Mercurial > cpython

view Lib/glob.py @ 102611:0de509a79181 2.7

- log
- graph
- tags
- bookmarks
- branches

- changeset
- browse

- **file**
- latest
- diff
- comparison
- annotate
- file log
- raw

- help

```
Issue #27714: For IDLE's test_textview, backport 3.x subclass with mocks
instead of overriding methods with mocks in original class and module.
This makes the 2.7 test_textview nearly identical to the 3.5/.6 test. [#27714]
```

author Terry Jan Reedy <tjreedy@udel.edu>
date Wed, 10 Aug 2016 19:41:39 -0400 (27 hours ago)
parents e430973149ed
children

line source **line wrap: on**

```python
"""Filename globbing utility."""

import sys
import os
import re
import fnmatch

try:
    _unicode = unicode
except NameError:
    # If Python is built without Unicode support, the unicode type
    # will not exist. Fake one.
    class _unicode(object):
        pass

__all__ = ["glob", "iglob"]

def glob(pathname):
    """Return a list of paths matching a pathname pattern.

    The pattern may contain simple shell-style wildcards a la
    fnmatch. However, unlike fnmatch, filenames starting with a
    dot are special cases that are not matched by '*' and '?'
    patterns.

    """
    return list(iglob(pathname))

def iglob(pathname):
    """Return an iterator which yields the paths matching a pathname pattern.

    The pattern may contain simple shell-style wildcards a la
    fnmatch. However, unlike fnmatch, filenames starting with a
    dot are special cases that are not matched by '*' and '?'
    patterns.

    """
    dirname, basename = os.path.split(pathname)
    if not has_magic(pathname):
        if basename:
            if os.path.lexists(pathname):
                yield pathname
        else:
            # Patterns ending with a slash should match only directories
            if os.path.isdir(dirname):
                yield pathname
        return
    if not dirname:
        for name in glob1(os.curdir, basename):
            yield name
        return
    # `os.path.split()` returns the argument itself as a dirname if it is a
    # drive or UNC path.  Prevent an infinite recursion if a drive or UNC path
    # contains magic characters (i.e. r'\\?\C:').
    if dirname != pathname and has_magic(dirname):
        dirs = iglob(dirname)
    else:
        dirs = [dirname]
    if has_magic(basename):
        glob_in_dir = glob1
    else:
        glob_in_dir = glob0
    for dirname in dirs:
        for name in glob_in_dir(dirname, basename):
            yield os.path.join(dirname, name)

# These 2 helper functions non-recursively glob inside a literal directory.
# They return a list of basenames. `glob1` accepts a pattern while `glob0`
# takes a literal basename (so it only has to check for its existence).

def glob1(dirname, pattern):
    if not dirname:
        dirname = os.curdir
    if isinstance(pattern, _unicode) and not isinstance(dirname, unicode):
        dirname = unicode(dirname, sys.getfilesystemencoding() or
                                   sys.getdefaultencoding())
    try:
        names = os.listdir(dirname)
    except os.error:
        return []
    if pattern[0] != '.':
        names = filter(lambda x: x[0] != '.', names)
    return fnmatch.filter(names, pattern)

def glob0(dirname, basename):
    if basename == '':
        # `os.path.split()` returns an empty basename for paths ending with a
        # directory separator.  'q*x/' should match only directories.
        if os.path.isdir(dirname):
            return [basename]
    else:
        if os.path.lexists(os.path.join(dirname, basename)):
            return [basename]
    return []


magic_check = re.compile('[*?[]')

def has_magic(s):
    return magic_check.search(s) is not None
```