Python Bootcamp & Masterclass

Sorting Lists

gknxt

sort( )

The `sort(key=None, reverse=False)` method sorts the list elements in place using `<` comparisons between items. If any comparison operations fail, the entire sort operation will fail (and the list will likely be left in a partially modified state) The optional *key* and *reverse* arguments can be used to customize the sorting.

```
a = [4, -1, 5, 3, 2]
a.sort()
a
```

```
[-1, 2, 3, 4, 5]
```

```
b = [4, -1, 5, 3, True]
b.sort()
b
```

```
[-1, True, 3, 4, 5]
```

```
c = [4, -1, 5, 3, None]
c.sort()
c
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_26432/2280025382.py in <module>
      1 c = [4, -1, 5, 3, None]
----> 2 c.sort()
      3 c

TypeError: '<' not supported between instances of 'NoneType' and 'int'
```

```
d = ['is', 'Life', 'beautiful']
d.sort() # compared lexicographically - uppercase letter comes before lower case letters
d
```

```
['Life', 'beautiful', 'is']
```

```
e = [['red', 1], ['blue', 2], ['red', 2], ['blue', 1]]
e.sort()
e
```

```
[['blue', 1], ['blue', 2], ['red', 1], ['red', 2]]
```

```
g = ['101.0', '20.9', '13.4', '106.4']
g.sort()
g
```

```
['101.0', '106.4', '13.4', '20.9']
```

The *sort(key=None, reverse=False)* method sorts the list elements in the ascending order by default. If the descending order is needed, the optional *reverse* arguments can be used *(reverse=True)*

```
a = [4, -1, 5, 3, 2]
a.sort(reverse=True)
a
```

```
[5, 4, 3, 2, -1]
```

```
b = [4, -1, 5, 3, True]
b.sort(reverse=True)
b
```

```
[5, 4, 3, True, -1]
```

```
f = ['is', 'Life', 'beautiful']
f.sort(reverse=True)
f
```

```
['is', 'beautiful', 'Life']
```

```
e = [['red', 1], ['blue', 2], ['red', 2], ['blue', 1]]
e.sort(reverse=True)
e
```

```
[['red', 2], ['red', 1], ['blue', 2], ['blue', 1]]
```

The *sort(key=None, reverse=False)* method's *key* argument can be used to specify a function (or other callable) to be called on each list element prior to making comparisons.

```
f = ['banana', 'apple', 'mango', 'kiwi', 'pineapple']
f.sort(key=len)
f
```

```
['kiwi', 'apple', 'mango', 'banana', 'pineapple']
```

```
f = ['is', 'Life', 'beautiful']
f.sort(key=str.lower)
f
```

```
['beautiful', 'is', 'Life']
```

```
g = ['101.0', '20.9', '13.4', '106.4']
g.sort(key=float)
g
```

```
['13.4', '20.9', '101.0', '106.4']
```

sorted( )

The `sorted(iterable, key=None, reverse=False)` built-in function sorts the `iterable` (list in this case) using `<` comparisons between items and returns a new sorted `list` (even if the iterable is not a list) If any comparison fails, the entire sort operation will fail.

```
s = ['is', 'Life', 'beautiful']
sorted(s)
```

```
['Life', 'beautiful', 'is']
```

```
t = [['red', 1], ['blue', 2], ['red', 2], ['blue', 1]]
sorted(t)
```

```
[['blue', 1], ['blue', 2], ['red', 1], ['red', 2]]
```

```
u = ['101.0', '20.9', '13.4', '106.4']
sorted(u)
```

```
['101.0', '106.4', '13.4', '20.9']
```

The `sorted(iterable, key=None, reverse=False)` function sorts the list elements in the ascending order by default. If the descending order is needed, the optional `reverse` arguments can be used `(reverse=True)`

```
v = [4, -1, 5, 3, 2]
sorted(v, reverse=True)
```

```
[5, 4, 3, 2, -1]
```

```
w = [4, -1, 5, 3, True]
sorted(w, reverse=True)
```

```
[5, 4, 3, True, -1]
```

```
x = [['red', 1], ['blue', 2], ['red', 2], ['blue', 1]]
sorted(x, reverse=True)
```

```
[['red', 2], ['red', 1], ['blue', 2], ['blue', 1]]
```

```
y = ['101.0', '20.9', '13.4', '106.4']
sorted(y, reverse=True)
```

```
['20.9', '13.4', '106.4', '101.0']
```

The `sorted(iterable, key=None, reverse=False)` function's *key* argument can be used to specify a function (or other callable) to be called on each list element prior to making comparisons.

```
a = ['banana', 'apple', 'mango', 'kiwi', 'pineapple']
sorted(a, key=len)
```

```
['kiwi', 'apple', 'mango', 'banana', 'pineapple']
```

```
b = ['is', 'Life', 'beautiful']
sorted(b, key=str.lower)
```

```
['beautiful', 'is', 'Life']
```

```
c = ['101.0', '20.9', '13.4', '106.4']
sorted(c, key=float)
```

```
['13.4', '20.9', '101.0', '106.4']
```

Online Resources

For best python resources, please visit:

gknxt.com/python/

gknxt

Python
Bootcamp
& Masterclass

Thank You

for your Rating & Review

gknxt