

October 2024

116 posts: [12 entries](#), [82 links](#), [22 quotes](#)

Oct. 1, 2024

[Reddit is] mostly ported over entirely to Lit now. There are a few straggling pages that we're still working on, but most of what everyday typical users see and use is now entirely Lit based. This includes both logged out and logged in experiences.

— [Jim Simon](#), Reddit

[12:09 am](#) / [web-components](#), [reddit](#), [lit-html](#), [javascript](#)

[Whisper large-v3-turbo model](#). It's [OpenAI DevDay](#) today. Last year they released a whole stack of new features, including GPT-4 vision and GPTs and their text-to-speech API, so I'm intrigued to see what they release today (I'll be at the San Francisco event).

Looks like they got an early start on the releases, with the first new Whisper model since November 2023.

Whisper Turbo is a new speech-to-text model that fits the continued trend of distilled models getting smaller and faster while maintaining the same quality as larger models.


large-v3-turbo is 809M parameters - slightly larger than the 769M medium but significantly smaller than the 1550M large. OpenAI claim its 8x faster than large and requires 6GB of VRAM compared to 10GB for the larger model.

The model file is a 1.6GB download. OpenAI continue to make Whisper (both code and model weights) available under the MIT license.

It's already supported in both Hugging Face transformers - [live demo here](#) - and in [mlx-whisper](#) on Apple Silicon, [via Awni Hannun](#):

```
import mlx_whisper
print(mlx_whisper.transcribe(
    "path/to/audio",
    path_or_hf_repo="mlx-community/whisper-turbo"
)["text"])
```

Awni reports:

 Transcribes 12 minutes in 14 seconds on an M2 Ultra (~50X faster than real time).

[3:13 pm](#) / [ai](#), [openai](#), [whisper](#), [mlx](#)

[OpenAI DevDay 2024 live blog](#)



I'm at [OpenAI DevDay](#) in San Francisco, and I'm trying something new: a live blog, where this entry will be updated with new notes during the event.

[... [68 words](#)]

[5:17 pm](#) / [blogging](#), [ai](#), [openai](#), [generative-ai](#), [llms](#)

[Oct. 2, 2024](#)

[Building an automatically updating live blog in Django](#). Here's an extended write-up of how I implemented the live blog feature I used for [my coverage of OpenAI DevDay](#) yesterday. I built the first version using Claude while waiting for the keynote to start, then upgraded it during the lunch break with the help of GPT-4o to add sort options and incremental fetching of new updates.

[# 3:42 pm](#) / [django](#), [javascript](#), [ai](#), [generative-ai](#), [chatgpt](#), [llms](#), [ai-assisted-programming](#), [claude](#)

[Ethical Applications of AI to Public Sector Problems](#). Jacob Kaplan-Moss developed this model a few years ago (before the generative AI rush) while working with public-sector startups and is publishing it now. He starts by outright dismissing the snake-oil infested field of “predictive” models:

It's not ethical to predict social outcomes — and it's probably not possible. Nearly everyone claiming to be able to do this is lying: their algorithms do not, in fact, make predictions that are any better than guesswork. [...] Organizations acting in the public good should avoid this area like the plague, and call bullshit on anyone making claims of an ability to predict social behavior.

Jacob then differentiates assistive AI and automated AI. Assistive AI helps human operators process and consume information, while leaving the human to take action on it. Automated AI acts upon that information without human oversight.

His conclusion: yes to assistive AI, and no to automated AI:

All too often, **AI algorithms encode human bias**. And in the public sector, failure carries real life or death consequences. In the private sector, companies can decide that a certain failure rate is OK and let the algorithm do its thing. But when citizens interact with their governments, they have an expectation of fairness, which, because AI judgement will always be available, it cannot offer.

On Mastodon [I said to Jacob](#):

I'm heavily opposed to anything where decisions with consequences are outsourced to AI, which I think fits your model very well

(somewhat ironic that I wrote this message from the passenger seat of my first ever Waymo trip, and this weird car is making extremely consequential decisions dozens of times a second!)

Which sparked an interesting conversation about why life-or-death decisions made by self-driving cars feel different from decisions about social services. My take on that:

I think it's about judgement: the decisions I care about are far more deep and non-deterministic than "should I drive forward or stop".

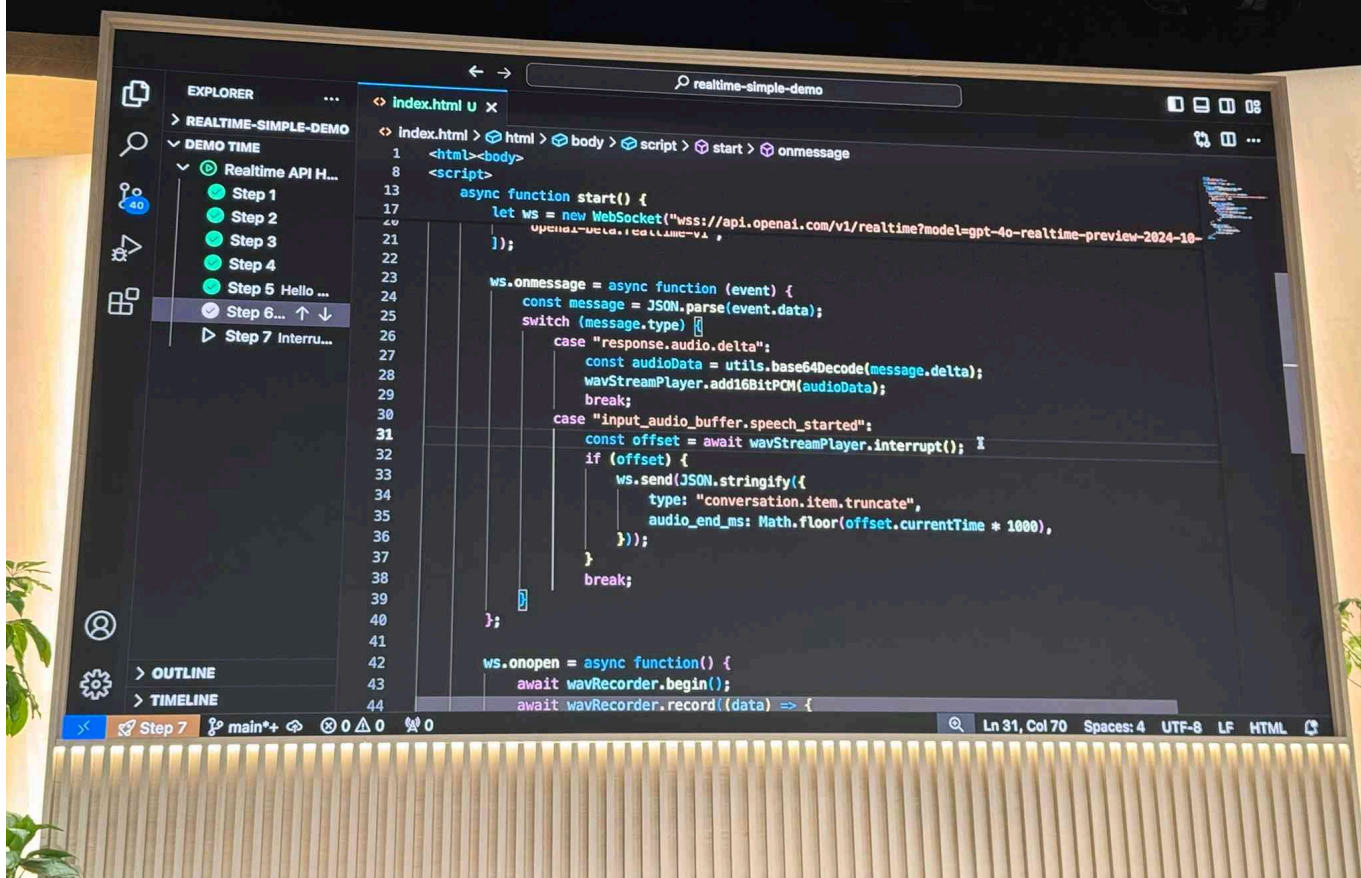
[Jacob](#):

Where there's moral ambiguity, I want a human to own the decision both so there's a chance for empathy, and also for someone to own the accountability for the choice.

That idea of ownership and accountability for decision making feels critical to me. A giant black box of matrix multiplication cannot take accountability for "decisions" that it makes.

[# 5:42 pm](#) / [ethics](#), [jacob-kaplan-moss](#), [ai](#), [ai-ethics](#)

[OpenAI DevDay: Let's build developer tools, not digital God](#)



I had a fun time [live blogging OpenAI DevDay yesterday](#)—I’ve now [shared notes](#) about the live blogging system I threw other in a hurry on the day (with assistance from Claude and GPT-4o). Now that the smoke has settled a little, here are my impressions from the event.

[... [2,090 words](#)]

[10:33 pm](#) / [websockets](#), [ai](#), [openai](#), [generative-ai](#), [llms](#), [prompt-caching](#)

[Oct. 3, 2024](#)

[Ask HN: What happens to “.io” TLD after UK gives back the Chagos Islands?](#) This morning on the BBC: [UK will give sovereignty of Chagos Islands to Mauritius](#). The Chagos Islands include the area that the UK calls [the British Indian Ocean Territory](#). The [.io ccTLD](#) uses the ISO-3166 two-letter country code for that designation.

As the owner of [datasette.io](#) the question of what happens to that ccTLD is suddenly very relevant to me.

This Hacker News conversation has some useful information. It sounds like there's a very real possibility that [.io](#) could be deleted after a few years notice - it's happened before, for ccTLDs such as [.zr](#) for Zaire (which renamed to [Democratic Republic of the Congo](#) in 1997, with [.zr](#) withdrawn in 2001) and [.cs](#) for Czechoslovakia, withdrawn in 1995.

Could [.io](#) change status to the same kind of TLD as [.museum](#), unaffiliated with any particular geography? The convention is for two letter TLDs to exactly match ISO country codes, so that may not be an option.

[# 5:25 pm](#) / [dns](#), [domains](#), [hacker-news](#)

[Announcing FLUX1.1 \[pro\] and the BFL API \(via\)](#) FLUX is the image generation model family from Black Forest Labs, a startup founded by members of the team that previously created Stable Diffusion.

Released today, FLUX1.1 [pro] continues the general trend of AI models getting both better and more efficient:

FLUX1.1 [pro] provides six times faster generation than its predecessor FLUX.1 [pro] while also improving image quality, prompt adherence, and diversity.

Black Forest Labs appear to have settled on a potentially workable business model: their smallest, fastest model FLUX.1 [schnell] is Apache 2 licensed. The next step up is FLUX.1 [dev] which is open weights for non-commercial use only. The [pro] models are closed weights, made available exclusively through their API or partnerships with other API providers.

I tried the new 1.1 model out using [black-forest-labs/flux-1.1-pro](#) on Replicate just now. Here's my prompt:

Photograph of a Faberge egg representing the California coast. It should be decorated with ornate pelicans and sea lions and a humpback whale.



The FLUX models have a reputation for being really good at following complex prompts. In this case I wanted the sea lions to appear in the egg design rather than looking at the egg from the beach, but I imagine I could get better results if I continued to iterate on my prompt.

The FLUX models are also better at applying text than any other image models I've tried myself.

[# 7:14 pm](#) / [ai](#), [stable-diffusion](#), [generative-ai](#), [replicate](#), [text-to-image](#)

At first, I struggled to understand why anyone would want to write this way. My dialogue with ChatGPT was frustratingly meandering, as though I were excavating an essay instead of crafting one. But, when I thought about the psychological experience of writing, I began to see the value of the tool. ChatGPT was not generating professional prose all at once, but it was providing starting points: interesting research ideas to

explore; mediocre paragraphs that might, with sufficient editing, become usable. For all its inefficiencies, this indirect approach did feel easier than staring at a blank page; “talking” to the chatbot about the article was more fun than toiling in quiet isolation. In the long run, I wasn’t saving time: I still needed to look up facts and write sentences in my own voice. But my exchanges seemed to reduce the maximum mental effort demanded of me.

— [Cal Newport](#)

7:43 pm / [writing](#), [generative-ai](#), [chatgpt](#), [ai](#), [llms](#)

[Gemini 1.5 Flash-8B is now production ready](#) ([via](#)) Gemini 1.5 Flash-8B is "a smaller and faster variant of 1.5 Flash" - and is now released to production, at half the price of the 1.5 Flash model.

It's really, really cheap:

- \$0.0375 per 1 million input tokens on prompts <128K
- \$0.15 per 1 million output tokens on prompts <128K
- \$0.01 per 1 million input tokens on cached prompts <128K

Prices are doubled for prompts longer than 128K.

I believe images are still charged at a flat rate of 258 tokens, which I think means a single non-cached image with Flash should cost 0.00097 cents - a number so tiny I'm doubting if I got the calculation right.

OpenAI's cheapest model remains GPT-4o mini, at \$0.15/1M input - though that drops to half of that for reused prompt prefixes thanks to their new prompt caching feature (or by half if you use batches, though those can't be combined with OpenAI prompt caching. Gemini also offer half-off for batched requests).

Anthropic's cheapest model is still Claude 3 Haiku at \$0.25/M, though that drops to \$0.03/M for cached tokens (if you configure them correctly).

I've released [llm-gemini 0.2](#) with support for the new model:

```
llm install -U llm-gemini
llm keys set gemini
# Paste API key here
llm -m gemini-1.5-flash-8b-latest "say hi"
```

8:16 pm / [google](#), [ai](#), [openai](#), [generative-ai](#), [llms](#), [llm](#), [anthropic](#), [gemini](#), [vision-llms](#), [llm-pricing](#), [prompt-caching](#), [llm-release](#)

[Oct. 4, 2024](#)

[Hybrid full-text search and vector search with SQLite](#). As part of Alex’s work on his [sqlite-vec](#) SQLite extension - adding fast vector lookups to SQLite - he’s been investigating hybrid search, where search results from both vector similarity and traditional full-text search are combined together.

The most promising approach looks to be [Reciprocal Rank Fusion](#), which combines the top ranked items from both approaches. Here’s Alex’s SQL query:

```
-- the sqlite-vec KNN vector search results
with vec_matches as (
  select
    article_id,
    row_number() over (order by distance) as rank_number,
    distance
  from vec_articles
```

```

where
    headline_embedding match lembded(:query)
    and k = :k
),
-- the FTS5 search results
fts_matches as (
    select
        rowid,
        row_number() over (order by rank) as rank_number,
        rank as score
    from fts_articles
    where headline match :query
    limit :k
),
-- combine FTS5 + vector search results with RRF
final as (
    select
        articles.id,
        articles.headline,
        vec_matches.rank_number as vec_rank,
        fts_matches.rank_number as fts_rank,
        -- RRF algorithm
        (
            coalesce(1.0 / (:rrf_k + fts_matches.rank_number), 0.0) * :weight_fts +
            coalesce(1.0 / (:rrf_k + vec_matches.rank_number), 0.0) * :weight_vec
        ) as combined_rank,
        vec_matches.distance as vec_distance,
        fts_matches.score as fts_score
    from fts_matches
    full outer join vec_matches on vec_matches.article_id = fts_matches.rowid
    join articles on articles.rowid = coalesce(fts_matches.rowid, vec_matches.article_id)
    order by combined_rank desc
)
select * from final;

```

I've been puzzled in the past over how to best do that because the distance scores from vector similarity and the relevance scores from FTS are meaningless in comparison to each other. RRF doesn't even attempt to compare them - it uses them purely for `row_number()` ranking within each set and combines the results based on that.

4:22 pm / [full-text-search](#), [search](#), [sql](#), [sqlite](#), [alex-garcia](#), [vector-search](#), [embeddings](#), [rag](#)

[Database Remote-Copy Tool For SQLite \(draft\)](#) ([via](#)) Neat new SQLite utilities often show up in branches of the SQLite repository. Here's a new one from last month: `sqlite3-rsync`, providing tools for efficiently creating and updating copies of WAL-mode SQLite databases on either the same machine or across remote machines via SSH.

The way it works is neat, inspired by `rsync` (hence the tool's name):

The protocol is for the replica to send a cryptographic hash of each of its pages over to the origin side, then the origin sends back the complete content of any page for which the hash does not match.

SQLite's default page size is 4096 bytes and a hash is 20 bytes, so if nothing has changed then the client will transmit 0.5% of the database size in hashes and get nothing back in return.

The tool takes full advantage of [SQLite's WAL mode](#) - when you run it you'll get an exact snapshot of the database state as it existed at the moment the copy was initiated, even if the source database continues to apply changes.

I wrote up [a TIL on how to compile it](#) - short version:

```

cd /tmp
git clone https://github.com/sqlite/sqlite.git

```



```
cd sqlite
git checkout sqlite3-rsync
./configure
make sqlite3.c
cd tool
gcc -o sqlite3-rsync sqlite3-rsync.c ../sqlite3.c -DSQLITE_ENABLE_DBPAGE_VTAB
./sqlite3-rsync --help
```

Update: It turns out you can now just run `./configure && make sqlite_rsync` in the root checkout.

Something I've worried about in the past is that if I want to make a snapshot backup of a SQLite database I need enough additional free disk space to entirely duplicate the current database first (using the backup mechanism or `VACUUM INTO`). This tool fixes that - I don't need any extra disk space at all, since the pages that have been updated will be transmitted directly over the wire in 4096 byte chunks.

I tried feeding the [1800 lines of C](#) through OpenAI's o1-preview with the prompt "Explain the protocol over SSH part of this" and [got a pretty great high level explanation](#) - [markdown copy here](#).

8:57 pm / [c](#), [sqlite](#), [o1](#)

Oct. 5, 2024

[Wikidata is a Giant Crosswalk File](#). Drew Breunig shows how to take the 140GB Wikidata JSON export, use `sed 's/,,$//'` to convert it to newline-delimited JSON, then use DuckDB to run queries and extract external identifiers, including a query that pulls out 500MB of latitude and longitude points.

3:45 pm / [json](#), [wikipedia](#), [duckdb](#), [drew-breunig](#)

[marimo v0.9.0 with mo.ui.chat](#). The latest release of the Marimo Python reactive notebook project includes a neat new feature: you can now easily embed a custom chat interface directly inside of your notebook.

Marimo co-founder Myles Scolnick [posted this intriguing demo](#) on Twitter, demonstrating a chat interface to my [LLM library](#) "in only 3 lines of code":

```
import marimo as mo
import llm

model = llm.get_model()
conversation = model.conversation()
mo.ui.chat(lambda messages: conversation.prompt(messages[-1].content))
```

I tried that out today - here's the result:

```
1 import marimo as mo
2 import llm
```



Hi there

Hello! How can I assist you today?

Three jokes about pelicans

Sure! Here are three pelican jokes for you: 1. **Why do pelicans always carry a suitcase?** Because they have a lot of “baggage” to handle! 2. **What do you call a pelican that can sing?** A “tune-ican”! 3. **Why did the pelican break up with his girlfriend?** She said he always had his head in the clouds and never “winged” it! Hope these made you smile!

Type your message...



```
1 model = llm.get_model()
2 conversation = model.conversation()
3 mo.ui.chat(lambda messages:
  conversation.prompt(messages[-1].content))
```

[marimo.ui.chat\(\)](#) takes a function which is passed a list of Marimo chat messages (representing the current state of that widget) and returns a string - or other type of renderable object - to add as the next message in the chat. This makes it trivial to hook in any custom chat mechanism you like.

Marimo also ships their own [built-in chat handlers](#) for OpenAI, Anthropic and Google Gemini which you can use like this:

```
mo.ui.chat(  
    mo.ai.llm.anthropic(  
        "claude-3-5-sonnet-20240620",  
        system_message="You are a helpful assistant.",  
        api_key="sk-ant-...",  
    ),  
    show_configuration_controls=True  
)
```

10:59 pm / [python](#), [ai](#), [generative-ai](#), [llms](#), [llm](#), [marimo](#)

UV with GitHub Actions to run an RSS to README project. Jeff Triplett demonstrates a very neat pattern for using [uv](#) to run Python scripts with their dependencies inside of GitHub Actions. First, add `uv` to the workflow using the [setup-uv action](#):

```
- uses: astral-sh/setup-uv@v3  
  with:  
    enable-cache: true  
    cache-dependency-glob: "*.py"
```

This enables the caching feature, which stores `uv`'s own cache of downloads from PyPI between runs. The `cache-dependency-glob` key ensures that this cache will be invalidated if any `.py` file in the repository is updated.

Now you can run Python scripts using steps that look like this:

```
- run: uv run fetch-rss.py
```

If that Python script begins with some dependency definitions ([PEP 723](#)) they will be automatically installed by `uv run` on the first run and reused from the cache in the future. From the start of [fetch-rss.py](#):

```
# /// script  
# requires-python = ">=3.11"  
# dependencies = [  
#     "feedparser",  
#     "typer",  
# ]  
# ///
```

`uv` will download the required Python version and cache that as well.

11:39 pm / [python](#), [github-actions](#), [jeff-triplett](#), [uv](#)

Oct. 6, 2024

Students who use AI as a crutch don't learn anything. It prevents them from thinking. Instead, using AI as co-intelligence is important because it increases your capabilities and also keeps you in the loop. [...]

AI does so many things that we need to set guardrails on what we don't want to give up. It's a very weird, general-purpose technology, which means it will affect all kinds of things, and we'll have to adjust socially.

— [Ethan Mollick](#)

3:26 pm / [ethan-mollick](#), [ai](#)

SVG to JPG/PNG. The latest in my [ongoing series](#) of interactive HTML and JavaScript tools written almost entirely by LLMs. This one lets you paste in (or open-from-file, or drag-onto-page) some SVG and then use that to render a JPEG or PNG image of your desired width.

SVG to JPEG/PNG

[Browse...](#) No file selected.

[Load example image](#)

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg id="svg2" xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 900 900" version="1.1">
  <g id="g4" fill="none"
transform="matrix(1.7656463,0,0,1.7656463,324.90716,255
.00942)">
    <a id="a6" stroke-width="0.17200001" stroke="#000"
```

☒ JPEG ☐ PNG

Background Color: ☐ Transparent

Output Width:

[Convert SVG](#)



[Download Image](#) (47.38 KB)

Base64 Image Tag:

Copy image tag

```

```

The source code is [on GitHub](#). It runs on Deno and Deno Deploy, and recently added per-Emoji hit counters powered by the Deno KV store, implemented in [db.ts](#) using this pattern:

```
export function incrementCount(emoji: string) {
  const VIEW_KEY = [`favicon`, `${emoji}`];
  return db.atomic().sum(
    VIEW_KEY, 1n
  ).commit(); // Increment KV by 1
}
```

[# 6:46 am](#) / [favicons](#), [javascript](#), [svg](#), [deno](#)

[Datasette 0.65](#). [Python 3.13](#) was released today, which broke compatibility with the Datasette 0.x series due to an issue with an underlying dependency. [I've fixed that problem](#) by vendoring and fixing the dependency and the new 0.65 release works on Python 3.13 (but drops support for Python 3.8, which is [EOL](#) this month). Datasette 1.0a16 added support for Python 3.13 [last month](#).

[# 6:07 pm](#) / [projects](#), [python](#), [datasette](#)

[What's New in Ruby on Rails 8](#) ([via](#))

Rails 8 takes SQLite from a lightweight development tool to a reliable choice for production use, thanks to extensive work on the SQLite adapter and Ruby driver.

With the introduction of the solid adapters discussed above, SQLite now has the capability to power Action Cable, Rails.cache, and Active Job effectively, expanding its role beyond just prototyping or testing environments. [...]

- Transactions default to IMMEDIATE mode to improve concurrency.

Also included in Rails 8: [Kamal](#), a new automated deployment system by 37signals for self-hosting web applications on hardware or virtual servers:

Kamal basically is Capistrano for Containers, without the need to carefully prepare servers in advance. No need to ensure that the servers have just the right version of Ruby or other dependencies you need. That all lives in the Docker image now. You can boot a brand new Ubuntu (or whatever) server, add it to the list of servers in Kamal, and it'll be auto-provisioned with Docker, and run right away.

More from the [official blog post about the release](#):

At 37signals, we're building a growing suite of apps that use SQLite in production with [ONCE](#). There are now thousands of installations of both [Campfire](#) and [Writebook](#) running in the wild that all run SQLite. This has meant a lot of real-world pressure on ensuring that Rails (and Ruby) is working that wonderful file-based database as well as it can be. Through proper defaults like WAL and IMMEDIATE mode. Special thanks to Stephen Margheim for [a slew of such improvements](#) and Mike Dalessio for [solving a last-minute SQLite file corruption issue](#) in the Ruby driver.

[# 7:17 pm](#) / [37-signals](#), [rails](#), [ruby](#), [sqlite](#), [docker](#), [sqlite-busy](#)

[What's New In Python 3.13](#). It's Python 3.13 release day today. The big signature features are a [better REPL](#) with improved error messages, an option to [run Python without the GIL](#) and the beginnings of [the new JIT](#). Here are some of the smaller highlights I spotted while perusing the release notes.

iOS and Android are both now [Tier 3 supported platforms](#), thanks to the efforts of Russell Keith-Magee and the [Beeware](#) project. Tier 3 [means](#) "must have a reliable buildbot" but "failures on these platforms do not block a release". This is still a really big deal for Python as a mobile development platform.

There's a whole bunch of smaller stuff relevant to SQLite.

Python's [dbm module](#) has long provided a disk-backed key-value store against multiple different backends. 3.13 introduces a new backend based on SQLite, and makes it the default.

```
>>> import dbm
>>> db = dbm.open("/tmp/hi", "c")
>>> db["hi"] = 1
```

The "c" option means "Open database for reading and writing, creating it if it doesn't exist".

After running the above, /tmp/hi was a SQLite database containing the following data:

```
sqlite3 /tmp/hi .dump
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE Dict (
  key BLOB UNIQUE NOT NULL,
  value BLOB NOT NULL
);
INSERT INTO Dict VALUES(X'6869',X'31');
COMMIT;
```

The `dbm.open()` function can detect which type of storage is being referenced. I found the implementation for that in the [whichdb\(filename\)](#) function.

I was hopeful that this change would mean Python 3.13 deployments would be guaranteed to ship with a more recent SQLite... but it turns out 3.15.2 is [from November 2016](#) so still quite old:

SQLite 3.15.2 or newer is required to build the [sqlite3](#) extension module. (Contributed by Erlend Aasland in [gh-105875](#).)

The `conn.iterdump()` SQLite method now accepts an optional `filter=` keyword argument taking a LIKE pattern for the tables that you want to dump. I found [the implementation for that here](#).

And one last change which caught my eye because I could imagine having code that might need to be updated to reflect the new behaviour:

[pathlib.Path.glob\(\)](#) and [rglob\(\)](#) now return both files and directories if a pattern that ends with "*" is given, rather than directories only. Add a trailing slash to keep the previous behavior and only match directories.

With the release of Python 3.13, Python 3.8 is [officially end-of-life](#). Łukasz Langa:

If you're still a user of Python 3.8, I don't blame you, it's a lovely version. But it's time to move on to newer, greater things. Whether it's typing generics in built-in collections, pattern matching, `except*`, low-impact monitoring, or a new pink REPL, I'm sure you'll find your favorite new feature in one of the versions we still support. So upgrade today!

7:36 pm / [android](#), [mobile](#), [python](#), [russell-keith-magee](#), [sqlite](#), [ios](#), [lukasz-langa](#), [beeware](#)

[Thoughts on the Treasurer Role at Tech NonProfits](#). Will Vincent, Django Software Foundation treasurer from 2020-2022, explains what's involved in the non-profit role with the highest level of responsibility and trust.

[10:41 pm](#) / [django](#), [dsf](#)

[Oct. 8, 2024](#)

[Django Commons](#). Django Commons is a really promising initiative started by Tim Schilling, aimed at the problem of keeping key Django community projects responsibly maintained on a long-term basis.

Django Commons is an organization dedicated to supporting the community's efforts to maintain packages. It seeks to improve the maintenance experience for all contributors; reducing the barrier to entry for new contributors and reducing overhead for existing maintainers.

I've stated recently that I'd love to see the Django Software Foundation take on this role - adopting projects and ensuring they are maintained long-term. Django Commons looks like it solves that exact problem, assuring the future of key projects beyond their initial creators.

So far the Commons has taken on responsibility for [django-fsm-2](#), [django-tasks-scheduler](#) and, as-of this week, [diango-typer](#).

Here's Tim [introducing the project](#) back in May. Thoughtful governance has been baked in from the start:

Having multiple administrators makes the role more sustainable, lessens the impact of a person stepping away, and shortens response time for administrator requests. It's important to me that the organization starts with multiple administrators so that collaboration and documentation are at the forefront of all decisions.

[3:27 am](#) / [django](#), [open-source](#)

[Anthropic: Message Batches \(beta\)](#) ([via](#)) Anthropic now have a batch mode, allowing you to send prompts to Claude in batches which will be processed within 24 hours (though probably much faster than that) and come at a 50% price discount.

This matches the batch models offered [by OpenAI](#) and [by Google Gemini](#), both of which also provide a 50% discount.

Update 15th October 2024: Alex Albert [confirms](#) that Anthropic batching and prompt caching can be combined:

Don't know if folks have realized yet that you can get close to a 95% discount on Claude 3.5 Sonnet tokens when you combine prompt caching with the new Batches API

[6:18 pm](#) / [ai](#), [openai](#), [generative-ai](#), [llms](#), [anthropic](#), [claude](#), [gemini](#), [alex-albert](#), [prompt-caching](#)

[If we had \\$1,000,000....](#) Jacob Kaplan-Moss gave my favorite talk at DjangoCon this year, imagining what the Django Software Foundation could do if it quadrupled its annual income to \$1 million and laying out a realistic path for getting there. Jacob suggests leaning more into large donors than increasing our small donor base:

It's far easier for me to picture convincing eight or ten or fifteen large companies to make large donations than it is to picture increasing our small donor base tenfold. So I think a major donor strategy is probably the most realistic one for us.

So when I talk about major donors, who am I talking about? I'm talking about four major categories: large corporations, high net worth individuals (very wealthy people), grants from governments (e.g. the Sovereign Tech Fund run out of Germany), and private foundations (e.g. the Chan Zuckerberg Initiative, who's given grants to the PSF in the past).

Also included: a TIL on [Turning a conference talk into an annotated presentation](#). Jacob used [my annotated presentation tool](#) to OCR text from images of keynote slides, extracted a Whisper transcript from the YouTube livestream audio and then cleaned that up a little with [LLM](#) and Claude 3.5 Sonnet ("Split the content of this transcript up into paragraphs with logical breaks. Add newlines between each paragraph.") before editing and re-writing it all into the final post.

7:59 pm / [django](#), [jacob-kaplan-moss](#), [whisper](#), [llm](#), [claude-3-5-sonnet](#), [dsf](#)

[Oct. 9, 2024](#)

[openai/openai-realtime-console](#). I got this OpenAI demo repository working today - it's an *extremely* easy way to get started playing around with the new Realtime voice API they announced [at DevDay](#) last week:

```
cd /tmp
git clone https://github.com/openai/openai-realtime-console
cd openai-realtime-console
npm i
npm start
```

That starts a `localhost:3000` server running the demo React application. It asks for an API key, you paste one in and you can start talking to the web page.

The demo handles voice input, voice output and basic tool support - it has a tool that can show you the weather anywhere in the world, including panning a map to that location. I tried [adding a `show_map\(\)` tool](#) so I could pan to a location just by saying "Show me a map of the capital of Morocco" - all it took was editing the `src/pages/ConsolePage.tsx` file and hitting save, then refreshing the page in my browser to pick up the new function.

Be warned, it can be quite expensive to play around with. I was testing the application intermittently for only about 15 minutes and racked up \$3.87 in API charges.

12:38 am / [javascript](#), [nodejs](#), [websockets](#), [ai](#), [react](#), [openai](#), [generative-ai](#), [llms](#)

[otterwiki](#) ([via](#)) It's been a while since I've seen a new-ish Wiki implementation, and this one by Ralph Thesen is really nice. It's written in Python (Flask + SQLAlchemy + [mistune](#) for Markdown + [GitPython](#)) and keeps all of the actual wiki content as Markdown files in a local Git repository.

The [installation instructions](#) are a little in-depth as they assume a production installation with Docker or systemd - I figured out [this recipe](#) for trying it locally using `uv`:

```
git clone https://github.com/redimp/otterwiki.git
cd otterwiki

mkdir -p app-data/repository
git init app-data/repository

echo "REPOSITORY='${PWD}/app-data/repository'" >> settings.cfg
echo "SQLALCHEMY_DATABASE_URI='sqlite:///${PWD}/app-data/db.sqlite'" >> settings.cfg
echo "SECRET_KEY='${echo $RANDOM | md5sum | head -c 16}'" >> settings.cfg

export OTTERWIKI_SETTINGS=$PWD/settings.cfg
uv run --with gunicorn gunicorn --bind 127.0.0.1:8080 otterwiki.server:app
```

3:22 pm / [flask](#), [git](#), [python](#), [sqlalchemy](#), [sqlite](#), [markdown](#), [wikis](#), [uv](#)

[The Fair Source Definition](#) ([via](#)) Fair Source ([fair.io](#)) is the new-ish initiative from Chad Whitacre and Sentry aimed at providing an alternative licensing philosophy that provides additional protection for the business models of companies that

release their code.

I like that they're establishing a new brand for this and making it clear that it's a separate concept from Open Source. Here's their definition:

Fair Source is an alternative to closed source, allowing you to safely share access to your core products. Fair Source Software (FSS):

- 1. is publicly available to read;
- 2. allows use, modification, and redistribution with minimal restrictions to protect the producer’s business model; and
- 3. undergoes delayed Open Source publication (DOSP).

They link to the [Delayed Open Source Publication](#) research paper published by [OSI in January](#). (I was frustrated that this is only available as a PDF, so I [converted it to Markdown](#) using Gemini 1.5 Pro so I could read it on my phone.)

The most interesting background I could find on Fair Source was [this GitHub issues thread](#), started in May, where Chad and other contributors fleshed out the initial launch plan over the course of several months.

6:17 pm / [licenses](#), [open-source](#), [pdf](#), [sentry](#), [chad-whitacre](#)

Free Threaded Python With Asyncio. Jamie Chang expanded [my free-threaded Python experiment](#) from a few months ago to explore the interaction between Python's `asyncio` and the new GIL-free build of Python 3.13.

The results look really promising. Jamie says:

Generally when it comes to Asyncio, the discussion around it is always about the performance or lack there of. Whilst performance is certain important, the ability to reason about concurrency is the biggest benefit. [...]

Depending on your familiarity with AsyncIO, it might actually be the simplest way to start a thread.

This code for running a Python function in a thread really is very pleasant to look at:

```
result = await asyncio.to_thread(some_function, *args, **kwargs)
```

Jamie also demonstrates [asyncio.TaskGroup](#), which makes it easy to execute a whole bunch of threads and wait for them all to finish:

```
async with TaskGroup() as tg:
    for _ in range(args.tasks):
        tg.create_task(to_thread(cpu_bound_task, args.size))
```

8:38 pm / [async](#), [gil](#), [python](#)

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27

