Fix ImportError: attempted relative import with no known parent package Issue

TechRounder PDF Edition

Live article:
https://www.techrounder.com/development/fix-importerror-attempted-relative-import-with-no-known-parent-package-issue/

By Vipin PG | Published July 6, 2024 | Updated March 9, 2026 | Format: Article | 4 min read

In brief

This error occurs when Python can't determine the parent package for a relative import, usually due to incorrect package structure, missing __init__.py files, or running the script from the wrong directory. Fix it by ensuring each package directory contains an __init__.

The ImportError: attempted relative import with no known parent package occurs when trying to perform a relative import in Python, but the interpreter is unable to determine the parent package. This error typically arises due to incorrect package structure, missing init.py files, or running the script from the wrong directory. Let's dive into the details and explore how to resolve this error.

Understanding Relative Imports

In Python, relative imports allow you to import modules or packages relative to the current module's location. Relative imports use dot notation to indicate the relative path. For example:

- from .module import function: Imports the function from the module in the same directory as the current module.
- from ..package import module: Imports the module from the package one level up from the current module.

Relative imports are useful for creating modular and organized code structures. However, they rely on the correct package hierarchy and the presence of init.py files to work properly.

Causes of the ImportError

The ImportError: attempted relative import with no known parent package can occur due to several reasons:

1. Incorrect Package Structure : Python expects a specific directory structure for packages and modules. If the structure is not followed correctly, the interpreter may fail to identify the parent package.
2. Missing init.py Files : In Python 2.x and earlier versions of Python 3.x, directories must contain an init .py file to be recognized as packages. Without this file, Python won't treat the directory as a package, leading to the ImportError.
3. Running the Script from the Wrong Directory : The script's location and the current working directory play a crucial role in resolving relative imports. If the script is executed from a directory that is not part of the package hierarchy, Python may not be able to determine the parent package.

Resolving the ImportError

To resolve the ImportError: attempted relative import with no known parent package, follow these steps:

Step 1: Verify the Package Structure

Ensure that your project follows the correct package structure. Here's an example of a valid structure:

my_package/ init.py module.py subpackage/ init.py submodule.py

In this structure, my_package is the top-level package, and subpackage is a subpackage within my_package. The init.py files are present in each directory to mark them as packages.

Step 2: Include init.py Files

If you are using Python 2.x or an earlier version of Python 3.x, make sure that each directory intended to be a package contains an init.py file. This file can be empty, but it must be present for Python to recognize the directory as a package.

In Python 3.3 and later versions, the init.py file is not strictly required, but it's still recommended to include it for backward compatibility and clarity.

Step 3: Run the Script from the Correct Directory

When executing a script that uses relative imports, ensure that you run it from the correct directory. The script should be run from the directory where the top-level package is located.

For example, if your script is located at my_package/subpackage/submodule.py and uses relative imports, navigate to the directory containing my_package and run the script from there:

cd /path/to/directory/containing/my_package python -m my_package.subpackage.submodule

Running the script using the -m flag ensures that Python treats the specified path as a module, allowing relative imports to work correctly.

Step 4: Use Absolute Imports as an Alternative

If relative imports are causing issues or if you prefer a more explicit approach, you can use absolute imports instead. Absolute imports specify the full path to the module or package, starting from the top-level package. For example, instead of using a relative import like from .module import function, you can use an absolute import: from my_package.module import function Absolute imports are less prone to errors and can provide better clarity, especially in larger projects. Example: Fixing the ImportError Let's consider an example to illustrate how to fix the ImportError. Suppose we have the following package structure: my_package/ init.py module.py subpackage/ init.py submodule.py In submodule.py, we want to import a function from module.py using a relative import: from ..module import function If we run submodule.py directly from the subpackage directory, we'll encounter the ImportError: attempted relative import with no known parent package. To fix this, we need to run the script from the directory containing the top-level package (my_package): cd /path/to/directory/containing/my_package python -m my_package.subpackage.submodule By running the script with the -m flag and specifying the full module path, Python will correctly resolve the relative import.

Conclusion

The ImportError: attempted relative import with no known parent package occurs when Python is unable to determine the parent package for a relative import. To resolve this error, ensure that your project follows the correct package structure, includes init.py files (if necessary), and run the script from the appropriate directory.

By understanding the causes of the error and following the steps outlined in this article, you can effectively fix the ImportError and use relative imports in your Python projects with confidence.

Remember, if relative imports are causing too much trouble, you can always switch to absolute imports for a more straightforward and explicit approach.