Python Package

A package is a container that contains various functions to perform specific tasks. For example, the math package includes the `sqrt()` function to perform the square root of a number.

While working on big projects, we have to deal with a large amount of code, and writing everything together in the same file will make our code look messy. Instead, we can separate our code into multiple files by keeping the related code together in packages.

Now, we can use the package whenever we need it in our projects. This way we can also reuse our code.

Package Model Structure in Python Programming

Suppose we are developing a game. One possible organization of packages and modules could be as shown in the figure below.

Package Model Structure

Game Package Model Structure

Note: A directory must contain a file named `__init__.py` in order for Python to consider it as a package. This file can be left empty but we generally place the initialization code for that package in this file.

Importing module from a package

In Python, we can import modules from packages using the dot (.) operator.

For example, if we want to import the `start` module in the above example, it can be done as follows:

```
import Game.Level.start
```

Now, if this module contains a function named `select_difficulty()`, we must use the full name to reference it.

```
Game.Level.start.select_difficulty(2)
```

Import Without Package Prefix

If this construct seems lengthy, we can import the module without the package prefix as follows:

```
from Game.Level import start
```

We can now call the function simply as follows:

```
start.select_difficulty(2)
```

Import Required Functionality Only

Another way of importing just the required function (or class or variable) from a module within a package would be as follows:

```
from Game.Level.start import select_difficulty
```

Now we can directly call this function.

```
select_difficulty(2)
```

Although easier, this method is not recommended. Using the full namespace avoids confusion and prevents two same identifier names from colliding.

While importing packages, Python looks in the list of directories defined in `sys.path`, similar as for module search path.

Table of Contents