B4

В этом модуле вы узнаете:

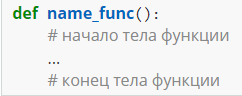
Что такое функция и как её определять.

Как вызывать функции с разными параметрами, в том числе запакованными.

Что такое рекурсия, как и для чего её использовать.

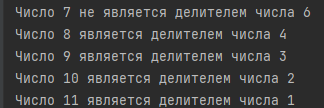
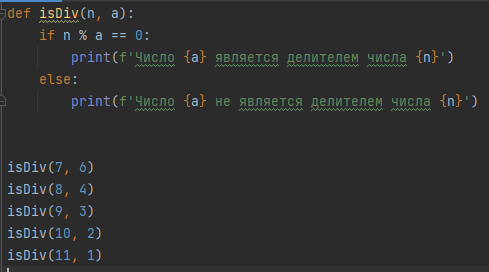
Что такое итераторы, генераторы и декораторы

**Функциональное программирование в Python. Понятие функции**



Если функция имеет аргументы, но их не передать или передать большее количество, то будет ошибка.



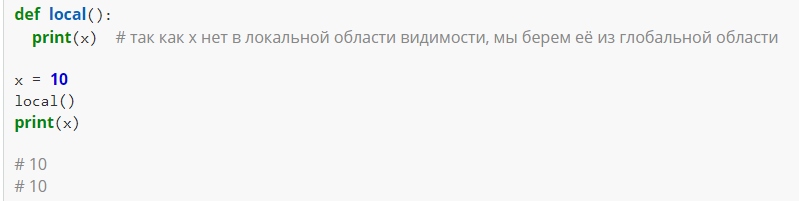


Любая функция в **Python** что-то возвращает. Если не указан return, то функция вернет **None.**

**Области видимости, запакованные переменные, аргументы по умолчанию и рекурсия**

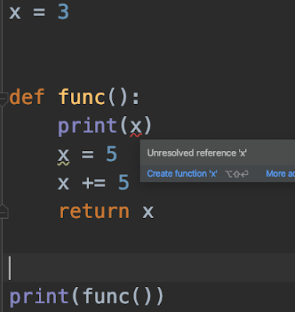
Области видимости определяют, в какой части программы мы можем работать с той или иной переменной, функцией и т. п., а от каких они «скрыты». Крайне важно понимать, как использовать только те значения и переменные, которые нам нужны, и как интерпретатор языка себя при этом ведет.

В python3 есть 3 области видимости. **Локальный, глобальная, нелокальная.**

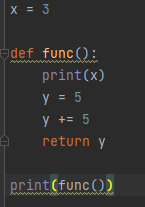


Тут в функцию local никакие переменные не передавались, тем не менее функция увидела переменную и напечатала ее. Просто сначала она проверил свою область видимости и когда не нашла там переменной, то обратилась уже к глобальной области видимости.

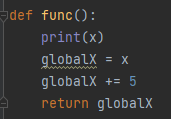
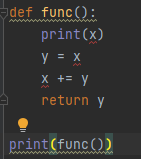
А такой код выполняться не может.



Так как в теле функции найдено объявление переменной, но осуществлена попытка распечатать ее до объявления. Вот как этот код можно исправить.

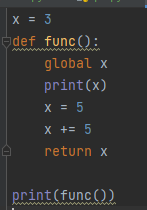
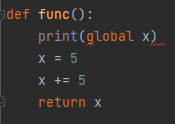


**Глобальные переменные внутри функций изменять нельзя.**

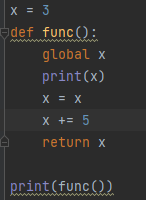
****

**Но обращаться к ним можно.**

Еще можно явно указать, что переменная является **глобальной**, тогда и проблем с названием не будет. Но нельзя сделать это внутри функции **print,** как минимум



**Это значение можно даже присвоить.**

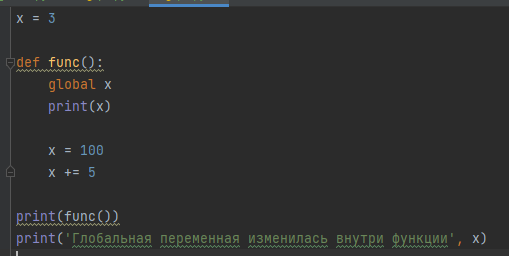
****

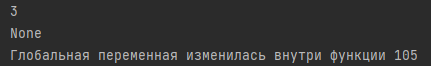
**Но так уже становится трудно воспринять этот код, так еще можно сделать еще хуже.** Этот код изменил переменную 3 на 8, вне функции.





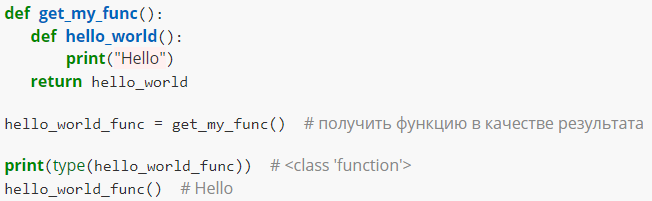
Да даже так, происходит изменение глобальной переменной!



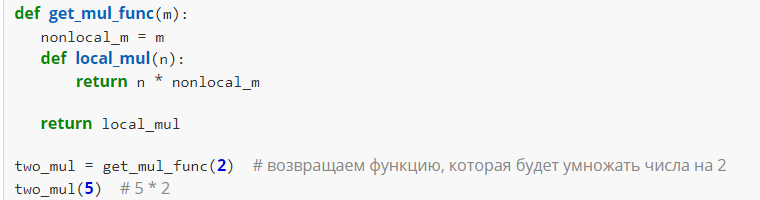


**Нелокальная область видимости**

Появилось это понятие в Python 3 вместе с ключевым словом nonlocal. Используется чаще всего во вложенных функциях, когда мы хотим дать интерпретатору понять, что для вложенной функции определенная переменная не является локальной, но она и не является глобальной в общем смысле. Вот пример функции, которая возвращает функцию.



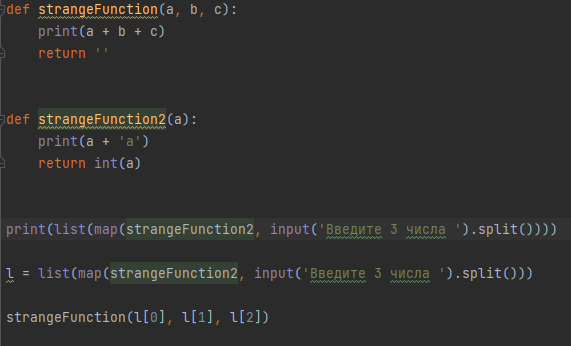
А nonlocal нужен вот для таких случаев.



**Использование переменных объявленных вне функции называется замыканием функций.**

Хочу написать функцию, в которую буду передавать распакованный список.

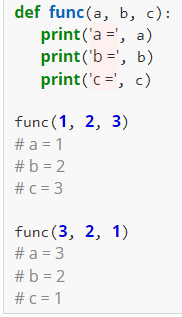
Пока получилось только так



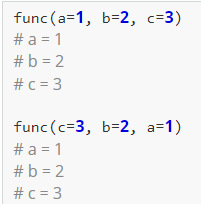
Как же мне распаковать список….

**Запакованные переменные, или что такое \*args и \*\*kwargs**

Это как раз по коду, который я не помню как оформить, тот что выше. Во первых в функцию можно отправлять значения просто через запятую. Это позиционный метод.

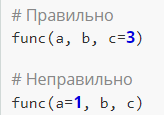


А можно строго по именам, именованный.

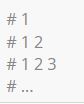
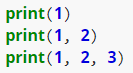


Используя именованный, порядок не нарушится.

**Важная особенность: все именованные аргументы должны идти строго после позиционных, как при объявлении функций, так и при их вызове.**

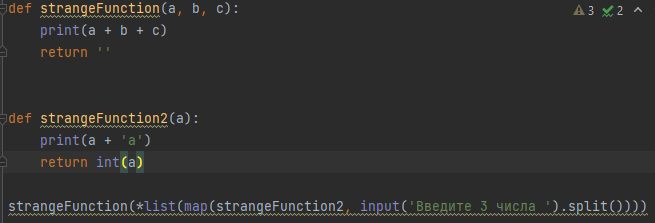
****

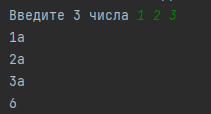
**А теперь. Как написать функцию, работающую как print()**



Оператор **\*** чаще всего ассоциируется с операцией умножения, но в Python он имеет и другой смысл. Этот оператор позволяет **«распаковывать»**

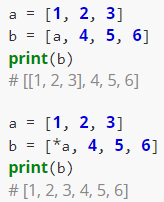
**О, то что надо.**





Хорошо, но ведь если я введу 4 числа, то жопа, верно? Нужно и в функцию передать их.

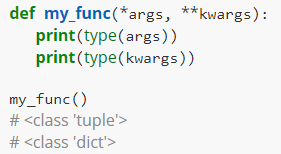
Чтобы было понятее, вот:



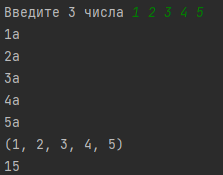
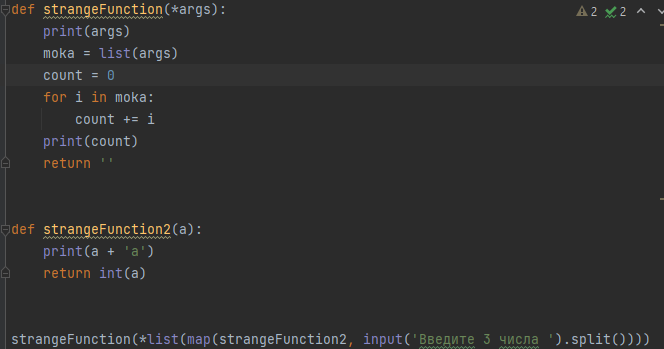
**Для неименованный аргументов \*args, для именованных \*\*kwargs**

Это не зарезервированные слова, просто хорошая практика. args — это сокращение от «arguments» (аргументы), а kwargs — сокращение от «keyword arguments» (именованные аргументы)

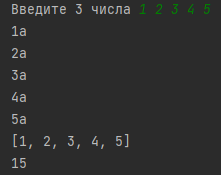
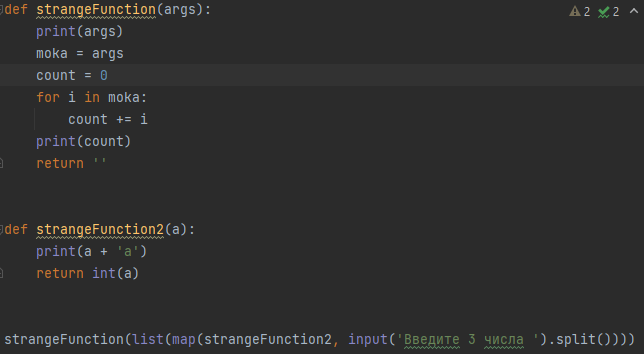
Разница в том, что одна обращает в список, другая в словарь.



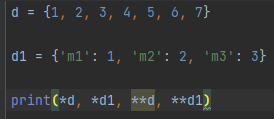
Ну вот:



Но если подумать, то вместо этого я просто мог бы отправлять в эту функцию список и все:

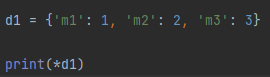


Ладно, просто попробовать распаковать словарь.



Ага, нужны именно две \*\*.

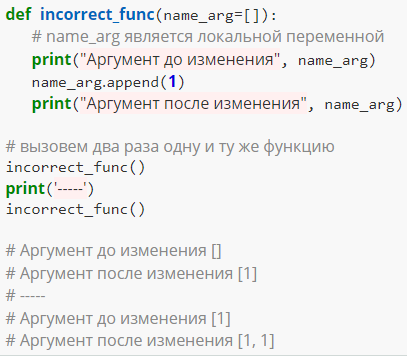
 Так тоже нельзя. Видимо только со словарем.

 Понятно, так можно получить ключи.

**Изменяемые типы данных как аргументы по умолчанию**

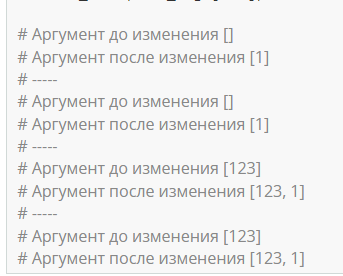
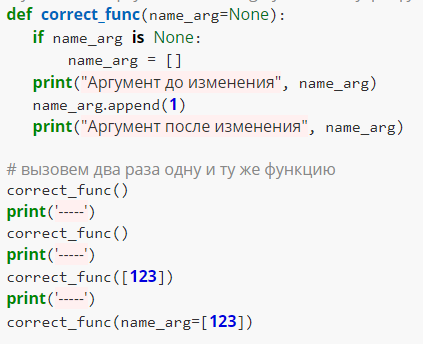
Если в качестве аргументов по умолчанию использовать изменяемые типы данных (списки, словари), то они создаются один раз и будут использоваться на протяжении времени жизни функции. Так как при передаче словаря передаются не все его значения, а указатель на первый его элемент.

**Вот пример неправильной функции.**

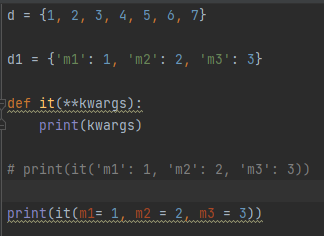
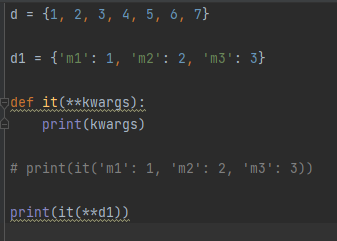
****

Функция при первом запуске создала в памяти пустой список. И он так и остался в памяти, поэтому после вызова функции второй раз используется все тот же список, а не новый.

Если в функции все же нужно использовать список, то следует писать так:



Кварги только так можно писать:





**Полезная информация.**

**Рекурсивные функции**