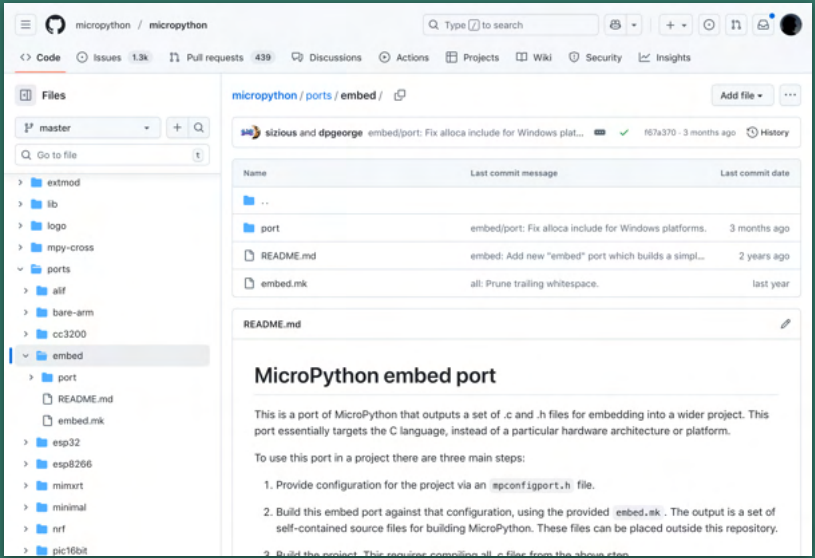
Christian Walther

Embedding MicroPython on Playdate

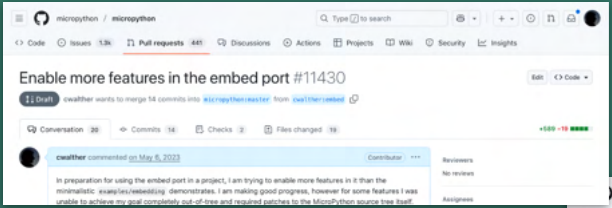
Next: Max Wittig

Embedding MicroPython on Playdate

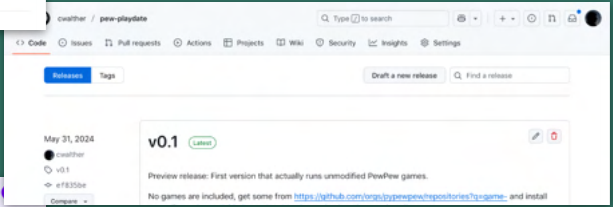
Christian Walther • cwalther@gmx.ch • mstdn.social/@isziaui



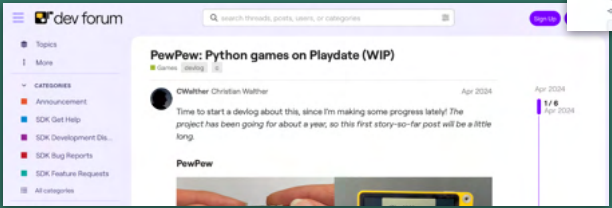
Christian Walther • cwalther@gmx.ch • mstdn.social/@isziaui



https://github.com/micropython/micropython/pull/11430/



https://github.com/cwalther/pew-playdate/releases



https://devforum.play.date/t/17379

Christian Walther • cwalther@gmx.ch • mstdn.social/@isziaui

Max Wittig

AI for developers by developers powered by Python

Next: Pavel Sulimov

AI for developers by developers powered by Python

Based on Open Source for all internal developers

SIEMENS

from contact import * # noqa

Max Wittig based in Zürich, Switzerland

- Senior **DevOps Engineer** at Siemens
- Working in the Developer Enablement Team
- Operating and developing on-premise GitLab (code.siemens.com)
  - with 80_000 users, 300_000 projects globally
  - Lots of **Python tooling to manage Gitlab** (user lifecycle, statistics, internal accounting etc.)
- Internal APIs based on Django Restframework
- **python-gitlab maintainer**
- Python Enthusiast

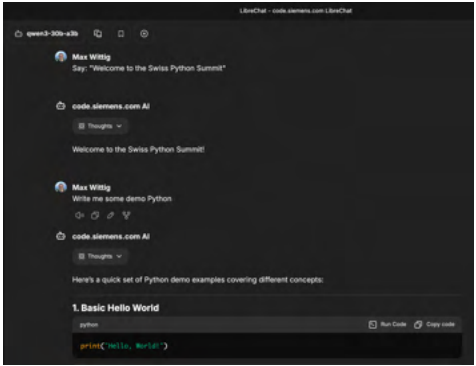
SIEMENS

What can (Python) developers do with AI?

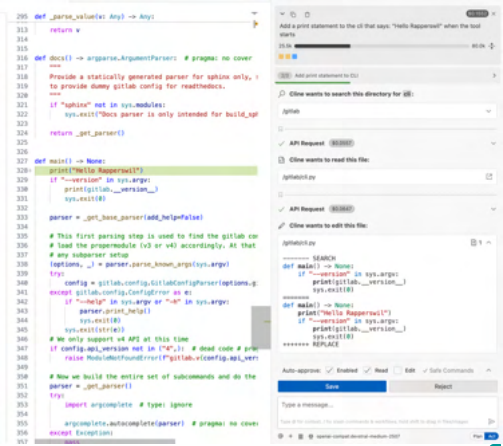
SIEMENS

AI (LLM) uses cases for developers (1/2)

Chat mode (e.g. using LibreChat)



Agentic Mode using Cline inside VS Code



SIEMENS

AI (LLM) uses cases for developers (2/2)

Code completions inside VS Code

```
172         """
173         data = {"scope": scope, "search": search}
174         path = f"/groups/{self.encoded_id}/search"
175         if kwargs.get("all", False):        You, 1 second
176             return self.manager.gitlab.http_list(path, query_
177
```

Embeddings, rerank etc using a REST API client

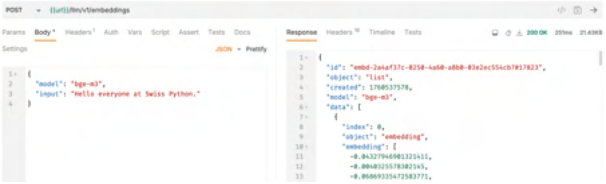
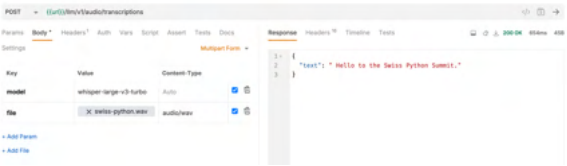


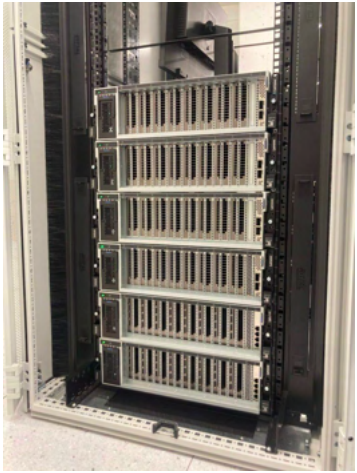
Image to text using Chatbox

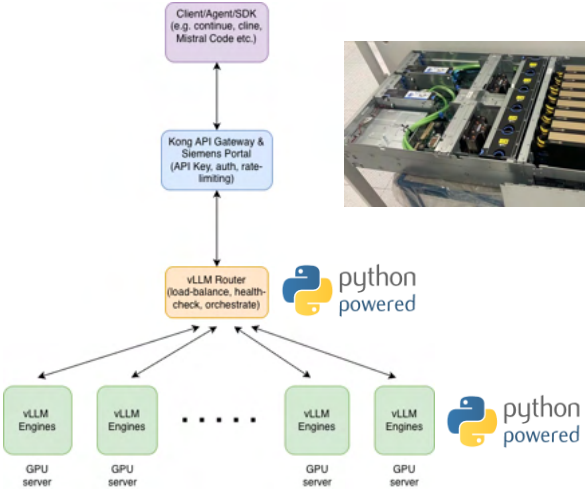
Audio to text using a REST API client



SIEMENS

How are we providing this?





SIEMENS

What does Python have to do with this? - vLLM

https://github.com/vllm-project/vllm

- Written in **Python**
- Provides **OpenAI-compatible API server** that hosts the models for inference
- Continuous batching of incoming requests
- Supports GPUs from multiple manufacturers
- We're running different **Open Weight models** on our hardware e.g.:
  - gpt-oss-120b
  - qwen3-30b-a3b-thinking-2507
  - qwen3-30b-a3b-instruct-2507
  - pixtral-12b-2409
  - whisper-large-v3-turbo
- We made contributions to allow usage counting for internal usage stats

SIEMENS

What does Python have to do with this? – vllm production-stack (vllm-router)

vLLM Production Stack

Powered by

LMCache Lab + vLLM

https://github.com/vllm-project/production-stack

- Also written in **Python**
- HTTP based, stateless
- Shards requests across GPU nodes
- Performs load-balancing
- Multiple different routing algorithms

- We made **multiple contributions**. E.g.
  - Model alias support
  - Dynamic human readable config
  - Real backend health checks
  - Post Request background callbacks

Thank you for listening!

SIEMENS

Contact

Email (private): max.wittig.ch@gmail.com
Email (Siemens): max.wittig@siemens.com

GitHub: max-wittig
GitLab: max-wittig
Linkedin: https://www.linkedin.com/in/max-wittig

SIEMENS

Questions & answers

SIEMENS

Links

Siemens Blog for details: https://blog.siemens.com/2025/10/our-sovereign-ai-journey-building-a-self-contained-sustainable-and-cost-effective-llm-platform/

LibreChat: https://github.com/danny-avila/LibreChat
vLLM: https://github.com/vllm-project/vllm
production-stack (vllm-router): https://github.com/vllm-project/production-stack/
ChatBox: https://github.com/ChatBoxAI/ChatBox
Cline: https://cline.bot/
Bruno: https://github.com/usebruno/bruno/
python-gitlab: https://github.com/python-gitlab/python-gitlab

SIEMENS

Sources

Siemens Blog: https://blog.siemens.com/2025/10/our-sovereign-ai-journey-building-a-self-contained-sustainable-and-cost-effective-llm-platform/

vLLM logo: https://docs.vllm.ai/en/latest/assets/logos/vllm-logo-text-dark.png
production-stack logo: https://docs.vllm.ai/projects/production-stack/en/latest/_images/prodstack.png
Python-powered logo: https://www.python.org/static/community_logos/python-powered-w.svg
Swiss Python summit logo: https://www.python-summit.ch/logo.svg

SIEMENS

Pavel Sulimov

From Events to Actions: Real-time CRM with Python

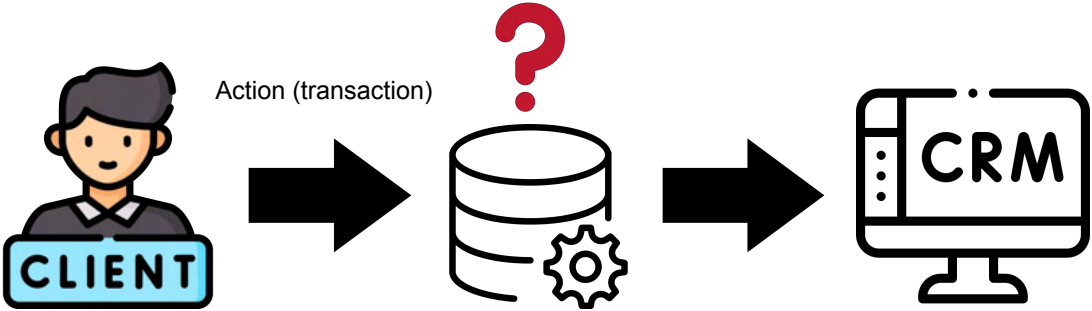
Next: Annabelle Wiegart

From Events to Actions: Real-time CRM with Python

16.10.2025

Pavel Sulimov, Sergey Eremeykin

Example from experience: Just Data. No Science



"Client/player act in seconds, but our campaigns take hours"

Why not to stream exactly in-memory?

School of Engineering

```
# What happens when Spark restarts?
streaming_query = (spark
    .readStream("kafka")
    .groupBy("customer_id")
    .agg(sum("bet_amount").alias("total_bets"))
    .writeStream
    .outputMode("complete")
    .format("memory")# ← ONLY IN MEMORY.queryName("customer_metrics")
    .start())

# After restart: ALL DATA GONE! 💥
spark.sql("SELECT * FROM customer_metrics").show()
# Empty! No customer history, no metrics
```

Why not to stream exactly into target database?

School of Engineering

```python
# Writing directly to database
def process_event(event):
    cursor.execute("""
        UPDATE customer_metrics
        SET total_bets = total_bets + ?,
            last_activity = NOW()
        WHERE customer_id = ?
    """, (event.amount, event.customer_id))

# Issues:
# 💥 Expensive at scale if each event = 1 database write
# 💥 Cannot perform fast rule-based aggregations if batching
```

Events→Spark (?)→Delta Lake (?)→CRM Actions

School of Engineering

```python
# Real-time customer metrics with Delta Lake
customer_metrics = (spark
  .readStream
  .format("kafka")
  .option("subscribe", "customer_events")
  .load()
  .selectExpr("CAST(value AS STRING)")
  .select(
    get_json_object("value", "$.customer_id").alias("customer_id"),
    get_json_object("value", "$.event_type").alias("event_type"),
    get_json_object("value", "$.amount").cast("double").alias("amount"),
    get_json_object("value", "$.timestamp").cast("timestamp").alias("timestamp")
  )
  .withWatermark("timestamp", "5 minutes")
  .groupBy("customer_id", window("timestamp", "1 hour"))
  .agg(
    sum("amount").alias("hourly_bet_amount"),
    count("event_type").alias("event_count"),
    approx_count_distinct("session_id").alias("session_count")
  ))

# Write to Delta Lake with checkpointing(customer_metrics
  .writeStream
  .format("delta")
  .outputMode("update")
  .option("checkpointLocation", "/checkpoints/crm")
  .start("/data/customer_metrics"))
```

More about Data Lakes:

https://docs.databricks.com/aws/en/delta/

- extends Parquet data files with a file-based transaction log
- optimized for Apache Spark, allowing to easily use a single copy of data for both batch and streaming operations and providing incremental processing at scale

You can even maybe use it together with CountMinSketch from the first talk from today (but I didn't try):

https://www.geeksforgeeks.org/dsa/count-min-sketch-in-python/

Real-time customer metrics

customer_id	window	hourly_bet_amount	session_count	
cust_001	[14:00, 15:00]	1250.50	3	
cust_007	[14:30, 15:30]	2100.75	2	⭐ VIP!
cust_042	[14:00, 15:00]	480.25	1	

VIP ALERT: cust_007 just became VIP with $2,100 in bets!
Send immediate bonus offer...

```python
# The best next step
def get_started():
    return "pip install pyspark delta-spark"
```

More stories with data and code are coming soon:

sulimov@data-pulse.site
eremeykin@data-pulse.site

Annabelle Wiegart

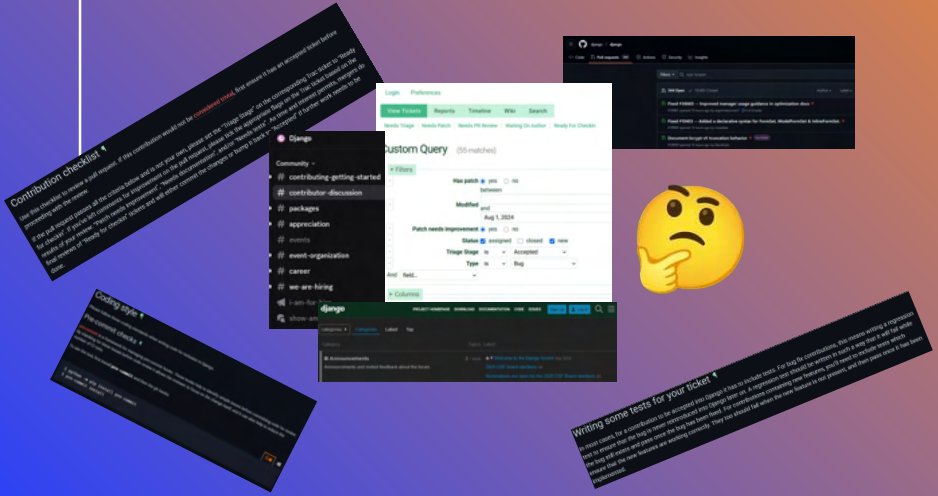
Venturing into Djangonaut Space

Next: David Halter

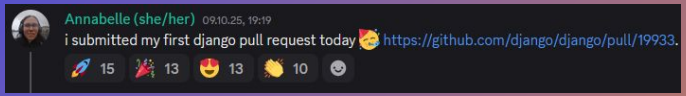
VENTURING INTO DJANGONAUT SPACE

HOW CAN I START CONTRIBUTING TO OPEN SOURCE?

CONTRIBUTING TO DJANGO...



...WITH DJANGONAUT SPACE

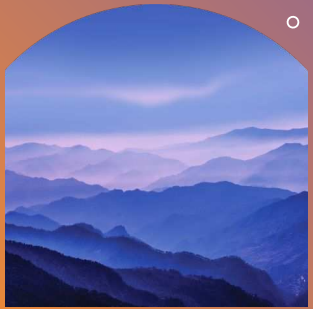


STRENGTHS OF DJANGONAUT SPACE

- Diversity in action
- Communication spaces are welcoming and supportive
- Regular check-ins with mentors
- Every small win is celebrated

THANK YOU

Annabelle Wiegart



Check out Djangonaut Space

David Halter

The new Type Checkers

Next: Stefan Schindler

Jedi Autocompletion

The New Type Checkers

Replace Mypy and PyRight

- Ty by Astral
- PyreFly by Meta
- Zuban by Me

Ty

- Most hyped, because of uv
- Lacks fundamental typing features
- Lacks IDE tools: Completions, Goto, Renames, etc.

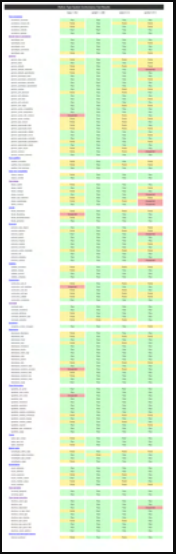
Pyrefly

- Mostly complete
- Lacks recursive type aliases
- Suports IDE tools: Completions, Goto, Renames, etc.

- Cutest Logo
- Mostly feature complete
- Suports IDE tools: Completions, Goto, Renames, etc.

Python Typing Conformance Tests

- Ensures type checkers work the same
- Ty is not in there yet, because they would look really bad



Performance

- All three are really fast
- Ty is fastest currently (threading)
- Zuban uses 2x less memory/CPU
- Depending on Notebook Zuban might be the fastest

Conclusion

- Choose any of these, but wait with Ty for a year

Questions?

- zubanls.com
- info@zubanls.com
- Twitter/GitHub: @ZubanLS

Stefan Schindler

Python like ease for numbers parsing

Next: Timon Erhart

parse_int

Numbers as easy as […]

by Stefan Schindler schindler@estada.ch
Blog: Estada.ch

Hosted by Python Summit

Who am I?

- Stefan Schindler @dns2utf8
- Looking for a job or paid projects
- OSS maintainer of crates like threadpool, parse_int, sudo, and others
- Opinions are my own
- Organiser of Rust Zürisee 🇨🇭 and RustFest 2024

Parsing hex numbers

```
decimal_value = int('0x1A', 16)
print(hex(decimal_value))
```

What is the output?

1A or **1a** or **0x1a** or **0x1A**

Parsing and printing but bigger

What if we make it more readable:

```
decimal_value = int('0x1A_00_00', 16)
print(hex(decimal_value))
```

Does it still print **0x1a0000** or **0x1a0_000** or **0x1a_00_00**?

parse_int

- why?
- A simple interface at runtime (via PyO3)
- Use one input parsing function for all your number needs

Installation

https://crates.io/crates/parse_int

Install

Run the following Cargo command in your project directory:

```
cargo add parse_int
```

Or add the following line to your Cargo.toml:

```
parse_int = "0.9.0"
```

Example with and without turbo-fish

```rust
use std::error::Error;
// import the function for multiple uses
use parse_int::parse;

fn main() -> Result<(), Box<dyn Error>> {
    // Detect the type automatically
    let d = parse_int::parse("42")?;
    assert_eq!(42_usize, d);

    // you can use underscores for more readable inputs,
    // just like in rust

    assert_eq!(1_111_638_594,
        parse::<isize>("0x42_42__42_42")?);
}
```

Handling floats

```
let pi = parse_int::parse("3.141_592_653_589_793")
    .expect("floats have a different error type");
assert_eq!(std::f64::consts::PI, pi);

assert_eq!(12.34_f32, parse("1_2.3_4").expect("static"));
assert_eq!(12.34_f64, parse("1_2.3_4").expect("static"));
```

Hex numbers

```
assert_eq!(128, parse::<isize>("0x80")?);
```

And negative hex

```
assert_eq!(-128, parse::<isize>("-0x80")?);
```

Integrate it with clap

lets introduce a subtile bug

```rust
use clap::Parser;

#[derive(Parser, Debug)]
#[command(name = "parse-int-with-clap", version, author)]
struct Args {
    /// left hand of addition
    #[arg(short = 'a', value_parser = parse_hex_u8)]
    a: u8,
    /// right hand of addition
    #[arg(short = 'b', value_parser = parse_hex_16)]
    b: i16,
}
```

Hand written parser

```
fn parse_hex_u8(s: &str) -> Result<u8, String> {
    let s = s.trim_start_matches("0x");
    u8::from_str_radix(s, 16).map_err(|e| e.to_string())
}
fn parse_hex_16(s: &str) -> Result<i16, String> {
    let s = s.trim_start_matches("0x");
    i16::from_str_radix(s, 16).map_err(|e| e.to_string())
}

fn main() {
    let args = Args::parse();    println!("{:?}", args);
    let Args { a, b } = args;

    println!("{} + {} = {}", a, b, a as i16 + b);
    println!("0x{:X} + 0x{:X} = 0x{:X}", a, b, a as i16 + b);
```

What actually happens

- Input a decimal like `10`
- It be translated to `0x10`
  - as if the user had specified `0xA`.

Fix it with parse_int

```
#[derive(Parser, Debug)]
#[command(name = "parse-int-with-clap", version, author)]
struct Args {
  /// left hand of addition
  #[arg(short='a', value_parser = parse_int::parse::<u8>)]
  a: u8,
  /// right hand of addition
  #[arg(short='b', value_parser = parse_int::parse::<i16>)]
  b: i16,
}
```

Parsing ranges

- experimental API
- Hidden behind a feature flag

a new Range type

```
pub enum Range<T: Num> {
    /// start..stop
    RangeExclusive { start: T, end: T, },
    /// start..=stop
    RangeInclusive { start: T, end: T, },

    RangeFrom { start: T  },
    RangeTo   { end: T    },

    RangeFull,
}
```

Full range

```
assert_eq!(Ok(RangeFull.into()), parse_range::<u8>("..."));

/// not the same type
assert_eq!(Ok(RangeFull.into()), parse_range::<i8>("..."));
```

Range from

```
let f32_r = parse_range::<f32>("-42.23..").expect("weird float error type");

assert_eq!(None, f32_r.as_full_range());
assert_eq!(None, f32_r.as_range_exclusive());
assert_eq!(None, f32_r.as_range_inclusive());
assert_eq!(None, f32_r.as_range_to());

assert_eq!(Some((-42.23..).into()), f32_r.as_range_from());
```

Formatting Decimal & Hex

```
assert_eq!("42_000", format_pretty_dec(42_000));

assert_eq!("-42_000", format_pretty_dec(-42_000));
```

```
assert_eq!("0xff_ff", format_pretty_hex(0xff_FF));
assert_eq!("0x4_20_00", format_pretty_hex(0x42_000));

assert_eq!("-0x42", format_pretty_hex(-0x42_i8));
assert_eq!("-0x42", format_pretty_hex(-0x42_i16));
```

Formatting floats

```
assert_eq!("42_230_123", format_pretty_dec(42_230_123));

assert_eq!("42.0", format_pretty_dec(42.0_f32));
```

parse_int

Numbers as easy as an Eclaire

by Stefan Schindler schindler@estada.ch
Blog: Estada.ch

Hosted by Python Summit

Timon Erhart

uv – An extremely fast Python package and project manager

Next: Florian Bruhin

uv

An extremely fast Python package and project manager

Timon Erhart
OST / IFS

What is it?

Highlights

- 🚀 A single tool to replace `pip`, `pip-tools`, `pipx`, `poetry`, `pyenv`, `twine`, `virtualenv`, and more.
- ⚡ 10-100x faster than `pip`.
- 🗂️ Provides comprehensive project management, with a universal lockfile.
- ❇️ Runs scripts, with support for inline dependency metadata.
- 🐍 Installs and manages Python versions.
- 🛠️ Runs and installs tools published as Python packages.
- 🔩 Includes a pip-compatible interface for a performance boost with a familiar CLI.
- 🏢 Supports Cargo-style workspaces for scalable projects.
- 💾 Disk-space efficient, with a global cache for dependency deduplication.
- ⏬ Installable without Rust or Python via `curl` or `pip`.
- 🖥️ Supports macOS, Linux, and Windows.

Why do I need it? - Case 1: Package manager

Projects with a requirements.txt

- Drop-in replacement for pip
- Can do everything pip does
  - ...but faster!
  - ...but more!
- Risk free
  - Try it out in an already existing project
  - Just write uv in front the pip command

```
[uv] pip install black
[uv] pip freeze

# but more! (pip doesn't)

uv pip tree [--depth 1]
uv pip sync requirements.txt
uv venv --python 3.11



# (Btw, did you know?)
python -m venv --system-site-packages .venv
```

Why do I need it? - Case 2: Project Manager

Projects with a pyproject.toml

- Replacement for Poetry
  - Similar interface
  - Way faster
  - Less errors (in my experience)

```
uv init hello-world
uv add black
uv run main.py
uv sync
uv publish
```

- Interlude: Why using pyproject.toml?
  - Clean dependency management
  - Version pinning (lock file)
  - Build backends and package distribution
  - Dev-tool settings, scripts etc…

More info: realpython.com/python-pyproject-toml

pyproject.toml

```toml
[project]
name = "hello-world"
version = "1.0.0"
dependencies = ["black>=25.1.0"]

[project.optional-dependencies]
+dev = ["black>=24.1.0", "isort>=5.13.0"]

[tool.black]
line-length = 88

[build-system]
requires = ["setuptools"]
```

Why do I need it? - Other cases ..

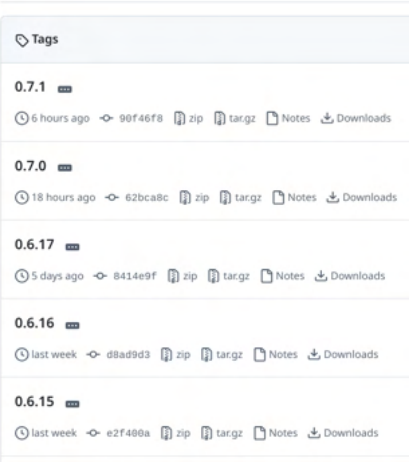
- Replace pipx / run tools
- Create project structures
- Handle virtual environments and install python version
- Execute standalone scripts

```
uv tool install black
uvx black / uv tool run black

uv init --app helloworld

uv venv --python 3.11
uv python list

uv add --script hello.py 'requests'
uv run hello.py
```

Read more: docs.astral.sh/uv/getting-started/features/

Is it stable?

- Pretty new
  - First release beginning of 2024
  - Very fast release cycle (almost every week)
- But
  - Fast growing community
  - Already 52k stars (Poetry has 33k)
  - From the creators of Ruff (Astral)
    - Ruff is a faster replacement for flake8 and black
  - Already used by (known) open source project
  - Good documentation
- IMHO: It will stay



Florian Bruhin

fstri.ng & pyte.st

fstring.help & pyte.st

Or: How I bought yet another (and yet another!) domain

Florian Bruhin

BRUHIN SOFTWARE

Swiss Python Summit 2025
October 16th

PyFormat

Using `%` and `.format()` for great good!

Contribute on GitHub

Python has had awesome string formatters for many years but the documentation on them is far too theoretic and technical. With this site we try to show you the most common use-cases covered by the old and new style string formatting API with practical examples.

All examples on this page work out of the box with with Python 2.7, 3.2, 3.3, 3.4, and 3.5 without requiring any additional libraries.

Further details about these two formatting methods can be found in the official Python documentation:

- old style
- new style

If you want to contribute more examples, feel free to create a pull-request on Github!

Basic formatting

Simple positional formatting is probably the most common use-case. Use it if the order of your arguments is not likely to change and you only have very few elements you want to concatenate.

Since the elements are not represented by something as descriptive as a name this simple style should only be used to format a relatively small number of elements.

Old

```
'%s %s' % ('one', 'two')
```

New

```
'{} {}'.format('one', 'two')
```

Output

```
one two
```

PEP 0498: f-Strings #24

Open **hut8** opened this issue on Oct 4, 2015 · 11 comments

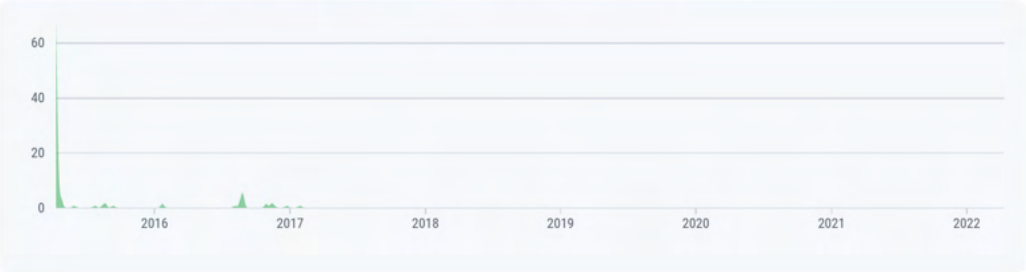
PEP 498 -- Literal String Interpolation #50

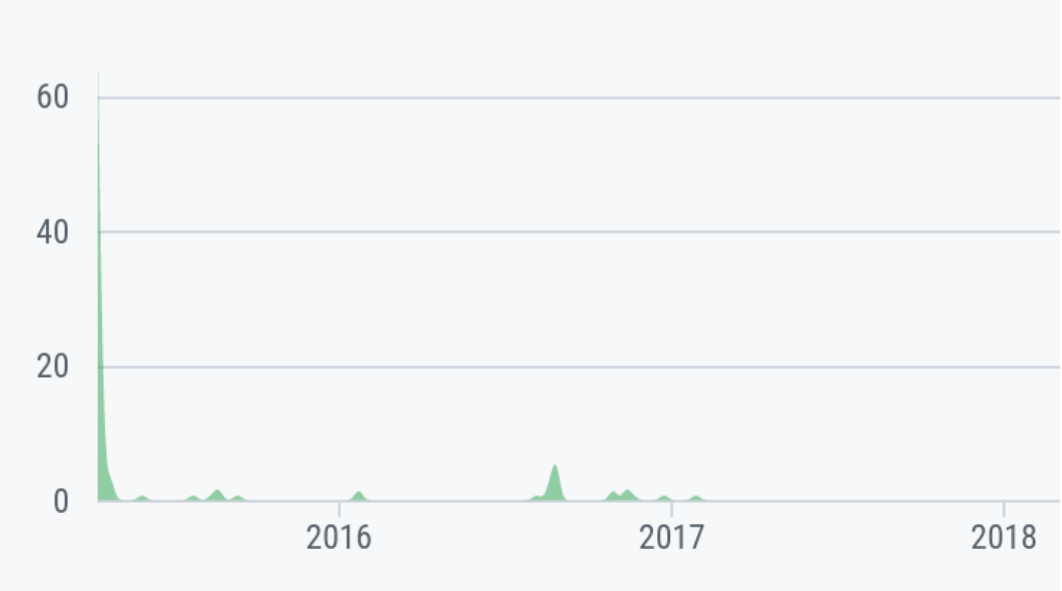
Open **danielniccoli** opened this issue on Dec 13, 2017 · 7 comments

Incorporating f-Strings (PEP 498) #56

Open **udincer** opened this issue on Apr 25, 2019 · 1 comment

Contributions to master, excluding merge commits and bot accounts




60
40
20
0
2016
2017
2018

master

Commits on Feb 4, 2017

Mention v2 in README

ulope committed on Feb 4, 2017

Commits on Dec 27, 2016

Slightly reword uneven center align padding note

v2

Commits on Jan 16, 2018

Remove Python 3.3 from testing

ulope committed on Jan 16, 2018 ✕

Fix circleci config

ulope committed on Jan 16, 2018 ✕

$\approx$ 2019: fstring.dev?

At PyConDE 2022:
OMG let's do it!
Lightning talks!!!1!

fstring.dev

DEV

Domain not available Whois

Domain Name: fstring.dev

...

Updated Date: 2022-11-01T17:46:43Z

Creation Date: **2019-09-17**T17:46:43Z

...

Registrant Name: REDACTED FOR PRIVACY

Registrant Organization: **Contact Privacy Inc. Customer 7151571251**

Unable to load page

Error while opening http://fstring.dev/
ERR_NAME_NOT_RESOLVED

Try again

fstri.ng

Nigeria

99.00 €

Verlängerung: 99.00 €

fstring.cat

CAT

PROMO 5.00 €

~~**13.00 €**~~

Verlängerung: 25.00 €

Restrictions [edit]

The .cat domain is not territorial, but applies to the whole Catalan-speaking community, whether or not a site is based in Catalonia. In order to be granted a .cat domain, one needs to belong to the Catalan linguistic and cultural community on the Internet. A person, organization or company is considered to belong if they either:[5]

[A] 1: NYAN.CAT!

Pick a language!

NON-STOP NYAN CAT

CREDITS STATS

CHECK OUT NYAN CAT ON FACEBOOK!

https://www.nyan.cat/index.php?cat=original[top][<][1/1]

fstring.help

HELP

32.80 €

Verlängerung: 32.80 €

Python f-strings

All currently supported Python versions (3.6+) support string-formatting via f-strings. While PEP 498 (Literal String Interpolation) as well as the Python documention (tutorial, syntax reference) have some information on their usage, I was missing a reference which is terse, but still verbose enough to explain the syntax.

Thus, fstring.help was born, made with ❤️ and jupyter (for now, as a quick hack at PyConDE 2022).

Created by BRUHIN SOFTWARE
Consulting, coaching and development for pytest, Qt and Python.

Some content is copied verbatim from pyformat.info (Copyright 2015 Ulrich Petri, Horst Gutmann). Thanks!

Repository on Github, contributions welcome! If you prefer an interactive version, launch binder.

Basic formatting

f-strings are strings with an `f` in front of them: `f"..."`. Inside the f-string, curly braces can be used to format values into it:

In [1]:
```
one = 1
two = 2
```

In [2]:
```
f"{one} {two}"
```

Out[2]:
```
'1 2'
```

Arbitrary code

You can put any Python code into f-strings:

Python f-string cheat sheets

See fstring.help for more examples and for a more detailed discussion of this syntax see this string formatting article.

All numbers

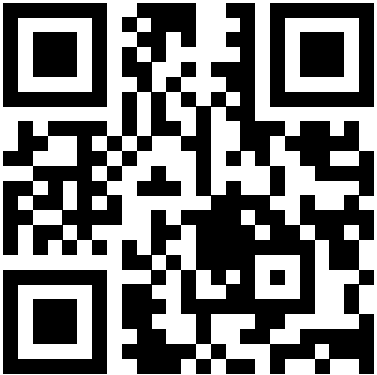
The below examples assume the following variables:

```
>>> number = 4125.6
>>> percent = 0.3738
```

Example Output	Replacement Field	Fill	Width	Grouping	Precision	Type
`'4125.60'`	`{number:.2f}`				`.2`	`f`
`'4,125.60'`	`{number:,.2f}`			`,`	`.2`	`f`
`'04125.60'`	`{number:08.2f}`	`0`	`8`		`.2`	`f`
`' 4125.60'`	`{number: 8.2f}`		`8`		`.2`	`f`
`'37%'`	`{percent:.0%}`				`.0`	`%`

These format specifications only work on all numbers (both `int` and `float`).

Short links for pytest docs: https://**pyte.st**



- `pyte.st/fixtures`
- `pyte.st/raises`
- `pyte.st/parametrize`
- ...and lots more!

https://**fstring.help**

https://github.com/
The-Compiler/fstring.help

https://**bruhin.software**
florian@bruhin.software