Сообщить об ошибке.

Функция all() в Python, все элементы True

Проверяет, что все элементы в последовательности True

Синтаксис:

```
all(iterable)
```

Параметры:

- `iterable` - итерируемый объект (список, кортеж, словарь).

Возвращаемое значение:

- bool - значение логического типа `True` или `False`.

Описание:

Функция `all()` возвращает значение `True` , если все элементы в итерируемом объекте - истинны, в противном случае она возвращает значение `False`.

Если передаваемая последовательность пуста, то функция `all()` также возвращает `True`.

Функция `all()` применяется для проверки на `True` ВСЕХ значений в последовательности и эквивалентна следующему коду:

```
def all(iterable):
    for element in iterable:
        if not element:
            return False
    return True
```

Дополнительно смотрите встроенную функцию any()

В основном функция `all()` применяется в сочетании с оператором ветвления программы `if ... else`. Работу функции `all()` можно сравнить с оператором `and` в Python, только `all()` работает с последовательностями:

```
>>> True and True and True
# True
>>> True and False and True
# False

>>> all([True, True, True])
# True
>>> all([True, False, True])
# False
```

Но между `and` и `all()` в Python есть два основных различия:

- Синтаксис.
- Возвращаемое значение.

Функция `all()` всегда возвращает `False` или `True` (значение bool)

```
>>> all([3, 1, 2, 6])
# True
>>> all([3, 0, 2, []])
# False
```

Оператор `and` , возвращает ПОСЛЕДНЕЕ истинное значение, при условии, что в выражении все значения `True` а если в выражении присутствует значение `False` (ложное значение), то ПЕРВОЕ ложное значение. Что бы добиться поведения как у функции `all()`, необходимо выражение с оператором `and` обернуть в функцию `bool()`.

```
>>> 3 and 1 and 2 and 6
# 6
>>> 3 and 0 and 3 and []
# 0

>>> bool(3 and 1 and 2 and 6)
# True
>>> bool(3 and 0 and 3 and [])
# False
```

Из всего сказанного можно сделать вывод, что для успешного использования функции `all()` необходимо в нее передавать последовательность, полученную в результате каких то вычислений/сравнений, элементы которого будут оцениваться как `True` или `False`. Это можно достичь применяя функцию `map()` или выражения-генераторы списков, используя в них встроенные функции или методы, возвращающие `bool` значения, операции сравнения, оператор вхождения `in` и оператор идентичности `is`.

```
num = [1, 2.0, 3.1, 4, 5, 6, 7.9]
# использование встроенных функций или
# методов на примере 'isdigit()'
>>> [str(x).isdigit() for x in num]
# [True, False, False, True, True, True, False]

# использование операции сравнения
>>> [x > 4 for x in num]
# [False, False, False, False, True, True, True]

# использование оператора вхождения `in`
>>> ['.' in str(x) for x in num]
# [False, True, True, False, False, False, True]

# использование оператора идентичности `is`
>>> [type(x) is int for x in num]
# [True, False, False, True, True, True, False]

# использование функции map()
>>> list(map(lambda x: x > 1, num))
# [False, True, True, True, True, True, True]
```

Примеры проводимых проверок функцией `all()`.

Допустим, у нас есть список чисел и для дальнейших операций с этой последовательностью необходимо знать, что все числа например положительные.

```
>>> num1 = range(1, 9)
>>> num2 = range(-1, 7)
>>> all([x > 0 for x in num1])
# True
>>> all([x > 0 for x in num2])
# False
```

Или проверить, что последовательность чисел содержит только ЦЕЛЫЕ числа.

```
>>> num1 = [1, 2, 3, 4, 5, 6, 7]
>>> num2 = [1, 2.0, 3.1, 4, 5, 6, 7.9]
>>> all([type(x) is int for x in num1])
# True
>>> all([type(x) is int for x in num2])
# False
```

Или есть строка с числами, записанными через запятую и нам необходимо убедится, что в строке действительно записаны только цифры. Для этого, сначала надо разбить строку на список строк по разделителю `','` и проверить каждый элемент полученного списка на десятичное число методом `str.isdigit()`. Что бы учесть правила записи десятичных чисел будем убирать точку перед проверкой строки на десятичное число.

```
>>> line1 = "1, 2, 3, 9.9, 15.1, 7"
>>> line2 = "1, 2, 3, 9.9, 15.1, 7, девять"
>>> all([x.replace('.', '').strip().isdigit() for x in line1.split(',')])
# True
>>> all([x.replace('.', '').strip().isdigit() for x in line2.split(',')])
# False
```

Еще пример со строкой. Допустим нам необходимо узнать, есть ли в строке наличие открытой И закрытой скобки?

```
>>> simbols = ['(', ')']
>>> line1 = "функция 'all()' всегда возвращает 'False' или 'True'"
>>> line2 = "функция any всегда возвращает значение bool"
>>> all([x in line1 for x in simbols])
# True
>>> all([x in line2 for x in simbols])
# False
```

Содержание раздела:

- ОБЗОРНАЯ СТРАНИЦА РАЗДЕЛА
- Функция abs(), абсолютное значение числа
- Функция all(), все элементы True
- Функция any(), хотя бы один элемент True
- Функция ascii(), преобразует строку в ASCII
- Функция bin(), число в двоичную строку
- Класс bool(), логическое значение объекта
- Функция breakpoint(), отладчик кода
- Класс bytearray(), преобразует в массив байтов
- Класс bytes(), преобразует в строку байтов
- Функция callable(), проверяет можно ли вызвать объект
- Функция chr(), число в символ Юникода
- Класс classmethod, делает функцию методом класса
- Функция compile() компилирует блок кода Python
- Класс complex(), преобразует в комплексное число
- Функция delattr(), удаляет атрибут объекта
- Класс dict() создает словарь
- Функция dir(), все атрибуты объекта
- Функция divmod(), делит числа с остатком
- Функция enumerate(), счетчик элементов последовательности
- Функция eval(), выполняет строку-выражение с кодом
- Функция exec(), выполняет блок кода
- Функция filter(), фильтрует список по условию
- Класс float(), преобразует в вещественное число
- Функция format(), форматирует значение переменной
- Класс frozenset(), преобразует в неизменяемое множество
- Функция getattr(), значение атрибута по имени
- Функция globals(), переменные глобальной области
- Функция hasattr(), наличие атрибута объекта
- Функция hash(), хэш-значение объекта
- Функция help(), справка по любому объекту
- Функция hex(), число в шестнадцатеричную строку
- Функция id(), идентификатор объекта
- Функция input(), ввод данных с клавиатуры
- Класс int(), преобразует в тип int
- Функция isinstance(), принадлежность экземпляра к классу
- Функция issubclass(), проверяет наследование класса

- Функция iter(), создает итератор
- Функция len(), количество элементов объекта
- Класс list()
- Функция locals(), переменные локальной области
- Функция map(), обработка последовательности без цикла
- Функция max(), максимальное значение элемента
- Класс memoryview(), ссылка на буфер обмена
- Функция min(), минимальное значение элемента
- Функция next(), следующий элемент итератора
- Класс object(), возвращает безликий объект
- Функция oct(), число в восьмеричную строку
- Функция open(), открывает файл на чтение/запись
- Функция ord(), число символа Unicode
- Функция pow(), возводит число в степень
- Функция print(), печатает объект
- Класс property(), метод класса как свойство
- Класс range(), генерирует арифметические последовательности
- Функция repr(), описание объекта
- Функция reversed(), разворачивает последовательность
- Функция round(), округляет число
- Класс set(), создает или преобразовывает в множество
- Функция setattr(), создает атрибут объекта
- Класс slice(), шаблон среза
- Функция sorted(), выполняет сортировку
- Декоратор staticmethod(), метод класса в статический метод
- Класс str(), преобразует объект в строку
- Функция sum(), сумма последовательности
- Функция super(), доступ к унаследованным методам
- Класс tuple(), создает или преобразует в кортеж
- Класс type(), возвращает тип объекта
- Функция vars(), словарь переменных объекта
- Функция zip(), объединить элементы в список кортежей
- Функция __import__(), находит и импортирует модуль
- Функция aiter(), создает асинхронный итератор
- Функция anext(), следующий элемент асинхронного итератора