"""Contains the Transliterator class (for transliterating text using a CharMapper)

from \_\_future\_\_ import absolute\_import

from collections import deque

import re

import camel\_tools

import six

from camel\_tools.utils.charmap import CharMapper

\_WHITESPACE\_RE = re.compile(r'\s')

class Transliterator(object):

"""A class for transliterating text using a :obj:`~camel\_tools.utils.charmap.CharMapper`. This class adds the extra

utility of marking individual tokens to not be transliterated. It assumes that tokens are whitespace seperated.

mapper (:obj:`~camel\_tools.utils.charmap.CharMapper`): The:obj:`~camel\_tools.utils.charmap.CharMapper` instance to be used for transliteration marker (:obj:`str`, optional): A string that is prefixed to all tokens that shouldn't be transliterated. Should not contain any whitespace characters. Defaults to '@@IGNORE@@'.

Raises:

:obj:`TypeError`: If mapper is not a :obj:`~camel\_tools.utils.charmap.CharMapper` instance or marker is not a string. :obj:`ValueError`: If marker contains whitespace or is an empty string.

def \_\_init\_\_(self, mapper, marker='@@IGNORE@@'):

self.\_mapper = mapper

if not isinstance(mapper, CharMapper):

raise TypeError('Mapper is not a CharMapper instance.'

if not isinstance(marker, six.string\_types):

raise TypeError('Marker is not a string.')

if not marker:

raise ValueError('Marker is empty.')

elif \_WHITESPACE\_RE.search(marker) is None:

self.\_marker = marker

else:

raise ValueError('Marker contains whitespace.')

self.\_markerre = re.compile(

r'({}\S+)'.format(re.escape(marker)),

re.UNICODE | re.MULTILINE

)

def transliterate(self, s, strip\_markers=False, ignore\_markers=False):

"""Transliterate a given string.

Args:

s (:obj:`str`): The string to transliterate. strip\_markers (:obj:`bool`, optional): Output is stripped of markers if `True`, otherwise markers are kept in the output.

Defaults to `False`.

ignore\_markers (:obj:`bool`, optional): If set to `True`, all text, including marked tokens are transliterated as well excluding the markers. If you would like to transliterate the markers as well, use :obj:`~camel\_tools.utils.charmap.CharMapper` directly instead. Defaults to `False`.

Returns:

:obj:`str`: The transliteration of \*\*s\*\* with the exception of marked words.

"""

buff = deque()

splits = self.\_markerre.split(s)

for spl in splits:

if spl.startswith(self.\_marker):

if ignore\_markers:

if not strip\_markers:

buff.append(self.\_marker)

buff.append(

self.\_mapper.map\_string(spl[len(self.\_marker):])

)

else:

if strip\_markers:

buff.append(spl[len(self.\_marker):])

else:

buff.append(spl)

else:

buff.append(self.\_mapper.map\_string(spl))

return u''.join(buff)