Язык Python - общая характеристика, преимущества и недостатки, краткая история. PEP’ы.

Преимущества:

* он читабельнее чем другие языки
* нужно писать меньше кода в сравнении с другими языками
* работает без изменений на разных ОС
* богатая стандартная библиотека с обширным функционалом
* можно интегрировать другие языки программирования
* более легок для изучения и запоминания чем другие языки

Минусы:

* работает медленнее чем другие языки программирования (к примеру, низкоуровневые)

PEP - (Python Enhancement Proposal) предложения по улучшению пайтон.

Python разработан математиком.

Работа интерпретатора. PVM. Байт-код. Выполнение и способы запуска кода и программ.

Интерпретатор – разновидность программы, которая читает и выполняет другие программы, является программной логикой между кодом и компьютером.

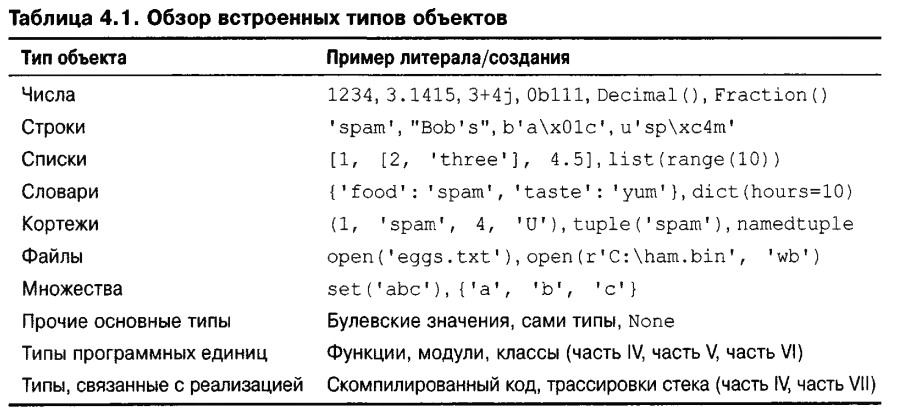
Сперва создается байт-код – это низкоуровневое и независимое от платформы представление исходного кода. Сохраняется файл байт-кода в новую папку \_\_pycache\_\_ с расширением .pyc, папка находится в каталоге с исходным кодом (до версии 3.2 все это делалось в текущем каталоге без доп папки). В последующем если не менять исходный код и запускать его с помощью такой же версии пайтона то пропустится компиляция и сразу запустится байт-код, иначе скомпилится новый байт-код.

PVM – виртуальная машина пайтон. Это большой закодированный цикл, который проходит по инструкциям байт-кода, чтобы выполнить операции. Это последний шаг интерпретатора.

Пять реализаций языка Python — CPython (пайтон + С), Jython (пайтон + джава), IronPython (пайтон + .нет), Stackless и PyPy. Все они направлены на те или иные задачи, например, на оптимизацию языка пайтон или интеграцию с другими языками и т.д.

Также код можно писать в интерактивной подсказке, это ради экспериментов либо для того, чтобы узнать, как работает неизвестная часть кода.

Основные примитивные типы данных. Реализация операций над ними.



Для создания объекта нового типа используются литералы – выражения с определенным синтаксисом, именно от синтаксиса литерального выражения зависит тип объекта. В питоне нет привычной инициализации как в языке С. Объекты строго типизированы, при объявлении определяется тип объекта, а дальше мы можем использовать только те операции, которые допустимы с типом объекта. Также типы делятся на неизменяемые и изменяемые. Первые – числа, строки, кортежи, вторые – списки, словари и множества. Для «изменения» неизменяемого типа нужно создать новый объект (можно такое же имя, при этом старый объект перезапишется)

Модули в питоне – библиотеки в С.

Числа – разные математические операции и так далее.

Строки – грубо говоря массив символов. Можем обращаться к отдельным символам по индексам, при этом порядок слева направо, первый индекс – 0. Можем использовать отрицательные индексы, тогда порядок справа налево, первый индекс -1. Формально отрицательный индекс добавляется к длине строки. То есть объект типа строка ‘spam’ с длинной 4 хранит букву ‘m’ по индексу 3, значит длина плюс отрицательный индекс это 4 – 1, то есть 3 индекс.  
Строки можно резать (срез). Примерно выглядит так:   
S = ‘spam’

S[1:3]

‘pa’

Другие примеры: [1:] – все после первого символа, [0:3] или [:3] – все кроме последнего элемента

Строки поддерживают конкатенацию – объединение строк в новую строку. Реализуется через +. Можно также вывести строку несколько раз по примеру «строка» \* кол-во раз.

В пайтон есть универсальные методы, которые могут применяться к разным типам, но методы, определенные для одного типа, применяются только к нему.

Команда dir(объект) показывает список методов доступных для данного типа.

Команда help(объект.метод) – документация определенного метода.

СПИСКИ

Синтаксис в квадратных скобках, схожи с массивами, только можно помещать туда неограниченное количество объектов разных типов, можно также увеличивать или уменьшать размер списка методами append/pop.

Невозможно работать с несуществующими объектами списка, то есть если мы скажем что в списке три элемента и попытаемся обратиться к элементу номер 4, то пайтон выдаст ошибку.

Списки можно вкладывать, делая неограниченную глубину.

Можно работать со столбцами, это называется списковые включения.

СЛОВАРИ

Не последовательности, а коллекции объектов с ключами.

Синтаксис в фигурных скобках {ключ:объект, и т.д.}

В словарях нет порядка по ключам, даже если ключи целочисленные объекты

Для сортировки можно использовать метод sorted.

КОРТЕЖИ

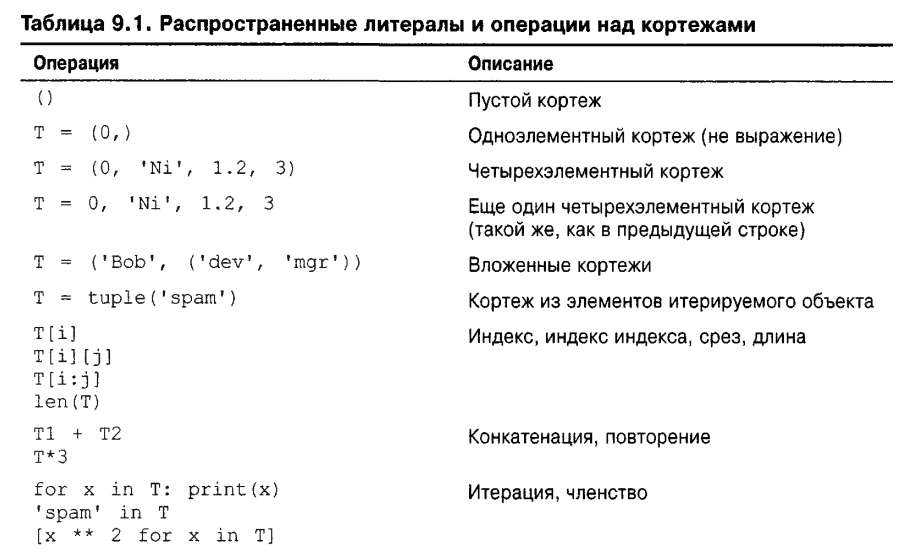
Они являются упорядоченными коллекциями произвольных объектов

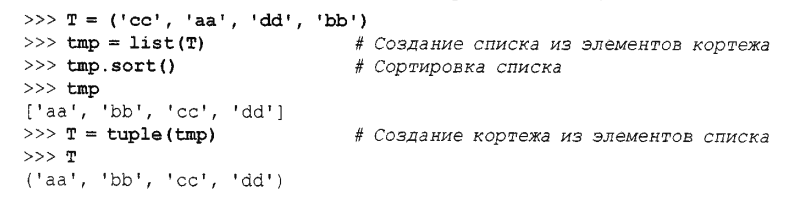
Они поддерживают доступ по смещению, индексу

Они относятся к категории “неизменяемая последовательность"

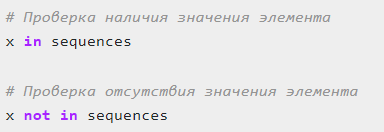
Они имеют фиксированную длину (так как неизменяемы), разнородны и допускают произвольно глубокое вложение

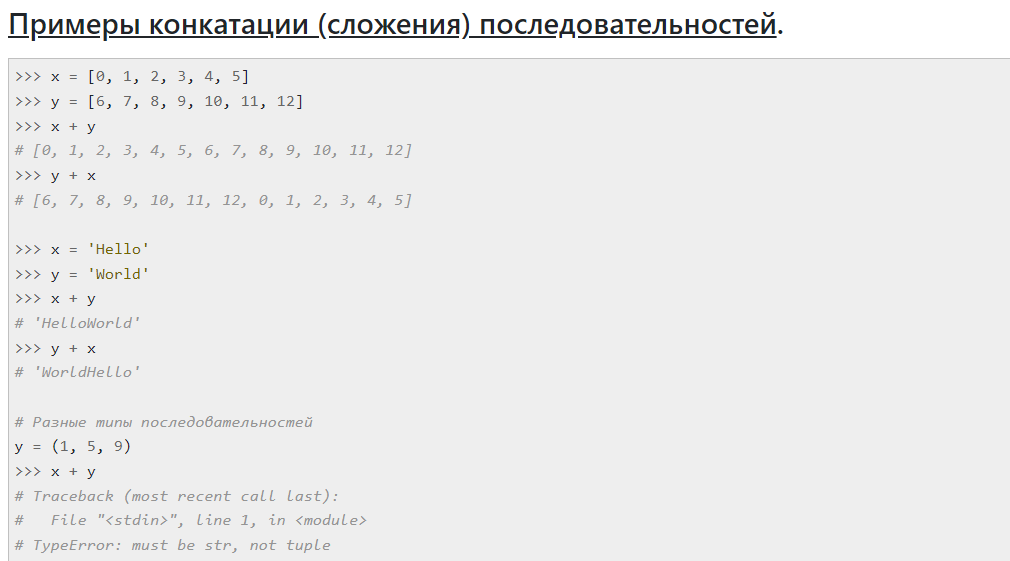
Они представляют собой массивы ссылок на объекты

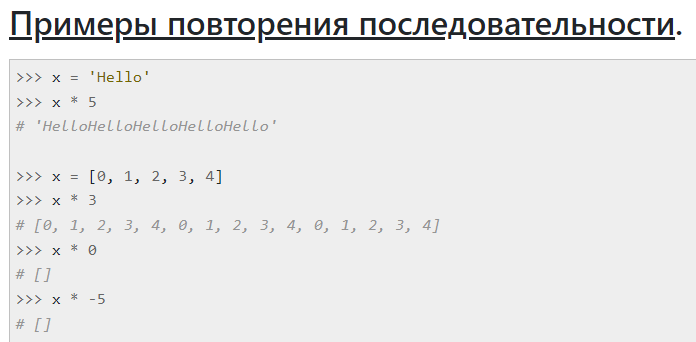


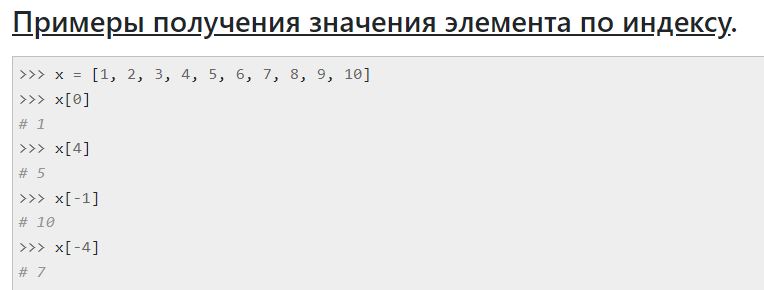


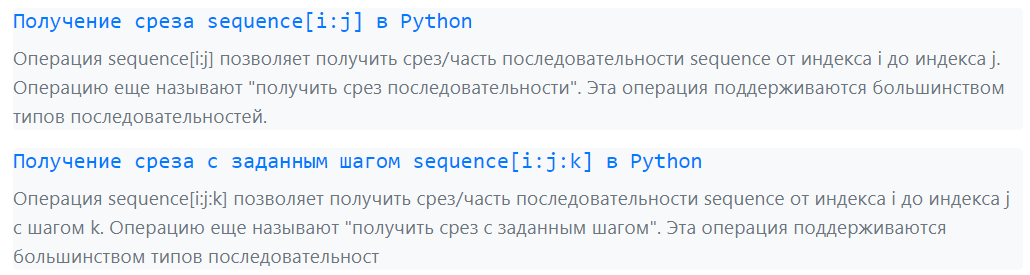
Последовательности. Операции над последовательностями

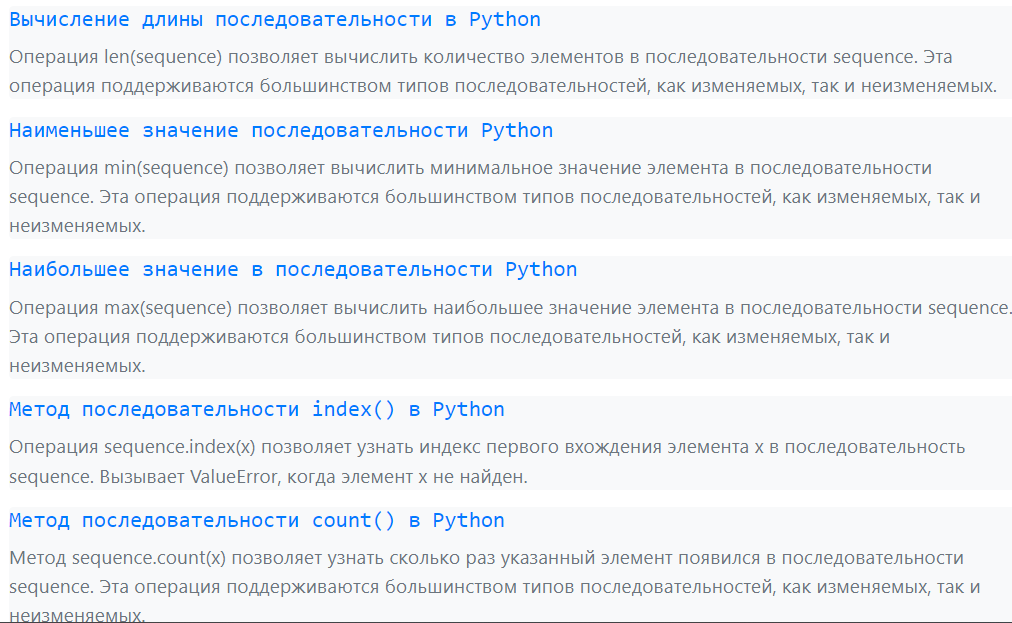




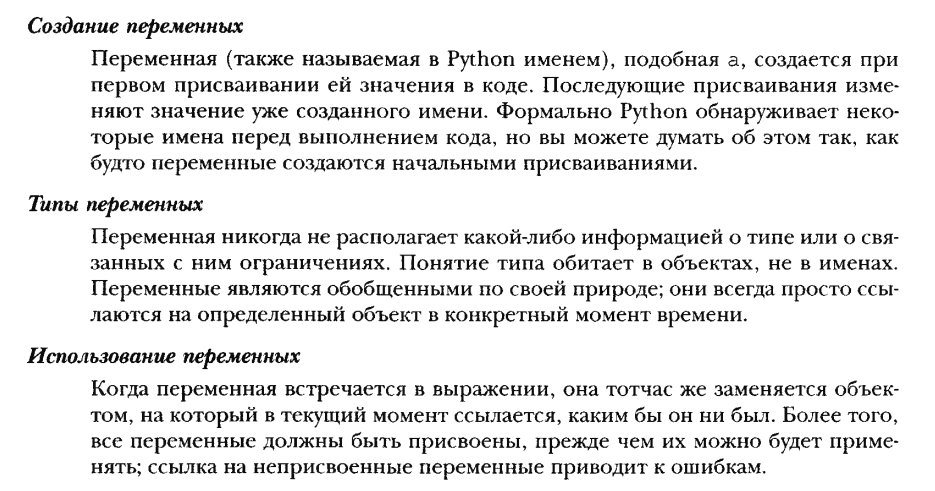


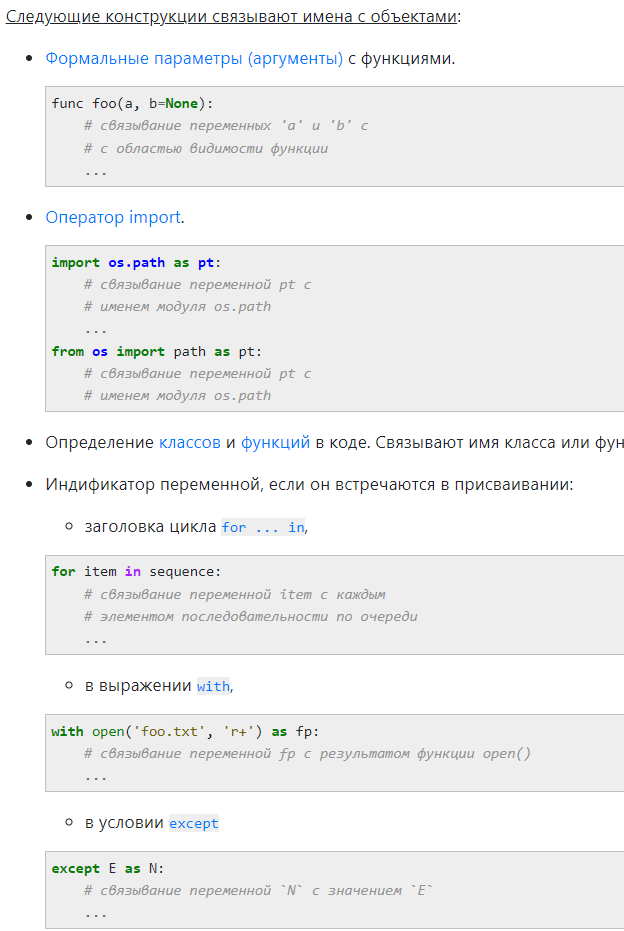






Операция связывания и динамическая типизация.

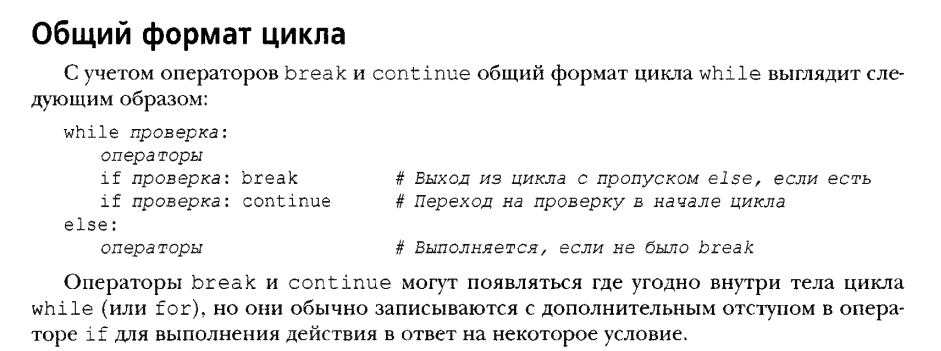


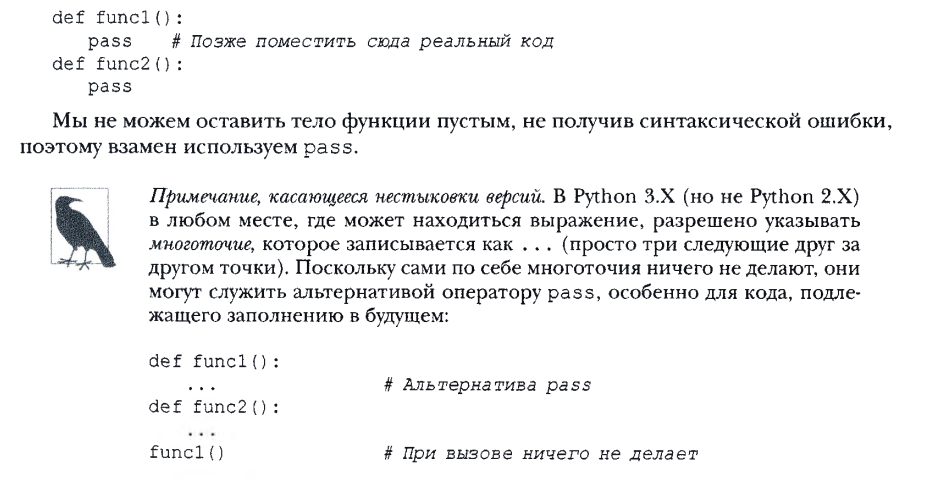


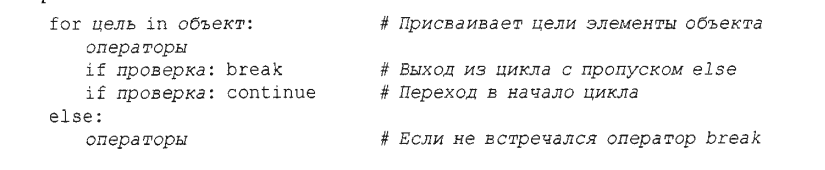
Условный оператор. Логические операции и логические сравнения.

Больше меньше и так далее, когда сравниваем несколько то ставится and

Циклы







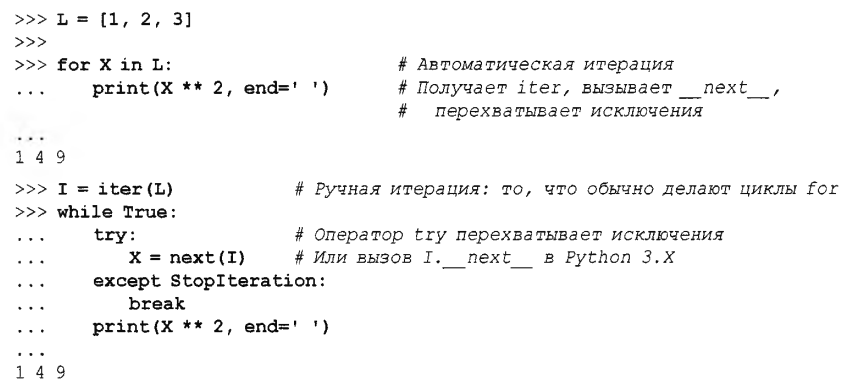
For присваивает цели элементы объекта это важно!

Можно использовать срезы

Итерации и включения. Протокол итерации

На примере файлов: можно читать строки файлов с помощью цикла for, используется метод \_\_next\_\_ который возвращает StopIteration

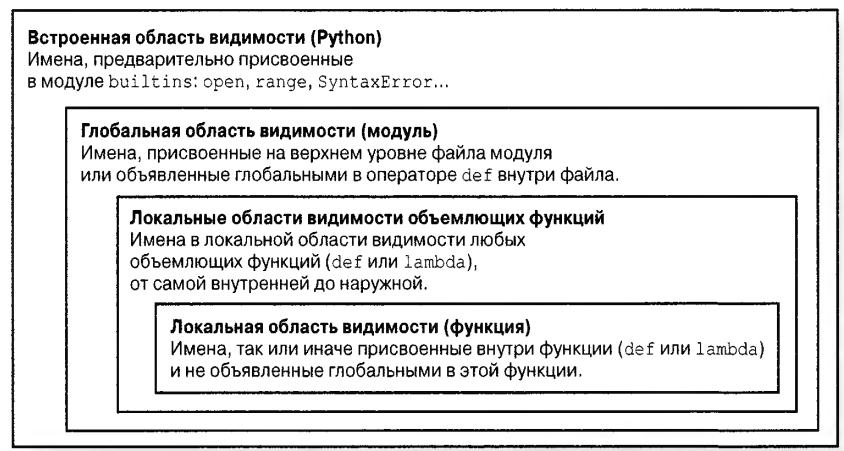
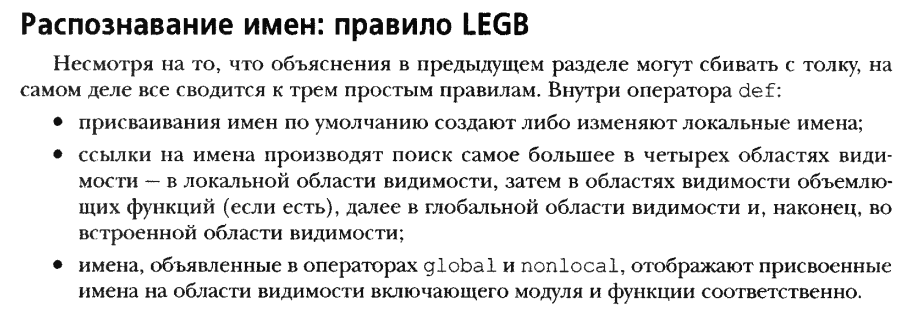
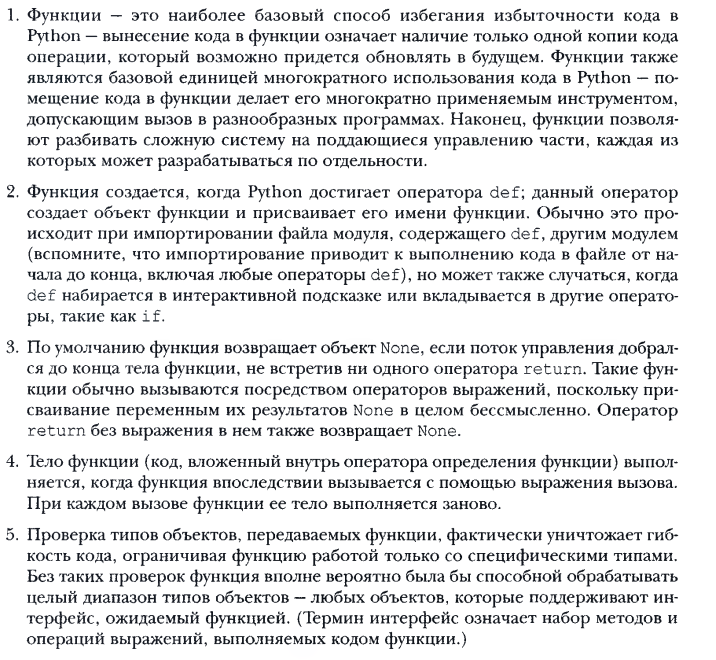
Полный протокол заключается в том, что для не итераторов мы обязаны вызывать метод \_\_iter\_\_, а для файлов – нет, так как они итераторы сами по себе, у них есть свой метод \_\_next\_\_

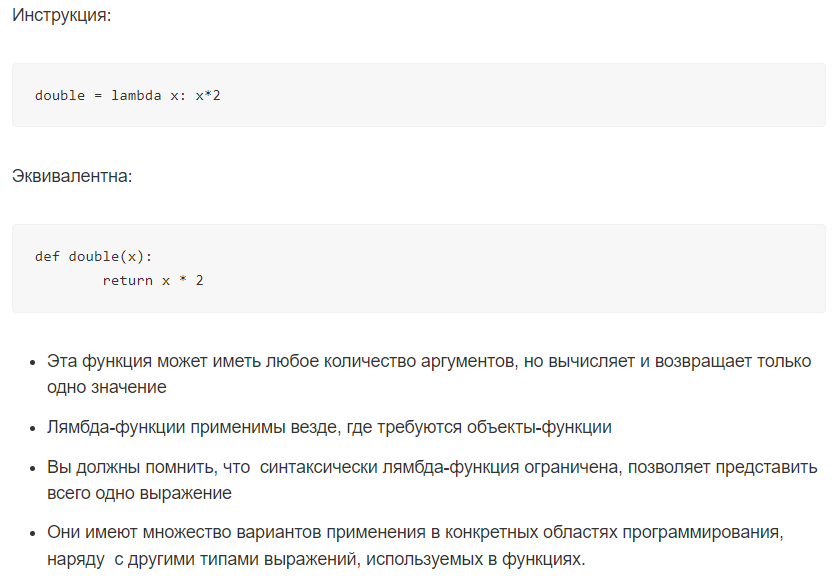


Функции. Чистота функций. Побочные эффекты. Области видимости. LEGB. global и nonlocal.

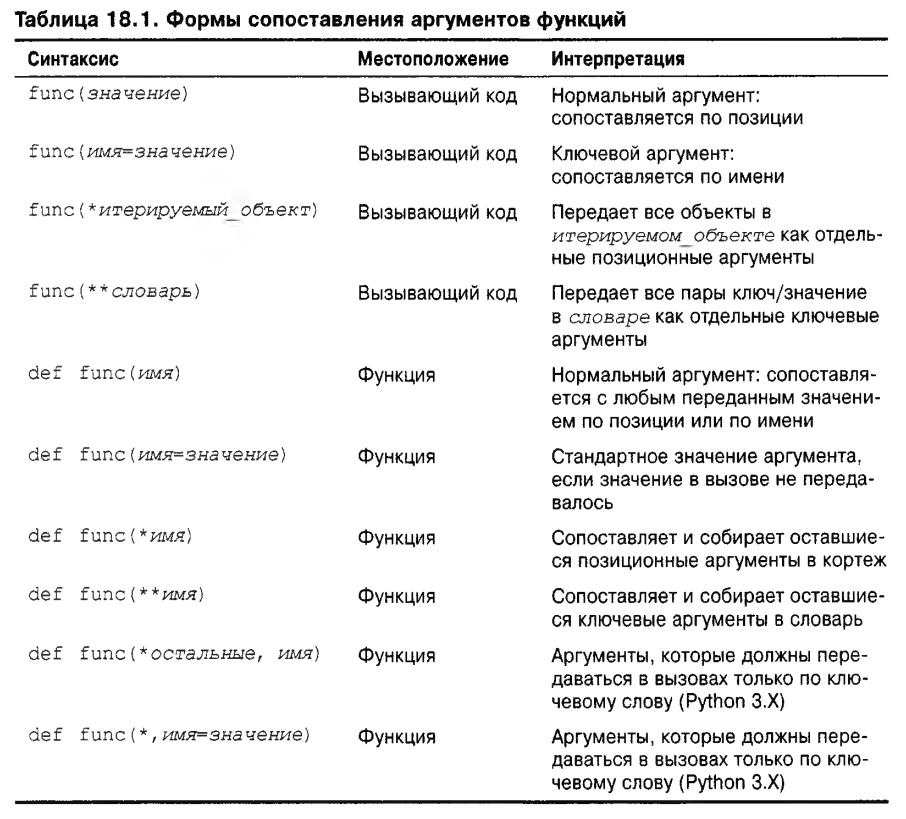
Def name (arguments):

Body..





Передача аргументов. Режимы сопоставления аргументов.



Функции первого класса. Функции в качестве аргументов. Возвращаемые функции Вложенные функции.

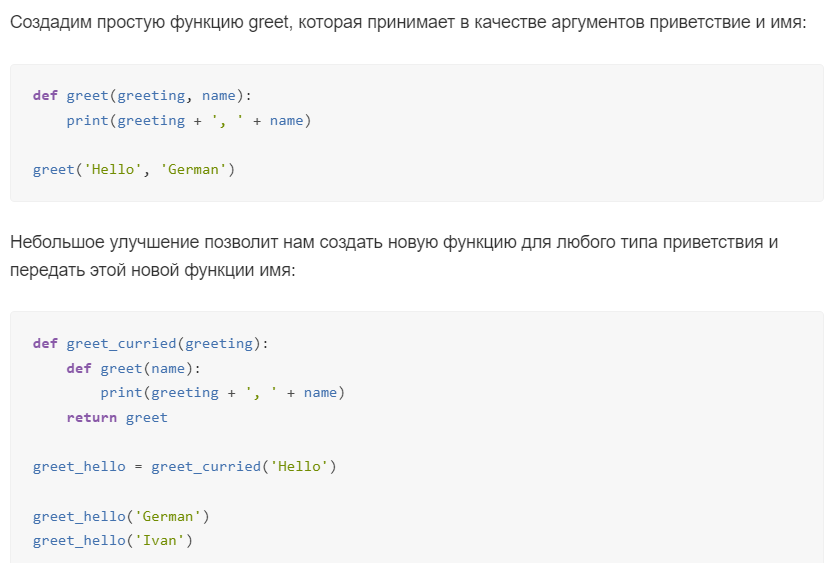
В Python функции являются объектами первого класса. Это означает, что их можно передавать в качестве аргументов другим функциям, возвращать как значения из других функций и хранить в переменных или структурах данных как любой другой объект. Это позволяет использовать функции как строительные блоки для более сложных программ.

Функции, которые принимают другие функции в качестве аргументов и/или возвращают функции в качестве результатов, называются **функциями высшего порядка** или **ФВП**. Их можно использовать для инкапсуляции многократно используемого поведения и создания более абстрактного кода, о котором легче рассуждать.

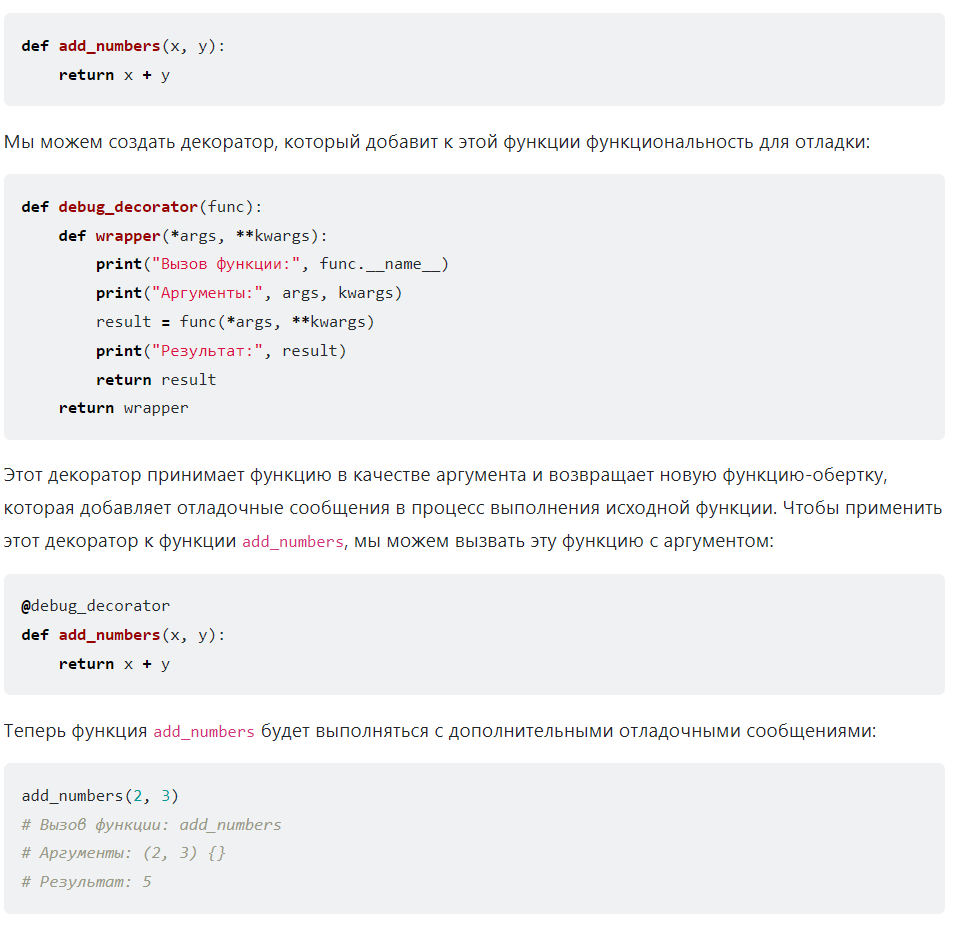
Например, встроенные функции map и filter в Python являются функциями высшего порядка, которые работают с итерируемыми объектами и применяют функцию к каждому элементу итерируемого объекта.

Функции первого класса. Каррирование и декаррирование

Карринг — это преобразование функции от многих аргументов в набор функций, каждая из которых является функцией от одного аргумента. Мы можем передать часть аргументов в функцию и получить обратно функцию, ожидающую остальные аргументы.



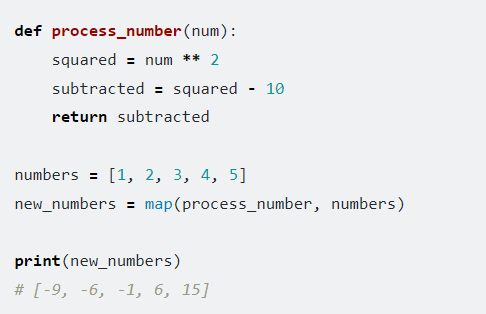
**Декораторы** в Python — это функции, которые принимают другую функцию в качестве аргумента, добавляют к ней некоторую дополнительную функциональность и возвращают функцию с измененным поведением.



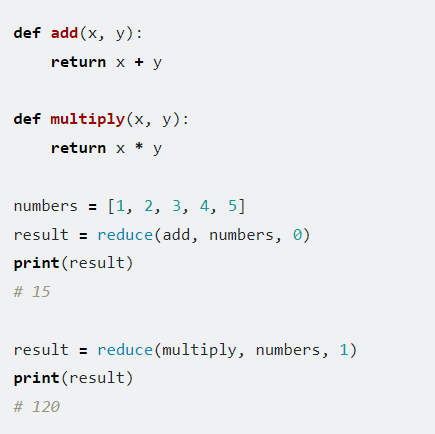
Анонимные функции. map, reduce, filter

Map

Получает на вход функцию, которая что-то делает со списком, сам список. Возвращает новый список с изменениями. Некая замена for



Reduce



Filter

Работает как map, но возвращает список по условию

