Сравнение операторов yield и return в Python (с примерами)

**ОПУБЛИКОВАНО**[**30.05.2021**](https://pythonist.ru/yield-vs-return-v-python/)[**ГЕНЕРАТОРЫ**](https://pythonist.ru/category/voprosy-s-sobesedovaniya-python/generatory/)**,**[**ФУНКЦИИ**](https://pythonist.ru/category/voprosy-s-sobesedovaniya-python/funkczii/)

В этой статье мы расскажем про основные различия между yield и return в Python. А для лучшего понимания этих различий приведем пару примеров.

Встроенное ключевое слово yield используется для создания функций-генераторов. (Про генераторы и их отличия от функций и списков можно подробнее прочитать [здесь](https://pythonist.ru/generatory-v-python/)).

Функция, содержащая yield, может генерировать сразу несколько результатов. Она приостанавливает выполнение программы, отправляет значение результата вызывающей стороне и возобновляет выполнение с последнего yield. Кроме того, функция, содержащая yield, отправляет сгенерированную *серию результатов в виде объекта-генератора*.

Return также является встроенным ключевым словом в Python. Он завершает функцию, а вызывающей стороне отправляет *значение*.

**Разница между yield и return**

Начнем с того, что между yield и return есть много заметных различий. Для начала давайте обсудим их.

| **RETURN** | **YIELD** |
| --- | --- |
| Оператор return возвращает только одно значение. | Оператор yield может возвращать серию результатов в виде объекта-генератора. |
| Return выходит из функции, а в случае цикла он закрывает цикл. Это последний оператор, который нужно разместить внутри функции. | Не уничтожает локальные переменные функции. Выполнение программы приостанавливается, значение отправляется вызывающей стороне, после чего выполнение программы продолжается с последнего оператора yield. |
| Логически, функция должна иметь только один return. | Внутри функции может быть более одного оператора yield. |
| Оператор return может выполняться только один раз. | Оператор yield может выполняться несколько раз. |
| Return помещается внутри обычной функции Python. | Оператор yield преобразует обычную функцию в функцию-генератор. |

**Пример 1**

Теперь давайте рассмотрим разницу между операторами return и yield на примерах.

В приведенном ниже коде мы использовали несколько операторов возврата. Вы можете заметить, что выполнение программы прекратится уже после первого оператора return. Весь код, идущий после, не будет выполнен.

num1 =10

num2=20

**def** mathOP():

**return** num1+num2

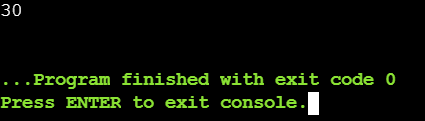
**return** num1-num2

**return** num1\*num2

**return** num1/num2

print(mathOP())

В выводе видно, что функция возвращает только первое значение, после чего программа завершается.



Чтобы выполнить аналогичную задачу с несколькими операторами return, нам нужно создать четыре разные функции для каждого типа арифметической операции.

num1 =10

num2=20

**def** sumOP():

**return** num1+num2

**def** subtractOP():

**return** num1-num2

**def** multiplicationOP():

**return** num1\*num2

**def** divisionOP():

**return** num1/num2

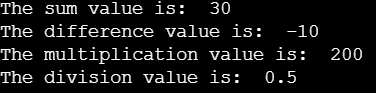
print("The sum value is: ",sumOP())

print("The difference value is: ",subtractOP())

print("The multiplication value is: ",multiplicationOP())

print("The division value is: ",divisionOP())

Запустив данный код, получим следующий результат:



Однако мы можем выполнить эти арифметические операции внутри одной функции-генератора, используя несколько операторов yield.

num1 =10

num2=20

**def** mathOP():

**yield** num1+num2

**yield** num1-num2

**yield** num1\*num2

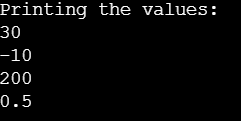
**yield** num1/num2

print("Printing the values:")

**for** i **in** mathOP():

print(i)

Получим результат:



**Пример 2**

Давайте рассмотрим еще один пример использования операторов return и yield.

Создадим список чисел и передадим его в функцию mod() в качестве аргумента. Далее, внутри функции, мы проверяем каждый элемент списка. Если он делится без остатка на 10, то мы его выводим.

Для начала давайте реализуем этот пример в нашем скрипте Python с использованием оператора return.

myList=[10,20,25,30,35,40,50]

**def** mod(myList):

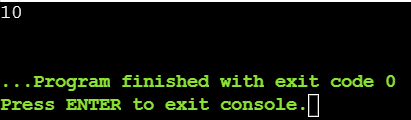
**for** i **in** myList:

**if**(i%10==0):

**return** i

print(mod(myList))

Оператор return возвращает только первое число, кратное 10, и завершает выполнение функции.



Теперь давайте реализуем тот же пример, используя оператор yield.

myList=[10,20,25,30,35,40,50]

**def** mod(myList):

**for** i **in** myList:

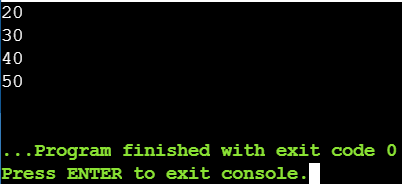
**if**(i%10==0):

**yield** i

**for** i **in** mod(myList):

print(i)

Получим следующий результат:



**Заключение**

В этой статье мы провели сравнение yield и return в Python, перечислили все заметные различия между ними и разобрали это на примерах.

Давайте подытожим.

И return, и yield являются встроенными ключевыми словами (или операторами) Python. Оператор return используется для возврата значения из функции. При этом он прекращает выполнение программы. А оператор yield создает объект-генератор и может возвращать несколько значений, не прерывая выполнение программы.