

Re: Flexible Collating (feedback please)

Re: Flexible Collating (feedback please)

Source: <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2006-10/msg03605.html>

- *From:* Ron Adam <rrr@xxxxxxxxxxxx>
 - *Date:* Wed, 18 Oct 2006 15:52:43 -0500
-

I made a number of changes ... (the new version is listed below)

These changes also resulted in improving the speed by about 3 times when all flags are specified.

Collating now takes about 1/3 (or less) time. Although it is still quite a bit slower than a bare list.sort(), that is to be expected as collate is locale aware and does additional transformations on the data which you would need to do anyways. The tests were done with Unicode strings as well.

Changed the flag types from integer values to a list of named strings. The reason for this is it makes finding errors easier and you can examine the flags attribute and get a readable list of flags.

A better regular expression for separating numerals. It now separates numerals in the middle of the string.

Changed flag COMMA_IN_NUMERALS to IGNORE_COMMAS, This was how it was implemented.

Added flag PERIOD_AS_COMMAS

This lets you collate decimal separated numbers correctly such as version numbers and internet address's. It also prevents numerals from being interpreted as floating point or decimal.

It might make more sense to implement it as PERIOD_IS_SEPARATOR. Needed?

Other minor changes to doc strings and tests were made.

Any feedback is welcome.

Cheers,
Ron

""""

Collate.py

A general purpose configurable collate module.

Re: Flexible Collating (feedback please)

Re: Flexible Collating (feedback please)

Collation can be modified with the following keywords:

CAPS_FIRST -> Aaa, aaa, Bbb, bbb
HYPHEN_AS_SPACE -> Don't ignore hyphens
UNDERSCORE_AS_SPACE -> Underscores as white space
IGNORE_LEADING_WS -> Disregard leading white space
NUMERICAL -> Digit sequences as numerals
IGNORE_COMMAS -> Allow commas in numerals
PERIOD_AS_COMMAS -> Periods can separate numerals.

* See doctests for examples.

Author: Ron Adam, ron@xxxxxxxxxxxx

```
"""
```

```
__version__ = '0.02 (pre-alpha) 10/18/2006'
```

```
import re  
import locale  
import string
```

```
locale.setlocale(locale.LC_ALL, "") # use current locale settings
```

```
# The above line may change the string constants from the string  
# module. This may have unintended effects if your program  
# assumes they are always the ascii defaults.
```

```
CAPS_FIRST = 'CAPS_FIRST'  
HYPHEN_AS_SPACE = 'HYPHEN_AS_SPACE'  
UNDERSCORE_AS_SPACE = 'UNDERSCORE_AS_SPACE'  
IGNORE_LEADING_WS = 'IGNORE_LEADING_WS'  
NUMERICAL = 'NUMERICAL'  
IGNORE_COMMAS = 'IGNORE_COMMAS'  
PERIOD_AS_COMMAS = 'PERIOD_AS_COMMAS'
```

```
class Collate(object):
```

```
    """ A general purpose and configurable collator class.
```

```
    """
```

```
    def __init__(self, flags=[]):  
        self.flags = flags  
        self.numrex = re.compile(r'([\d\.]*|D*)', re.LOCALE)  
        self.txttable = []  
        if HYPHEN_AS_SPACE in flags:  
            self.txttable.append('-', ' ')  
        if UNDERSCORE_AS_SPACE in flags:  
            self.txttable.append('_', ' ')  
        if PERIOD_AS_COMMAS in flags:  
            self.txttable.append('.', ',')  
        if IGNORE_COMMAS in flags:  
            self.txttable.append(',', '')  
        self.flags = flags
```

Re: Flexible Collating (feedback please)

Re: Flexible Collating (feedback please)

```
def transform(self, s):
    """ Transform a string for collating.
    """
    if not self.flags:
        return locale.strxfrm(s)
    for a, b in self.txttable:
        s = s.replace(a, b)
    if IGNORE_LEADING_WS in self.flags:
        s = s.strip()
    if CAPS_FIRST in self.flags:
        s = s.swapcase()
    if NUMERICAL in self.flags:
        slist = self.numrex.split(s)
        for i, x in enumerate(slist):
            try:
                slist[i] = float(x)
            except:
                slist[i] = locale.strxfrm(x)
        return slist
    return locale.strxfrm(s)

def __call__(self, a):
    """ This allows the Collate class work as a sort key.
```

```
USE: list.sort(key=Collate(flags))
"""
```

```
return self.transform(a)
```

```
def collate(slist, flags=[]):
    """ Collate list of strings in place.
    """
    slist.sort(key=Collate(flags).transform)
```

```
def collated(slist, flags=[]):
    """ Return a collated list of strings.
    """
    return sorted(slist, key=Collate(flags).transform)
```

```
def _test():
    """
```

DOC TESTS AND EXAMPLES:

Sort (and sorted) normally order all words beginning with caps before all words beginning with lower case.

```
>>> t = ['tuesday', 'Tuesday', 'Monday', 'monday']
>>> sorted(t) # regular sort
['Monday', 'Tuesday', 'monday', 'tuesday']
```

Locale collation puts words beginning with caps after words

Re: Flexible Collating (feedback please)

Re: Flexible Collating (feedback please)

beginning with lower case of the same letter.

```
>>> collated(t)
['monday', 'Monday', 'tuesday', 'Tuesday']
```

The CAPS_FIRST option can be used to put all words beginning with caps before words beginning in lowercase of the same letter.

```
>>> collated(t, [CAPS_FIRST])
['Monday', 'monday', 'Tuesday', 'tuesday']
```

The HYPHEN_AS_SPACE option causes hyphens to be equal to space.

```
>>> t = ['a-b', 'b-a', 'aa-b', 'bb-a']
>>> collated(t)
['aa-b', 'a-b', 'b-a', 'bb-a']

>>> collated(t, [HYPHEN_AS_SPACE])
['a-b', 'aa-b', 'b-a', 'bb-a']
```

The IGNORE_LEADING_WS and UNDERSCORE_AS_SPACE options can be used together to improve ordering in some situations.

```
>>> t = ['sum', '__str__', 'about', ' round']
>>> collated(t)
[' round', '__str__', 'about', 'sum']

>>> collated(t, [IGNORE_LEADING_WS])
['__str__', 'about', ' round', 'sum']

>>> collated(t, [UNDERSCORE_AS_SPACE])
[' round', '__str__', 'about', 'sum']

>>> collated(t, [IGNORE_LEADING_WS, UNDERSCORE_AS_SPACE])
['about', ' round', '__str__', 'sum']
```

The NUMERICAL option orders sequences of digits as numerals.

```
>>> t = ['a5', 'a40', '4abc', '20abc', 'a10.2', '13.5b', 'b2']
>>> collated(t, [NUMERICAL])
['4abc', '13.5b', '20abc', 'a5', 'a10.2', 'a40', 'b2']
```

The IGNORE_COMMAS option prevents commas from separating numerals.

```
>>> t = ['a5', 'a4,000', '500b', '100,000b']
>>> collated(t, [NUMERICAL, IGNORE_COMMAS])
['500b', '100,000b', 'a5', 'a4,000']
```

Re: Flexible Collating (feedback please)

The `PERIOD_AS_COMMAS` option can be used to sort version numbers and other decimal separated numbers correctly.

```
>>> t = ['5.1.1', '5.10.12', '5.2.2', '5.2.19' ]
>>> collated(t, [NUMERICAL, PERIOD_AS_COMMAS])
['5.1.1', '5.2.2', '5.2.19', '5.10.12']
```

Collate also can be done in place by using `collate()` instead of `collated()`.

```
>>> t = ['Fred', 'Ron', 'Carol', 'Bob']
>>> collate(t)
>>> t
['Bob', 'Carol', 'Fred', 'Ron']
```

```
.....
```

```
import doctest
doctest.testmod()
```

```
if __name__ == '__main__':
    _test()
```

```
.
```