Developing AI Applications with Python and Flask

Table of Contents

[MODULE 1 Application Development and Packaging using Python 2](#_Toc180778264)

[Python with Flask for Large-Scale Projects 2](#_Toc180778265)

[Static Code Analysis 2](#_Toc180778266)

[Unit Testing 2](#_Toc180778267)

[Python Packaging 3](#_Toc180778268)

[Python Coding Practices and Packaging Concepts 3](#_Toc180778269)

[Module 1 Cheatsheet: Python Coding Practices and Packaging Concepts 3](#_Toc180778270)

[MODULE 2 Web Application Deloyment Using Flask 4](#_Toc180778271)

[Get Started with Flask Basics 4](#_Toc180778272)

[MODULE 3 Creating AI Applications and Deploy using Flask 5](#_Toc180778273)

## **MODULE 1 Application Development and Packaging using Python**

## Python with Flask for Large-Scale Projects

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-PY0103EN-SkillsNetwork/labs/Python_and_Flask_for_large_scale_projects.md.html?origin=www.coursera.org>

## Static Code Analysis

Pylint test.py

## Unit Testing

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTURldmVsb3BlclNraWxsc05ldHdvcmstUFkwMjIyRU4tU2tpbGxzTmV0d29yay9sYWJzL21vZHVsZV8xL1VuaXRfVGVzdGluZy5tZCIsInRvb2xfdHlwZSI6InRoZWlhIiwiYWRtaW4iOmZhbHNlLCJpYXQiOjE3MjEwMzM1ODB9.J7otfk4QG00ra7BOlNaLtoxx4GcbnBD01gy-EC92AOU>

## Python Packaging

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTURldmVsb3BlclNraWxsc05ldHdvcmstUFkwMjIyRU4tU2tpbGxzTmV0d29yay9sYWJzL21vZHVsZV8xL0NyZWF0aW5nX2FfUHl0aG9uX1BhY2thZ2UubWQiLCJ0b29sX3R5cGUiOiJ0aGVpYSIsImFkbWluIjpmYWxzZSwiaWF0IjoxNzIxMDM1NTI1fQ.lZUt6-ycknYw5b0Lp_q7TnZms3fglm3oPHwunT8Q5c8>

## Python Coding Practices and Packaging Concepts

* The application development lifecycle has seven phases, including:
* Requirement Gathering: You collect user, business, and technical requirements for the app
* Analysis: You analyze the requirements
* Design: You design the complete solution
* Code and test: You build and test the different components of the app
* User and system test: Users test the app for usability, and you perform system integration testing and performance testing
* Production: The application is available to all end users
* Maintenance: You upgrade or fix any user or system issues
* All web apps are APIs, but not all APIs are web apps. Both share data between apps, but not all APIs require networks like web apps do.
* The PEP8 guidelines for code readability include the following:
* Four spaces for indentation
* Blank lines to separate functions and classes
* Spaces around operators and after commas
* The PEP8 coding conventions for consistency and manageability include:
* Add larger blocks of code inside functions
* Name functions and files using lowercase with underscores
* Name classes using CamelCase
* Name constants in capital letters with underscores separating words
* To ensure that your code adheres to the predefined style and standard without executing the code, you can use the Static code analysis method.
* Unit testing is a method to validate if code units are operating as designed. You must test every unit before integration with the final codebase.
* To create a package:
* Create a folder with the package name
* Create an empty \_\_init\_\_.py file
* Create the required modules
* In the \_\_init\_\_.py file, add code to reference the modules needed in the package
* You can verify the package via the bash terminal in a Python shell.

## Module 1 Cheatsheet: Python Coding Practices and Packaging Concepts

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-PY0224EN-Coursera/labs/reading/m1/Cheatsheet.md.html?origin=www.coursera.org>

## **MODULE 2 Web Application Deloyment Using Flask**

## Get Started with Flask Basics

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-CD0320EN-SkillsNetwork/labs/practice_with_flask.md.html>

Building an API with Flask: Route Creation, Error Handling, and HTTP Requests

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-CD0320EN-SkillsNetwork/labs/Practice_with_Flask_Part2.md.html>

Decorators in Flask

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0222EN-SkillsNetwork/labs/module_1/Decorators_Flask.md.html?origin=www.coursera.org>

Building and Deploying a Web App using Flask App

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-PY0223EN-SkillsNetwork/labs/module_2/Building_and_Deploying_a_Web_App_using_Flask2.md.html>

Additional Features in Flask

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/kRI7CNfBPXrkEkL6HkCV0A.md.html?origin=www.coursera.org>

Exploring Additional Features Flask

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/nIlEoQmUUaC_50bCKkU5iA.md.html>

Module 2 Lesson Summary: Web App Deployment using Flask

* Python libraries are like toolkits. Each library has specific tools to simplify and expedite certain programming tasks. Frameworks are predefined structures for application development. Framework enables you to build the complete application, while libraries aid with specific functionality.
* Flask is a microframework that ships with minimal dependencies. To build websites, Flask has features like debugging servers, routing, templates, and error handling. Flask can be installed as a python package. Django is a full-stack framework compared to Flask. You can create a server by instantiating the Flask class.
* Flask provides a Request and a Response object for each client call. You can get additional information from the Flask Request, like headers. You can parse the Request Object to get query parameters, body, and other arguments. You can even set status on Response objects before sending a response back to the client.
* You can use dynamic routes to create RESTful endpoints.
* There are multiple classes of HTTP status codes showing success, user error, or server error. Flask implicitly returns a success code of 200 with the response. You can also provide status codes explicitly. Flask also provides application-level error handlers.
* Flask is a microframework for creating web applications and supports CRUD.
* Install the Flask package using pip.
* To create a web application using Flask:
  + Import Flask
  + Instantiate Flask
  + Run the app
* You can render both static and dynamic templates with Flask.

Module 2 Cheatsheet: Web App Deployment using Flask

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-PY0224EN-Coursera/labs/reading/m2/Cheatsheet.md.html?origin=www.coursera.org>

## **MODULE 3 Creating AI Applications and Deploy using Flask**

Practice Project: Sentiment Analysis

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-PY0224EN-Coursera/labs/v1/practice_project_AI_embAI_test.md.html>

Final Project: Emotion Detector

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-PY0224EN-Coursera/labs/v1/practice_project_AIwebapp_using_Flask.md.html>

Module 3 Summary: Creating AI Application and Deploy using Flask

* Watson AI libraries are embeddable libraries that are preinstalled in the Cloud IDE framework. These embeddable AI libraries provide a variety of NLP and speech-based functions based on popular AI models.
* In the projects, you developed web apps with incorporated AI functionality using IBM Watson libraries.
* Both projects followed the same application creation steps including:
  + Create a function for the application
  + Package the application function
  + Run unit testing of the function by calling it as a package
  + Deploy the application over the web using Flask
  + Incorporate Error handling to the application
  + Run static code analysis on the codes created