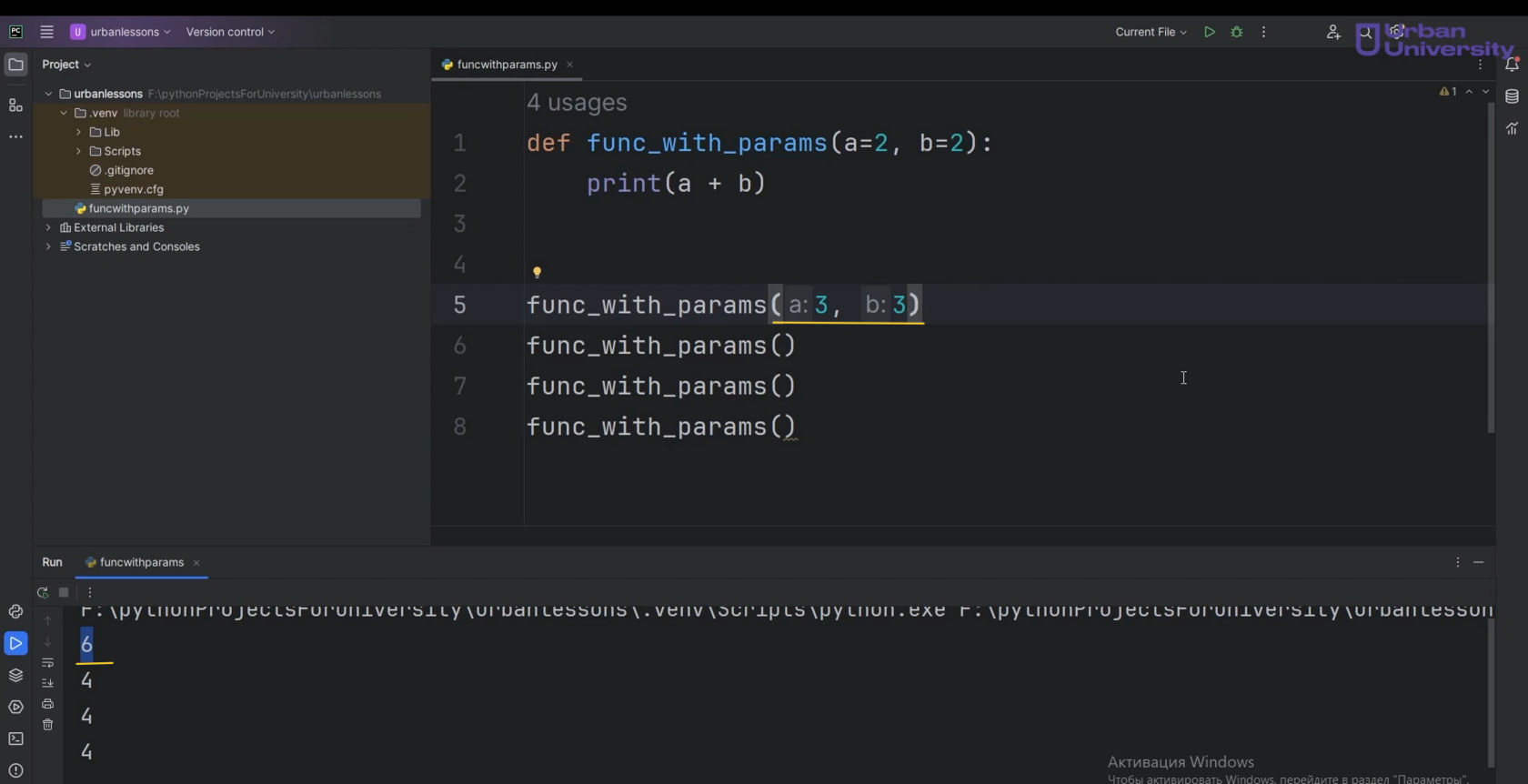


Рис.3

**Это достаточно удобно, но есть ряд важных моментов, о которых нужно помнить:**

* Значение параметров по умолчанию указываются только для конечных параметров. Что это значит? Вот если посмотреть, у нас есть параметр “а” и параметр “b”, они оба имеют значение по умолчанию. Но если указать, что первый параметр принимает значение по умолчанию, а второй - нет, мы получим ошибку: non-default argument follows default argument(рис.4), то есть аргумент без первоначального значения идёт за аргументом со значением по умолчанию.

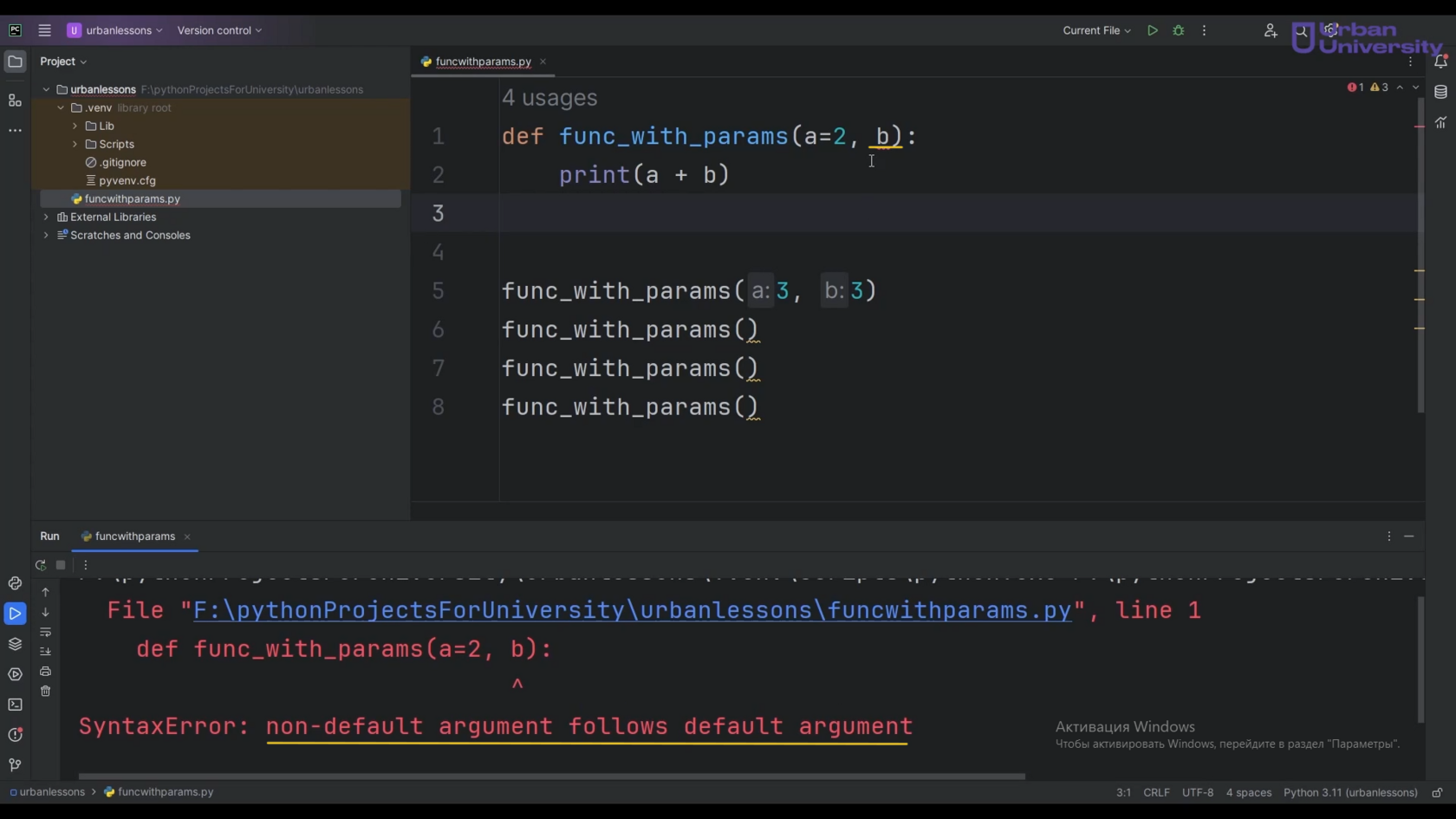


Рис.4

То есть, если переставить, что “а” не имеет знания по умолчанию, а “b” имеет, тогда всё сработает как нужно. Только тут, раз у нас “а” не имеет значения по умолчанию, соответственно, вызовы функции мы должны это “а” передать. Запускаем, видим результат 5, 5, 5, 5(рис.5), то есть первое значение передали, а второе подставилось по умолчанию.

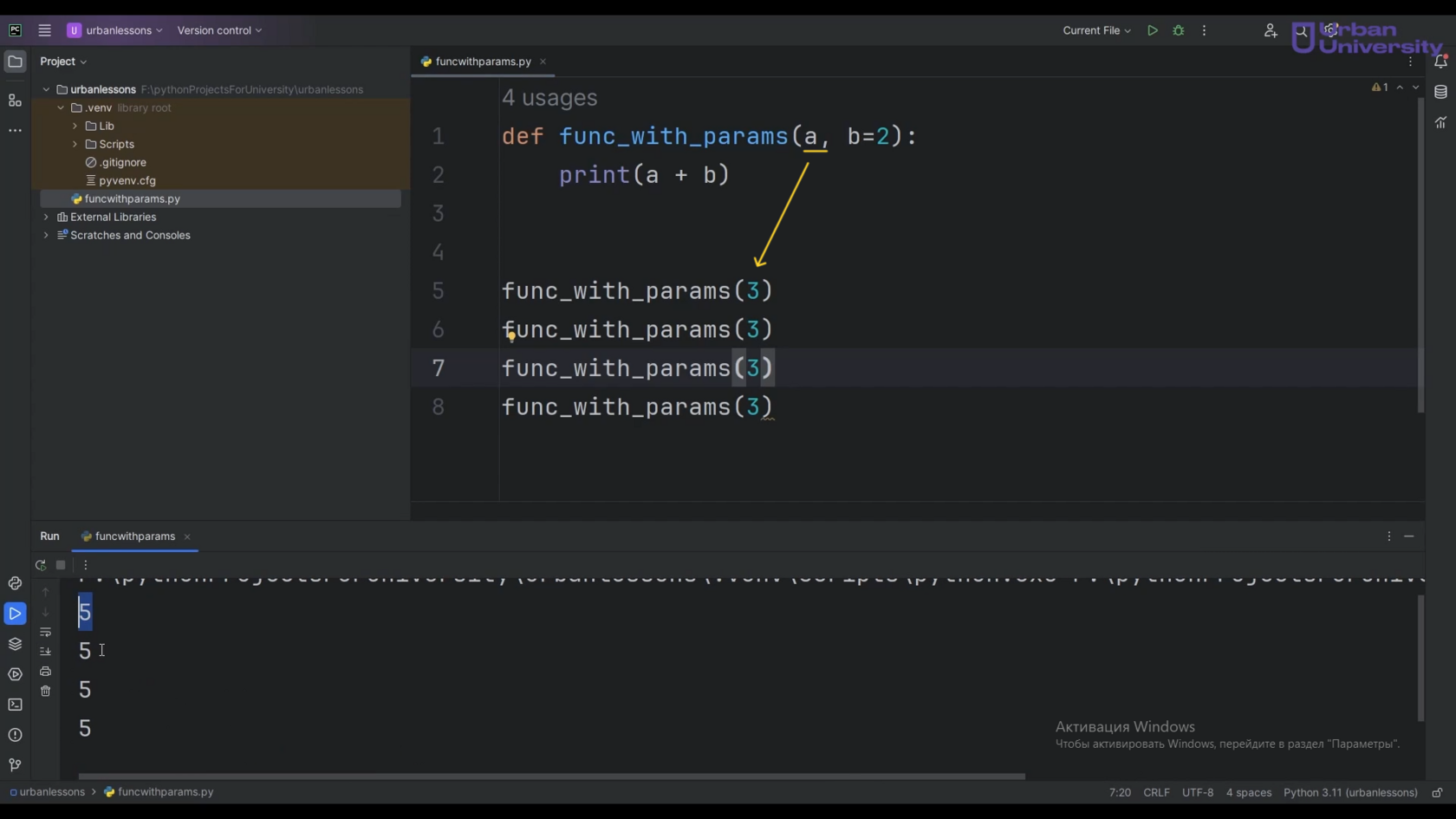


Рис.5

Но это не мешает нам точно также переопределять и второе значение и получать интересующий нас результат(рис.6). Здесь задумка в том то, что мы по умолчанию указываем конечные параметры, те, которые имеют значение по умолчанию, всегда идут после тех, у которых значение по умолчанию отсутствует.

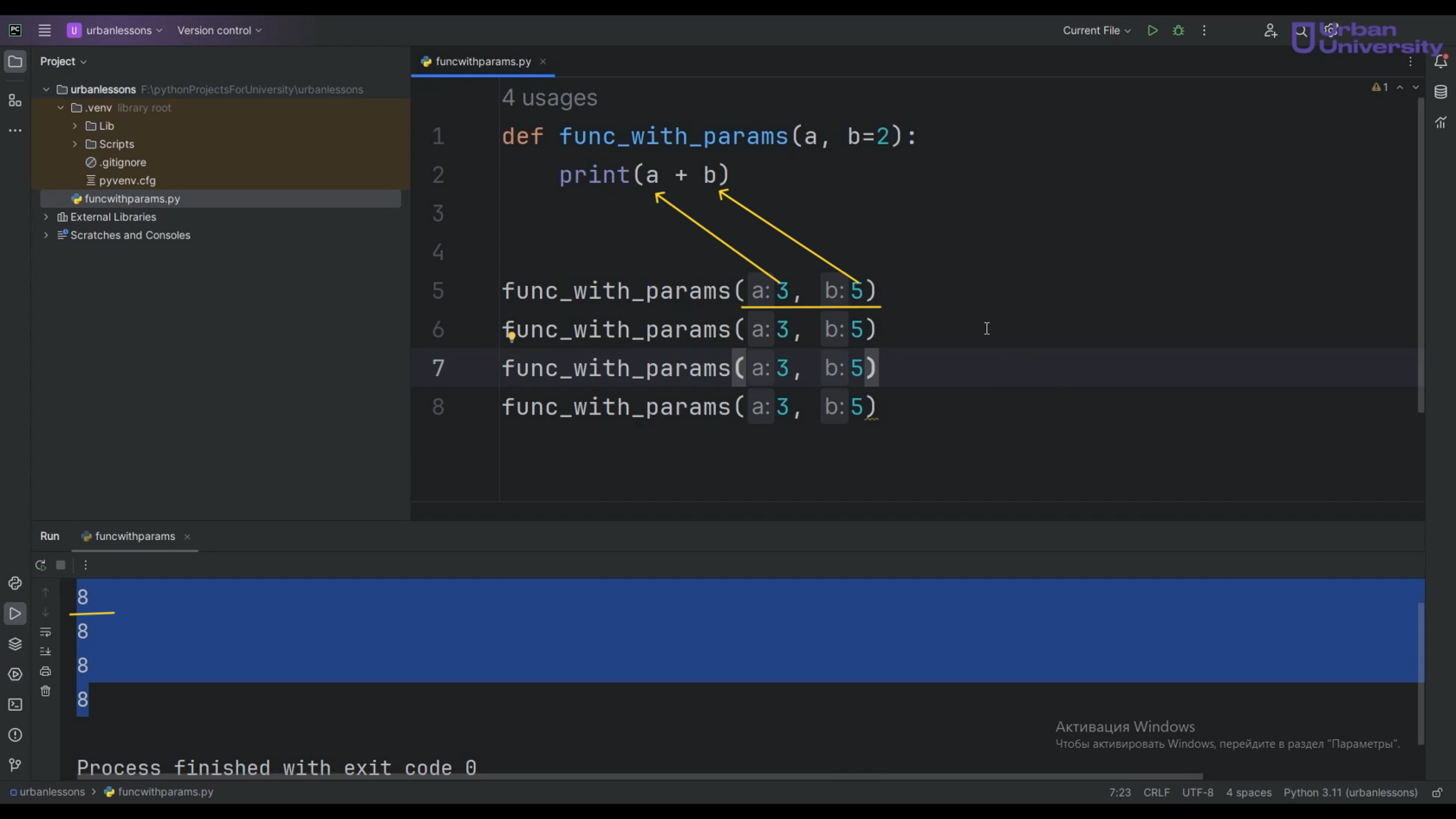


Рис.6

Ну, соответственно, если бы здесь параметров было бы больше, то это работало бы вот таким образом: идёт параметр по умолчанию, параметр по умолчанию, здесь нет параметра по умолчанию. Ошибок никаких нет. Запускаем, видим результат. Мы передали всего два значения: “а” обязательно, потому что у него никакого значения нет, а “b” переопределили по умолчанию. Ответ: 3 + 5 + 3 = 11(рис.7). Всё верно, работает.

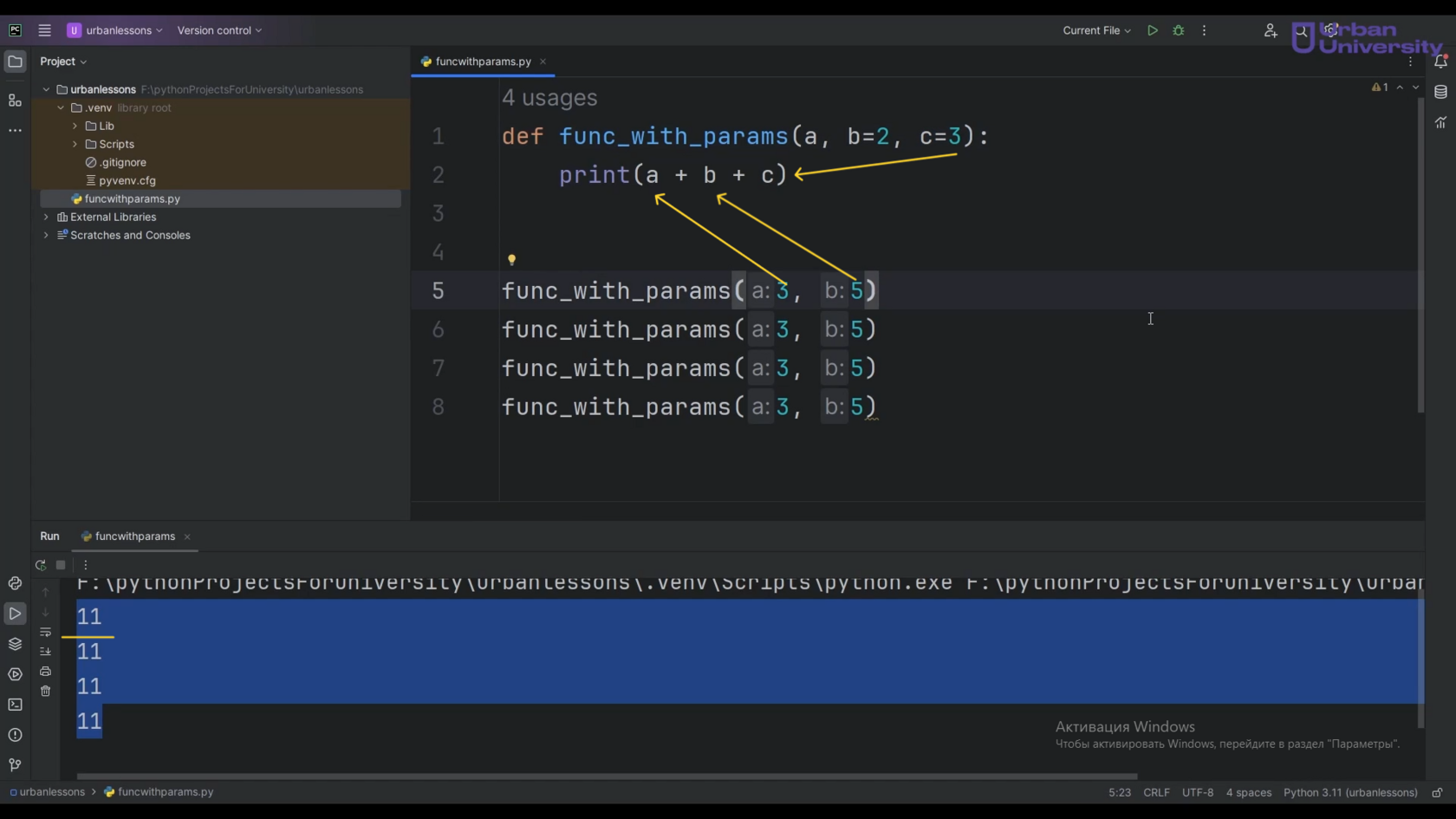


Рис.7

* Когда мы работаем с параметрами по умолчанию, в качестве этих параметров мы указываем неизменяемые объекты. Нужно помнить, что эти значения параметров создаются не в момент, когда мы вызываем функцию, а создаются при её определении. Если мы в качестве какого-то изменяемого параметра точнее в качестве параметра по умолчанию укажем, например, список, то есть изменяемый объект. Давайте немного изменим функцию: она, например, в этот объект “с”, то есть в наш список будет добавлять параметр “а” и выводить этот список. Получается "а” обязательно должны передать. Если запустим, то увидим, что эти значения, которые мы передаем, добавляются не в новый список, а в уже существующий(рис.8). Этот список был создан в момент определения функции func\_with\_params, и каждый вызов просто добавляем, в уже существующий список, новый элемент.

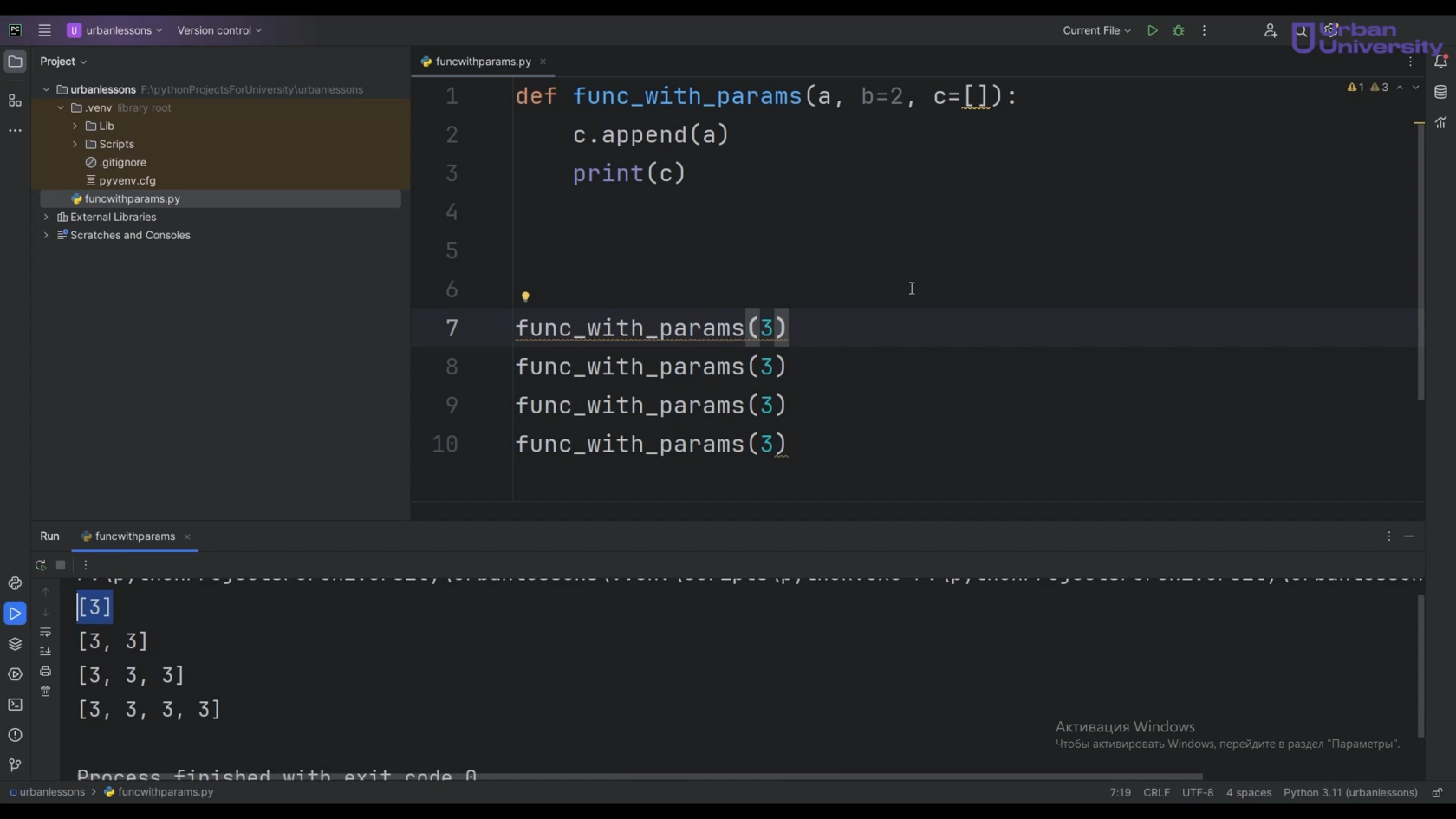


Рис.8

Если нам необходимо избежать такого, например, работать в моменте, мы можем в “с” указать значение по умолчанию “None” и в функции проверить, что если “с”- “None”, тогда мы говорим, что “с” будет списком, и в этот список добавим “а”, потом выведем этот список(рис.9). Запускаем. В таких случаях у нас список “с” будет переопределяться. Если мы будем менять объекты, результат тоже будет, соответственно разным(рис.10).

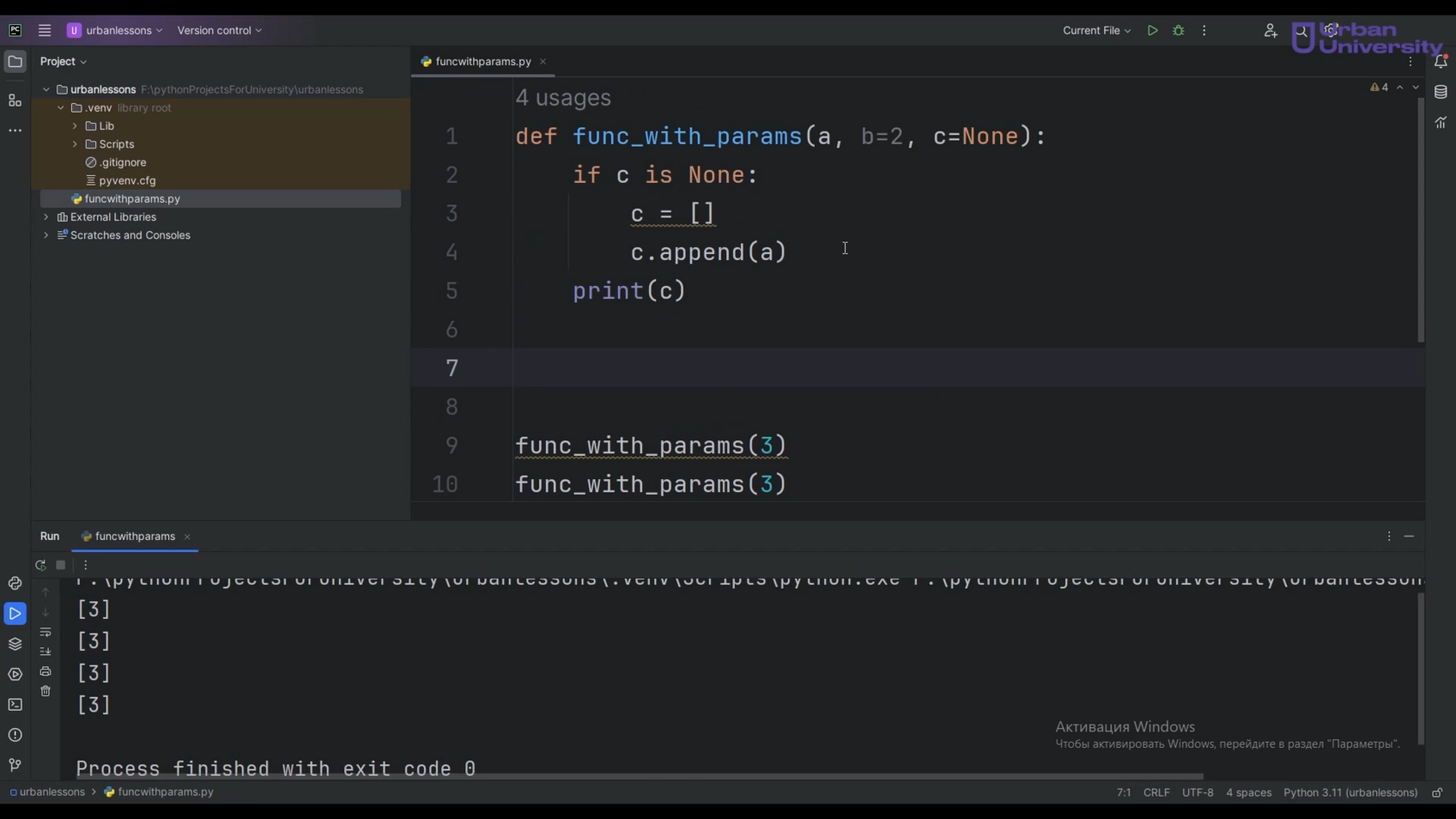


Рис.9

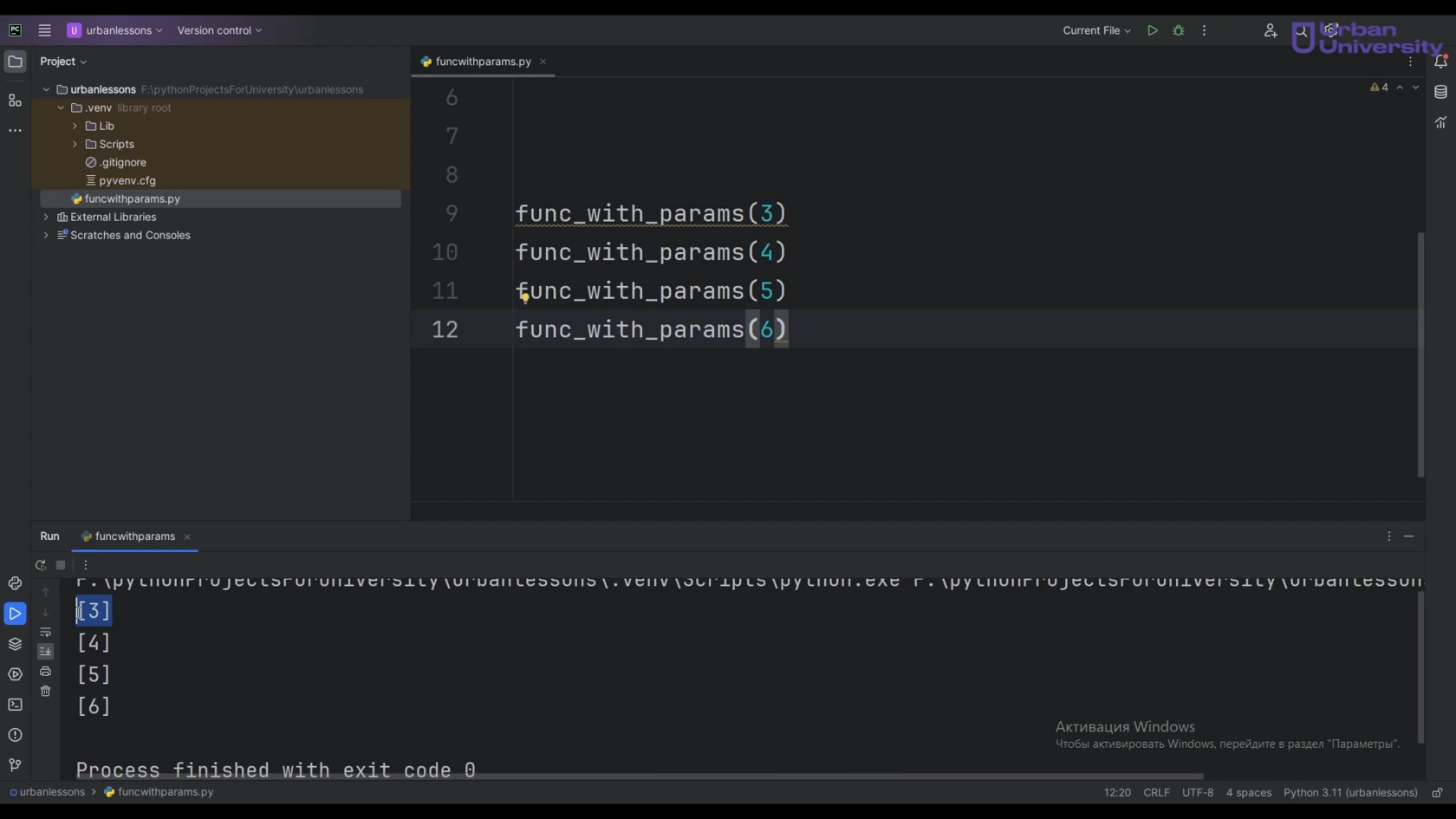


Рис.10

#### Подведём краткий итог.

У нас функция имеет возможность принимать параметры по умолчанию, что позволяет нам в момент вызова не передавать никаких аргументов. Но, если мы работаем с параметрами по умолчанию, мы их задаём либо конечным элементам, либо уж всем, которые присутствуют у нас в функции. И в качестве этих самых значений по умолчанию мы используем неизменяемые объекты.