CS50 Python for Web Programming

Overview

In addition to using Python for writing local programs and algorithms, Python is often used for web programming. In web programming, Python is used for **server-side scripting**. In other words, it's used on the back end of a web application to implement web servers. While Python is one of many languages used for back end web programming, its readability, simplicity, and convenience all contribute to its popularity. In particular, Python has a built in HTTP server library called http.server that includes functions for listening, managing, and responding to HTTP requests. Because Python has a history of being used for web programming, many external web programming tools are built using and compatible with Python, such as Django and Flask. Python also offers easy integration with other tools and programming languages.

Key Terms

- server-side scripting
- Flask
- web framework
- micro framework
- Jinja

Flask

Flask is a micro web framework written in Python that provides programmers with tools to easily and quickly implement web applications. A **web framework** is software that provides tools, libraries, and extra technologies to build web applications, and a **micro framework** is a framework that is not highly dependent upon external resources. Because of tools like Flask, it is unnecessary for web programmers today to build web servers from scratch. Instead, by abstracting away lower level details, web programmers can focus on the logic of their specific web applications.

The code on the right shows the Python file of a simple web application. In the first line, we import some functionality from the Flask package. Then, we use Flask syntax create a new web application, allowing Flask to set up some low level details. Below that, we tell our applciation to go to the function index when the "/" route is requested. Within index, we've called the Flask function render_template to send the file index.html to our user's browser.

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route("/")
def index():
    return render_template("index.html")
```

As you can see, building a web application with Flask is incredibly simple. In addition to this basic functionality, Flask offers many other features that are useful for building web applications. To use these features, simply check out Flask's documentation.

layout.html

```
<!DOCTYPE html>

<html>
	<head>
		<title>Hello</title>
	</head>
	<body>
		{% block body %}{% endblock %}
	</body>
</html>
```

message.html

```
{% extends "layout.html" %}

{% block body %}
	Hello, world!
{% endblock %}
```

Jinja

Jinja is a template engine built into Flask. One of its features is a templating language that allows you to use dynamic elements (variables, loops, conditions) in static HTML/CSS files. Jinja expressions are very similar to Python expressions, making Jinja even more convenient to use. Jinja also enables inheritance of HTML/CSS files, minimizing rewritten code.

In the code on the left, the file message.html is using Jinja to inherit code from layout.html. Inheritance of templates demonstrates better programming design because it reduces repeated code, maintains consistency, and makes templates more convenient to update.

Like Flask, you can learn more about features of Jinja by looking at the Jinja documentation.