Возможности и примеры функции sorted в Python

Функция `sorted()` возвращает новый отсортированный список итерируемого объекта (списка, словаря, кортежа). По умолчанию она сортирует его по возрастанию.

Сортировка строк осуществляется по ASCII-значениям.

- Возвращаемое значение — `List` (список).
- Синтаксис: `sorted(iterable,key=None,reverse=False)`.
- `iterable`: строка, список, кортеж, множество, словарь
- `key` (необязательный параметр): если указать ключ, то сортировка будет выполнена по функции этого ключа.
- `reverse` (необязательный параметр): по умолчанию сортировка выполняется по возрастанию. Если указать `reverse=True`, то можно отсортировать по убыванию.

КОПИРОВАТЬ

```
# Сортировка строки
s2="hello"
print(sorted(s2))  # Вывод:['e', 'h', 'l', 'l', 'o']
print(sorted(s2, reverse=True))   # Вывод:['o', 'l',

# Сортировка списка
l1=[1, 4, 5, 2, 456, 12]
print(sorted(l1))   # Вывод:[1, 2, 4, 5, 12, 456]
print(sorted(l1, reverse=True))   # Вывод:[456, 12,

# Сортировка кортежа
t1=(15, 3, 5, 7, 9, 11, 42)
print(sorted(t1))   # Вывод:[3, 5, 7, 9, 11, 15, 42]
print(sorted(t1, reverse=True))   # Вывод:[42, 15, 1

# Сортировка списка кортежей
t2=[(1, 2), (11, 12), (0, 2), (3, 2)]
print(sorted(t2))   # Вывод:[(0, 2), (1, 2), (3, 2),
print(sorted(t2, reverse=True))   # Вывод:[(11, 12),

# Сортировка множества
s1={1, 4, 3, 6, 2, 8, 11, 32}
print(sorted(s1))   # Вывод:[1, 2, 3, 4, 6, 8, 11, 3
print(sorted(s1, reverse=True))   # Вывод:[32, 11, 8

# Сортировка словаря
d1={2: 'red', 1: 'green', 3: 'blue'}
# Вернется список отсортированных ключей
print(sorted(d1))   # Вывод:[1, 2, 3]
```


НОВОЕ В БЛОГЕ

- Нахождение делителей числа с помощью Python
- Лямбда-функции и анонимные функции в Python
- Когда стоит использовать yield вместо return в Python
- Как извлечь кубический корень в Python

```
# Вернется список отсортированных значений
print(sorted(d1.values()))   # Вывод:['blue', 'greer

# Вернется список кортежей (ключ, значение), отсорти
print(sorted(d1.items()))   # Вывод:[(1, 'green'), (

# Сортировка словаря в обратном порядке
print(sorted(d1, reverse=True))   # Вывод:[3, 2, 1]
print(sorted(d1.values(), reverse=True))   # Вывод:[
print(sorted(d1.items(), reverse=True))   # Вывод:[(
```

Параметр key

Итерируемый объект можно также отсортировать по функции, указанной в параметре `key`. Это может быть:

- Встроенная функция,
- Определенная пользователем функция,
- Лямбда-функция,
- itemgetter,
- attrgetter.

1. Встроенная функция

`len()` — посчитает длину объекта. Если указать `len` в виде параметра key, то сортировка будет выполнена по длине.

КОПИРОВАТЬ

```
# Сортировка словаря на основе функции len
l1 = {'carrot': 'vegetable', 'red': 'color', 'apple'
# Возвращает список ключей, отсортированных по функц
print(sorted(l1, key=len))
# Вывод: ['red', 'apple', 'carrot']

# Возвращает список значений, отсортированных на осн
print(sorted(l1.values(), key=len))
# Вывод: ['fruit', 'color', 'vegetable']

# Сортировка списка на основе функции len
l1 = ['blue', 'green', 'red', 'orange']
print(sorted(l1, key=len))
# Вывод: ['red', 'blue', 'green', 'orange']
```

`abs()` вернет абсолютно значение числа. Если задать `abs` для `key`, то сортировка будет основана на абсолютном значении.

КОПИРОВАТЬ

```
# Сортировка списка по абсолютному значению
l1 = [1, -4, 5, -7, 9, 2]
```


СОДЕРЖАНИЕ

НОВОЕ В БЛОГЕ

- Нахождение делителей числа с помощью Python
- Лямбда-функции и анонимные функции в Python
- Когда стоит использовать yield вместо return в Python
- Как извлечь кубический корень в Python

```
print(sorted(l1, key=abs))
# Вывод: [1, 2, -4, 5, -7, 9]
```

`str.lower()` — конвертирует все символы в верхнем регистре в нижний регистр. В таком случае список будет отсортирован так, будто бы все символы в нижнем регистре.

```
КОПИРОВАТЬ

s1 = "Hello How are you"

# Разбивает строку и сортирует по словам
print(sorted(s1.split()))
# Вывод: ['Hello', 'How', 'are', 'you']

# Разбивает строку и сортирует после применения str.
print(sorted(s1.split(), key=str.lower))
# Вывод: ['are', 'Hello', 'How', 'you']

d1 = {'apple': 1, 'Banana': 2, 'Pears': 3}
# Возвращает список ключей, отсортированный по значе
print(sorted(d1))
# Вывод: ['Banana', 'Pears', 'apple']

# Возвращает список ключей, отсортированный после пр
print(sorted(d1, key=str.lower))
# Вывод: ['apple', 'Banana', 'Pears']
```

2. Пользовательские функции

Также можно указывать свои функции.

Пример №1: по умолчанию сортировка кортежа происходит по первому элементу в нем. Добавим функцию, которая будет возвращать второй элемент кортежа. Теперь и сортировка будет опираться на соответствующий элемент.

```
КОПИРОВАТЬ

# напишем функцию для получения второго элемента
def sort_key(e):
    return e[1]

l1 = [(1, 2, 3), (2, 1, 3), (11, 4, 2), (9, 1, 3)]
# По умолчанию сортировка выполняется по первому эле
print(sorted(l1))
# Вывод: [(1, 2, 3), (2, 1, 3), (9, 1, 3), (11, 4, 2

# Сортировка по второму элементу с помощью функции s
print(sorted(l1, key=sort_key))
# Вывод: [(2, 1, 3), (9, 1, 3), (1, 2, 3), (11, 4, 2
```

СОДЕРЖАНИЕ

НОВОЕ В БЛОГЕ

- Нахождение делителей числа с помощью Python
- Лямбда-функции и анонимные функции в Python
- Когда стоит использовать yield вместо return в Python
- Как извлечь кубический корень в Python

Пример №2: можно сортировать объекты класса с помощью функций. Вот как это происходит:

- У класса `Studen` есть три атрибута: `name`, `rollno`, `grade`.
- Создаются три объекта этого класса: `s1`, `s2`, `s3`.
- Создается список `s4`, который содержит все три объекта.
- Дальше происходит сортировка по этому списку.
- Определенная функция (`sort_key`) возвращает атрибут `rollno`.
- В функции sorted эта функция задана для параметра `key`.
- Теперь `sorted()` возвращает список объектов `Student`, которые отсортированы по значению `rollno`.

```
КОПИРОВАТЬ

class Student:
    def __init__(self, name, rollno, grade):
        self.name = name
        self.rollno = rollno
        self.grade = grade

    def __repr__(self):
        return f"{self.name}-{self.rollno}-{self.gra

# Создание объектов
s1 = Student("Paul", 15, "second")
s2 = Student("Alex", 12, "fifth")
s3 = Student("Eva", 21, "first")
s4 = [s1, s2, s3]

# Сортировка списка объектов
# Создание функции, которая вернет rollno объекта
def sort_key(s):
    return s.rollno

# сортировка списка объектов без ключевого параметра
print(sorted(s4))
# Вывод: TypeError: '>' not supported between instar

# Сортировка списка объектов по атрибуту: rollno
s5 = sorted(s4, key=sort_key)
print(s5)
# Вывод: [Alex-12-fifth, Paul-15-second, Eva-21-firs
```

3. Лямбда-функция

Также в качестве ключа можно задать лямбда-функцию. Сортировка будет выполняться по ней.

Пример №1: сортировка списка объектов класса на основе лямбда-функции, переданной для параметра `key`.

```
КОПИРОВАТЬ
Основы  Уроки  Примеры  Библиотеки  База знаний
```

СОДЕРЖАНИЕ

НОВОЕ В БЛОГЕ

- Нахождение делителей числа с помощью Python
- Лямбда-функции и анонимные функции в Python
- Когда стоит использовать yield вместо return в Python
- Как извлечь кубический корень в Python

```
class Student:
    def __init__(self, name, rollno, grade):
        self.name = name
        self.rollno = rollno
        self.grade = grade

    def __repr__(self):
        return f"{self.name}-{self.rollno}-{self.gra

# Создание объектов
s1 = Student("Paul", 15, "second")
s2 = Student("Alex", 12, "fifth")
s3 = Student("Eva", 21, "first")
s4 = [s1, s2, s3]

# Сортировка списка объектов
# Создание lambda-функции, которая вернет rollno об
s5 = sorted(s4, key=lambda x: x.rollno)
print(s5)
# Вывод: [Alex-12-fifth, Paul-15-second, Eva-21-firs
```

Пример №2: сортировка кортежей по второму элементу. Лямбда-функция просто возвращает его для каждого переданного объекта.

```
КОПИРОВАТЬ

l1 = [( 1, 2, 3), (3, 1, 1), (8, 5, 3), (3, 4, 2)]

# Сортировка по второму элементу в кортеже.
# Lambda-функция возвращает второй элемент в кортеже
print(sorted(l1, key=lambda x: x[1]))
# Вывод: [(3, 1, 1), (1, 2, 3), (3, 4, 2), (8, 5, 3)
```

4. itemgetter

`operator` — это встроенный модуль, включающий много операторов. `itemgetter(n)` создает функцию, которая принимает итерируемый объект (список, кортеж, множество) и возвращает n-элемент из него.

Пример №1: сортировка списка кортежей по третьему элементу в каждом из них. `itemgetter()` возвращает функцию, которая получает третий элемент в кортеже. По умолчанию если `key` не указать, то сортировка будет выполнена по первому элементу кортежа.

```
КОПИРОВАТЬ

from operator import itemgetter

l1 = [(1, 2, 3), (3, 1, 1), (8, 5, 3), (3, 4, 2)]

# Сортировка по третьему элементу в кортеже
```


СОДЕРЖАНИЕ

НОВОЕ В БЛОГЕ

- Нахождение делителей числа с помощью Python
- Лямбда-функции и анонимные функции в Python
- Когда стоит использовать yield вместо return в Python
- Как извлечь кубический корень в Python

```
print(sorted(l1, key=itemgetter(2)))
# Вывод: [(3, 1, 1), (1, 2, 3), (3, 4, 2), (8, 5, 3
```

Пример №2: сортировка списка словарей по ключу `age`.

```
from operator import itemgetter

# Создание списка словарей
d1 = [{'name': 'Paul', 'age': 30},
      {'name': 'Alex', 'age': 7},
      {'name': 'Eva', 'age': 3}]

# Сортировка по key(age) в словаре
print(sorted(d1, key=itemgetter('age')))
# Вывод: [{'name': 'Eva', 'age': 3}, {'name': 'Alex'
```

Пример №3: сортировка значений словаря с помощью `itemgetter`.

```
from operator import itemgetter

d = {'a': [1, 2, 3], 'b': [3, 4, 5], 'c': [1, 1, 2]}
print(sorted(d.values(), key=itemgetter(1)))
# Вывод: [[1, 1, 2], [1, 2, 3], [3, 4, 5]]
```

5. attrgetter

```
operator.attrgetter(attr)
operator.attrgetter(*attrs)
```

Возвращает вызываемый объект, который получает `attr` операнда. Если требуется больше одного аргумента, то возвращается кортеж атрибутов. Названия также могут включать троеточия.

Пример: сортировка списка объекта классов с помощью `attrgetter`. Функция `attrgetter` может получить атрибут объекта, а функция `sorted` будет выполнять сортировку на основе этого атрибута.

```
from operator import attrgetter

class Student:
    def __init__(self, name, rollno, grade):
```


НОВОЕ В БЛОГЕ

- Нахождение делителей числа с помощью Python
- Лямбда-функции и анонимные функции в Python
- Когда стоит использовать yield вместо return в Python
- Как извлечь кубический корень в Python

```
        self.name = name
        self.rollno = rollno
        self.grade = grade

    def __repr__(self):
        return f"{self.name}-{self.rollno}-{self.gra

# Создание объектов
s1 = Student("Paul", 15, "second")
s2 = Student("Alex", 12, "fifth")
s3 = Student("Eva", 21, "first")
s4 = [s1, s2, s3]

# Сортировка списка объектов по атрибуту: rollno
s5 = sorted(s4, key=attrgetter('rollno'))
print(s5)
# Вывод: [Alex-12-fifth. Paul-15-second. Eva-21-firs
```

Многоуровневая сортировка

- С помощью `itemgetter` можно также выполнять сортировку на нескольких уровнях.

В следующем примере сортировка значений словаря будет выполнена по второму элементу. Если же некоторые из них будут равны, то сортировка пройдет по третьему элементу.

КОПИРОВАТЬ

```
from operator import itemgetter

d = {'a': [1, 2, 9], 'b': [3, 2, 5], 'c': [1, 1, 2]}

# Сортировка по второму, затем по третьему элементу
print(sorted(d.values(), key=itemgetter(1, 2)))
# Вывод: [[1, 1, 2], [3, 2, 5], [1, 2, 9]]
```

- Многоуровневая сортировка с помощью лямбда-функции в `key`.

А в этом примере сортировка кортежа будет проходить по второму элементу. В случае равенства используется третий.

КОПИРОВАТЬ

```
l1 = [(1, 2, 3), (4, 2, 0), (2, 1, 7)]
# Сортировка по второму элементу
# Если второй элемент равен, сортировка по третьему
print(sorted(l1, key=lambda x: x[1:3]))
```

Сортировка смешанных типов данных

СОДЕРЖАНИЕ

НОВОЕ В БЛОГЕ

- Нахождение делителей числа с помощью Python
- Лямбда-функции и анонимные функции в Python
- Когда стоит использовать yield вместо return в Python
- Как извлечь кубический корень в Python

Функция `sorted` не поддерживает смешанные типы данных. При попытке выполнить сортировку она вернет ошибку TypeError.

```
# список, содержащий смешанные типы данных
l1 = [1, 'a', 2]
print(sorted(l1))
# Вывод: TypeError: '>' not supported between instar
```

КОПИРОВАТЬ

Выводы

1. При сортировке списка объектов класса параметр `key` является обязательным. Если его не указать, то вернется ошибка TypeError.
2. Для сортировки списка объектов класса можно использовать в качестве параметра `key` определенную пользователем функцию, лямбда-функцию или `attrgetter`.
3. Функция `sorted()` возвращает список.
4. Метода `sort` отсортирует сам оригинальный список. Но функция `sorted` возвращает новый список, не изменяя оригинальный.
5. Метод `sort` используется только для сортировки списка. Его нельзя использовать для строки, кортежа или словаря.

теги **для начинающих**

Максим

Я создал этот блог в 2018 году, чтобы распространять полезные учебные материалы, документации и уроки на русском. На сайте опубликовано множество статей по основам python и библиотекам, уроков для начинающих и примеров написания программ.
Мои контакты: Почта

Статьи по теме

СОДЕРЖАНИЕ

НОВОЕ В БЛОГЕ

- Нахождение делителей числа с помощью Python
- Лямбда-функции и анонимные функции в Python
- Когда стоит использовать yield вместо return в Python
- Как извлечь кубический корень в Python

Лямбда-функции и анонимные функции в Python

Когда стоит использовать yield вместо return в Python

Как извлечь кубический корень в Python

Python цикл for — for i in range

Полное руководство по замене элементов списка на Python

Функции в Python

СОДЕРЖАНИЕ

НОВОЕ В БЛОГЕ

- Нахождение делителей числа с помощью Python
- Лямбда-функции и анонимные функции в Python
- Когда стоит использовать yield вместо return в Python
- Как извлечь кубический корень в Python