

Navigation

- [index](#)
- [modules](#) |
- [next](#) |
- [previous](#) |
- [Python v2.6.4 documentation](#) »
- [The Python Standard Library](#) »
- [31. Importing Modules](#) »

31.4. pkgutil — Package extension utility

New in version 2.3.

This module provides functions to manipulate packages:

```
pkgutil.extend_path(path, name)
```

Extend the search path for the modules which comprise a package. Intended use is to place the following code in a package's `__init__.py`:

```
from pkgutil import extend_path
__path__ = extend_path(__path__, __name__)
```

This will add to the package's `__path__` all subdirectories of directories on `sys.path` named after the package. This is useful if one wants to distribute different parts of a single logical package as multiple directories.

It also looks for `*.pkg` files beginning where `*` matches the `name` argument. This feature is similar to `*.pth` files (see the [site](#) module for more information), except that it doesn't special-case lines starting with `import`. A `*.pkg` file is trusted at face value: apart from checking for duplicates, all entries found in a `*.pkg` file are added to the path, regardless of whether they exist on the filesystem. (This is a feature.)

If the input path is not a list (as is the case for frozen packages) it is returned unchanged. The input path is not modified; an extended copy is returned. Items are only appended to the copy at the end.

It is assumed that `sys.path` is a sequence. Items of `sys.path` that are not (Unicode or 8-bit) strings referring to existing directories are ignored. Unicode items on `sys.path` that cause errors when used as filenames may cause this function to raise an exception (in line with [os.path.isdir\(\)](#) behavior).

```
pkgutil.get_data(package, resource)
```

Get a resource from a package.

This is a wrapper for the PEP 302 loader [get_data\(\)](#) API. The package argument should be the name of a package, in standard module format (`foo.bar`). The resource argument should be in the form of a relative filename, using `/` as the path separator. The parent directory name `..` is not allowed, and nor is a rooted name (starting with a `/`).

The function returns a binary string that is the contents of the specified resource.

For packages located in the filesystem, which have already been imported, this is the rough equivalent of:

```
d = os.path.dirname(sys.modules[package].__file__)
data = open(os.path.join(d, resource), 'rb').read()
```

If the package cannot be located or loaded, or it uses a PEP 302 loader which does not support [get_data\(\)](#), then `None` is returned.

Previous topic

[31.3. zipimport — Import modules from Zip archives](#)

Next topic

[31.5. modulefinder — Find modules used by a script](#)

This Page

- [Show Source](#)

Navigation

- [index](#)
- [modules](#) |
- [next](#) |

- [previous](#) |
- [Python v2.6.4 documentation](#) »
- [The Python Standard Library](#) »
- [31. Importing Modules](#) »

© [Copyright](#) 1990-2010, Python Software Foundation.

The Python Software Foundation is a non-profit corporation. [Please donate.](#)

Last updated on Feb 26, 2010. Created using [Sphinx](#) 0.6.3.