**Sorting Lists, Tuples, and Objects**

lunes, 3 de octubre de 2022

10:53 a. m.

Source: [Python Tutorial: Sorting Lists, Tuples, and Objects](https://www.youtube.com/watch?v=D3JvDWO-BY4)

There are mainly two ways to sort a *List*, via *List*.sort( *key = kwargs, reverse = False* ) method and the built-in sorted( *iter, key = kwargs, reverse = False*) function.

The difference between the two are that *List*.sort() returns a None, meaning that this list method sorts in place the list but does not return anything. And the sorted() function actually returns a new object sorted leaving the original object intact.

Now, as *List*.sort() is a class method, to sort other iterables like tuples, dictionaries or anything else the sorted() function is in place.

sorted() would turn any iterable into a list sorted with the given parameters.

Both sorted and sort have the *key* optional argument that essentially would apply the same criteria to each element in order to be sorted out.

Sorting a list of numbers in ascending order ignoring the sign

import math

li = [1, -2, 15, -78, math.pi, 100 ]

print(sorted(li, key = abs, reverse = False)) = [1, -2, 3.141592653589793, 15, -78, 100]

Sorting a String in descending order ignoring case sensitivity

li = "This is a Test to check how Sorted() function works"

li = sorted(li, key = str.lower, reverse = True)

print(li) = ['w', 'w', 'u', 'T', 'T', 't', 't', 't', 't', 's', 's', 's', 'S', 's', 'r', 'r', 'o', 'o', 'o', 'o', 'o', 'n', 'n', 'k', 'k', 'i', 'i', 'i', 'h', 'h', 'h', 'f', 'e', 'e', 'e', 'd', 'c', 'c', 'c', 'a', ')', '(', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']

Sorting a list of lists with elements sorted

import random

li = [ [random.randint(1, 5) for x in range(5)], [random.randint(1, 5) for x in range(5)], [random.randint(1, 5) for x in range(5)]]

print(li) = [[4, 3, 4, 4, 4], [3, 1, 4, 5, 3], [5, 1, 2, 2, 1]]

for i in li:

    i.sort()

print(li) = [[3, 4, 4, 4, 4], [1, 3, 3, 4, 5], [1, 1, 2, 2, 5]]

li = sorted(li, reverse=False)

print(li) = [[1, 1, 2, 2, 5], [1, 3, 3, 4, 5], [3, 4, 4, 4, 4]]

Sorting a Class objects list by attributes

class Employee():

    def \_\_init\_\_(self, name, age, salary):

        self.name = name

        self.age = age

        self.salary = salary

    def \_\_repr\_\_(self):

        return '({}, {}, ${})'.format(self.name, self.age, self.salary)

e1 = Employee('Juan', 30, 30000)

e2 = Employee('Jerry', 29, 0)

e3 = Employee('Vic', 31, 7000)

employees = [e1, e2, e3]

s\_employees = sorted(employees)  # if we try this, an error would be raised

**Se produjo una excepción: TypeError**

'<' not supported between instances of 'Employee' and 'Employee'

File "C:\Users\USUARIO\GR\Software Development\Learning\Python\List, Tuples and Sets\Sorting iterables.py", line 59, in <module> s\_employees = sorted(employees)

This means that the interpreter does not how to compare one element to the other without a key.

To tackle this down, we're creating a new little function to call for one of the attributes of those objects and rewrite the sorted function at the end.

def e\_sort(emp):

    return emp.name

s\_employees = sorted(employees, key = e\_sort)

print(s\_employees) = [(Jerry, 29, $0), (Juan, 30, $30000), (Vic, 31, $7000)]

The criteria of the sort by attribute could be changed to another attribute just changing the e\_sort function. Let's try by age in descending order

def e\_sort(emp):

    return emp.age

s\_employees = sorted(employees, key = e\_sort, reverse = True)

print(s\_employees) = [(Vic, 31, $7000), (Juan, 30, $30000), (Jerry, 29, $0)]

Now, let's try by salary in ascending order

def e\_sort(emp):

    return emp.salary

s\_employees = sorted(employees, key = e\_sort, reverse = False)

print(s\_employees) = [(Jerry, 29, $0), (Vic, 31, $7000), (Juan, 30, $30000)]

Now, the same sorting a Class objects list by attributes, but without defining a intermediate function but using lambda functions instead.

s\_employees = sorted(employees, key = lambda x: x.name, reverse = False)

print(s\_employees) = [(Jerry, 29, $0), (Juan, 30, $30000), (Vic, 31, $7000)]

or

s\_employees = sorted(employees, key = lambda x: x.salary, reverse = True)

print(s\_employees) = [(Juan, 30, $30000), (Vic, 31, $7000), (Jerry, 29, $0)]

Finally there is a function from the module operator that could also be useful to do the same.

from operator import attrgetter

s\_employees = sorted(employees, key = attrgetter('age'), reverse = True)

print(s\_employees) = [(Vic, 31, $7000), (Juan, 30, $30000), (Jerry, 29, $0)]