11.7. glob — Unix style pathname pattern expansion

Source code: Lib/glob.py

The glob module finds all the pathnames matching a specified pattern according to the rules used by the Unix shell, although results are returned in arbitrary order. No tilde expansion is done, but *, ?, and character ranges expressed with [] will be correctly matched. This is done by using the os.scandir() and fnmatch.fnmatch() functions in concert, and not by actually invoking a subshell. Note that unlike fnmatch.fnmatch(), glob treats filenames beginning with a dot (.) as special cases. (For tilde and shell variable expansion, use os.path.expanduser() and os.path.expandvars().)

For a literal match, wrap the meta-characters in brackets. For example, '[?]' matches the character '?'.

> **See also:** The pathlib module offers high-level path objects.

glob.**glob**(*pathname*, *, *recursive=False*)

Return a possibly-empty list of path names that match *pathname*, which must be a string containing a path specification. *pathname* can be either absolute (like /usr/src/Python-1.5/Makefile) or relative (like ../../Tools/*/*.gif), and can contain shell-style wildcards. Broken symlinks are included in the results (as in the shell).

If *recursive* is true, the pattern "**" will match any files and zero or more directories and subdirectories. If the pattern is followed by an os.sep, only directories and subdirectories match.

> **Note:** Using the "**" pattern in large directory trees may consume an inordinate amount of time.

Changed in version 3.5: Support for recursive globs using "**".

glob.**iglob**(*pathname*, *, *recursive=False*)

Return an iterator which yields the same values as glob() without actually storing them all simultaneously.

glob.**escape**(*pathname*)

Escape all special characters (`'?'`, `'*'` and `'['`). This is useful if you want to match an arbitrary literal string that may have special characters in it. Special characters in drive/UNC sharepoints are not escaped, e.g. on Windows `escape ('//?/c:/Quo vadis?.txt')` returns `'//?/c:/Quo vadis[?].txt'`.

New in version 3.4.

For example, consider a directory containing the following files: `1.gif`, `2.txt`, `card.gif` and a subdirectory `sub` which contains only the file `3.txt`. `glob()` will produce the following results. Notice how any leading components of the path are preserved.

```
>>> import glob
>>> glob.glob('./[0-9].*')
['./1.gif', './2.txt']
>>> glob.glob('*.gif')
['1.gif', 'card.gif']
>>> glob.glob('?.gif')
['1.gif']
>>> glob.glob('**/*.txt', recursive=True)
['2.txt', 'sub/3.txt']
>>> glob.glob('./**/', recursive=True)
['./', './sub/']
```

If the directory contains files starting with `.` they won't be matched by default. For example, consider a directory containing `card.gif` and `.card.gif`:

```
>>> import glob
>>> glob.glob('*.gif')
['card.gif']
>>> glob.glob('.c*')
['.card.gif']
```

See also:

Module `fnmatch`
Shell-style filename (not path) expansion