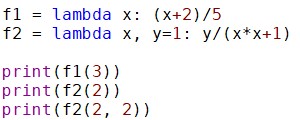
## Анонимные функции

Очень часто в качестве аргумента для функций высшего порядка требуется совсем простая функция. Причем нередко такая функция нужна в программе только в одном месте, поэтому ей необязательно даже иметь имя. Такие короткие безымянные (анонимные) функции можно создавать инструкцией **lambda** *параметры*: *выражение*

Такая инструкция создаст ссылку на функцию, принимающую указанный список параметров и возвращающую результат вычисления выражения. Эту ссылку можно сохранить в переменной или передать в качестве параметра в другую функцию. Вызывается эта функция как обычная функция.

Тело лямбда-функции состоит из одного выражения. Инструкция return подразумевается, но не пишется. Скобки вокруг параметров отсутствуют, аргументы от выражения отделяет двоеточие. Параметры могут иметь значения по умолчанию.



Пример анонимной функции с проверкой условия:



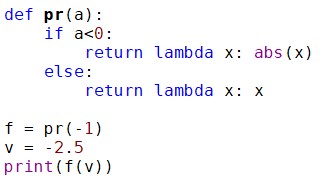
Эта функция возвращает -1 для отрицательных значений, 0 – для нулей, 1 – для положительных.

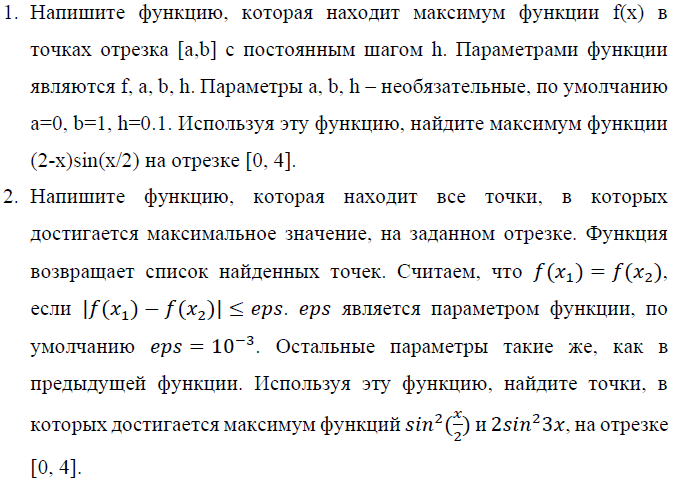
Анонимные функции используются:

* в качестве параметра функции:



* в качестве возвращаемого значения:





## Встроенные функции высшего порядка

Ниже рассматриваются некоторые встроенные функции высшего порядка, которые часто используются на практике. На самом деле их достаточно много. Описание следует искать в документации и специальной литературе.

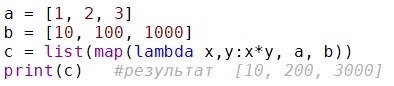
map(f, \*args) – преобразует элементы заданных последовательностей с помощью функции f в новый объект, поддерживающий итерацию. Если при вызове задана одна последовательность, то функция f по очереди выполняется для каждого элемента исходной последовательности (элемент является параметром функции f). Если последовательностей несколько, то при вычислении функции f берутся по одному элементу из каждой последовательности. Этот набор значений будет параметрами функции f. Вычисленные значения образуют новую последовательность, которая и является результатом функции map(). Полученный объект не является списком. Преобразовать его в список можно с помощью функции list().

Примеры:

* получаем список длин строк для заданного списка строк:



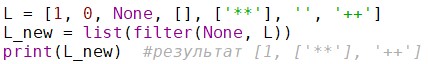
* получаем список, составленный из попарных произведений двух исходных списков:



filter(f, seq) – позволяет отобрать элементы последовательности, удовлетворяющие условию. Функция f должна возвращать True для элементов, включаемых в результат, и False – в противном случае. Второй параметр – исходная последовательность. Если в первом параметре указать значение None вместо имени функции, то каждый

элемент исходной последовательности будет проверен на соответствие значению True. Как и в случае с map() функция возвращает итерируемый объект. Для преобразования его в список используйте list(). Примеры:

* удалим «пустые» значения из списка (нули, пустые строки, пустые списки):

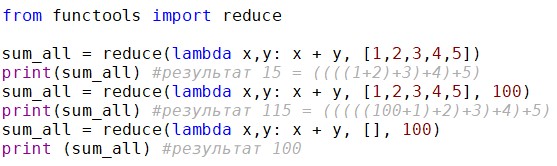


* найдем все числа от 1 до 99, которые при делении на 15 дают 2 в остатке:



reduce(f, seq[, initial]) – применяет указанную функцию f к парам элементов и накапливает результат. Функция f должна иметь два параметра. При первом обращении к f ее параметрами являются два первых элемента последовательности. При последующих обращениях в качестве первого параметра используется значение, которое f вернула на предыдущем шаге, а в качестве второго – очередной элемент последовательности. Значение initial, если оно задано, при вычислении добавляется перед элементами последовательности seq, а также является значением по умолчанию, если последовательность seq пуста. Функция находится в модуле functools. Примеры:

* вычислим сумму элементов списка:



* вычислим, сколько раз символ «\*» встречается в списке строк:

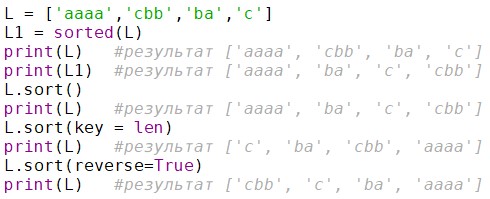


* вычислим наибольший элемент списка с помощью reduce^

 sorted(seq, key=None, reverse=False) – возвращает новый список, содержащий все элементы исходной последовательности в порядке возрастания значений элементов (по умолчанию). Если параметр reverse=True, то элементы располагаются в порядке убывания значений. В параметре key можно указать ссылку на функцию с одним параметром, возвращающую значение. Тогда сортировка выполняется следующим образом. Для каждого элемента последовательности вычисляется значение функции, указанной в key (элемент последовательности является параметром функции). Элементы исходной последовательности переставляются таким образом, чтобы вычисленные для них значения были отсортированы.

Для списков в языке определен метод sort(), который очень похож на функцию sorted(). Параметры метода key и reverse имеют тот же смысл. Отличие метода от функции состоит в том, что он применяется только для списков, и он не создает новый список, а сортирует имеющийся. Примеры:

* отсортируем список строк с использованием различных значений параметров:



* отсортируем список слов по алфавиту. Если слова содержат символы в разных регистрах, то сортировка получается неправильной, так как коды символа в разных регистрах разные. Чтобы получить правильный результат, нужно привести все символы к одному регистру. Для этого можно, например, воспользоваться методом lower() у строк. Обратите внимание, как можно сослаться на метод в параметре key:



* сформируем последовательность чисел от 1 до 9 такую, чтобы остатки от деления на 3 этих чисел шли в порядке возрастания:



1. При помощи функций map/filter/reduce возвести в квадрат числа от 1 до 100, и рассчитать их сумму, не включая в сумму числа, кратные 9.

2. При помощи функций map/filter/reduce превратить список целых чисел в строку, содержащую строковое представление этих чисел, разделенных пробелами.

3. При помощи функций map/filter/reduce из несколько одинаковых подряд идущих элементов списка оставить только один. Например, [1, 2, 3, 4, 4, 4, 5, 6, 6, 7, 6, 1, 1] -> [1, 2, 3, 4, 5, 6, 7, 6, 1]

4. При помощи функций map/filter/reduce из списка списков извлечь элементы, содержащиеся во вложенных списках по индексу 1. Например, [[1, 2, 3], [2, 3, 4], [0, 1 , 1 , 1], [0, 0]] -> [2, 3, 1, 0]

5. Дан список А, состоящий из 2N элементов. Разбейте его на списки В и С по N элементов каждый так, чтобы каждый элемент В не превосходил каждого элемента С.

6. В списке 2n + 1 различных элементов. Найдите средний элемент списка. Под средним элементом понимают такой, для которого в списке n элементов больше его и n элементов меньше.

7. Дан список из N элементов. Вывести индексы списка в том порядке, в котором соответствующие им элементы образуют возрастающую последовательность.

8. Для списков вида [[‘Иванов’, 3,5,4], [‘Петров’, 4, 5, 5], [‘Сидоров’, 3, 3, 3], [‘Николаев’, 4, 4, 3]] напишите функцию, которая выводит на экран этот список в виде таблицы в отсортированном виде. Параметрами функции являются список, функция, которая определяет порядок сортировки (значение параметра key в sort), и параметр, определяющий, выполняется сортировка по возрастанию или убыванию. Функция не возвращает значение и не изменяет исходный список.

Используя эту функцию, выведите исходный список, отсортированный:

• по фамилиям в алфавитном порядке;

• в порядке возрастания первой оценки;

• в порядке убывания суммы баллов.