

Using Python libraries

Chapter 19

This chapter covers

- Managing various data types—strings, numbers, and more
- Manipulating files and storage
- Accessing operating system services
- Using internet protocols and formats
- Developing and debugging tools
- Accessing PyPI (a.k.a. “The Cheese Shop”)
- Installing Python libraries and virtual environments using pip and venv

“Batteries included”: The standard library

- In Python, what's considered to be the library consists of several components, including built-in data types and constants that can be used without an import statement, such as numbers and lists, as well as some built-in functions and exceptions.
- The largest part of the library is an extensive collection of modules.
- If you have Python, you also have libraries to manipulate diverse types of data and files, to interact with your operating system, to write servers and clients for many internet protocols, and to develop and debug your code.

String services modules

Modules	Description and possible uses
<code>string</code>	Compare with string constants, such as digits or whitespace; format strings (see chapter 6)
<code>re</code>	Search and replace text using regular expressions (see chapter 16)
<code>struct</code>	Interpret bytes as packed binary data, and read and write structured data to/from files
<code>difflib</code>	Use helpers for computing deltas, find differences between strings or sequences, and create patches and diff files
<code>textwrap</code>	Wrap and fill text, and format text by breaking lines or adding spaces

Data types modules

Modules	Description and possible uses
<code>datetime,</code> <code>calendar</code>	Date, time, and calendar operations
<code>collections</code>	Container data types
<code>enum</code>	Allows creation of enumerator classes that bind symbolic names to constant values
<code>array</code>	Efficient arrays of numeric values
<code>sched</code>	Event scheduler
<code>queue</code>	Synchronized queue class
<code>copy</code>	Shallow and deep copy operations
<code>pprint</code>	Data pretty printer
<code>Typing</code>	Support for annotating code with hints as to the types of objects, particularly of function parameters and return values

Numeric and mathematical modules

Modules	Description and possible uses
<code>numbers</code>	Numeric abstractbase classes
<code>math</code> , <code>cmath</code>	Mathematical functions for real and complex numbers
<code>decimal</code>	Decimal fixed-point and floating-point arithmetic
<code>statistics</code>	Functions for calculating mathematical statistics
<code>fractions</code>	Rational numbers
<code>queue</code>	Synchronized queue class
<code>random</code>	Generate pseudorandom numbers and choices, and shuffle sequences
<code>itertools</code>	Functions that create iterators for efficient looping
<code>functools</code>	Higher-order functions and operations on callable objects
<code>operator</code>	Standard operators as functions

File and storage modules

Modules	Description and possible uses
<code>os.path</code>	Perform common pathname manipulations
<code>pathlib</code>	Deal with pathnames in an object-oriented way
<code>fileinput</code>	Iterate over lines from multiple input streams
<code>filecmp</code>	Compare files and directories
<code>tempfile</code>	Generate temporary files and directories
<code>glob</code> , <code>fnmatch</code>	Use UNIX-style pathname and filename pattern handling
<code>linecache</code>	Gain random access to text lines
<code>csv</code>	Read and write CSV files
<code>sqlite3</code>	Work with a DB-API 2.0 interface for SQLite databases
<code>zlib</code> , <code>gzip</code> , <code>bz2</code> , <code>zipfile</code> , <code>tarfile</code>	Work with archive files and compressions

Operating system modules

Modules	Description and possible uses
os	Miscellaneous operating system interfaces
io	Core tools for working with streams
time	Time access and conversions
optparse	Powerful command-line option parser
logging	Logging facility for Python
getpass	Portable password input
curses	Terminal handling for character-cell displays
platform	Access to underlying platform's identifying data
ctypes	Foreign function library for Python
select	Waiting for I/O completion

Modules supporting internet protocols and formats

Modules	Description and possible uses
<code>socket, ssl</code>	Low-level networking interface and SSL wrapper for socket Objects
<code>email</code>	Email and MIME handling package
<code>json</code>	JSON encoder and decoder
<code>html.parser,</code> <code>html.entities</code>	Parse HTML and XHTML
<code>cgi, cgiitb</code>	Common Gateway Interface support
<code>urllib.request,</code> <code>urllib.parse</code>	Open and parse URLs
<code>socketserver</code>	Framework for network servers
<code>http.server</code>	HTTP servers

Development, debugging, and runtime modules

Modules	Description and possible uses
<code>pydoc</code>	Documentation generator and online help system
<code>doctest</code>	Test interactive Python examples
<code>unittest</code>	Unit testing framework
<code>test.support</code>	Utility functions for tests
<code>pdb</code>	Python debugger
<code>trace</code>	Trace or track Python statement execution
<code>sys</code>	System-specific parameters and functions
<code>gc</code>	Garbage collector interface
<code>inspect</code>	Inspect live objects

Installing Python libraries using pip and venv

- Python offers pip as the current solution to both problems. pip tries to find the module in the Python Package index (more about that soon), downloads it and any dependencies, and takes care of the installation.
- The basic syntax of pip is quite simple.
- To install the popular requests library from the command line, for example, all you have to do is

```
python3.6 -m pip install requests
```

```
python3.6 -m pip install --upgrade requests
```

```
python3.6 -m pip install requests==2.11.1
```

```
python3.6 -m pip install --user requests
```

Virtual environments

- You have another, better option if you need to avoid installing libraries in the system Python.
- This option is called a virtual environment (virtualenv).
- A virtual environment is a self-contained directory structure that contains both an installation of Python and its additional packages.
- Because the entire Python environment is contained in the virtual environment, the libraries and modules installed there can't conflict with those in the main system or in other virtual environments, allowing different applications to use different versions on both Python and its packages.

```
python3.6 -m venv test-env  
test-env\Scripts\activate.bat  
pip install requests
```

PyPI (a.k.a. “The Cheese Shop”)

- Although distutils packages get the job done, there’s one catch: You have to find the correct package, which can be a chore.
- And after you’ve found a package, it would be nice to have a reasonably reliable source from which to download that package.
- To meet this need, various Python package repositories have been made available over the years.
- Currently, the official (but by no means the only) repository for Python code is the Python Package Index, or PyPI (formerly also known as “The Cheese Shop,” after the Monty Python sketch) on the Python website.
- You can access it from a link on the main page or directly at <https://pypi.python.org>.
- PyPI is the logical next stop if you can’t find the functionality you want with a search of the standard library.

Summary

- Python has a rich standard library that covers more common situations than many other languages, and you should check what's in the standard library carefully before looking for external modules.
- If you do need an external module, prebuilt packages for your operating system are the easiest option, but they're sometimes older and often hard to find.
- The standard way to install from source is to use pip, and the best way to prevent conflicts among multiple projects is to create virtual environments with the venv module.
- Usually, the logical first step in searching for external modules is the Python Package Index (PyPI).