Arjun Bamba

Professor Liv d’Aliberti

Modern Software Concepts in Python

September 28, 2025

# Module 5

## Dependency Analysis

The pydeps visualization shows the app.py file making use of the Flask and psycopg ecosystems along with custom modules from homework_sample_code. It shows app.py importing the core Flask package which also brings in plenty of submodules such as flask.app, flask.ctx, flask.globals, flask.wrappers, flask.helpers, flask.json, and flask.blueprints. These modules help the app by making it possible for it to use features such as request/response handling, global objects, JSON serialization, and template rendering. Additionally, it shows the app also making use of flask.signals and flask.config which help it manage component configurations and facilitate signaling between components.

On the db side, app.py imports psycopg which also brings in a large network of submodules including psycopg.Connection, psycopg.cursor, psycopg.types, psycopg.errors, and psycopg.pq. These modules allows it to use features such as database connections, query execution, error handling, and asynchronous execution. Furthermore, app.py also incorporates our custom modules, such as homework_sample_code. As we know, this includes the script to query and load our data and then further down in the course_app/ directory, it includes my sql util components as well as all my logic to scrape/clean data. These psycopg and my custom components is what allows the application-specific logic and SQL integration logic to function.

Through using Flask for the web interface, psycopg for db access, and our custom homework modules for its business logic, the pydeps visualization shows the dependencies of app.py which help us get a better sense of the layers/features of the overall application.