```python
# -*- coding: utf-8 -*-
import six

QUOTES = {u'"', u"'"}


def give_another_quote(q):
    """When you pass a quote character, returns you an another one if possible"""
    for qc in QUOTES:
        if qc != q:
            return qc
    else:
        raise ValueError(u'Could not find a different quote for {}'.format(q))


def escape_filter(o):
    """Tries to escape the values that are passed to filter as correctly as possible.

    No standard way is followed, but at least it is simple.
    """
    if o is None:
        return u'NULL'
    if isinstance(o, int):
        return str(o)
    if not isinstance(o, six.string_types):
        raise ValueError('Filters take only None, int or a string type')
    if not o:
        # Empty string
        return u"''"
    # Now enforce unicode
    o = unicode_process(o)
    if u'"' not in o:
        # Simple case, just put the quote that does not exist in the string
        return u'"' + o + u'"'
    elif u"'" not in o:
        # Simple case, just put the quote that does not exist in the string
        return u"'" + o + u"'"
    else:
        # Both are there, so start guessing
        # Empty strings are sorted out, so the string must contain something.
        # String with length == 1 are sorted out because if they have a quote, they would be quoted
        # with the another quote in preceeding branch. Therefore the string is at least 2 chars long
        # here which allows us to NOT check the length here.
        first_char = o[0]
        last_char = o[-1]
        if first_char in QUOTES and last_char in QUOTES:
            # The first and last chars definitely are quotes
            if first_char == last_char:
                # Simple, just put another ones around them
                quote = give_another_quote(first_char)
                return quote + o + quote
            else:
                # I don't like this but the nature of the escape is like that ...
                # Since now it uses both of the quotes, just pick the simple ones and surround it
                return u"'" + o + u"'"
        elif first_char not in QUOTES and last_char not in QUOTES:
            # First and last chars are not quotes, so a simple solution
            return u"'" + o + u"'"
        else:
            # One of the first or last chars is not a quote
            if first_char in QUOTES:
                quote = give_another_quote(first_char)
            else:
                # last_char
                quote = give_another_quote(last_char)
            return quote + o + quote


def unicode_process(s):
    if not isinstance(s, six.string_types):
        if six.PY2:
            if hasattr(s, '__unicode__'):
                s = unicode(s)
            else:
                s = str(s)
        else:
            s = str(s)
    if (six.PY3 and isinstance(s, bytes)) or (six.PY2 and isinstance(s, str)):
        s = s.decode('utf-8')
    # Here we have Unicode!
    return s
```