imptools

Danijar Hafner

Nov 30, 2021

CONTENTS

Tools for improving Python imports.

CHAPTER

ONE

IMPORT_PATH

imptools.**import_path**(*path*, *name=None*, *notfound='error'*, *reload=False*)

Import a module from any path on the filesystem.

Usually, this would be achieved by adding the parent directory of the module to `sys.path` or the `PYTHONPATH` environment variable and then importing it normally. However, this pollutes Python's import path, which can lead to accidentally importing the wrong modules. The function `import_path()` avoids this problem by importing a package from a path on the filesystem without modifying the Python import path.

The module can be either a directory containing `__init__.py` or a single file.

Relative paths are resolved relative to the directory of the source file that calls `import_path()`.

```
import imptools

my_module = imptools.import_path('../path/to/my_module')
```

```
import imptools

my_module = imptools.import_path(
'../path/to/my_module',  # Path to a module directory or single file.
notfound='error',        # Raise 'error' or 'ignore' if not found.
reload=False,            # Whether to import if already available.
)

import my_module  # Import statement also works.
```

Parameters

- **path** (*str or Path*) – The filesystem path to the module to import. This should be either a directory containing an `__init__.py` file or a single file with `.py` extension. The path can be relative.
- **name** (*str, optional*) – A name that is compared to the directory name of `path` and must match it. Useful for relative paths.
- **notfound** (*str*) – A string indicating what to do if the module is not found. Values can be `error` to raise an exception or `ignore` to return `None` from this function.
- **reload** (*bool*) – A boolean indicating whether to reload the package if it is already loaded.

Raises **ModuleNotFoundError** – If `path` does not point to a module or if `name` is specified and dose not match the directory name of `path`.

Returns The module or `None` if it was not found.

CHAPTER

TWO

ENABLE_RELATIVE

imptools.**enable_relative**()

Enable relative imports for scripts that are not executed as module.

Usually, scripts that are part of a module and use relative imports must be run as `python3 -m module.script`. However, this requires being in the correct working directory and can be annoying. The `enable_relative()` function allows to execute those scripts normally as `python3 script.py`.

Since PEP 366, this can be achieved by setting the `__package__` variable in the script and importing the package or making it available on the Pyhton import path. The `enable_relative()` function hides this behind a simple function that can be imported and called inside the script, before any relative imports.

```
import imptools

imptools.enable_relative()

# Relative imports...
```

Raises **ModuleNotFoundError** – If the parent directory of the script that calls this function is not a module.

INDEX

E

I