**Chat Bot Gemini Menggunakan Bahasa Pemrograman Python dengan API**

1. Apa itu API ?

Api (Aplication Programming Interface) adalah sebuah perantara yang memungkinkan dua aplikasi atau perangkat lunak untuk saling berkomunikasi dan bertukar data.

1. Bagaimana cara kerja API?

API bekerja dengan cara mendefinisikan serangkaian aturan dan protokol yang harus diikuti oleh aplikasi untuk dapat berinteraksi satu sama lain. Aturan dan protokol ini biasanya didokumentasikan dalam bentuk spesifikasi API.

Berikut cara kerja API secara umum:

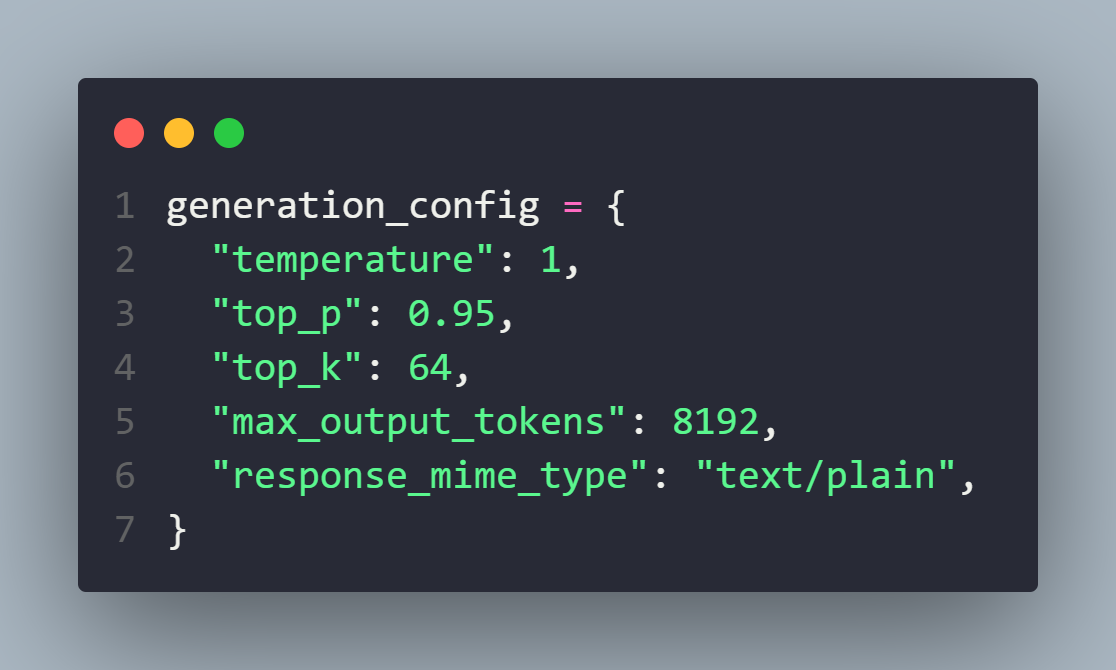
* Aplikasi klien membuat permintaan: Aplikasi klien, yang ingin mengakses data atau layanan dari aplikasi lain, mengirimkan permintaan ke API. Permintaan ini biasanya dikirim melalui HTTP (Hypertext Transfer Protocol), dan berisi informasi tentang data atau layanan yang diinginkan, serta autentikasi dan otorisasi untuk mengaksesnya.
* Server API menerima permintaan: Server API, yang merupakan aplikasi yang menyediakan data atau layanan, menerima permintaan dari aplikasi klien. Server API kemudian memproses permintaan dan menghasilkan respons.
* Server API mengirimkan respons: Server API mengirimkan respons ke aplikasi klien. Respons ini biasanya berisi data atau layanan yang diminta, serta informasi status dan pesan kesalahan jika ada.
* Aplikasi klien memproses respons: Aplikasi klien memproses respons dari server API dan menggunakan data atau layanan yang diterima.





load\_dotenv(): Kode ini memanggil fungsi load\_dotenv untuk memuat variabel dari file .env

genai.configure(api\_key=os.getenv("GOOGLE\_API\_KEY")): Baris ini mengonfigurasi library genai dengan kunci API Anda yang diambil dari variabel getenv.



generation\_config: Dictionary ini mendefinisikan parameter konfigurasi untuk proses pembuatan obrolan.

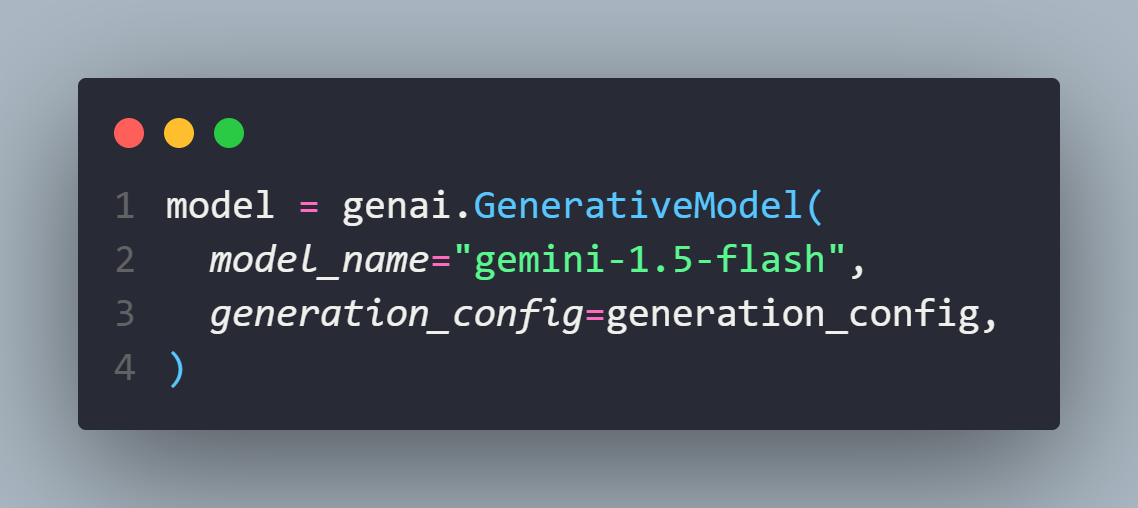
 temperature: Mengontrol keacakan teks yang dihasilkan. Nilai 1.0 menghasilkan output yang lebih seimbang antara kata umum dan tidak umum.

 top\_p: Parameter ini memfilter kosakata yang digunakan untuk pembuatan, berfokus pada p kata teratas yang paling mungkin pada setiap langkah. Di sini, memfilter 5% yang paling kecil kemungkinannya.

 top\_k: Parameter ini hanya mempertimbangkan k kata teratas yang paling mungkin pada setiap langkah selama pembuatan. Di sini, mempertimbangkan 64 kata teratas.

 max\_output\_tokens: Menetapkan jumlah maksimum token (kata) yang akan dihasilkan model dalam sebuah respons.

 response\_mime\_type: Menentukan format respons model, diatur ke teks biasa di sini.



Model = genai.GenerativeModel(model\_name="gemini-1.5-flash", generation\_config=generation\_config): Baris ini membuat objek GenerativeModel dari pustaka genai. Ini menentukan nama model sebagai "gemini-1.5 flash" (yang mengacu pada versi spesifik model Gemini) dan menggunakan generation\_config yang didefinisikan sebelumnya untuk interaksi obrolan.



history = []: Baris ini menginisialisasi daftar kosong bernama history yang akan menyimpan riwayat percakapan antara pengguna dan model.



while True: code ini memulai loop yang berlanjut hingga secara eksplisit dihentikan.

* user\_input = input("Anda: "): code ini meminta masukan pengguna dan menyimpannya dalam variabel user\_input.
* if user\_input.strip() == '':: Pernyataan kondisional ini memeriksa apakah input pengguna kosong (setelah menghapus spasi putih di depan/di belakang). Jika demikian, loop berhenti, mengakhiri obrolan.
* chat\_session = model.start\_chat(history=history): Baris ini memulai sesi obrolan baru dengan model, memberikan riwayat percakapan untuk konteks.
* response = chat\_session.send\_message(user\_input): Baris ini mengirimkan input pengguna ke model dan menyimpan objek respons dalam variabel response.
* model\_response = response.text: Baris ini mengekstrak konten teks dari respons model dan menyimpannya dalam variabel model\_response.
* print(f'bot: {model\_response}'): Baris ini mencetak respons bot ke konsol.
* history.append({"role": "user", "parts": [user\_input]}): Baris ini menambahkan input pengguna ke riwayat percakapan dengan peran yang disetel ke "pengguna".
* history.append({"role":"model","parts":[model\_response]}): Baris ini menambahkan respons model ke riwayat percakapan dengan peran yang disetel ke "model".

