# Developing AI Applications with Python and Flask

Maulana Hafidz Ismail

16 Oktober 2025

# Contents

# 1 Python Coding Practices and Packaging Concepts

## 1.1 Introduction

Python dengan Flask adalah framework aplikasi web yang ringan dan fleksibel. Framework ini dikenal karena kesederhanaan, minimalis, dan kemudahan penggunaannya. Dirancang sebagai micro-framework yang menyediakan struktur ringan yang memudahkan pengembang membangun aplikasi web dengan cepat dan mudah, tanpa mengorbankan efisiensi dan kemampuan untuk meningkatkan skala proyek dari skala kecil ke aplikasi yang lebih besar dan kompleks.

## 1.2 Python with Flask for large-scale development

Flask adalah pilihan yang baik untuk aplikasi yang lebih kecil dan sederhana. Namun, istilah "mikro" lebih berkaitan dengan apa itu Flask, alih-alih membatasi potensi skalabilitasnya. Flask dapat digunakan untuk sistem berskala besar dan aplikasi yang lebih kompleks dengan memperhatikan persyaratan dan batasan spesifik, perencanaan yang cermat, arsitektur yang baik, dan desain modular. Namun, Flask mungkin memerlukan upaya pengelolaan dan penskalaan yang lebih besar dibandingkan dengan kerangka kerja yang lebih tangguh dan kaya fitur.

Ekosistemnya yang kaya dan tangguh menyediakan alat, pustaka, dan fungsionalitas bagi pengembang untuk menangani tugas-tugas pengembangan web seperti perutean, penanganan permintaan, rendering templat, atau tugas serupa. Caching, penyeimbangan beban, replikasi, dan penyimpanan data Anda secara skalabel dapat membantu mencapai hasil yang optimal.

## 1.3 Key Flask Capabilities

- **Extensibility and integration**: Flask dapat diperluas dan pengembang dapat menambahkan atau menghapus fitur yang memungkinkan kustomisasi. Flask terintegrasi secara mulus dengan pustaka dan kerangka kerja Python lainnya, memungkinkan pengembang untuk menggabungkan fungsionalitasnya dengan alat dan teknologi lain, sehingga meningkatkan kemampuannya.

- **Transparent documentation**: Dokumentasi Flask diterbitkan, memungkinkan pengembang untuk menggunakan API dan utilitas internalnya serta menemukan titik kait, penggantian, dan sinyal sesuai kebutuhan.

- **Custom implementation**: Kustomisasi bawaan dan kelas khusus dapat digunakan untuk hal-hal seperti objek permintaan dan respons. Kelas Flask memiliki banyak metode yang dirancang untuk subkelas. Anda dapat dengan cepat menambahkan atau menyesuaikan perilaku dengan membuat subkelas Flask dan menggunakan subkelas tersebut di mana pun Anda membuat instance kelas aplikasi.

- **Scaling considerations**: Anda dapat menggunakan penskalaan sedemikian rupa sehingga jika Anda menggandakan jumlah server, Anda akan mendapatkan kinerja sekitar dua kali lipat. Hanya ada satu faktor pembatas terkait penskalaan di Flask, yaitu penggunaan proksi lokal konteks. Proksi ini bergantung pada konteks yang dalam Flask didefinisikan sebagai thread, proses, atau greenlet. Jika server Anda menggunakan jenis konkurensi yang tidak berbasis thread atau greenlet, Flask tidak akan lagi dapat mendukung proksi global ini.
- **Modular development**: Carilah cara-cara agar proyek Anda dapat direfaktor menjadi kumpulan utilitas dan ekstensi Flask. Jelajahi berbagai ekstensi di komunitas dan cari pola untuk membangun ekstensi Anda sendiri jika Anda tidak menemukan alat yang dibutuhkan. Cara terbaik untuk meningkatkan alat untuk aplikasi yang lebih besar adalah dengan mendapatkan umpan balik dari pengguna.

## 1.4 Real-world applications

Saat ini, Python dengan Flask telah menjadi pilihan populer di kalangan perusahaan besar karena kesederhanaan, fleksibilitas, fleksibilitas, serta kemudahan pembelajaran dan penggunaannya. Desainnya yang minimalis dan sifatnya yang dapat dikustomisasi menjadikannya adaptif, efektif, dan andal untuk kebutuhan pengembangan web skala besar di berbagai industri dan sektor.

Beberapa perusahaan terkemuka, termasuk Netflix, Reddit, Lyft, LinkedIn, Pinterest, dan Uber, memanfaatkan Python dengan Flask dalam tumpukan teknologi mereka untuk layanan atau fungsionalitas backend tertentu. Python Flask menguntungkan perusahaan besar untuk berbagai tujuan seperti pengembangan API, layanan backend, pengembangan cepat, dan pembuatan prototipe, sementara ekstensibilitasnya memfasilitasi penambahan fungsionalitas ke dalam infrastruktur mereka. Hal ini menunjukkan bahwa Python Flask dapat menjadi bagian dari arsitektur yang skalabel jika dikombinasikan dengan strategi dan alat yang tepat.

## 1.5 Fase dalam Application Development Lifecycle

- Requirement Gathering:

  Mengumpulkan kebutuhan dari pengguna, bisnis, dan teknis. Contoh: Pengguna harus dapat melihat berbagai kamar dan fasilitas yang tersedia.

- Analysis:

  Menganalisis kebutuhan yang telah dikumpulkan untuk merancang solusi yang mungkin. Penting untuk mendokumentasikan semua pembaruan dalam desain.

- Design:

  Merancang solusi lengkap berdasarkan analisis yang dilakukan. Dokumentasi yang jelas dan ringkas diperlukan untuk fase berikutnya.

- Code and Test:

  Menggunakan dokumentasi desain untuk mengkode, menguji, dan merevisi aplikasi hingga memenuhi semua spesifikasi. Unit Testing: Pengujian pada level unit untuk memastikan setiap bagian kode berfungsi sesuai harapan.

- User and System Test:

  Pengujian dari sudut pandang pengguna dan pengujian sistem untuk memastikan aplikasi berfungsi dengan baik. Integration Testing: Memastikan semua program berfungsi setelah diintegrasikan. Performance Testing: Mengukur kecepatan, skalabilitas, dan stabilitas aplikasi.

- Production:

  Aplikasi tersedia untuk pengguna akhir. Aplikasi harus dalam keadaan stabil dan tidak mengalami perubahan besar.

- Maintenance:

  Aplikasi mungkin memerlukan pembaruan atau penambahan fitur baru. Fitur baru harus melalui semua fase sebelumnya sebelum diintegrasikan ke dalam aplikasi yang sudah ada.

## 1.6 Praktik Terbaik dalam Pengkodean

- Menggunakan beberapa file untuk mengkode berbagai fungsionalitas.
- Membuat program pusat yang memanggil file-file individual untuk menjalankan fungsi tertentu.

## 1.7 Web Applications

- Definisi: Web application adalah program yang disimpan di server jarak jauh dan diakses melalui internet menggunakan browser. Contoh: situs e-commerce, webmail.
- Komponen: Terdapat tiga komponen utama untuk memproses permintaan klien:
  - Web Server: Mengelola permintaan yang diajukan.
  - App Server: Menjalankan tugas yang diminta.
  - Database: Menyimpan informasi yang diperlukan untuk menyelesaikan tugas.
- Pengembangan: Kode ditulis untuk front-end menggunakan JavaScript, HTML, atau CSS, dan untuk back-end menggunakan Python, Java, atau Ruby.

## 1.8 Keuntungan Web Applications

- Versi aplikasi yang sama dapat diakses oleh banyak pengguna secara bersamaan.
- Pengguna dapat menggunakan aplikasi dari berbagai platform (desktop, laptop, mobile).
- Aplikasi dapat diakses melalui browser tanpa perlu instalasi.

## 1.9 Application Programming Interface (API)

- Definisi: API adalah komponen perangkat lunak yang memungkinkan dua aplikasi yang tidak terhubung untuk berkomunikasi. API memiliki aturan dan fungsi standar untuk menentukan data yang dapat diambil atau dimodifikasi.
- Contoh: Aplikasi cuaca yang meminta data dari weather API untuk memberikan ramalan cuaca.
- Arsitektur: API dapat dibangun menggunakan arsitektur seperti **Representational State Transfer (REST)** atau **Simple Object Access Protocol (SOAP)**.
- Manfaat API :
  - Meningkatkan konektivitas antara aplikasi.
  - Mendukung tindakan CRUD (Create, Read, Update, Delete).
  - Bekerja dengan HTTP verbs seperti PUT, POST, DELETE, dan GET.
  - Dapat disesuaikan karena berbasis HTTP.

## 1.10 Perbandingan Web Application dan APIs

- Semua web applications adalah APIs, tetapi tidak semua APIs adalah web applications.
- Contoh: Dalam layanan belanja e-commerce, browser bertindak sebagai API yang menghubungkan pengguna dengan web application.

## 1.11 PEP-8 Guidelines untuk Keterbacaan Kode

- Indentasi: Gunakan empat spasi untuk setiap level indentasi. Ini penting karena editor teks dapat menginterpretasikan tab dengan cara yang berbeda, yang dapat menyebabkan kesalahan format.
- Baris Kosong: Gunakan baris kosong untuk memisahkan fungsi dan kelas. Ini membantu dalam memahami struktur kode dengan lebih baik.

## 1.12 Konvensi Pengkodean untuk Konsistensi

- Fungsi Terpisah: Buat fungsi terpisah untuk blok kode yang lebih besar. Ini meningkatkan kecepatan eksekusi dan mendukung penggunaan kembali blok kode. Contoh sintaks:

```
def function_one(a, b):
return a + b
```

- Penamaan Fungsi dan File: Gunakan huruf kecil dengan garis bawah (lowercase with underscores) untuk nama fungsi dan file. Ini mengikuti konvensi Python yang umum. Contoh: `def calculate_area()`
- Penamaan Kelas: Gunakan CamelCase untuk nama kelas. Ini membantu membedakan antara kelas dan fungsi. Contoh: `class MyClass`
- Konstanta: Gunakan huruf kapital dengan garis bawah untuk nama konstanta. Ini menunjukkan tujuan konstanta tersebut. Contoh: `MAX_FILE_UPLOAD_SIZE`

## 1.13 Analisis Kode Statis

Analisis kode statis, atau analisis statis, adalah aktivitas verifikasi efisiensi kode aplikasi yang menganalisis kode sumber untuk kualitas, keandalan, dan keamanan tanpa mengeksekusi kode tersebut. Analisis kode statis merupakan bagian penting dari setiap siklus pengembangan aplikasi dan tersedia sebagai bagian dari berbagai kerangka kerja dengan Python.

Salah satu kerangka kerja yang paling populer adalah PyLint. PyLint pada dasarnya mengevaluasi kode berdasarkan kepatuhan terhadap panduan gaya pengkodean PEP8 dan menghasilkan komentar setiap kali menemukan masalah.

- Static Code Analysis: Metode untuk mengevaluasi kode terhadap gaya dan standar yang telah ditentukan tanpa mengeksekusi kode. Ini membantu menemukan masalah seperti kesalahan pemrograman dan pelanggaran standar pengkodean Contoh alat: **PyLint** digunakan untuk memeriksa kepatuhan kode Python terhadap pedoman PEP-8.

## 1.14

-

# 2 Web App Deployment using Flask

# 3 Creating AI Application and Deploy using Flask