16.3. Optional key parameter

If you want to sort things in some order other than the “natural” or its reverse, you can provide an additional parameter, the key parameter. For example, suppose you want to sort a list of numbers based on their absolute value, so that -4 comes after 3? Or suppose you have a dictionary with strings as the keys and numbers as the values. Instead of sorting them in alphabetic order based on the keys, you might like to sort them in order based on their values.

First, let’s see an example, and then we’ll dive into how it works.

First, let’s define a function absolute that takes a number and returns its absolute value. (Actually, python provides a built-in function `abs` that does this, but we are going to define our own, for reasons that will be explained in a minute.)

Save & Run 3/18/2020, 4:04:16 PM - 3 of 3 Show in CodeLens

```
L1 = [1, 7, 4, -2, 3]

def absolute(x):
    if x >= 0:
        return x
    else:
        return -x

print(absolute(3))
print(absolute(-119))

for y in L1:
    print(absolute(y))

```

```
3
119
1
7
4
2
3
```

ActiveCode (ac18_3_1)

Now, we can pass the absolute function to sorted in order to specify that we want the items sorted in order of their absolute value, rather than in order of their actual value.

Save & Run 3/18/2020, 4:04:18 PM - 3 of 3 Show in CodeLens

```
L1 = [1, 7, 4, -2, 3]

def absolute(x):
    if x >= 0:
        return x
    else:
        return -x

L2 = sorted(L1, key=absolute)
print(L2)

#or in reverse order
print(sorted(L1, reverse=True, key=absolute))

```

```
[1, -2, 3, 4, 7]
[7, 4, 3, -2, 1]
```

ActiveCode (ac18_3_2)

What's really going on there? We've done something pretty strange. Before, all the values we have passed as parameters have been pretty easy to understand: numbers, strings, lists, Booleans, dictionaries. Here we have passed a function object: absolute is a variable name whose value is the function. When we pass that function object, it is *not* automatically invoked. Instead, it is just bound to the formal parameter key of the function sorted.

We are not going to look at the source code for the built-in function sorted. But if we did, we would find somewhere in its code a parameter named key with a default value of None. When a value is provided for that parameter in an invocation of the function sorted, it has to be a function. What the sorted function does is call that key function once for each item in the list that's getting sorted. It associates the result returned by that function (the absolute function in our case) with the original value. Think of those associated values as being little post-it notes that decorate the original values. The value 4 has a post-it note that says 4 on it, but the value -2 has a post-it note that says 2 on it. Then the sorted function rearranges the original items in order of the values written on their associated post-it notes.

To illustrate that the absolute function is invoked once on each item, during the execution of sorted, I have added some print statements into the code.

Save & Run Original - 1 of 1 Show in CodeLens

```
L1 = [1, 7, 4, -2, 3]

def absolute(x):
    print("--- figuring out what to write on the post-it note for " + st
```

```
    if x >= 0:
        return x
    else:
        return -x

print("About to call sorted")
L2 = sorted(L1, key=absolute)
print("Finished execution of sorted")
print(L2)

```

```
About to call sorted
--- figuring out what to write on the post-it note for 1
--- figuring out what to write on the post-it note for 7
--- figuring out what to write on the post-it note for 4
--- figuring out what to write on the post-it note for -2
--- figuring out what to write on the post-it note for 3
Finished execution of sorted
[1, -2, 3, 4, 7]
```

ActiveCode (ac18_3_3)

Note that this code never explicitly calls the absolute function at all. It passes the absolute function as a parameter value to the sorted function. Inside the sorted function, whose code we haven't seen, that function gets invoked.

Note

It is a little confusing that we are reusing the word *key* so many times. The name of the optional parameter is `key` . We will usually pass a parameter value using the keyword parameter passing mechanism. When we write `key=some_function` in the function invocation, the word key is there because it is the name of the parameter, specified in the definition of the sort function, not because we are using keyword-based parameter passing.

Check Your Understanding

1. You will be sorting the following list by each element's second letter, a to z. Create a function to use when sorting, called `second_let` . It will take a string as input and return the second letter of that string. Then sort the list, create a variable called `sorted_by_second_let` and assign the sorted list to it. Do not use lambda.

Save & Run 3/18/2020, 5:36:10 PM - 32 of 32 Show in CodeLens

```
ex_lst = ['hi', 'how are you', 'bye', 'apple', 'zebra', 'dance']

def second_let(s):
    return s[1]

sorted_by_second_let = sorted(ex_lst, key=second_let)
print(sorted_by_second_let)
```

```
['dance', 'zebra', 'hi', 'how are you', 'apple', 'bye']
```

ActiveCode (ac18_3_4)

Expand Differences Expand Differences

Result	Actual Value	Expected Value	Notes
Pass	['dan...bye']	['dan...bye']	Testing that sorted_by_second_let has the correct value.
Pass	'1'	'1'	Testing that the second_let function returns the second letter in a string.
Pass	'lambda'	'ex_ls..._let)'	Checking that you did *not* use a lambda (Don't worry about actual and expected values).

You passed: 100.0% of the tests

2. Below, we have provided a list of strings called `nums` . Write a function called `last_char` that takes a string as input, and returns only its last character. Use this function to sort the list `nums` by the last digit of each number, from highest to lowest, and save this as a new list called `nums_sorted` .

Save & Run 3/18/2020, 5:46:52 PM - 14 of 14 Show in CodeLens

```
nums = ['1450', '33', '871', '19', '14378', '32', '1005', '44', '8907',

def last_char(x):
    return x[-1]

nums_sorted = sorted(nums, reverse = True, key=last_char)
print(last_char)
```

```
<function last_char>
```

ActiveCode (ac18_3_5)

Expand Differences

Result	Actual Value	Expected Value	Notes
Pass	['19'...450']	['19'...450']	Testing that nums_sorted was created correctly.
Pass	's'	's'	Testing the function last_char on input 'pants'.

You passed: 100.0% of the tests

3. Once again, sort the list `nums` based on the last digit of each number from highest to lowest. However, now you should do so by writing a lambda function. Save the new list as `nums_sorted_lambda` .

Save & Run 3/18/2020, 5:51:44 PM - 7 of 7 Show in CodeLens

```
nums = ['1450', '33', '871', '19', '14378', '32', '1005', '44', '8907', '1
def last_char(x):
    return x[-1]
nums_sorted_lambda = sorted(nums, reverse = True, key=lambda x:x[-1])
print(last_char)

```

```
<function last_char>
```

ActiveCode (ac18_3_6)

Expand Differences Expand Differences

Result	Actual Value	Expected Value	Notes
Pass	['19'...450']	['19'...450']	Testing that nums_sorted_lambda was created correctly.
Pass	'lambda'	'nums ...har)\n'	Testing your code (Don't worry about actual and expected values).

You passed: 100.0% of the tests