**Gunicorn**

Gunicorn is a web server gateway interface (WSGI) server that's often used with Flask to deploy Python web applications because it's well-suited for production traffic:

* Scalability: Gunicorn can help deploy scalable web applications that can handle high traffic levels efficiently.
* Easy to install: Gunicorn is easy to install and doesn't require additional dependencies or compilation.
* Integrates with hosting platforms: Gunicorn integrates easily with hosting platforms.
* Built-in async worker support: Gunicorn has built-in async worker support using gevent or eventlet.

**Flask**

[Flask](http://flask.pocoo.org/) is a small and lightweight Python web framework that provides useful tools and features that make creating web applications in Python easier. It gives developers flexibility and is a more accessible framework for new developers since you can build a web application quickly using only a single Python file. Flask is also extensible and doesn’t force a particular directory structure or require complicated boilerplate code before getting started.

As part of this tutorial, you’ll use the [Bootstrap toolkit](https://getbootstrap.com/) to style your application so it is more visually appealing. Bootstrap will help you incorporate responsive web pages in your web application so that it also works well on mobile browsers without writing your own HTML, CSS, and JavaScript code to achieve these goals. The toolkit will allow you to focus on learning how Flask works.

Flask uses the [Jinja template engine](http://jinja.palletsprojects.com/) to dynamically build HTML pages using familiar Python concepts such as variables, loops, lists, and so on. You’ll use these templates as part of this project.

Reference: https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3

**Jinja Template engine**

Jinja is a fast, expressive, extensible templating engine. Special placeholders in the template allow writing code similar to Python syntax. Then the template is passed data to render the final document.

It includes:

* Template inheritance and inclusion.
* Define and import macros within templates.
* HTML templates can use autoescaping to prevent XSS from untrusted user input.
* A sandboxed environment can safely render untrusted templates.
* Async support for generating templates that automatically handle sync and async functions without extra syntax.
* I18N support with Babel.
* Templates are compiled to optimized Python code just-in-time and cached, or can be compiled ahead-of-time.
* Exceptions point to the correct line in templates to make debugging easier.
* Extensible filters, tests, functions, and even syntax.

Jinja’s philosophy is that while application logic belongs in Python if possible, it shouldn’t make the template designer’s job difficult by restricting functionality too much.

Reference: <https://jinja.palletsprojects.com/en/3.1.x/intro/>

**aniso8601**

The aniso8601 library is used to parse ISO 8601 strings into datetime format. It's a Python library that offers the following features:

* Pure Python implementation: Aniso8601 is implemented in pure Python.
* No regular expressions: Aniso8601 doesn't use regular expressions.
* UTC offset: Aniso8601 represents UTC offset as fixed-offset tzinfo.
* Logical behavior: Aniso8601 has logical behavior, such as parsing a time to get a datetime.time.
* Separate parser from representation: Aniso8601 has a separate parser from representation.
* Python 3 support: Aniso8601 supports Python 3.
* No extra dependencies: Aniso8601 doesn't require any extra dependencies.

You can use aniso8601 to parse times, dates, durations, intervals, and repeating intervals.

**Blinker**

Blinker provides fast & simple object-to-object and broadcast signaling for Python objects. The core of Blinker is quite small but provides powerful features: a global registry of named signals.

Reference: <https://blinker.readthedocs.io/en/stable/>

**Click**

To the best of the developer's knowledge, Click is the first Python library that aims to create a level of composability of applications that goes beyond what the system itself supports. Docopt, for instance, acts by parsing your help pages and then parsing according to those rules.

**Itsdangerous**

Sometimes you want to send some data to untrusted environments, then get it back later. To do this safely, the data must be signed to detect changes.

Given a key only you know, you can cryptographically sign your data and hand it over to someone else. When you get the data back you can ensure that nobody tampered with it.

The receiver can see the data, but they can not modify it unless they also have your key. So if you keep the key secret and complex, you will be fine.

Reference : <https://itsdangerous.palletsprojects.com/en/2.2.x/>

**MarkupSafe**

MarkupSafe implements a text object that escapes characters so it is safe to use in HTML and XML. Characters that have special meanings are replaced so that they display as the actual characters. This mitigates injection attacks, meaning untrusted user input can safely be displayed on a page.

Reference: https://pypi.org/project/MarkupSafe/

**Packaging**

Reusable core utilities for various Python Packaging [interoperability specifications](https://packaging.python.org/specifications/).

This library provides utilities that implement the interoperability specifications which have clearly one correct behaviour (eg: [PEP 440](https://peps.python.org/pep-0440)) or benefit greatly from having a single shared implementation (eg: [PEP 425](https://peps.python.org/pep-0425)).

The packaging project includes the following: version handling, specifiers, markers, requirements, tags, utilities.

Reference : https://pypi.org/project/packaging/

**python-dotenv**

Python-dotenv reads key-value pairs from a .env file and can set them as environment variables. It helps in the development of applications

Reference : https://pypi.org/project/python-dotenv/

**Pytz**

pytz brings the Olson tz database into Python. This library allows accurate and cross platform timezone calculations using Python 2.4 or higher. It also solves the issue of ambiguous times at the end of daylight saving time, which you can read more about in the Python Library Reference (datetime.tzinfo).

Almost all of the Olson timezones are supported.

Reference : https://pypi.org/project/pytz/

**PyYAML**

YAML is a data serialization format designed for human readability and interaction with scripting languages. PyYAML is a YAML parser and emitter for Python.

PyYAML features a complete YAML 1.1 parser, Unicode support, pickle support, capable extension API, and sensible error messages. PyYAML supports standard YAML tags and provides Python-specific tags that allow to represent an arbitrary Python object.

PyYAML is applicable for a broad range of tasks from complex configuration files to object serialization and persistence.

Reference : https://pypi.org/project/PyYAML/

**Werkzeug**

*werkzeug* German noun: "tool". Etymology: *werk* ("work"), *zeug* ("stuff")

Werkzeug is a comprehensive [WSGI](https://wsgi.readthedocs.io/en/latest/) web application library. It began as a simple collection of various utilities for WSGI applications and has become one of the most advanced WSGI utility libraries.

It includes:

* An interactive debugger that allows inspecting stack traces and source code in the browser with an interactive interpreter for any frame in the stack.
* A full-featured request object with objects to interact with headers, query args, form data, files, and cookies.
* A response object that can wrap other WSGI applications and handle streaming data.
* A routing system for matching URLs to endpoints and generating URLs for endpoints, with an extensible system for capturing variables from URLs.
* HTTP utilities to handle entity tags, cache control, dates, user agents, cookies, files, and more.
* A threaded WSGI server for use while developing applications locally.
* A test client for simulating HTTP requests during testing without requiring running a server.

Werkzeug doesn't enforce any dependencies. It is up to the developer to choose a template engine, database adapter, and even how to handle requests. It can be used to build all sorts of end user applications such as blogs, wikis, or bulletin boards.

[Flask](https://www.palletsprojects.com/p/flask/) wraps Werkzeug, using it to handle the details of WSGI while providing more structure and patterns for defining powerful applications.

Reference : https://pypi.org/project/Werkzeug/

**Six**

Six is a Python 2 and 3 compatibility library. It provides utility functions for smoothing over the differences between the Python versions with the goal of writing Python code that is compatible on both Python versions. See the documentation for more information on what is provided.

Six supports Python 2.7 and 3.3+. It is contained in only one Python file, so it can be easily copied into your project. (The copyright and license notice must be retained.)

Online documentation is at <https://six.readthedocs.io/>.

Bugs can be reported to <https://github.com/benjaminp/six>. The code can also be found there.