

Applied Python



Overview

- getting interpretable results from big data
- genome assemblies, annotations, expression data, ...
- BLAST result analysis
- Concept of plots via axes in matplotlib
- General figure types: plot, boxplot, barplot
 - Genome figures: scatter plots, box plots, chromosome plots, gene cluster plots
- Statistics in Python: tests and theoretical background
 - t-test, W-test, U-test, cor-test, X^2 -test

Analyze problem

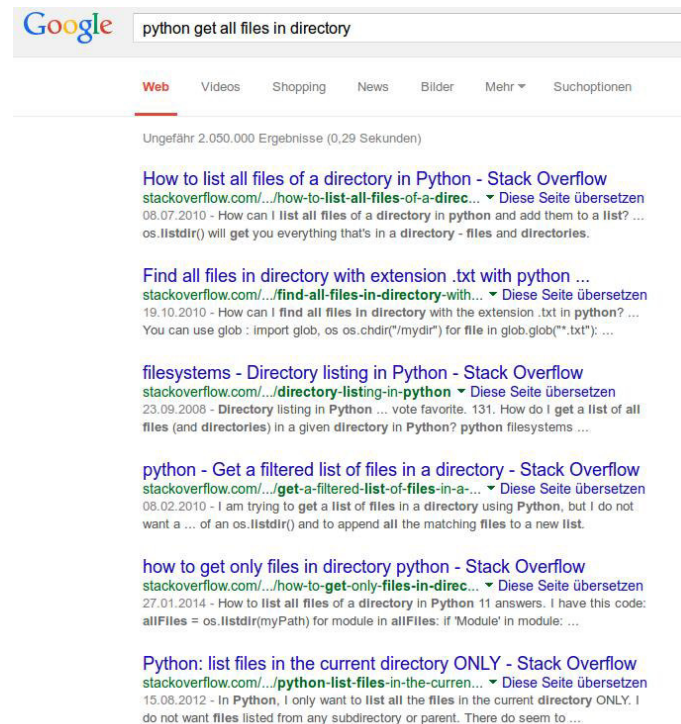
- Split problem into parts => solutions for small parts are more generic and thus often available
- Precise description of problem is necessary

stackoverflow

- Best hit in approximately 90-95% of all cases
- Stackoverflow is chat forum for programmers (different languages)
- Problems are described in questions
- Answers often contain examples which can be directly applied
- Answers are judged by members => look for green marking and the number of positive votes

Example

- Problem: get paths of all files in a certain directory
- Search expression: “python get all files in directory”



Example

533 user like
this answer

This answers
solved the
question

14 Answers

active oldest votes

▲
533
▼
✓

You can use `glob`:

```
import glob, os
os.chdir("/mydir")
for file in glob.glob("*.txt"):
    print(file)
```

or simply `os.listdir`:


```
import os
for file in os.listdir("/mydir"):
    if file.endswith(".txt"):
        print(file)
```

or if you want to traverse directory, use `os.walk`:


```
import os
for root, dirs, files in os.walk("/mydir"):
    for file in files:
        if file.endswith(".txt"):
            print(os.path.join(root, file))
```

share improve this answer

edited Apr 22 at 17:44

 Tarantula
4,794 ● 3 ● 22 ● 39

answered Oct 19 '10 at 1:12

 ghostdog74
102k ● 17 ● 116 ● 188

1 Using solution #2, How would you create a file or list with that info? – Merlin Oct 19 '10 at 3:48

35 @ghostdog74: In my opinion it would more appropriate to write `for file in f` than for `for files in f` since what is in the variable is a single filename. Even better would be to change the `f` to `files` and then the for loops could become `for file in files`. – martineau Oct 26 '10 at 14:18

18 @computermaggyver: No, `file` is not a reserved word, just the name of a predefined function, so it's quite possible to use it as a variable name in your own code. Although it's true that generally one should avoid collisions like that, `file` is a special case because there's hardly ever any need to use it, so it is often consider an exception to the guideline. If you don't want to do that, PEP8 recommends appending a single underscore to such names, i.e. `file_`, which you'd have to agree is still quite readable. – martineau Oct 14 '12 at 19:04

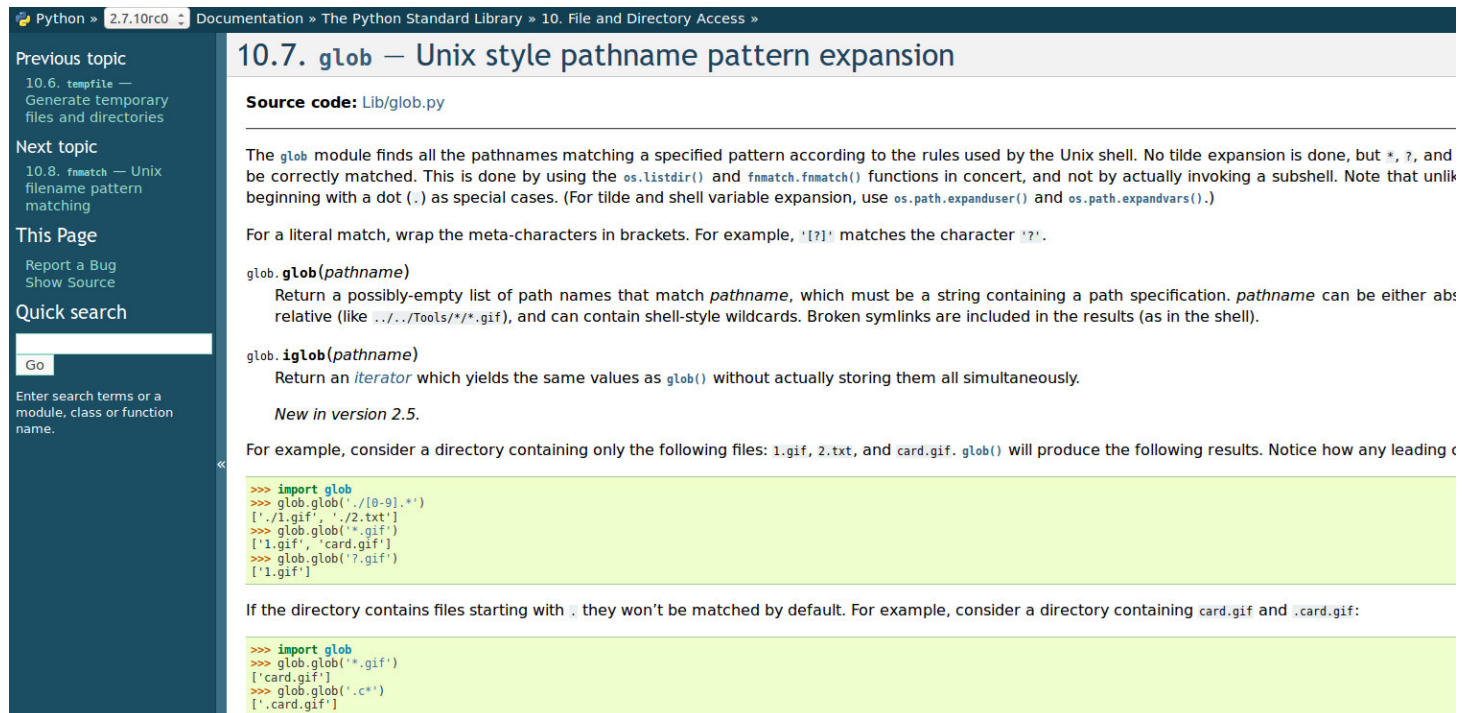
2 Thanks, martineau, you're absolutely right. I jumped too quickly to conclusions. – computermaggyver Oct 15 '12 at 19:53

1 Really cool answer, you could replace `r,d,f` by `r_,d_,f_` to avoid unused variable declaration. – AsTeR Mar 8 '13 at 20:16

Three different
ways to solve this
problem are
described

Official Python documentation

- Systematic documentation of all functions in a module with all possible arguments
- Sometimes examples are given as well



The screenshot shows the official Python documentation for the `glob` module. The page is titled "10.7. glob — Unix style pathname pattern expansion". It includes a sidebar with navigation links for previous and next topics, a search bar, and a list of links for reporting bugs or showing source code. The main content area provides a detailed description of the `glob` module, including its source code location (`Lib/glob.py`), a brief overview of its functionality, and examples of how to use it. The examples show how to find files matching specific patterns, such as `l.gif`, `card.gif`, and `l.gif?`.

Python » 2.7.10rc0 » Documentation » The Python Standard Library » 10. File and Directory Access »

10.7. 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. No tilde expansion is done, but `*`, `?`, and `[]` are correctly matched. This is done by using the `os.listdir()` and `fnmatch.fnmatch()` functions in concert, and not by actually invoking a subshell. Note that unlike `find`, `glob` does not begin 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 `'?'`.

glob.glob(pathname)

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 or relative (like `../Tools/*/*.gif`), and can contain shell-style wildcards. Broken symlinks are included in the results (as in the shell).

glob.iglob(pathname)

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

New in version 2.5.

For example, consider a directory containing only the following files: `l.gif`, `2.txt`, and `card.gif`. `glob()` will produce the following results. Notice how any leading `.` is not included in the results.

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

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('.*')
['.card.gif']
```

Human dataset - Exercise 1

- Input: genes_human_genenames_duplicates.txt
- Count number of unique genes in file and write gene names in new file.