cmp_to_key() Python3

Maria Assiduo

Introduction

After installing PyCBC with the new environment conda igwn-py39-20210916, trying to run banksim I got the error:

TypeError: 'cmp' is an invalid keyword argument for sort()

This is an incompatibility error of the cmp function between versions of Python 2 → 3

Next, some slides on the syntax and explanation of the use of the sorted() and cmp functions in Python, and the changes they have undergone to be functional in Python3.

Python sorted()

The sorted() function sorts the elements of an iterable data into a specific order and returns the sorted iterable as a list.

Syntax

sorted (iterable, key = function, reverse = optional)

- iterable: a sequence (string, tuple, list) or collection or any other iterator.
- key: a function that serves as a key for sorting comparison
- reverse (optional): if True the sorted list is inverted (or sorted in descending order). The default is False.

In Python 2 sort() or sorted() functions had a cmp parameter which determines the sort order. The argument for cmp is a function that returns a negative, zero, or positive result depending on the order of its two arguments.

```
def cmp(x, y):

    """
    Replacement for built-in function cmp that was removed in Python 3

    Compare the two objects x and y and return an integer according to
    the outcome. The return value is negative if x < y, zero if x == y
    and strictly positive if x > y.
    """“


    return (x > y) - (x < y)
```

https://portingguide.readthedocs.io/en/latest/comparisons.html

In Python 3.0 and above, the function cmp has been deprecated and a new function cmp_to_key() has been introduced.

Definition

- **cmp_to_key()** uses a key to compare elements
- It is built into **functools module**, thus functools has to be imported first in order to use the function
- Used with tools that accept key functions such as min(), max(), sorted() etc.
- **Takes only one argument which strictly should be a callable**
- **This function returns a special key that can be used to compare elements**

Syntax

functools.cmp_to_key(callable) → I added the line from functools import cmp_to_key on top of the code

- Each element is compared with every other element of the list until a sorted list is obtained
- Every element apparently calls mycmp() function with the other element of the list
- mycmp() function returns a key after comparing the numbers
- This key eventually help sorted() to arrange elements in ascending (or discending) order

In the case of BANKSIM pycbc_splitbank script, we have:

tt = template_bank_table → is the list

tt = sorted (template_bank_table, key = cmp_to_key (*)) → here the sorted() function uses cmp_to_key() to sort elements in the template_bank_table list

(*) = once passed is Mchirp_sort() and once frequency_cutoff_sort()

Interpretation

- Mchirp_sort (x, y) compares the elements corresponding to x and y that are passed to it with the cmp function.
- Since Mchirp_sort() is passed to cmp_to_key(), the latter passes x and y from the template_bank_table list at each interaction, compares and sorts them.
- At the end template_bank_table should be an ordered list with the masses (and frequencies) in ascending order, since in tt = sorted() there is no option reverse=True.
- Idem for frequency_cutoff_sort().

Original code lines

```
def mchirp_sort(x, y):
   mc1, e1 = pycbc.pnutils.mass1_mass2_to_mchirp_eta(x.mass1, x.mass2)
   mc2, e2 = pycbc.pnutils.mass1_mass2_to_mchirp_eta(y.mass1, y.mass2)
   return cmp(mc1, mc2)

def frequency_cutoff_sort(x, y):
   p1 = pycbc.pnutils.frequency_cutoff_from_name(args.sort_frequency_cutoff,
                                 x.mass1, x.mass2,
                                 x.spin1z, x.spin2z)
   p2 = pycbc.pnutils.frequency_cutoff_from_name(args.sort_frequency_cutoff,
                                 y.mass1, y.mass2,
                                 y.spin1z, y.spin2z)
   return cmp(p1, p2)

tt = template_bank_table

if args.sort_frequency_cutoff:
   tt = sorted(template_bank_table, cmp=frequency_cutoff_sort)

if args.sort_mchirp:
   tt = sorted(template_bank_table, cmp=mchirp_sort)
```

TypeError: 'cmp' is an invalid keyword argument for sort()

Code lines with my changes


```
from functools import cmp_to_key                    # required in Python3

def cmp(a,  b):
return (a > b) - (a < b)                            # look back slide 3

def mchirp_sort(x, y):
   mc1, e1 = pycbc.pnutils.mass1_mass2_to_mchirp_eta(x.mass1, x.mass2)
   mc2, e2 = pycbc.pnutils.mass1_mass2_to_mchirp_eta(y.mass1, y.mass2)
   return cmp(mc1, mc2)

def frequency_cutoff_sort(x, y):
   p1 = pycbc.pnutils.frequency_cutoff_from_name(args.sort_frequency_cutoff,
                                 x.mass1, x.mass2,
                                 x.spin1z, x.spin2z)
   p2 = pycbc.pnutils.frequency_cutoff_from_name(args.sort_frequency_cutoff,
                                 y.mass1, y.mass2,
                                 y.spin1z, y.spin2z)
   return cmp(p1, p2)

tt = template_bank_table

if args.sort_frequency_cutoff:
   tt = sorted(template_bank_table, key = cmp_to_key(frequency_cutoff_sort))     # required in Python3

if args.sort_mchirp:
   tt = sorted(template_bank_table, key = cmp_to_key(mchirp_sort))               # required  in Python3
```

How are the variables mc1, mc2, p1, p2 calculated? The following functions are defined in the pnutils module of PyCBC:

- **pycbc.pnutils.mass1_mass2_to_mchirp_eta**

```
def mass1_mass2_to_mchirp_eta(mass1, mass2):
m_chirp =
conversions.mchirp_from_mass1_mass2(mass1, mass2)
eta = conversions.eta_from_mass1_mass2(mass1,
mass2)
return m_chirp,eta
```

- **pycbc.pnutils.frequency_cutoff_from_name**

```
def frequency_cutoff_from_name(name, m1, m2, s1z, s2z):

 """ Returns the result of evaluating the frequency cutoff function
specified by 'name' on a template with given parameters.”””

params = {"mass1":m1, "mass2":m2, "spin1z":s1z, "spin2z":s2z}
return named_frequency_cutoffs[name](params)
```

Source

https://pycbc.org/pycbc/latest/html/pycbc.html#pycbc.pnutils.mass1_mass2_to_mchirp_eta

https://pycbc.org/pycbc/latest/html/pycbc.html#pycbc.pnutils.frequency_cutoff_from_name