External Packages

Python has a vast ecosystem of external packages that you can use to extend the functionality of your programs. These packages are collections of related modules that provide a wide range of features, from scientific computing to web development, machine learning, and more.

In this tutorial section, we'll cover how to use external packages in Python, including how to install them, import them into your code, and use their modules, functions, and classes.

Installing External Packages

Before you can use an external package in your Python code, you need to install it. You can use pip, which is included with Python, to install external packages.

To install a package using pip, open a terminal or command prompt and run the following command:

```
pip install <package_name>
```

Replace `<package_name>` with the name of the package you want to install. For example, to install the latest version of the NumPy package, you would run:

```
pip install numpy
```

Installing a Specific Version

If you want to install a specific version of a package, you can do so by appending the version number to the package name, like this:

```
pip install numpy==1.19.3
```

This will install version 1.19.3 of the NumPy package.

Installing a Minimum Version

You can also use the `>=` operator to install a minimum version of a package, like this:

```
pip install numpy>=1.19.3
```

This will install the latest version of the NumPy package that is greater than or equal to version 1.19.3.

Installing a Maximum Version

You can also use the `>=` operator to install a maximum version of a package, like this:

```
pip install numpy<=1.19.3
```

This will install the latest version of the NumPy package that is less than or equal to version 1.19.3.

Installing a Range of Versions

You can also use the `>=` operator to install between a range of versions of a package, like this:

```
pip install numpy>=1.19.3,<=1.19.5
```

This will install the latest version of the NumPy package that is greater than or equal to version 1.19.3 and less than or equal to version 1.19.5.

Installing Multiple Packages

You can also install multiple packages at once by separating them with spaces, like this:

```
pip install numpy pandas matplotlib
```

This will install the latest versions of the `NumPy`, `pandas`, and `Matplotlib` packages.

Installing from a Requirements File

You can also install packages from a text file that lists the packages you want to install. To do this, create a text file named `requirements.txt` and add the names of the packages you want to install to it, one per line, using the above conventions. Then, run the following command:

```
pip install -r requirements.txt
```

This will install all of the packages listed in the `requirements.txt` file.

There are many other ways to install external packages, including using a package manager like Anaconda or Miniconda. For more information, see the Installing Packages tutorial on the Python Packaging User Guide.

Importing External Packages

Once you have installed a package, you can import it into your Python code using the `import` statement. For example, to import the `NumPy` package, you would run:

```
import numpy
```

This will import the `NumPy` package into your code and make it available for use. You can then use the modules, functions, and classes that are included in the package.

Package Aliases

You can also use an alias when importing a package. For example, to import the `NumPy` package and use the alias `np`, you would run:

```
import numpy as np
```

This will import the `NumPy` package into your code and make it available for use, but you will need to use the alias `np` to access the package's modules, functions, and classes.

Using External Packages

To use the modules, functions, and classes provided by an external package, you need to understand its API (Application Programming Interface). The API describes the modules, functions, classes, and objects that the package provides and how to use them.

To use a module from a package, you need to import it using the import statement. For example, to use the random module from the `NumPy` package to generate random numbers, you would write:

```
import numpy as np

x = np.random.rand(5)
print(x)
```

This will generate an array of five random numbers between 0 and 1 and print it to the console.

Importing Specific Functions and Classes

To import a specific function or class from a module, you can use the `from` keyword. For example:

```
from numpy import random

x = random.rand(5)
print(x)
```

This method is useful if you only need to use a few functions or classes from a module. It also makes it easier to read your code, because you don't need to include the module name when you use the function or class.

Summary

In this tutorial section, you learned the basics on how to use external packages in Python, including how to install them, import them into your code, and use their modules, functions, and classes. By using external packages, you can extend the functionality of your Python programs and save time and effort by reusing code written by others.

Further Reading

Check out the following resources to learn more about external packages in Python:

- Python Modules and Packages – An Introduction - A guide to Python modules and packages by Real Python.
- Installing Packages on the Python Packaging User Guide
- Python Package Index - The official Python Package Index
- Anaconda - A package manager for Python and R
- Miniconda - A minimal version of Anaconda