16.5. Breaking Ties: Second Sorting

What happens when two items are "tied" in the sort order? For example, suppose we sort a list of words by their lengths. Which five letter word will appear first?

The answer is that the python interpreter will sort the tied items in the same order they were in before the sorting.

What if we wanted to sort them by some other property, say alphabetically, when the words were the same length? Python allows us to specify multiple conditions when we perform a sort by returning a tuple from a key function.

First, let's see how python sorts tuples. We've already seen that there's a built-in sort order, if we don't specify any key function. For numbers, it's lowest to highest. For strings, it's alphabetic order. For a sequence of tuples, the default sort order is based on the default sort order for the first elements of the tuples, with ties being broken by the second elements, and then third elements if necessary, etc. For example,

Save & Run Original - 1 of 1 Show in CodeLens

```
tups = [('A', 3, 2),
        ('C', 1, 4),
        ('B', 3, 1),
        ('A', 2, 4),
        ('C', 1, 2)]
for tup in sorted(tups):
    print(tup)

```

```
('A', 2, 4)
('A', 3, 2)
('B', 3, 1)
('C', 1, 2)
('C', 1, 4)
```

Activity: 1 -- ActiveCode (ac18_5_0)

In the code below, we are going to sort a list of fruit words first by their length, smallest to largest, and then alphabetically to break ties among words of the same length. To do that, we have the key function return a tuple whose first element is the length of the fruit's name, and second element is the fruit name itself.

Save & Run Original - 1 of 1 Show in CodeLens

```
fruits = ['peach', 'kiwi', 'apple', 'blueberry', 'papaya', 'mango', 'pear']
new_order = sorted(fruits, key=lambda fruit_name: (len(fruit_name), fruit_name))
for fruit in new_order:
    print(fruit)

```

```
kiwi
pear
apple
mango
peach
papaya
blueberry
```

Activity: 2 -- ActiveCode (ac18_5_1)

Here, each word is evaluated first on it's length, then by its alphabetical order. Note that we could continue to specify other conditions by including more elements in the tuple.

What would happen though if we wanted to sort it by largest to smallest, and then by alphabetical order?

Save & Run Original - 1 of 1 Show in CodeLens

```
fruits = ['peach', 'kiwi', 'apple', 'blueberry', 'papaya', 'mango', 'pear']
new_order = sorted(fruits, key=lambda fruit_name: (len(fruit_name), fruit_name), re
for fruit in new_order:
    print(fruit)

```

```
blueberry
papaya
peach
mango
apple
pear
kiwi
```

Activity: 3 -- ActiveCode (ac18_5_2)

Do you see a problem here? Not only does it sort the words from largest to smallest, but also in reverse alphabetical order! Can you think of any ways you can solve this issue?

One solution is to add a negative sign in front of `len(fruit_name)`, which will convert all positive numbers to negative, and all negative numbers to positive. As a result, the longest elements would be first and the shortest elements would be last.

Save & Run | Original - 1 of 1 | Show in CodeLens

```
fruits = ['peach', 'kiwi', 'apple', 'blueberry', 'papaya', 'mango', 'pear']
new_order = sorted(fruits, key=lambda fruit_name: (-len(fruit_name), fruit_name))
for fruit in new_order:
    print(fruit)

```

```
blueberry
papaya
apple
mango
peach
kiwi
pear
```

Activity: 4 -- ActiveCode (ac18_5_3)

We can use this for any numerical value that we want to sort, however this will not work for strings.

Check Your Understanding

sort-5-1: What will the sorted function sort by?

```
weather = {'Reykjavik': {'temp':60, 'condition': 'rainy'},
           'Buenos Aires': {'temp': 55, 'condition': 'cloudy'},
           'Cairo': {'temp': 96, 'condition': 'sunny'},
           'Berlin': {'temp': 89, 'condition': 'sunny'},
           'Caloocan': {'temp': 78, 'condition': 'sunny'}}

sorted_weather = sorted(weather, key=lambda w: (w, weather[w]['temp']))
```

- (•) A. first city name (alphabetically), then temperature (lowest to highest)
- ( ) B. first temperature (highest to lowest), then city name (alphabetically)
- ( ) C. first city name (alphabetically), then temperature (highest to lowest)
- ( ) D. first temperature (lowest to highest), then city name (alphabetically)

Check me | Compare me

✔ Correct! First we sort alphabetically by city name, then by the temperature, from lowest to highest.

Activity: 5 -- Multiple Choice (question18_5_1)

sort-5-2: What how will the following data be sorted?

```
weather = {'Reykjavik': {'temp':60, 'condition': 'rainy'},
           'Buenos Aires': {'temp': 55, 'condition': 'cloudy'},
           'Cairo': {'temp': 96, 'condition': 'sunny'},
           'Berlin': {'temp': 89, 'condition': 'sunny'},
           'Caloocan': {'temp': 78, 'condition': 'sunny'}}

sorted_weather = sorted(weather, key=lambda w: (w, -weather[w]['temp']), reverse=True)
```

- (•) A. first city name (reverse alphabetically), then temperature (lowest to highest)

B. first temperature (highest to lowest), then city name (alphabetically)

C. first city name (reverse alphabetically), then temperature (highest to lowest)

D. first temperature (lowest to highest), then city name (alphabetically)

E. first city name (alphabetically), then temperature (lowest to highest)

Check me Compare me

✔ Correct! In this case, the reverse parameter will cause the city name to be sorted reverse alphabetically instead of alphabetically, and it will also negate the negative sign in front of the temperature.

Activity: 6 -- Multiple Choice (question18_5_2)

You have attempted 7 of 6 activities on this page

16.4. Sorting a Dictionary">
16.6. When to use a Lambda Expression">
orting a Dictionary">
✔ Completed.
16.6. When to use a Lambda Expression">Next Section - 16.6. When to use a Lambda Expression

| Back to top