**2023/10/06 00:00|Лекция. Распаковка позиционных параметров**

## Распаковка параметров.

Поговорим об операторе “звездочка”- \*.

* Одна “звездочка” используется для распаковки и запаковки именно позиционных параметров, которые содержат один элемент: списки, кортежи, множества и т.д.;
* Две “звездочки” используются для запаковки и распаковки именованных параметров(рис.1). В Python такая коллекция одна - словари. В них элементы хранятся парами.

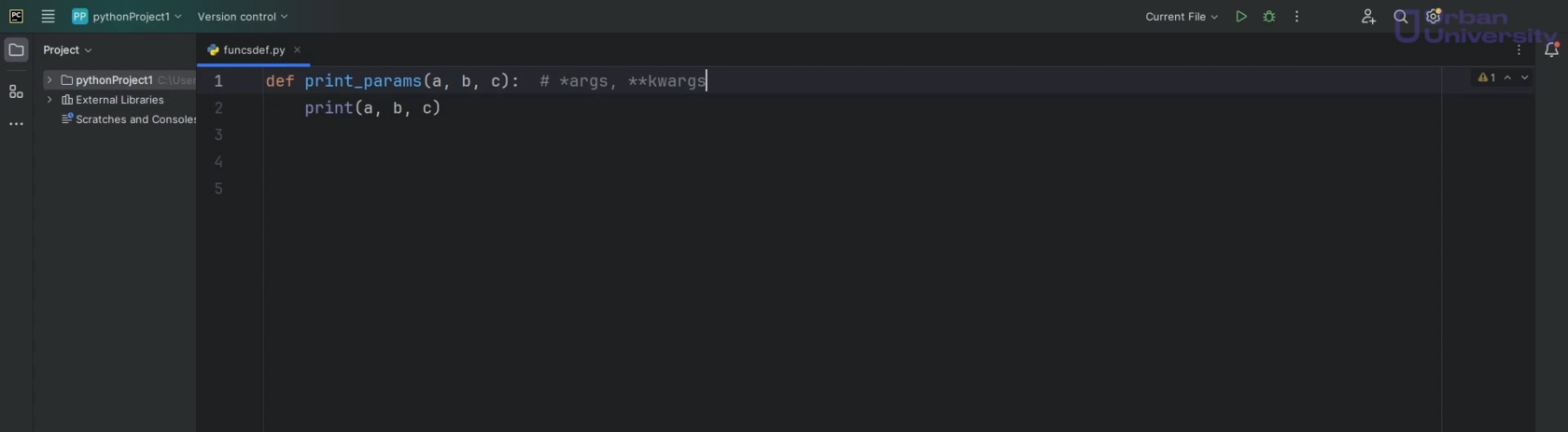


Рис.1

#### Позиционные параметры.

Начнем с**примера.** Нам необходимо функции передать переменное количество позиционных параметров. Мы не знаем сколько их будет. Для этого мы можем указать “\*args”(название можно поменять, у нас params). Все параметры, которые мы будем передавать при вызове функции print(params), у нас будут упакованы в кортеж(рис.2). Но если мы поставим “звездочку” ( print(\*params)), то все параметры будут распакованы(рис.3).

Рис.2

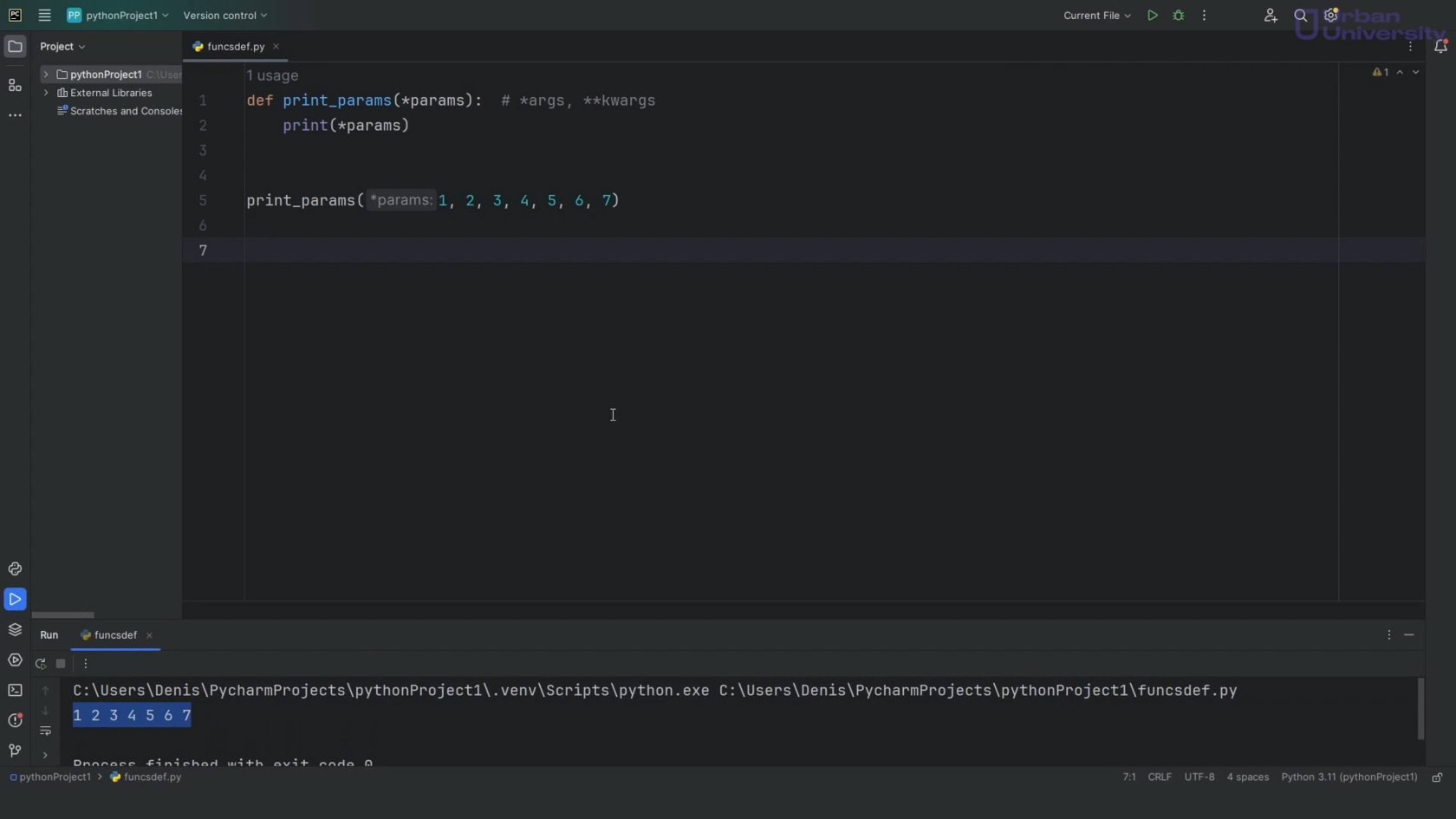


Рис.3

То есть, одну “звездочку” мы можем поставить тогда, когда не уверены или не знаем сколько у нас будет параметров. Как правило, если функция принимает больше 7 параметров, используют оператор “\*args”.

**Рассмотрим**, как работает **распаковка.** У нас есть функция, которая принимает параметры a, b, c и распечатывает их. Допустим у нас есть список list, в котором находятся значения 1, 2, 3. Мы можем передать список целиком, но при запуске мы получим ошибку(рис.4), потому что список встал на место первого параметра “a “, а у нас их три.

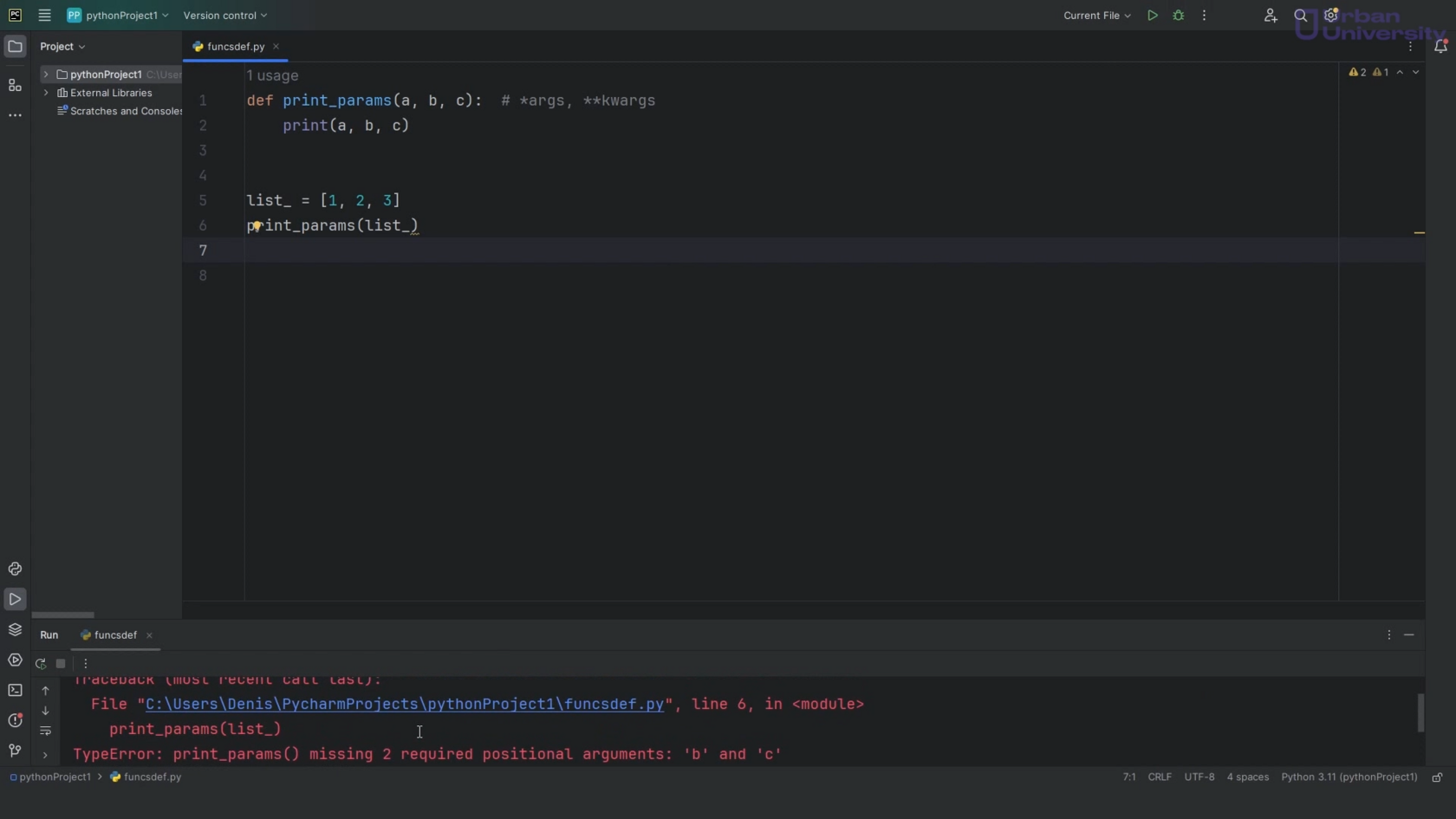


Рис.4

Из предыдущих уроков мы знаем, что сколько функция принимает параметров, столько она и должна получить. Поэтому, мы должны присвоить значение каждому параметру(рис.5).

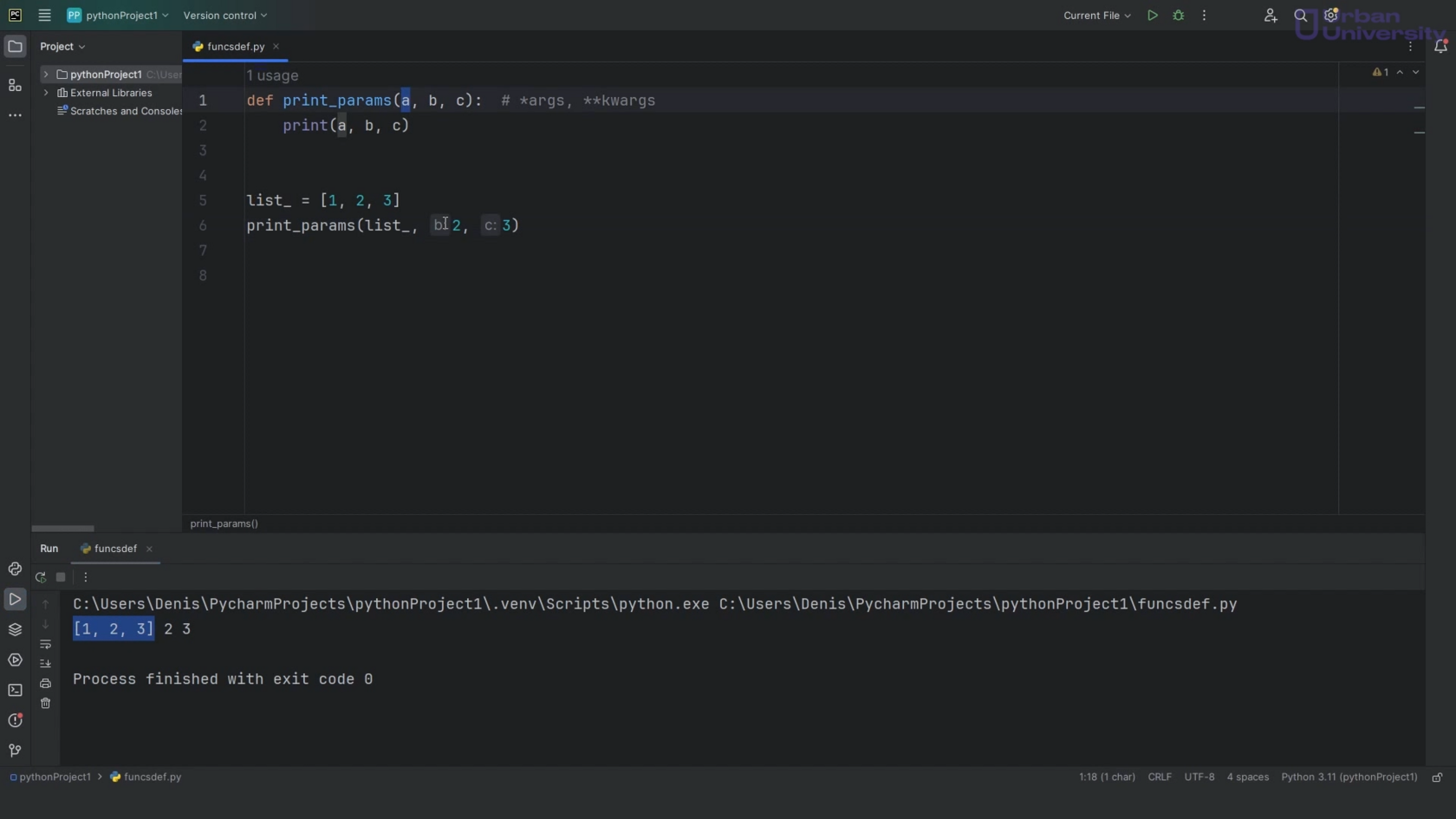


Рис.5

Что сделать, чтобы список list распаковался и значения этого списка встали на место параметров, которые функция ожидает, пока мы ей их передадим?

Для этого мы должны поставить “звездочку” перед списком list в строке параметров и при запуске мы увидим значения из списка, то есть 1, 2, 3(рис.6). Это произошло по причине того, что элементы их списка list\_ = [1, 2, 3] встали на доступные им места вместо параметров a, b, c соответственно, опуская при этом квадратные скобки.

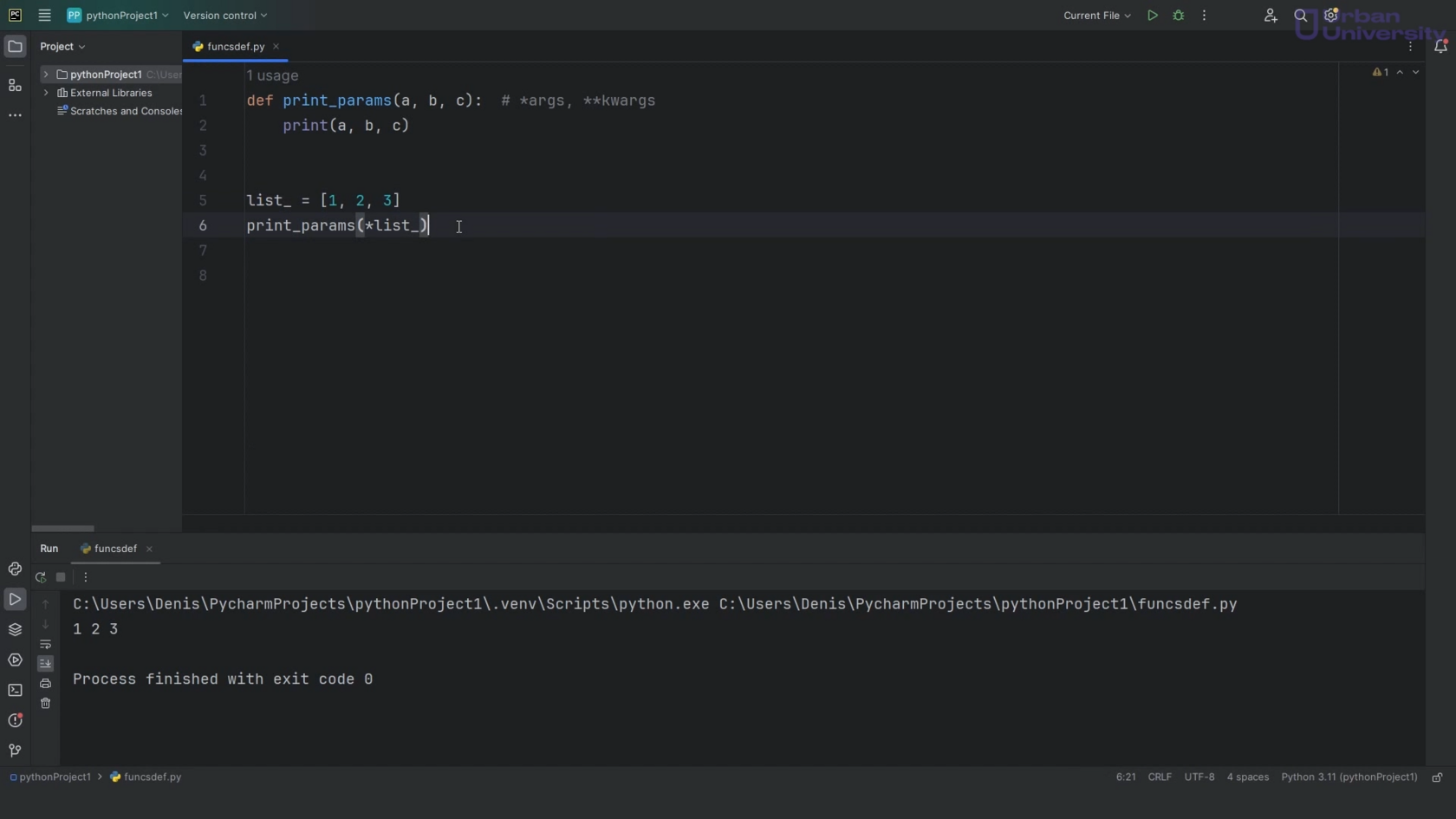


Рис.6

**!Запомните**, что если вы используете данный прием в своих параметрах, то всегда обращайте внимание на количество значений внутри ваших объектов. То есть, если у вас в списке 3 значения, функция принимает 3 параметра, то тогда вы можете использовать список и распаковать его, чтобы передать все элементы в качестве параметров. Если их тут будет меньше, то при запуске выйдет ошибка, так как функция, например, ожидала 3 параметра, а вы ей дали всего 2. При этом всегда можно дополнить, так как в данном примере это позиционные параметры, а не именованные.

#### Именованные параметры.

**Пример.** У нас есть словарь dict и в словаре есть значения с ключами ‘a’ :1, ‘b’ :2, ‘c’ :3. Получается при вызове сейчас, мы должны передать три параметра в эту функцию. Если мы начали использовать в качестве параметров словарь, в котором 3 элемента, то мы поставим не одну “звездочку”, а две. При запуске мы видим 1 2 3(рис.7).

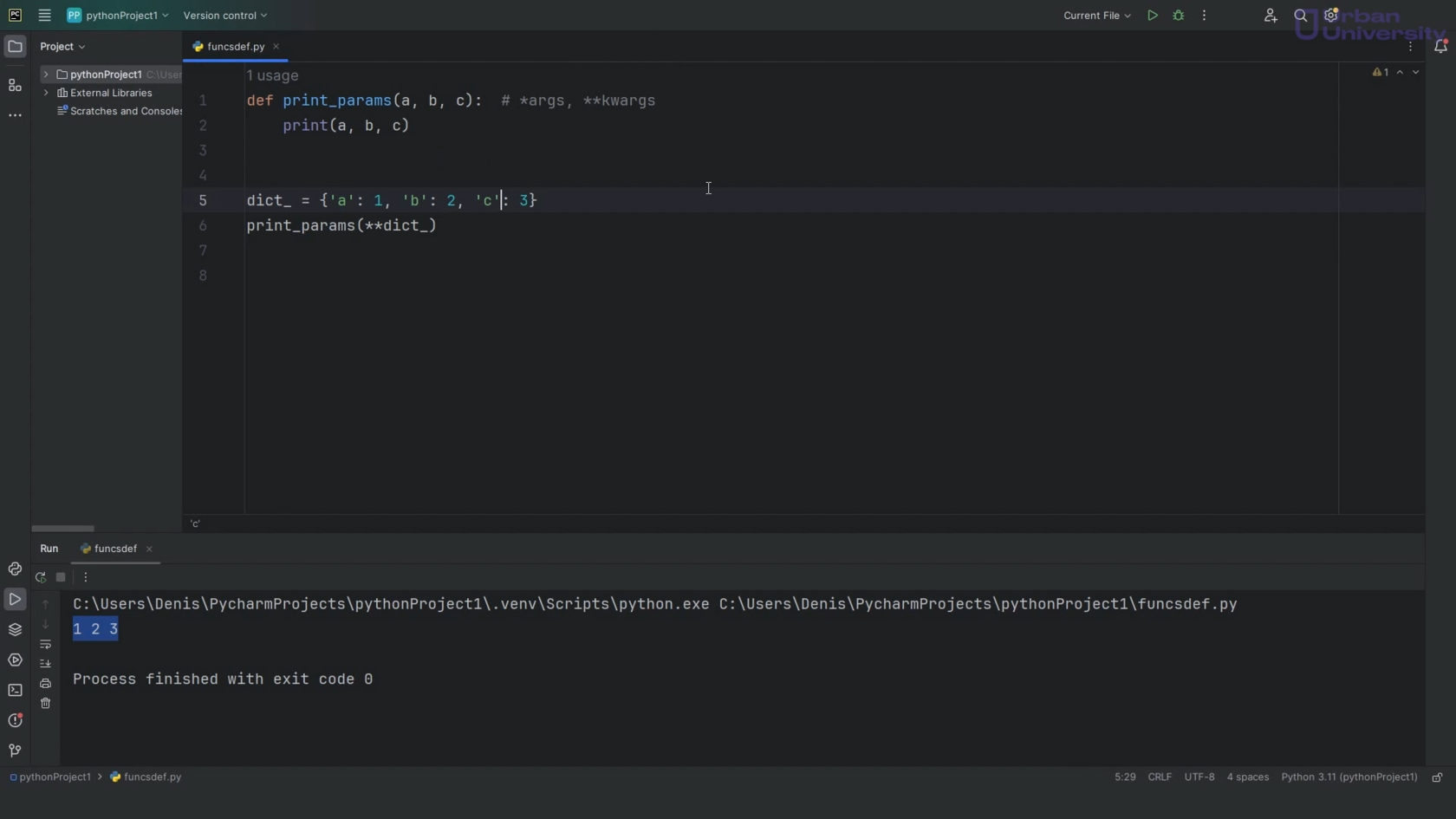


Рис.7

**!Заметьте**, что ключи должны соответствовать названиям параметров функции, иначе выдаст ошибку. Если же вы не хотите придавать конкретики названиям ключей, то вместо a, b, c , мы можем написать “\*\*kwargs”(рис.8). Но когда явно указаны параметры, эти ключи должны соответствовать.

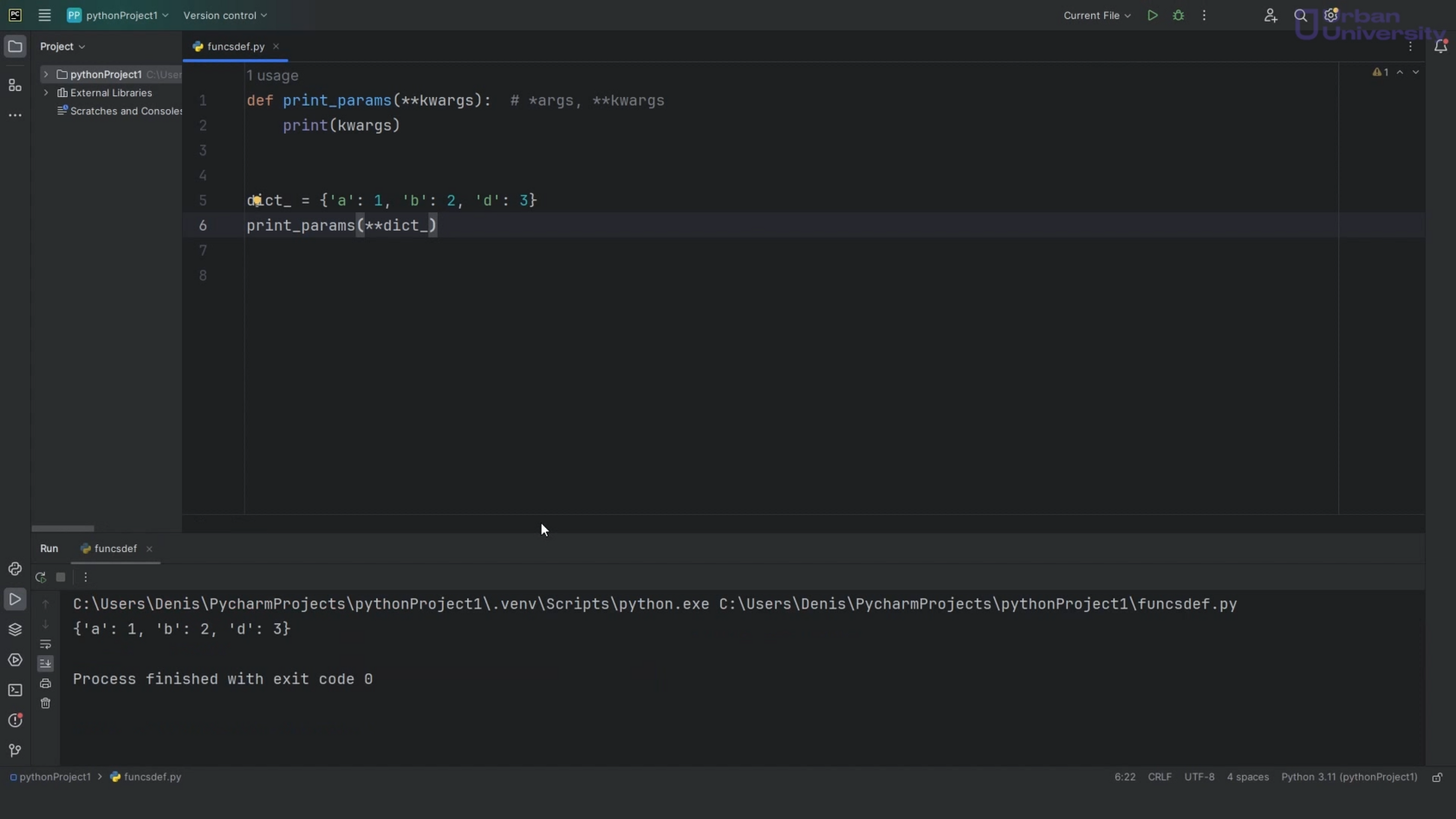


Рис.8

Здесь же происходит то же самое, что происходило и при распаковке позиционных параметров, с помощью списка. Единственное отличие, что здесь у нас именованные параметры. Также “\*\*kwargs” часто используется только после “\*args”. Как правило, именованные параметры идут после позиционных.

Кроме того, раз у нас есть значение параметров, мы можем, например, с помощью цикла for вывести сами ключи(рис.9). Можно пройтись по ключам и значениям этих параметров, только используя метод items для словарей, и вывести ключ и значение(рис.10).

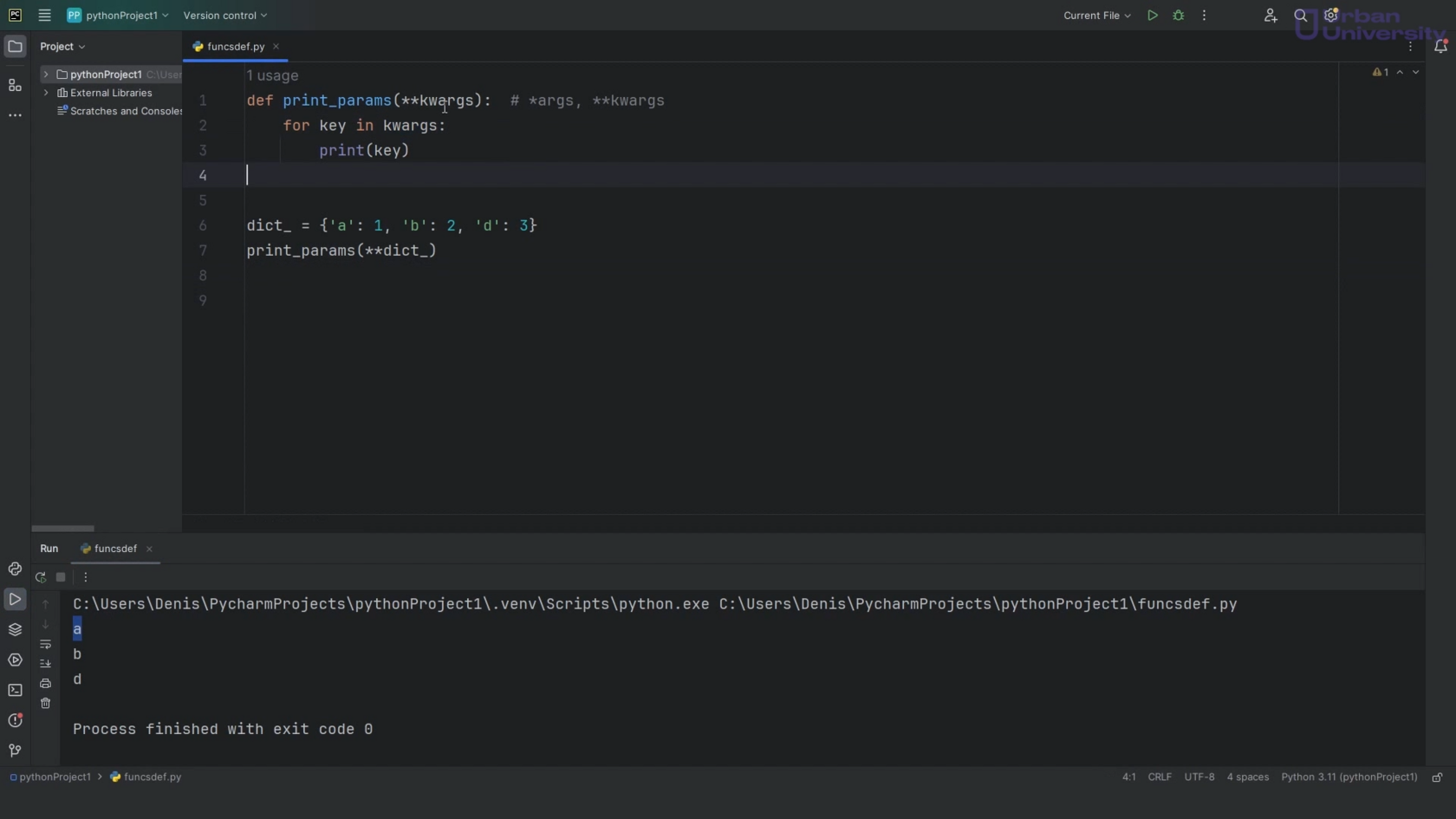


Рис.9

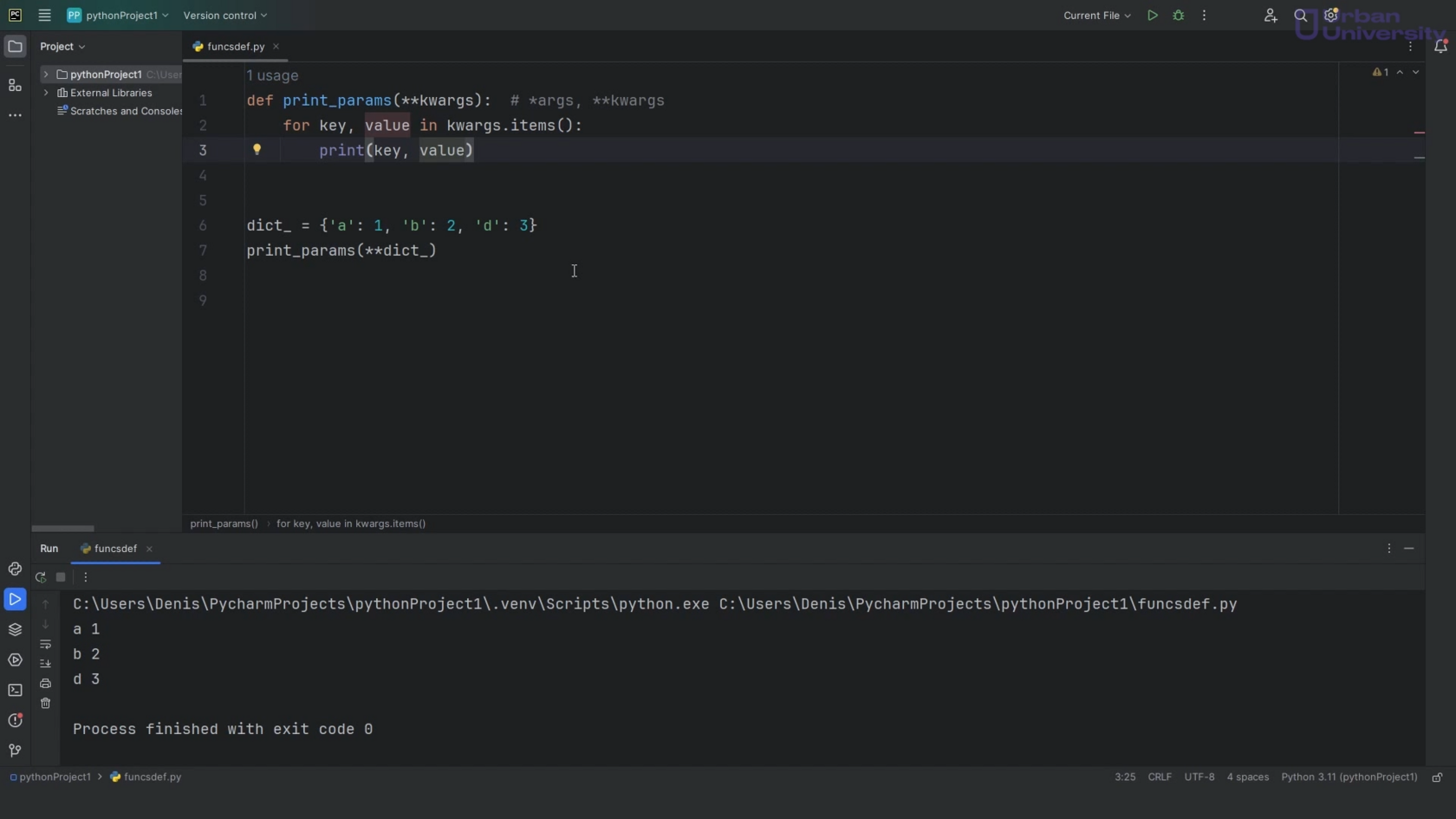


Рис.10

Все эти вызовы мы также можем комбинировать между собой, то есть можно еще совместить распаковку, например, с помощью списка и с помощью словаря. Но важным моментом является то, что всегда необходимо следить за тем, сколько элементов у вас принимает функция и сколько параметров она ожидает.

Функция print\_params в примере ожидает всего 3 параметра, это - a, b, c, значит их мы и должны передать. Если у нас есть и список, и словарь, то мы должны проследить за тем, что в списке у нас будет два значения, а в словаре одно. То есть, мы берем список, распаковываем его, при этом 1 у нас занимает место “a”, 2 займет место параметра “b”, а “c” мы возьмем из словаря, который мы распакуем. При запуске мы видимо ответ - 1 2 3(рис.11).

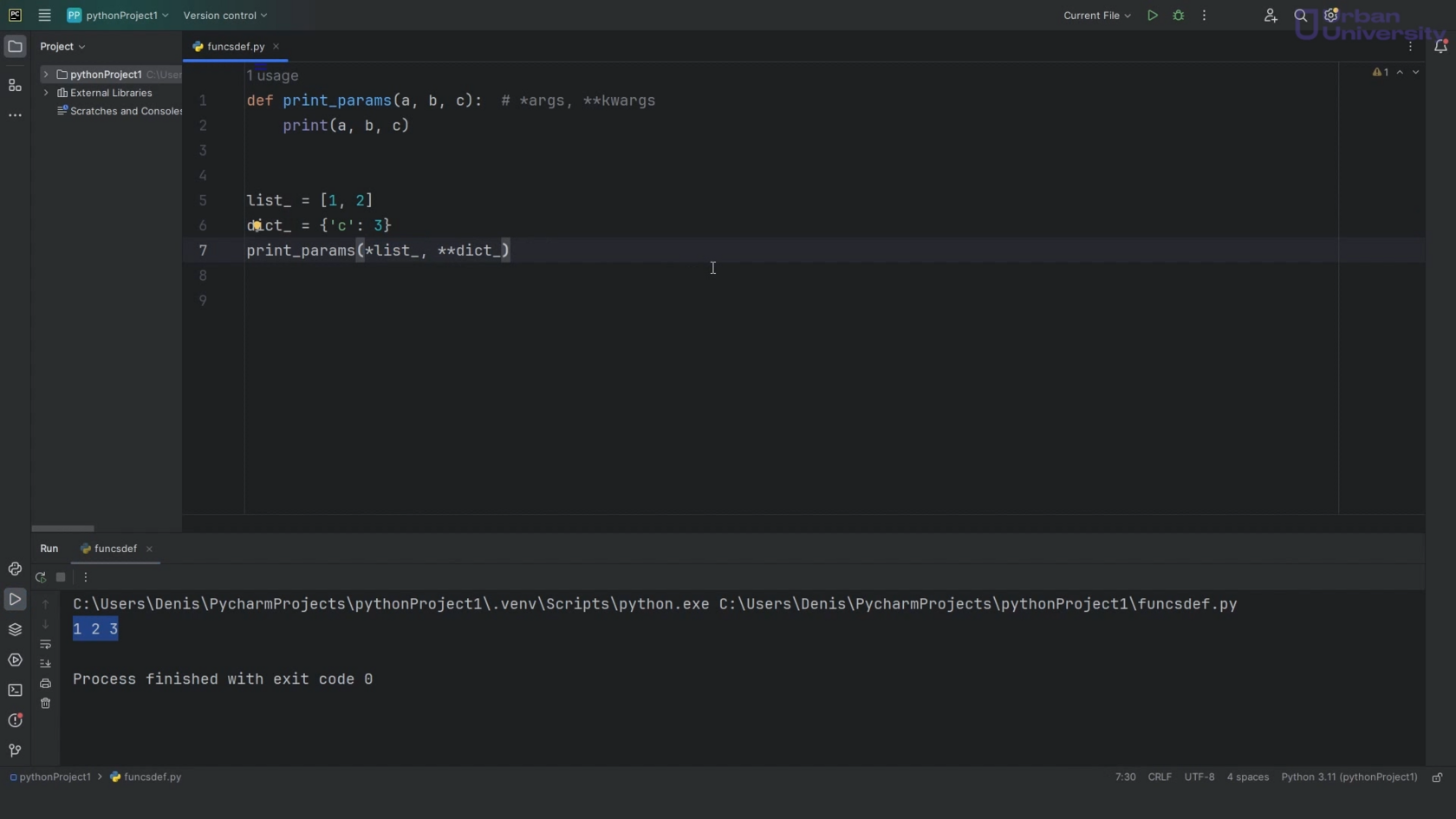


Рис.11