Сообщить об ошибке.

# Функция reduce() в Python



## Синтаксис:

```
from functools import reduce

reduce(function, iterable[, initializer])
```

## Параметры:

- `function` - пользовательская функция, принимающая 2 аргумента,
- `iterable` - итерируемая последовательность,
- `initializer` - начальное значение.

## Возвращаемое значение:

- требуемое единственное значение.

## Описание:

Функция `reduce()` модуля `functools` кумулятивно применяет функцию `function` к элементам итерируемой `iterable` последовательности, сводя её к единственному значению.

Аргумент `function` это функция которую требуется применить к элементам последовательности. Должна принимать два аргумента, где первый аргумент - аккумулированное ранее значение, а второй аргумент следующий элемент последовательности.

Аргумент `iterable` представляет собой последовательность, элементы которой требуется свести к единственному значению. Если последовательность пуста и не задан аргумент `initializer`, то возбуждается исключение `TypeError`.

Например `reduce(lambda x, y: x+y, [1, 2, 3, 4, 5])` вычисляет `((((1 + 2) +3) +4) +5)`. Левый аргумент `x` - это накопленное значение, а правый аргумент `y` - это следующий элемент `iterable`.

Если присутствует необязательный `initializer`, он помещается перед элементами `iterable` в вычислении. Другими словами это базовое значение, с которого требуется начать отсчёт. Аргумент `initializer`, так же служит значением по умолчанию, когда `iterable` является пустым.

Функция `reduce()` эквивалентна следующему коду:

```
def reduce(function, iterable, initializer=None):
    it = iter(iterable)
    if initializer is None:
        value = next(it)
    else:
        value = initializer
    for element in it:
        value = function(value, element)
    return value
```

## Примеры использования:

Функция `functools.reduce()` может быть полезна в различных сценариях, когда необходимо агрегировать данные или преобразовывать их кумулятивным способом.

Вычисление суммы всех элементов списка при помощи модуля `operator` и функции `functools.reduce()`:

```
>>> from functools import reduce
>>> import operator
# допустим имеем список чисел
>>> numbers = list(range(1, 11))
>>> reduce(operator.add, numbers)
```

```
# 55
>>> reduce(operator.sub, numbers)
# -53
>>> reduce(operator.mul, numbers)
# 3628800
>>> reduce(operator.truediv, numbers)
# 2.7557319223985893e-07
```

Вычисление суммы всех элементов списка при помощи reduce() и lambda-функции:

```
>>> from functools import reduce
>>> items = [10, 20, 30, 40, 50]
>>> reduce(lambda x,y: x + y, items)
# 150
```

Вычисление наибольшего элемента в списке при помощи reduce():

```
>>> from functools import reduce
>>> items = [1, 24, 17, 14, 9, 32, 2]
>>> reduce(lambda a,b: a if (a > b) else b, items)
# 32
```

## Содержание раздела: