sorted() function in Python

What is the sorted function in Python used for? Here's everything you need to know about the sort function in Python.

n .

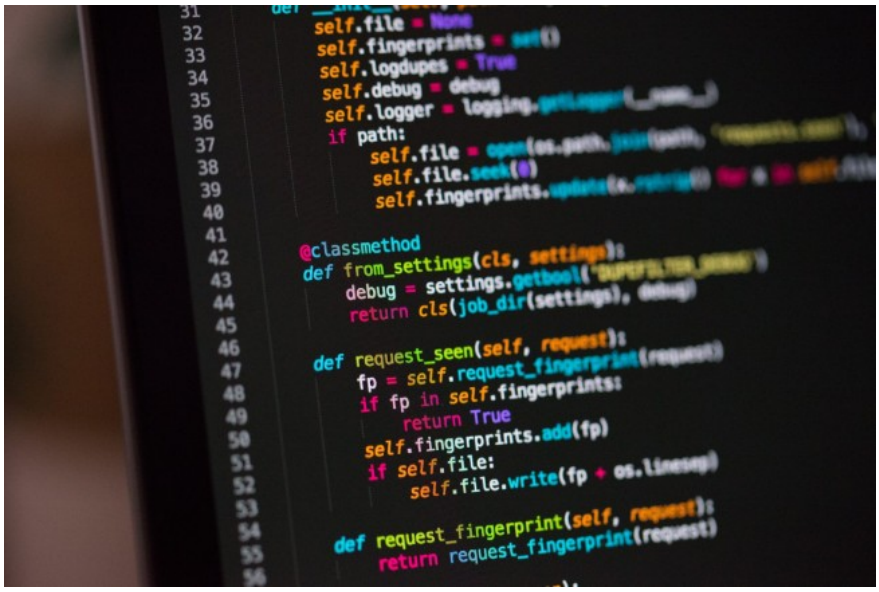
Syntax of the sorted() function

```
sorted(??i t??ng, key=None, reverse=False)
```

Parameters of the sorted() function

sorted() can have up to 3 parameters:

1. Object: Each string (string, , list) or a collection (set, dictionary,) or any object.
2. reverse (optional): If value is True, the list will be sorted in reverse (or descending) fashion. If not declared, the default value is False.
3. **key** (optional): If present, the **sorted()** function compares the value with the key then sorts. The default value is None.



Advantages of the sorted() function in Python

1. **Ease of use** : The sorted function is very easy to use. It only needs one required parameter, which can be repeated, and other parameters, including key and reverse, can be used to customize the sort.

2. **Immutability** : The sorted function in Python creates a new sorted list that does not change the passed string. This feature can be useful when we want the data to be sorted and not affect the original data.
3. **Classification customization** : The key parameter can be used to set classification criteria. By using this feature, you can easily customize the sorting process. In addition, programmers can also use the reverse parameter to reverse order the data.

Applications of the sorted() function in Python:

1. Data analysis helps users easily create overview reports.
2. Customize classification according to specific criteria.
3. Display results in a specific sorted manner.

Example 1: Sort string, lish and tube

Please see the following code:

```
# S?p x?
p list py_list = ['q', 'u', 'a', 'n', 't', 'r', 'i', 'm', 'a', 'n', 'g'] print(s
?p x?p string py_string = 'TipsMake' print(sorted(py_string)) # S?p x?
p tuple py_tuple = ('q', 'u', 'a', 'n', 't', 'r', 'i', 'm', 'a', 'n', 'g') print
```

When running the program, the results obtained are:

```
['a', 'a', 'g', 'i', 'm', 'n', 'n', 'q', 'r', 't', 'u'] ['Q', 'a', 'a', 'g', 'i',
```

Note: A list also has a s **sort()** function that works similarly to **the sorted()** function . The only difference is that the **sort()** function does not return any value and changes the original list.

Example 2: Sort in descending order

The sorted() function has a reverse parameter, and when you set reverse=True, the list will be sorted in descending order.

Consider the following lines of code:

```
# S?p x?
p set py_set = {'q', 'u', 'a', 'n', 't', 'r', 'i', 'm', 'a', 'n', 'g'} print(sor
?p x?
p dictionary py_dict = {'e': 1, 'a': 2, 'u': 3, 'o': 4, 'i': 5} print(sorted(py_
?p x?
p frozen set frozen_set = frozenset(('q', 'u', 'a', 'n', 't', 'r', 'i', 'm', 'a',
```

When running the program, the returned results are:

```
['u', 't', 'r', 'q', 'n', 'm', 'i', 'g', 'a'] ['u', 'o', 'i', 'e', 'a'] ['u', 't
```

The key parameter of the sorted() function in Python

If you want to sort your own style, the **sorted()** function also accepts a **key** as an optional parameter.

You can sort objects based on the returned value of **key** . For example:

```
sorted(iterable, key=len)
```

Here, **len()** is a built-in Python function that counts the length of an object.

The list is sorted based on the length of the element, from lowest number to highest.

Example 3: Sort a list using the sorted() function containing the key

Please see the code:

```
# X??p xê?p d??a trên phâ?n t?? th??
 2 def take_second(elem): return elem[1] # list ngâ?
u nhiên random = [(2, 2), (3, 4), (4, 1), (1, 3)] # s??p xê?p list v??
i key sorted_list = sorted(random, key=take_second) # hiê?n thi?
 list print('List ?a? ????c s??p xê?p:', sorted_list)
```

After running the program, the results obtained are:

```
List ?a? ????c s??p: [(4, 1), (2, 2), (1, 3), (3, 4)]
```

Example 4: Sorting with multiple keys

Initially we have a list like this:

```
# List kê?t qua? thi ho?c ky? cu?a ca?c sinh viên # Ca?c tha?nh phâ?n cu?
a list: (Tên sinh viên, ?iê?m sô? ?a?t ????c theo thang 100, Tuô?
i) participant_list = [ ('Lê An', 50, 18), ('Trâ?n Bi?nh', 75, 12), ('Trâ?
n Tâm', 75, 20), ('Thanh Lam', 90, 22), ('Vu? Lan', 45, 12) ]
```

Now we need to sort the list of students by score from high to low. In this case, if students have the same score, the younger student will be placed in front.

We can do this with the multi-key **sorted()** function by putting the numbers into a **tuple** .

Two **tuples** can be compared by comparing elements from first to last. If there is an equal factor, the second factor will be compared.continued until the end.

```
>>> (1,3) > (1, 4) False >>> (1, 4)  (2,2) True >>> (1, 4, 1)  (2, 1) True
```

Using this approach, we can build the comparison code as follows:

```
# List kê?t qua? thi ho?c ky? cu?a ca?c sinh viên # Ca?c tha?nh phâ?n cu?
a list: (Tên sinh viên, ?iê?m sô? ?a?t ????c theo thang 100, Tuô?
i) participant_list = [ ('Lê An', 50, 18), ('Trâ?n Bi?nh', 75, 12), ('Trâ?
n Tâm', 75, 20), ('Thanh Lam', 90, 22), ('Vu?
 Lan', 45, 12) ] def sorter(item): error = 100 - item[1] age = item[2] return (e
```

When running the program, the results obtained are:

```
[('Thanh Lam', 90, 22), ('Trâ?n Bi?nh', 75, 12), ('Trâ?
n Tâm', 75, 20), ('Lê An', 50, 18), ('Vu? Lan', 45, 12)
```

Because the sort function is short and only requires one line, the lambda function is used inside the key instead of passing it out as a separate function.

The above code can be rewritten using a lambda function in the following way:

```
# List kê?t qua? thi ho?c ky? cu?a ca?c sinh viên # Ca?c tha?nh phâ?n cu?
a list: (Tên sinh viên, ?iê?m sô? ?a?t ????c theo thang 100, Tuô?
i) participant_list = [ ('Lê An', 50, 18), ('Trâ?n Bi?nh', 75, 12), ('Trâ?
n Tâm', 75, 20), ('Thanh Lam', 90, 22), ('Vu?
 Lan', 45, 12) ] sorted_list = sorted(participant_list, key=lambda item: (100-it
```

The results obtained are still:

```
[('Thanh Lam', 90, 22), ('Trâ?n Bi?nh', 75, 12), ('Trâ?
n Tâm', 75, 20), ('Lê An', 50, 18), ('Vu? Lan', 45, 12)
```

Hope you soon get familiar with the sorted() function in Python and don't forget to visit the website below to learn more useful Python functions.