:mod:`importlib.resources` -- Resources

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]importlib.resources.rst, line 1); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]importlib.resources.rst, line 4)

Unknown directive type "module".

```
.. module:: importlib.resources
    :synopsis: Package resource reading, opening, and access
```

Source code: :source:`Lib/importlib/resources.py`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]importlib.resources.rst, line 7); *backlink*

Unknown interpreted text role "source".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]importlib.resources.rst, line 11)

Unknown directive type "versionadded".

```
.. versionadded:: 3.7
```

This module leverages Python's import system to provide access to *resources* within *packages*. If you can import a package, you can access resources within that package. Resources can be opened or read, in either binary or text mode.

Resources are roughly akin to files inside directories, though it's important to keep in mind that this is just a metaphor. Resources and packages **do not** have to exist as physical files and directories on the file system.

> **Note**
>
> This module provides functionality similar to pkg_resources Basic Resource Access without the performance overhead of that package. This makes reading resources included in packages easier, with more stable and consistent semantics.
>
> The standalone backport of this module provides more information on using importlib.resources and migrating from pkg_resources to importlib.resources and migrating legacy usage.

Loaders that wish to support resource reading should implement a `get_resource_reader(fullname)` method as specified by :class:`importlib.abc.ResourceReader`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]importlib.resources.rst, line 40); *backlink*

Unknown interpreted text role "class".

The following types are defined.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]importlib.resources.rst, line 46)

Unknown directive type "data".

```
.. data:: Package

    The ``Package`` type is defined as ``Union[str, ModuleType]``.  This means
    that where the function describes accepting a ``Package``, you can pass in
    either a string or a module.  Module objects must have a resolvable
    ``__spec__.submodule_search_locations`` that is not ``None``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]importlib.resources.rst, line 53)

Unknown directive type "data".

```
.. data:: Resource

    This type describes the resource names passed into the various functions
    in this package.  This is defined as ``Union[str, os.PathLike]``.
```

The following functions are available.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]importlib.resources.rst, line 62)

Unknown directive type "function".

```
.. function:: files(package)

    Returns an :class:`importlib.resources.abc.Traversable` object
    representing the resource container for the package (think directory)
    and its resources (think files). A Traversable may contain other
    containers (think subdirectories).

    *package* is either a name or a module object which conforms to the
    ``Package`` requirements.

    .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]importlib.resources.rst, line 74)

Unknown directive type "function".

```
.. function:: as_file(traversable)

    Given a :class:`importlib.resources.abc.Traversable` object representing
    a file, typically from :func:`importlib.resources.files`, return
    a context manager for use in a :keyword:`with` statement.
    The context manager provides a :class:`pathlib.Path` object.

    Exiting the context manager cleans up any temporary file created when the
    resource was extracted from e.g. a zip file.

    Use ``as_file`` when the Traversable methods
    (``read_text``, etc) are insufficient and an actual file on
    the file system is required.

    .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]importlib.resources.rst, line 90)

Unknown directive type "function".

```
.. function:: open_binary(package, resource)

    Open for binary reading the *resource* within *package*.

    *package* is either a name or a module object which conforms to the
    ``Package`` requirements.  *resource* is the name of the resource to open
    within *package*; it may not contain path separators and it may not have
    sub-resources (i.e. it cannot be a directory).  This function returns a
    ``typing.BinaryIO`` instance, a binary I/O stream open for reading.

   .. deprecated:: 3.11
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-

main\Doc\library\[cpython-main][Doc][library]importlib.resources.rst, line 103)

Unknown directive type "function".

```
.. function:: open_text(package, resource, encoding='utf-8', errors='strict')

    Open for text reading the *resource* within *package*.  By default, the
    resource is opened for reading as UTF-8.

    *package* is either a name or a module object which conforms to the
    ``Package`` requirements.  *resource* is the name of the resource to open
    within *package*; it may not contain path separators and it may not have
    sub-resources (i.e. it cannot be a directory).  *encoding* and *errors*
    have the same meaning as with built-in :func:`open`.

    This function returns a ``typing.TextIO`` instance, a text I/O stream open
    for reading.

   .. deprecated:: 3.11
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]importlib.resources.rst, line 120)

Unknown directive type "function".

```
.. function:: read_binary(package, resource)

    Read and return the contents of the *resource* within *package* as
    ``bytes``.

    *package* is either a name or a module object which conforms to the
    ``Package`` requirements.  *resource* is the name of the resource to open
    within *package*; it may not contain path separators and it may not have
    sub-resources (i.e. it cannot be a directory).  This function returns the
    contents of the resource as :class:`bytes`.

   .. deprecated:: 3.11
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]importlib.resources.rst, line 134)

Unknown directive type "function".

```
.. function:: read_text(package, resource, encoding='utf-8', errors='strict')

    Read and return the contents of *resource* within *package* as a ``str``.
    By default, the contents are read as strict UTF-8.

    *package* is either a name or a module object which conforms to the
    ``Package`` requirements.  *resource* is the name of the resource to open
    within *package*; it may not contain path separators and it may not have
    sub-resources (i.e. it cannot be a directory).  *encoding* and *errors*
    have the same meaning as with built-in :func:`open`.  This function
    returns the contents of the resource as :class:`str`.

   .. deprecated:: 3.11
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]importlib.resources.rst, line 149)

Unknown directive type "function".

```
.. function:: path(package, resource)

    Return the path to the *resource* as an actual file system path.  This
    function returns a context manager for use in a :keyword:`with` statement.
    The context manager provides a :class:`pathlib.Path` object.

    Exiting the context manager cleans up any temporary file created when the
    resource needs to be extracted from e.g. a zip file.

    *package* is either a name or a module object which conforms to the
```

```
        ``Package`` requirements.  *resource* is the name of the resource to open
        within *package*; it may not contain path separators and it may not have
        sub-resources (i.e. it cannot be a directory).

       .. deprecated:: 3.11
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]importlib.resources.rst, line 166)

Unknown directive type "function".

```
   .. function:: is_resource(package, name)

       Return ``True`` if there is a resource named *name* in the package,
       otherwise ``False``.  Remember that directories are *not* resources!
       *package* is either a name or a module object which conforms to the
       ``Package`` requirements.

      .. deprecated:: 3.11
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]importlib.resources.rst, line 176)

Unknown directive type "function".

```
   .. function:: contents(package)

       Return an iterable over the named items within the package.  The iterable
       returns :class:`str` resources (e.g. files) and non-resources
       (e.g. directories).  The iterable does not recurse into subdirectories.

       *package* is either a name or a module object which conforms to the
       ``Package`` requirements.

      .. deprecated:: 3.11
```