Everything about Gemini

What is Gemini?

↳ a LLM (large language Model)

What is LLM?

↳ LLM = L + LM
= Large + Language Model

What is Language Model?

↳ LM = L + M = Language Model

A model is a representation of something, how does that work from insite.

So, LM is a model of **Language**

What does LM do?

↳ Help us know the intricacies of a given Language (Eng, Hindi Etc)

↳ Predict the next word given a set of words

~~P(Hello~~

P(you | thank) → ? (High)

P(he | thank) → ? (Low)

↳ we can compute the prob by taking any text (wiki, Books)

↳ what if we take the entire **Internet**

So, LLM = a Language Model which is trained on the entire Internet Language Data.

How is it trained? → **Next time**

↳ variants of Gemini →

(1) Gemini 1.0
(2) Gemini 1.5
(3) Gemini 2.0 (latest)

} for each of these we have,
↳ Pro
↳ Flash

↳ How do these differ? which one to choose?

↳ Context Length → the no of (tokens | words) present in your prompt | question

↳ Input Data Types → Audio, Text, Video, Images

↳ RPM → Request Per Minute

↳ TPM → Tokens | Words Per minute

↳ RPD → RP[Day]

↳ TPD → TP[Day].

100 tokens ≈ 60–80 English words

	2.0 (F)	1.5 (F)	1.5 (F) - 8B	1.5 Pro	1.0 Pro
CL(I)	1 Mil	1 Mil	1 mil	2 mil	—
CL(O)	8K	8K	8K	8K	8K
IDT	ALL	ALL	ALL	ALL	Text
RPM	10	15 | 2000	15 | 4000	2 | 1000	15 | 360
TPM	4 mill	1 | 4 mil	1 | 4 Mil	32K | 4 mil	32K | 120K
RPD	1500	1500	1500	50	1.5K | 32K
ODT	Text (More in future)	Text	Text	Text	Text

↳ API Accessible for

↳ Python

↳ JS (via NodeJS)

↳ Simple endpoint → callable via CURL, Python requests etc

↳ Installation :

↳ Python

↳ pip install -U google-generativeai

↳ A simple generation → Demo

↳ Image prompting ~~generation~~ → Demo

↳ Chat-interface → Demo

↳ Generation configuration → Demo

↳ A way to control the generation process via

a). Max output tokens (max_output_tokens)

b). setting temperature

c). providing the schema for JSON parsing

↳ Streaming rather than generation.

↳ Getting Structured Outputs

a) LLMs are by nature probabilistic models.

b) Same question / prompt two different times would yield two different results

c) How to get predictive responses?

↳ Intelligent prompting and then parsing

↳ Using gemini's built in generation config and supplying schema

```
model.generate_content( response_mime_type = "application/json",
                        response_schema = list[Recipe])
```

Typed Dict Representation of Schema

↳ What is streaming?

↳ By default, the Gemini is going to respond when the entire generation is completed

↳ We can control this and keep the interaction with Gemini Interactive by setting **stream = True**