

Основы языка Python. Интерактивный курс



Определение функции. Встроенные функции

План

- Определение функции.
- Зачем нужны функции.
- Функции, которые мы уже использовали.
- Другие полезные встроенные функции.
- Примеры.



Определение функции

Фрагмент программного кода (подпрограмма), к которому можно обратиться из другого места программы (Википедия).

- Функции обычно имеют имя.
- В Python можно создавать функции и без имени.



Зачем нужны функции

- Для повторного использования кода.
- Для создания более логичной структуры программы.
- Для объединения нескольких небольших действий в одно.



Полезные встроенные функции

- `abs` - модуль числа
- `min`, `max` - минимальное, максимальное значение
- `round` - округление числа
- `sum` - сумма элементов последовательности
- `enumerate` - нумерация последовательности



Пример

- Пользователь вводит 3 числа. Найти минимальное из них, максимальное из них, их сумму и вывести результат на экран.



Python



Создание собственных функций

План

- Зачем писать свои функции.
- Как объявить функцию в коде.
- Использование своих функций.
- Примеры.



Зачем писать свои функции

- Для повторного использования своего кода.
- Для создания функционала, которого нет в стандартной библиотеке.



Параметры и результат

- Простой разделитель - нет параметров, нет возврата.
- Меняем знак разделителя - 1 параметр, нет возврата.
- Меняем знак и длину - 2 параметра, нет возврата.
- Как использовать разделитель в тексте вместо того, чтобы печатать его в консоль? 2 параметра и результат (возвращаемое значение).



Операции со множествами

- объединение \cup
- пересечение \cap
- разность $-$
- существуют и другие методы и операции



Python



Аргументы функции

План

- Параметры (аргументы) функции.
- Функции без параметров и с параметрами.
- Передача параметров по порядку, по имени.
- Значения параметров по умолчанию.
- args, kwargs (любое кол-во параметров).



Параметры функции

- `def my_func(параметр1, параметр2, ...)`



Передача параметров

- По порядку
- `greeting('Leo', 'Hello')`
- По имени
- `greeting(say='Hello', who='Leo')`



Значения по умолчанию

- Можно указать у параметра значение по умолчанию
- `def greeting(who, say='Hello')`
- Если мы не передадим параметр `say` при вызове `greeting('Max')`, функция сработает со значением по умолчанию.



args, kwargs

Иногда нужно реализовать передачу любого количества аргументов:

- `def greeting('Hello', 'Leo', 'Max', 'Kate', ...)`
- `args` - передача любого количества по порядку
- `kwargs` - передача любого количества по имени



Python



Области видимости

План

- Определение области видимости.
- Локальные и глобальные переменные.
- Чем плохи глобальные переменные.
- Примеры.



Область видимости (википедия)

- В программировании обозначает область программы, в пределах которой идентификатор (имя) некоторой переменной продолжает быть связанным с этой переменной и возвращать её значение.
- За пределами области видимости тот же самый идентификатор может быть связан с другой переменной, либо быть свободным (не связанным ни с какой из них).



Область видимости объекта

- Набор функций или модулей, внутри которых допустимо использование имени этого объекта.



Локальные и глобальные переменные

- Глобальными называют объекты, объявление которых дано вне функции. Они доступны (видимы) во всем файле, в котором объявлены.
- Локальными называют объекты, объявление которых дано внутри блока или функции. Эти объекты доступны только внутри того блока, в котором объявлены.



global

- При желании можно изменить глобальную переменную.
- Для этого в функции нужно указать, что она глобальная.
- `global my_var.`

Этот механизм лучше не использовать. Мы рискуем, изменив значение переменной в одной функции, получить неверное значение в другой функции.



ОТНОСИТЕЛЬНОСТЬ ОБЛАСТИ ВИДИМОСТИ

В программе могут встречаться вложенные друг в друга функции. Тогда мы рассматриваем область видимости относительно какой-либо функции:

- `def a():`
 - `def b():`
 - `def c():`



Python



Передача функции
параметром,
lambda-функции

План

- Функция как объект.
- Передача функции как параметра.
- Применение.
- `lambda`-функции.



Функция - тоже объект

- Её можно записать в переменную.
- Её можно передавать параметром в другие функции.



Применение

- возможность не только входных данных, но и входных функций
- внутри функции переменными являются
- алгоритм
- последовательность действий
- сами действия



lambda-функции

- применяются для создания анонимных функций по месту их использования
- lambda входные параметры: результат



Python



sorted, filter, map

План

- sorted
- filter
- map
- примеры использования



sorted

- сортировка последовательности
- `sorted(iterable, *, key=None, reverse=False)`
- аргументы: последовательность, ключ для сортировки, порядок



filter

- фильтрация последовательности
- `filter(function, iterable)`
- аргументы: функция фильтрации, последовательность



map

- применение функции к каждому элементу последовательности
- `map(func, iterable, ...)`
- аргументы: функция, последовательность



sorted, filter, map

- применяются к последовательности
- имеют параметр - функцию
- позволяют создавать код быстро и удобно

