

Python Type System & Tooling Survey

2026 Results

1,794
responses

20
questions

Aug-Sep
Aug 5 – Sep 2, 2026

Key findings

Typing is mainstream: 58% always use type hints and 28% often do — 85% combined.

Static checkers: Mypy and Pyright tied: 46.1% and 46.0%, with ty at 37.0%. Ruff lint rules (56.8%) and Pydantic (49.3%) top the wider tool list, but neither is a static checker.

Docs do the teaching: official docs (73.2%) top learning; Python Typing docs (65.9%) top help source, with LLM chat tools close behind (41.7%).

Adoption keeps growing: 44% say type hints are used more than a year ago; under 1% say less.

Speed leads checker choice: 56.5% chose their checker for speed/performance — the top reason.

Top pain: untyped libraries: 65.1% cite third-party libraries lacking stubs; 45.7% want better Callable syntax, 44.1% want a fully annotated stdlib.

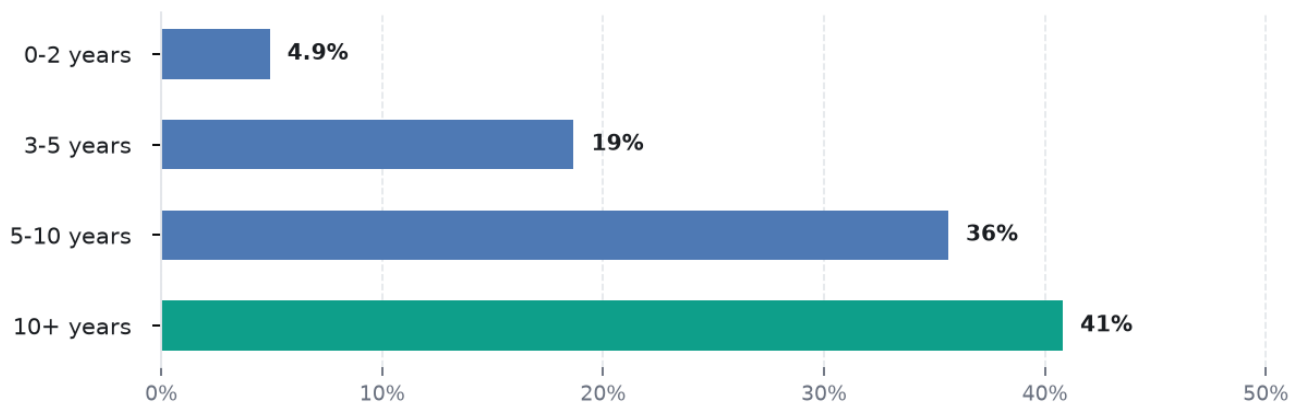
Percentages use per-question respondents as the base (shown as *n* under each chart). Multi-select questions sum past 100%. See “About this survey” for branching and drop-off details.

1 Python & typing experience

85% use type hints often or always

How many years of experience do you have using Python (typed or untyped)?

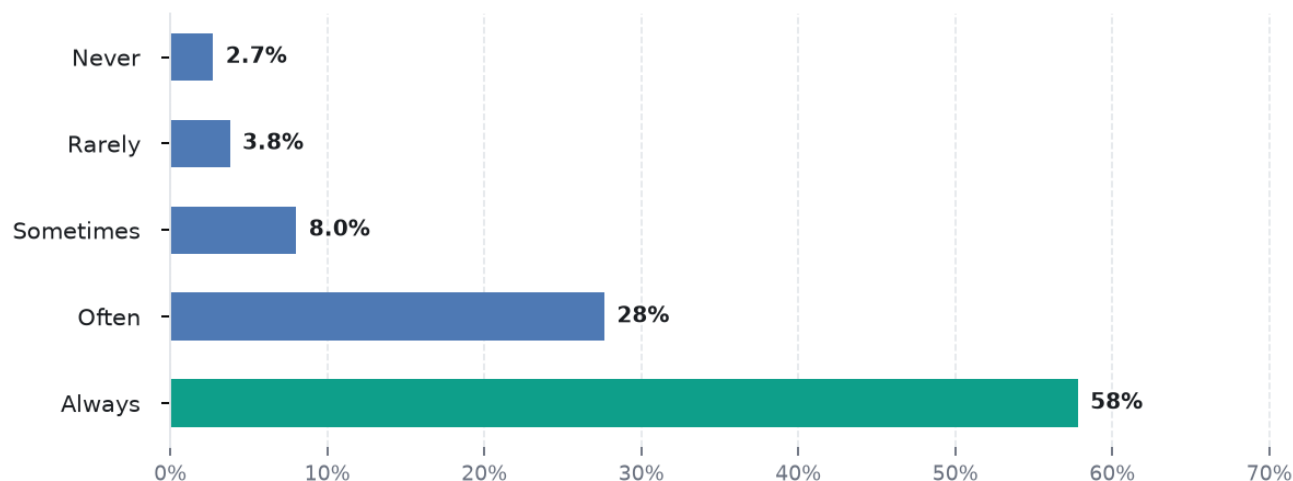
n = 1,794



Top answer: **10+ years** — 40.8%.

How often do you use type hints (static types) in your Python projects?

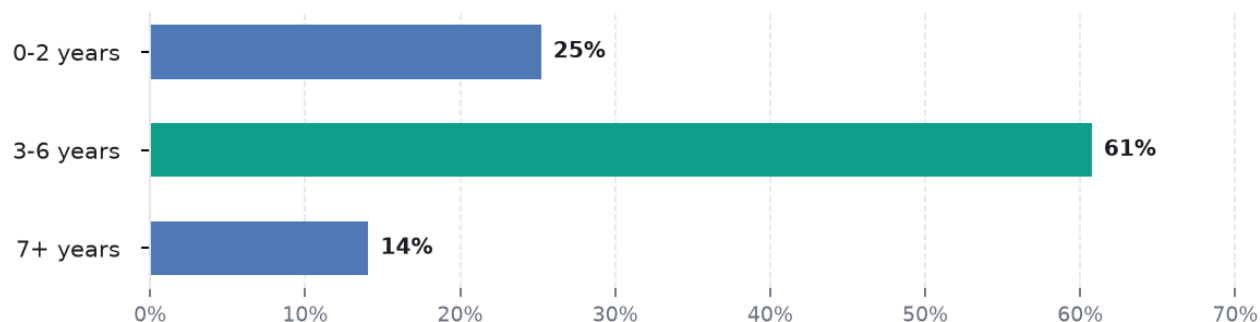
n = 1,794



Top answer: **Always** — 57.8%.

How many years of experience do you have using type hints in Python?

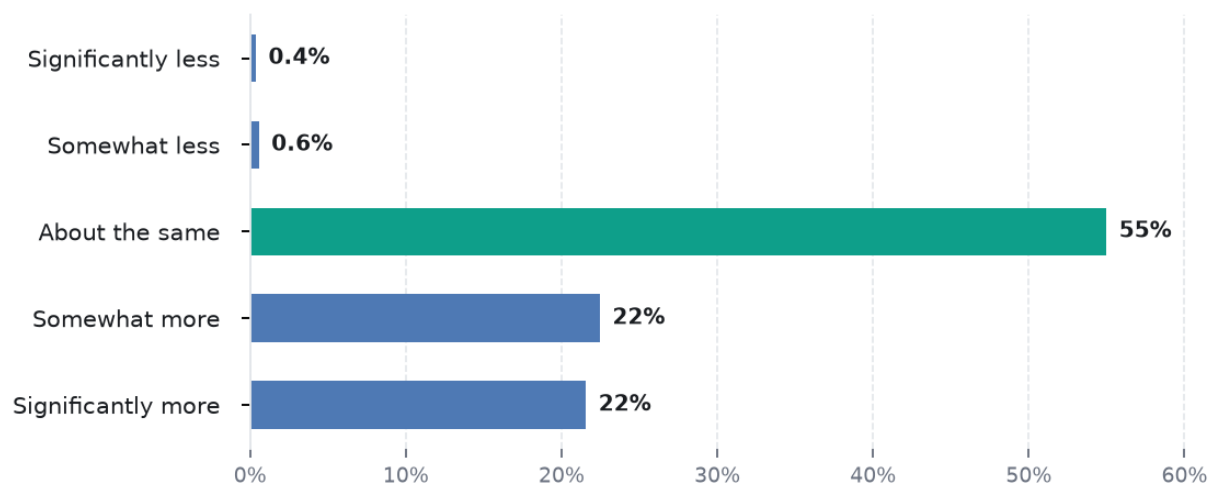
n = 1,571 • only respondents who use types (Q2 ≠ Never)



Top answer: **3-6 years** — 60.7%.

Compared to a year ago, how has the use of type hints in your projects changed?

n = 1,571 • only respondents who use types (Q2 ≠ Never)



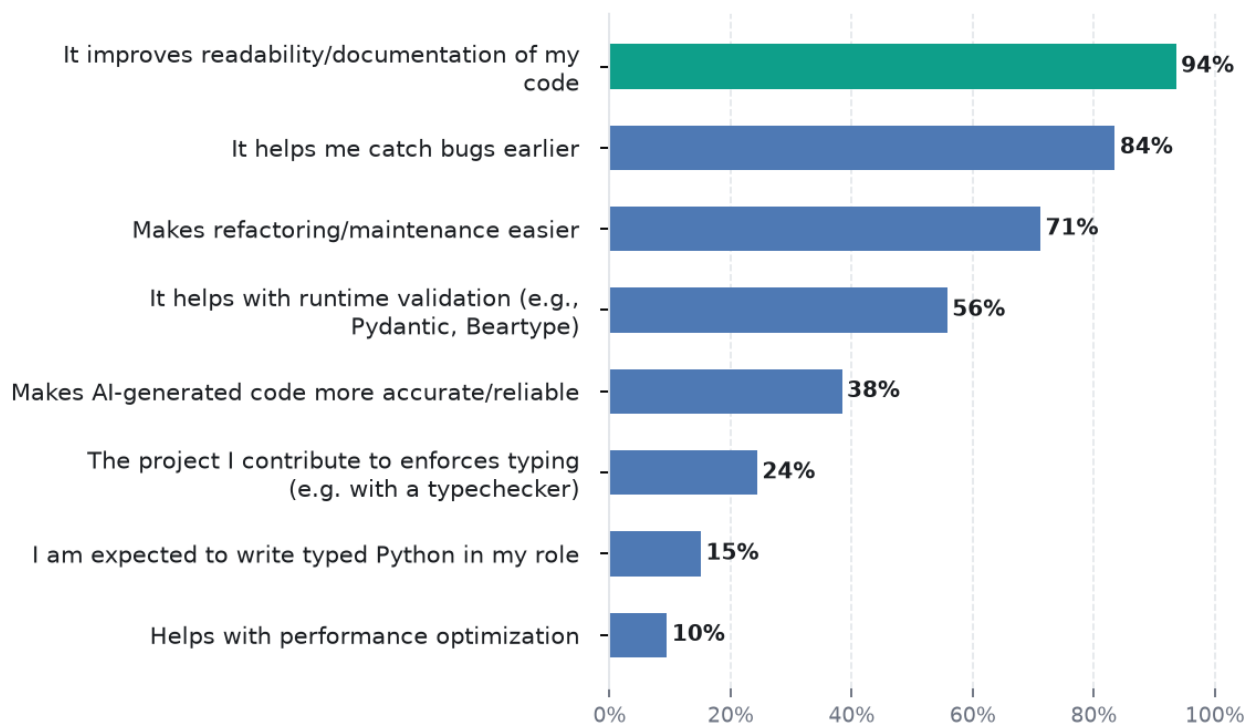
Top answer: **About the same** — 55.0%.

2 Why types — or why not

94% type for readability & documentation

If you DO use static types, what are the reasons?

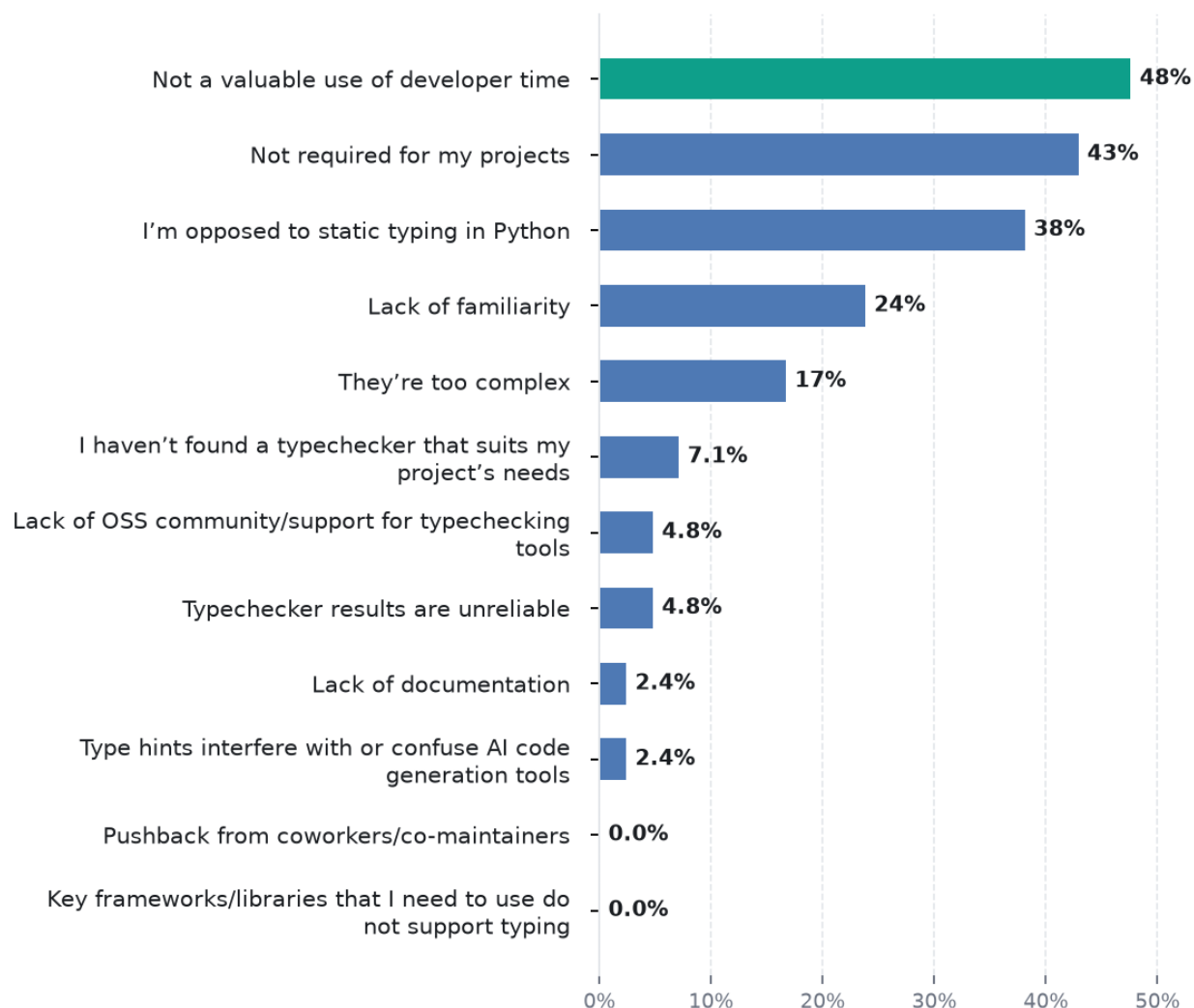
n = 1,571 (select all that apply) • only respondents who use types (Q2 ≠ Never)



Top answer: **It improves readability/documentation of my code** — 93.6%. *Other*: 103 (6.6%). Examples: “it’s the right thing to do.”; “IDE support”; “it’s a fun puzzle to add types to an existing project”.

If you DO NOT use static types, what are the reasons?

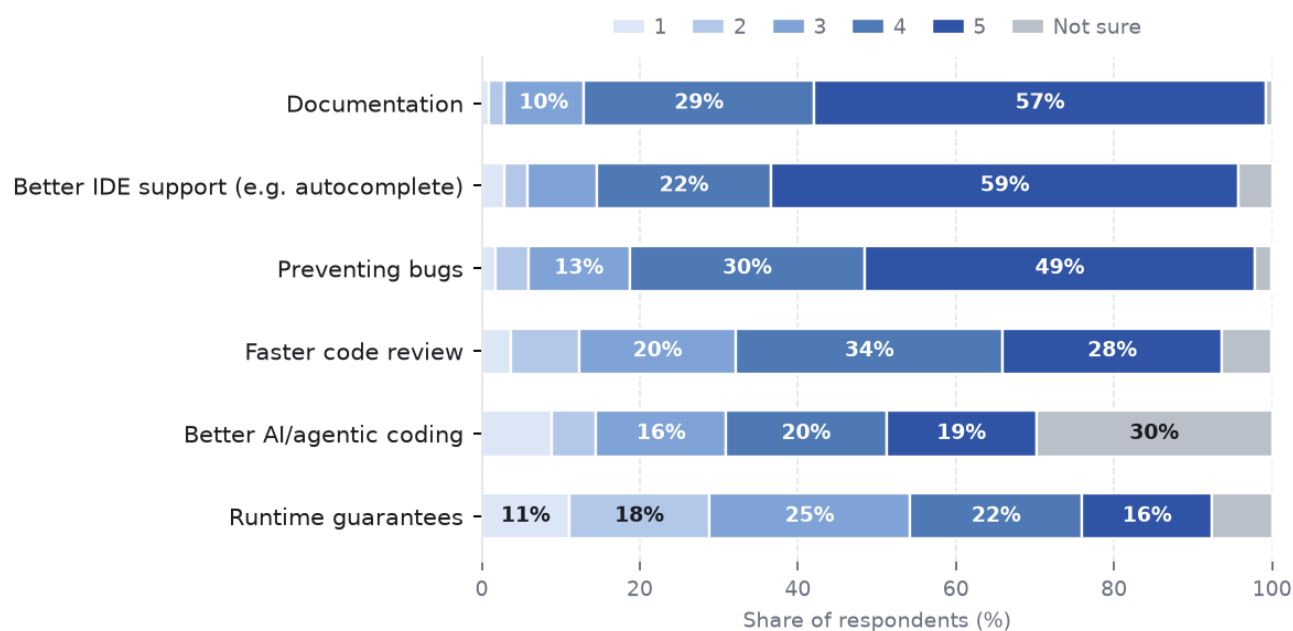
$n = 42$ (select all that apply) • only respondents who answered “Never” in Q2 — small base, read with care



Top answer: **Not a valuable use of developer time** — 20 of 42 (47.6%). *Other*: 17 (40.5%). Examples: “Type hints reduce readability/understanding”; “For research programming it’s not so necessary only until I publish”; “Static typing is not compatible with main idea of python language ...”.

How useful do you find the following benefits of typed Python code? (1-5)

n ≈ 1,560 per benefit • only respondents who use types



Benefit	Mean	4-5	Not sure
Documentation	4.41	86.3%	0.9%
Better IDE support (e.g. autocomplete)	4.38	81.2%	4.3%
Preventing bugs	4.23	79.1%	2.1%
Faster code review	3.78	61.6%	6.2%
Better AI/agent coding	3.50	39.3%	29.8%
Runtime guarantees	3.16	38.2%	7.7%

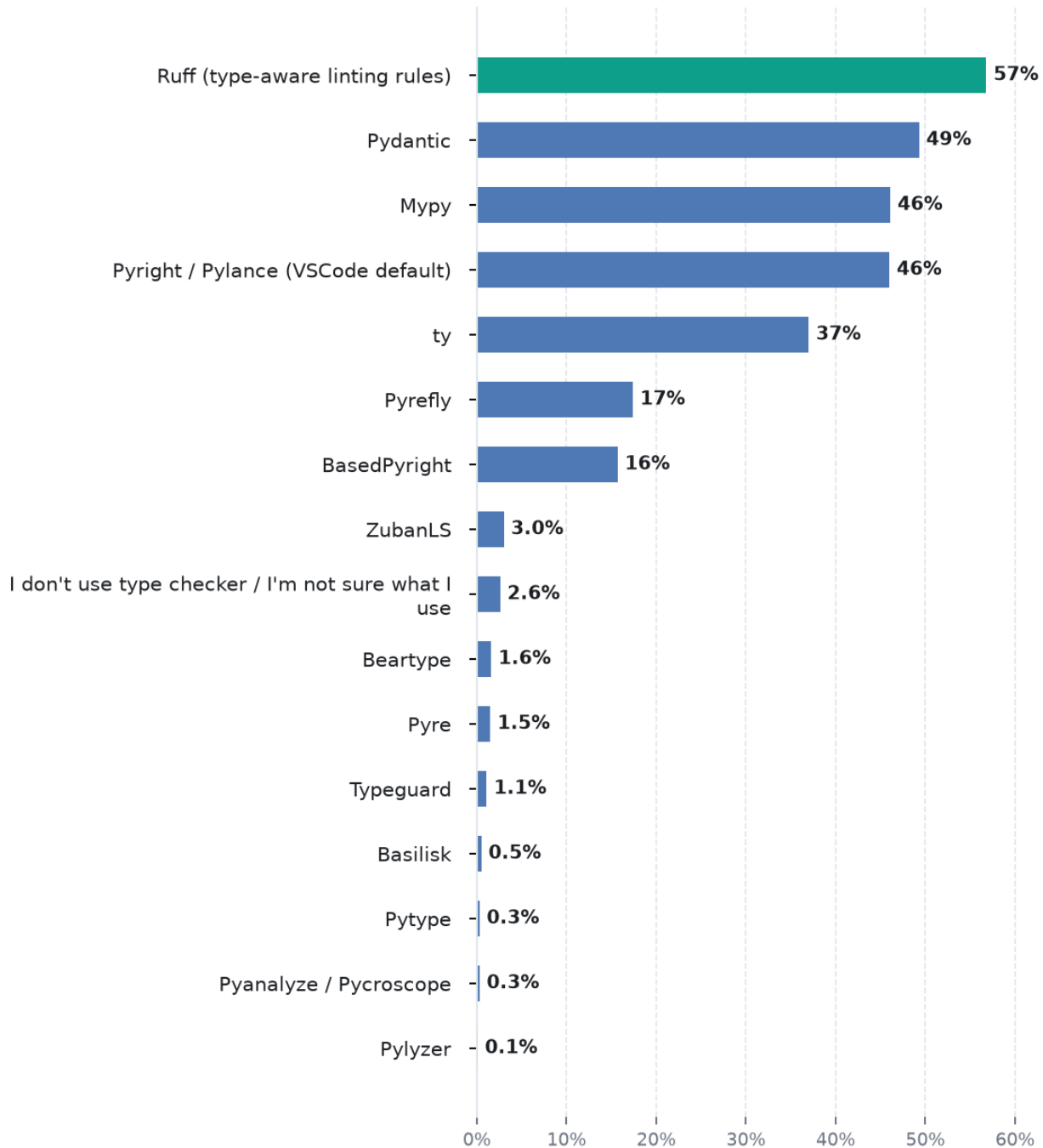
4-5 shares use all row answerers as the base (including “I’m not sure”); means exclude “I’m not sure”. Documentation (4.41) and IDE support (4.38) top the list; runtime guarantees (3.16) trail.

3 Type checkers

46% use Mypy — neck and neck with Pyright

What type checking tools do your projects use?

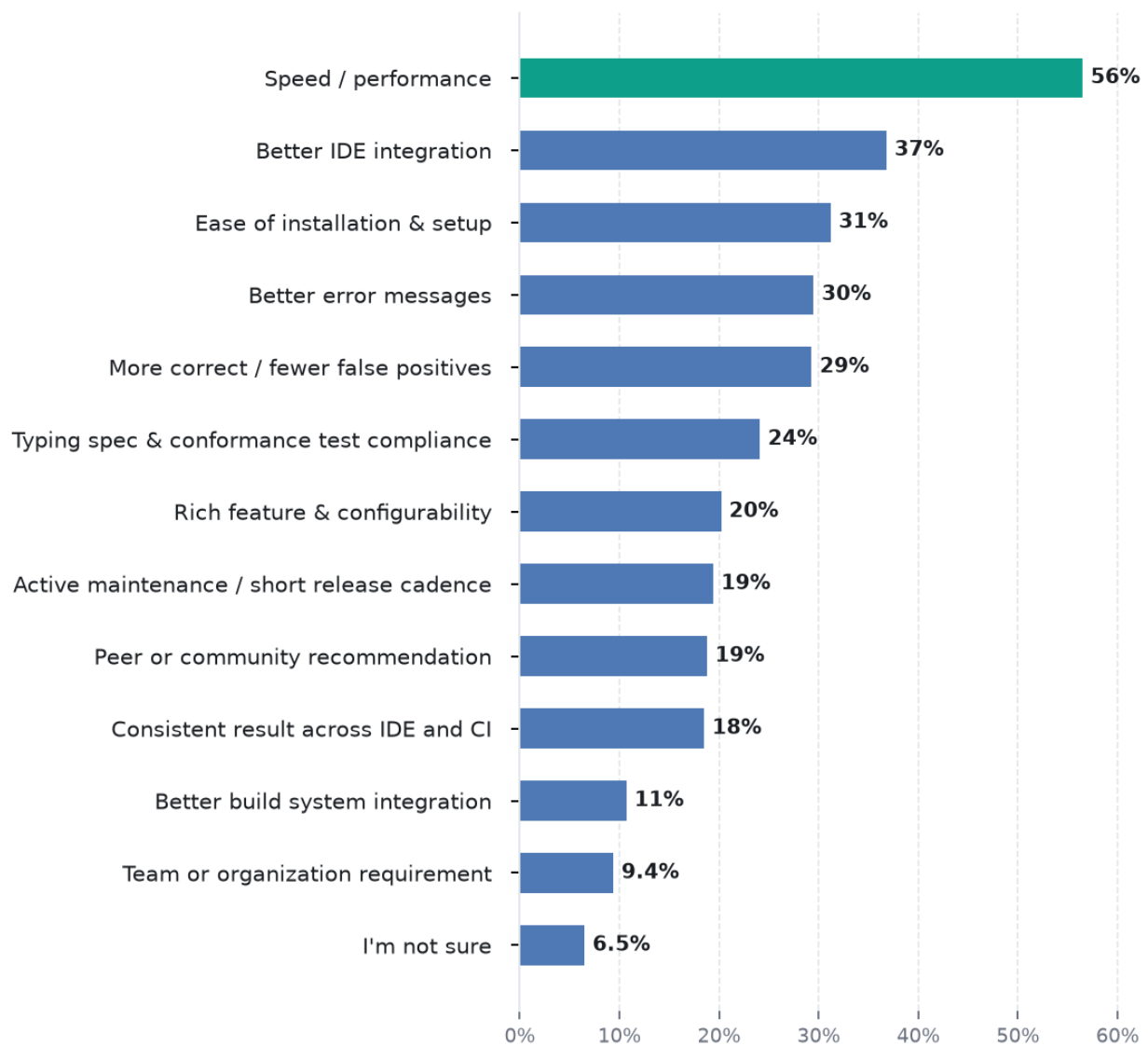
n = 1,517 (select all that apply) • Ruff is a linter and Pydantic a runtime validator — neither is a static checker



Top answer: **Ruff (type-aware linting rules)** — 56.8%. Other: 37 (2.4%). Examples: “Pycharm's default”; “flake8 plugins”; “mypy-baseline”.

Why did you choose / switch to your current type checker(s)?

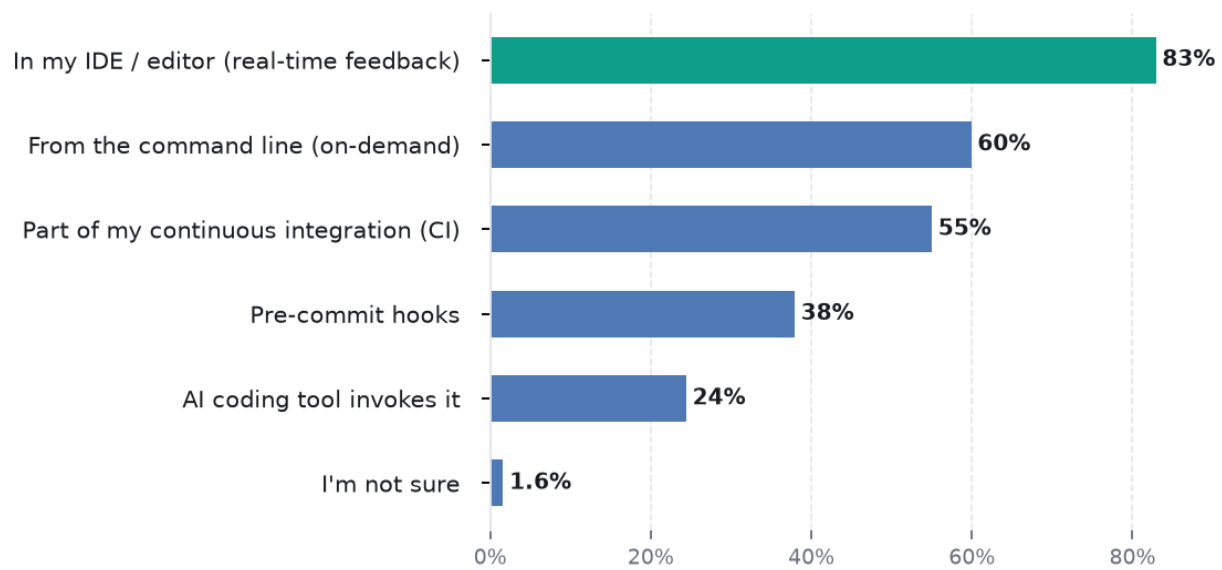
n = 1,517 (select all that apply)



Top answer: **Speed / performance** — 56.5%. *Other*: 123 (8.1%). Examples: “Added perspective”; “Appreciation of the work by the astral team”; “Strictness”.

How do you run your type checker?

n = 1,517 (select all that apply)



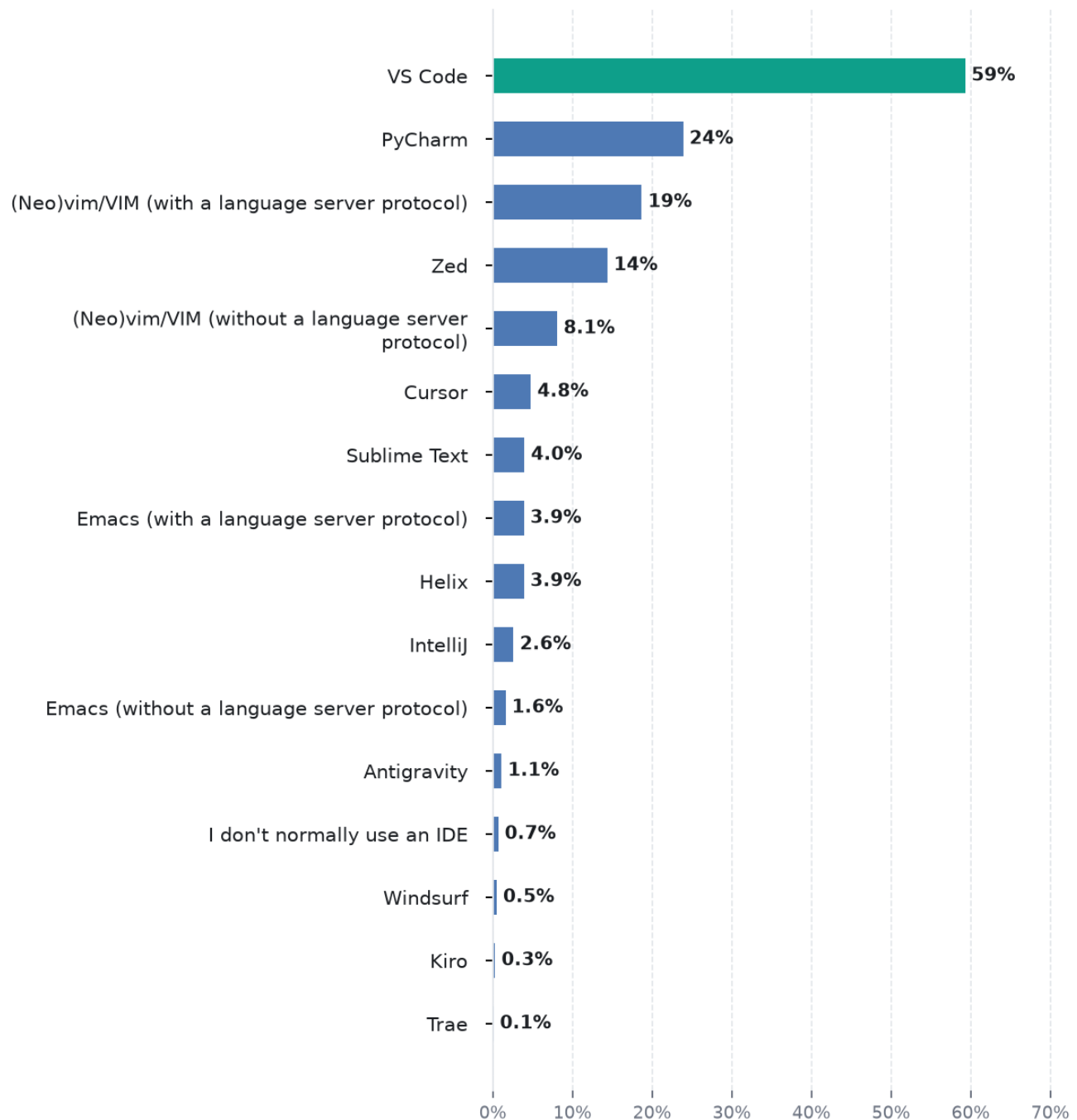
Top answer: **In my IDE / editor (real-time feedback)** — 82.9%. Other: 14 (0.9%). Examples: “Manually”; “tox”; “custom bazel-based sanitizer tool”.

4 Editors & IDEs

59% mainly code in VS Code

Which editors/IDEs do you mainly use for Python development?

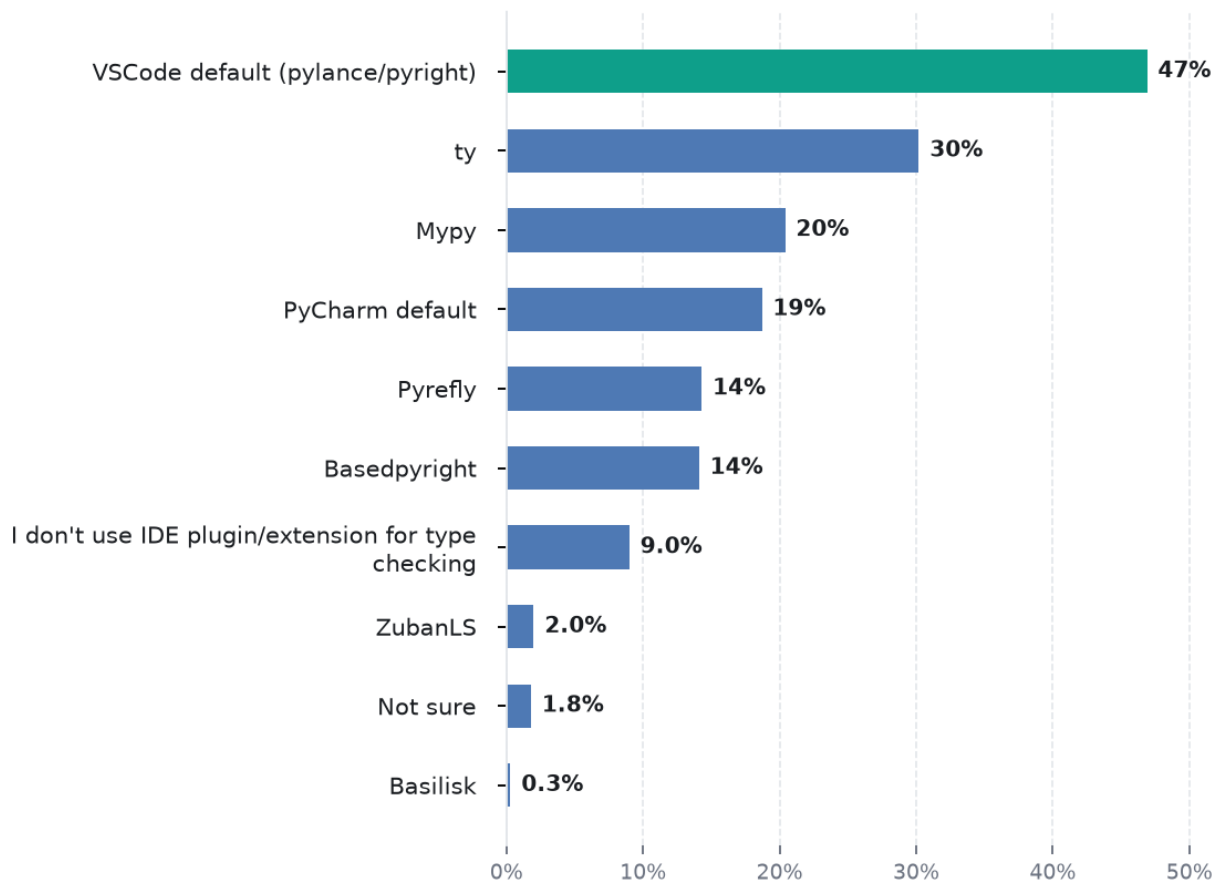
n = 1,517 (select all that apply)



Top answer: **VS Code** — 59.3%. *Other*: 75 (4.9%). Examples: “gedit”; “KDE's Kate”; “Jupyter for prototyping”.

Which IDE plugins/extensions do you use for typing?

n = 1,517 (select all that apply)



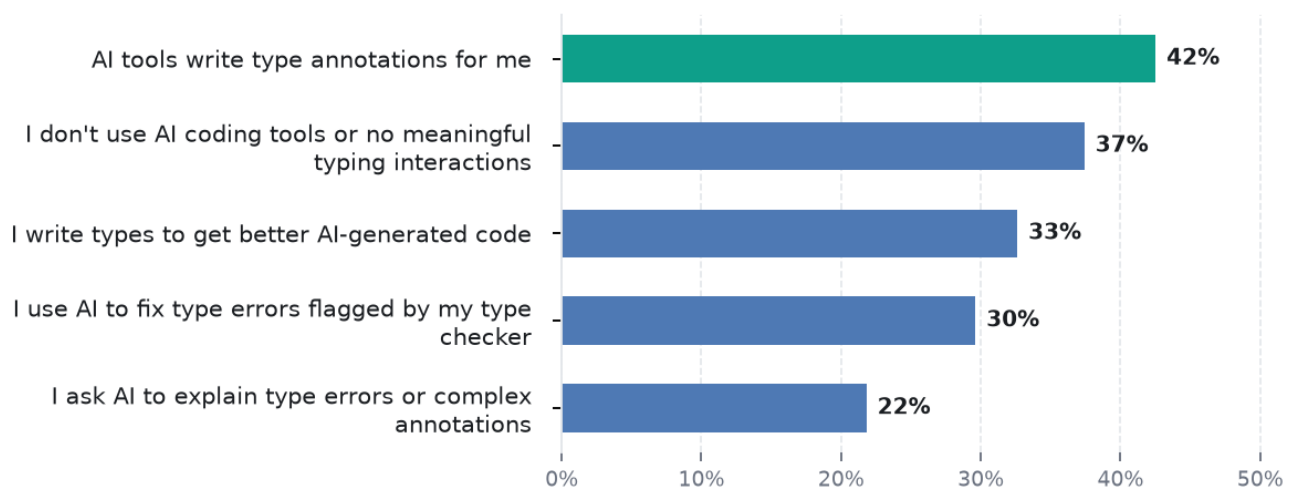
Top answer: **VSCoDe default (pylance/pyright)** — 46.9%. Other: 32 (2.1%). Examples: “JetBrains builtin”; “nvim-lint, nvim-lspconfig”; “neovim + lsp”.

5 AI & typing

42% have AI write their annotations

How do AI coding tools interact with your use of type annotations?

n = 1,517 (select all that apply)



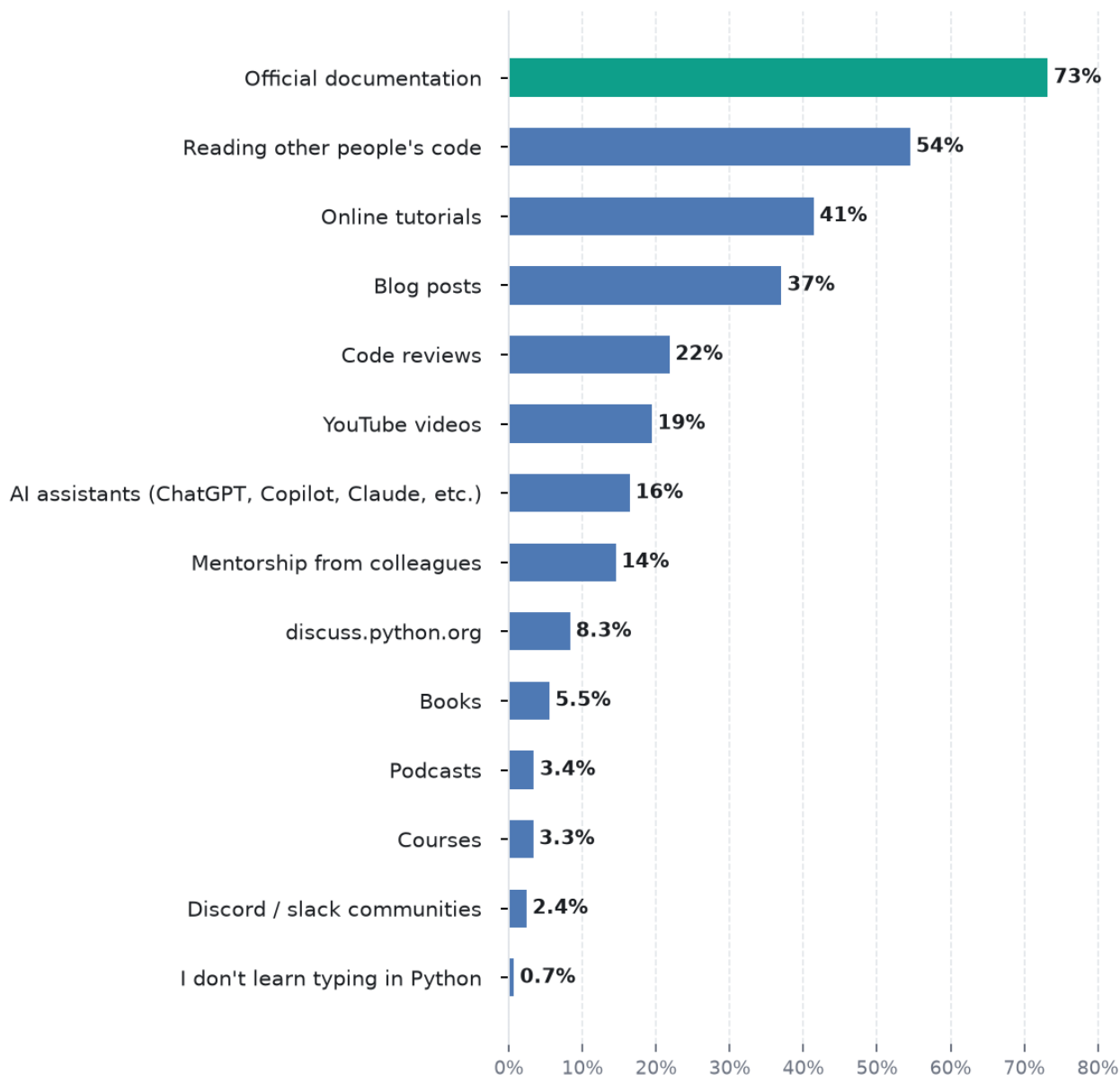
Top answer: **AI tools write type annotations for me** — 42.5%. Other: 34 (2.2%). Examples: “If using AI, I ask it to be mypy —strict compliant with an integrated ...”; “AI tools are requested to generate fully type-checked code for all ...”; “For reviewing and prioritization”.

6 Learning & staying current

73% learned typing from official docs

How did you learn to use Python typing?

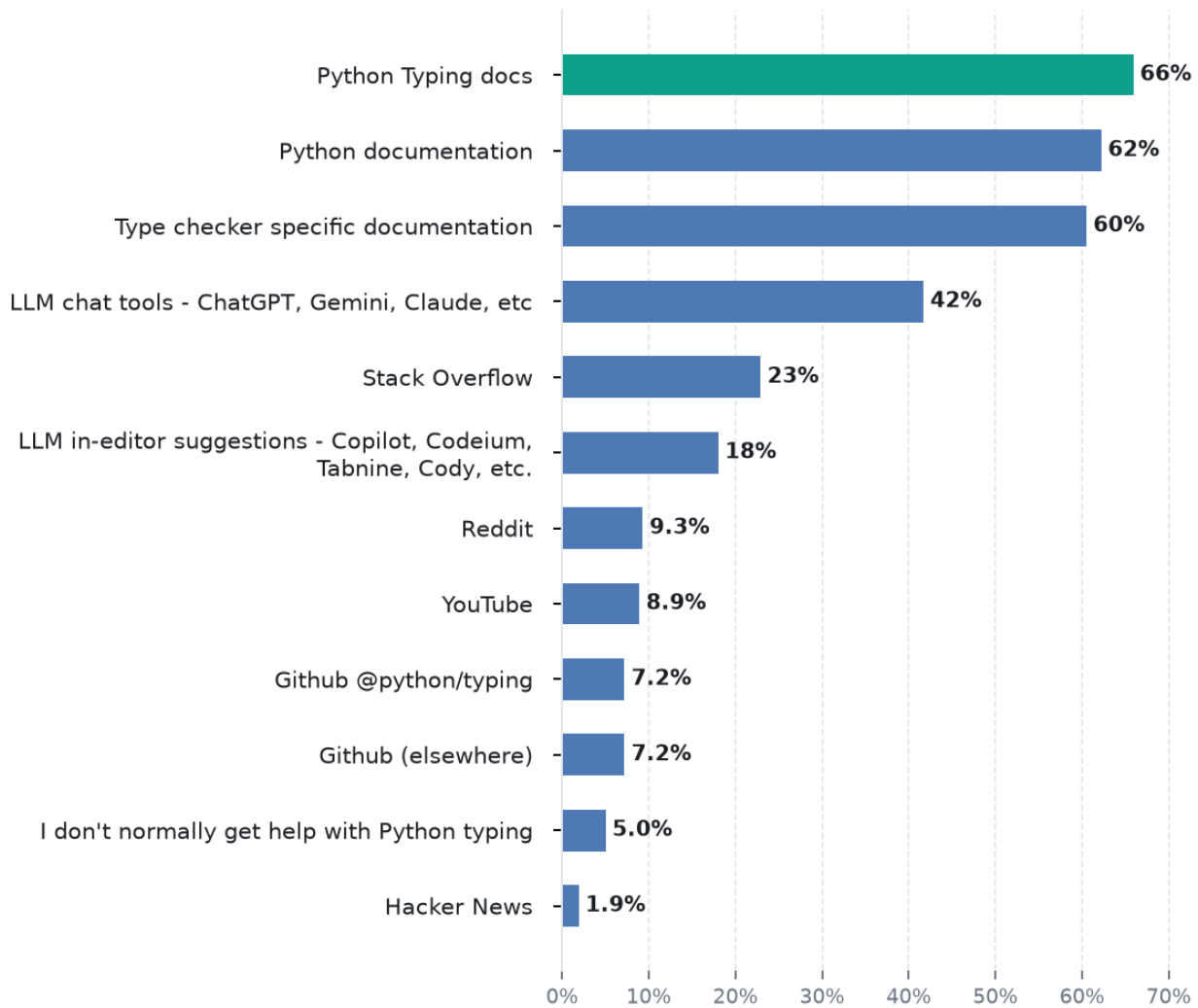
n = 1,480 (select all that apply)



Top answer: **Official documentation** — 73.2%. *Other*: 72 (4.9%). Examples: “Randomly online”; “I used only the Mypy documentation, which included one or more ...”; “University”.

Where do you get help using Python typing?

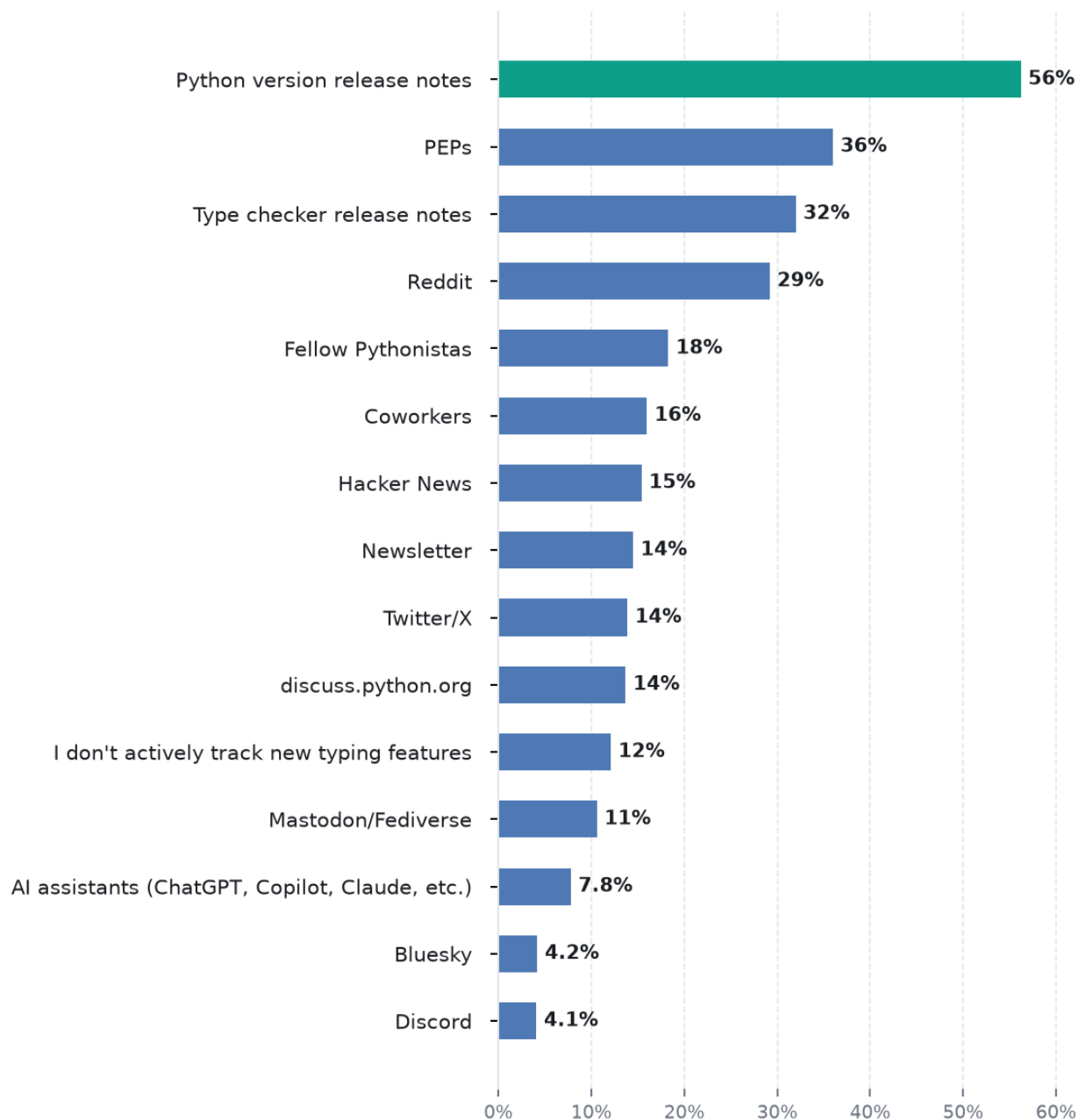
n = 1,480 (select all that apply)



Top answer: **Python Typing docs** — 65.9%. Other: 70 (4.7%). Examples: “Coworker”; “Co-workers”; “Python Discord”.

Where do you hear about new type system features?

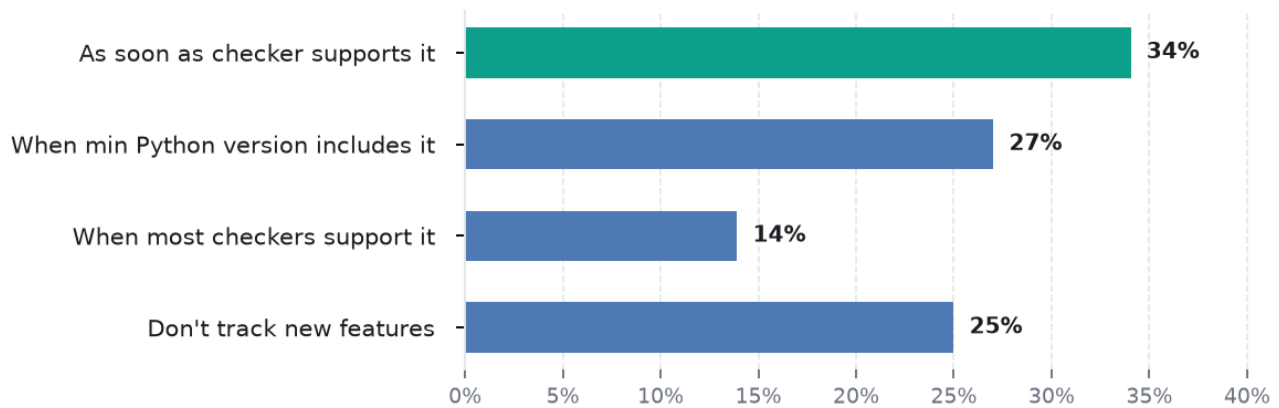
n = 1,480 (select all that apply)



Top answer: **Python version release notes** — 56.2%. Other: 96 (6.5%). Examples: “reading PEPs”; “anthonywritescode on YouTube”; “blog posts”.

How quickly do you adopt new type system features?

n = 1,480



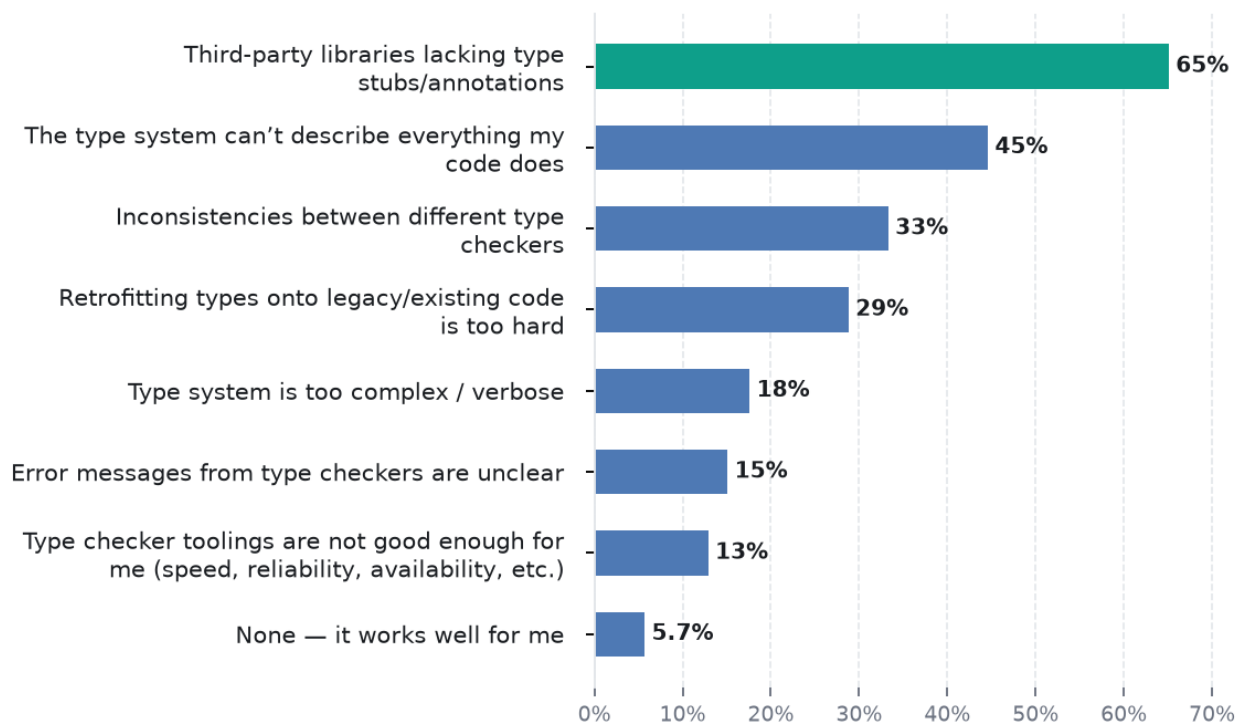
Top answer: **As soon as my type checker supports them (even via ``typing_extensions``)** — 34.1%.

7 Pain points & missing features

65% are blocked by untyped third-party libraries

What are the hardest parts about using the Python type system?

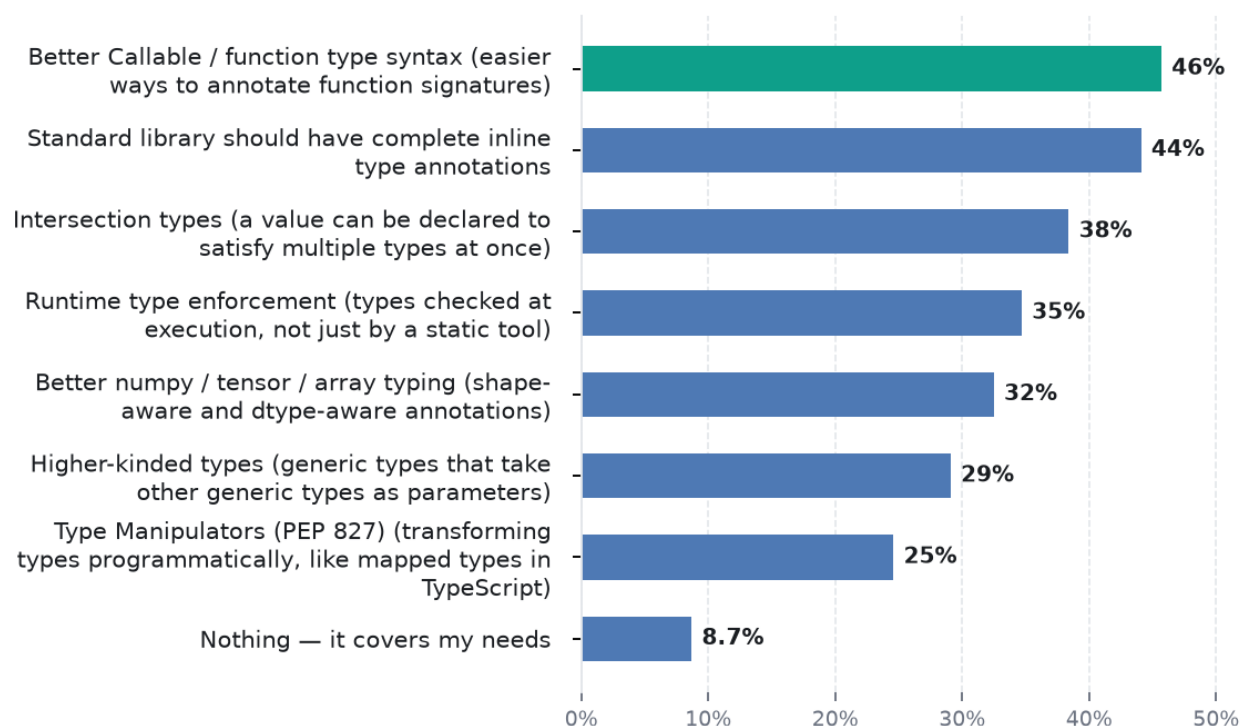
n = 1,486 (select all that apply)



Top answer: **Third-party libraries lacking type stubs/annotations** — 65.1%. Other: 156 (10.5%). Examples: “Sqlalchemy is a mess with typing. It has mypy extension, but some ...”; “Type system is missing features TypeScript has”; “I am lazy, and been used to not typing for so long.”.

What features do you think are missing from the Python type system?

n = 1,486 (select all that apply)

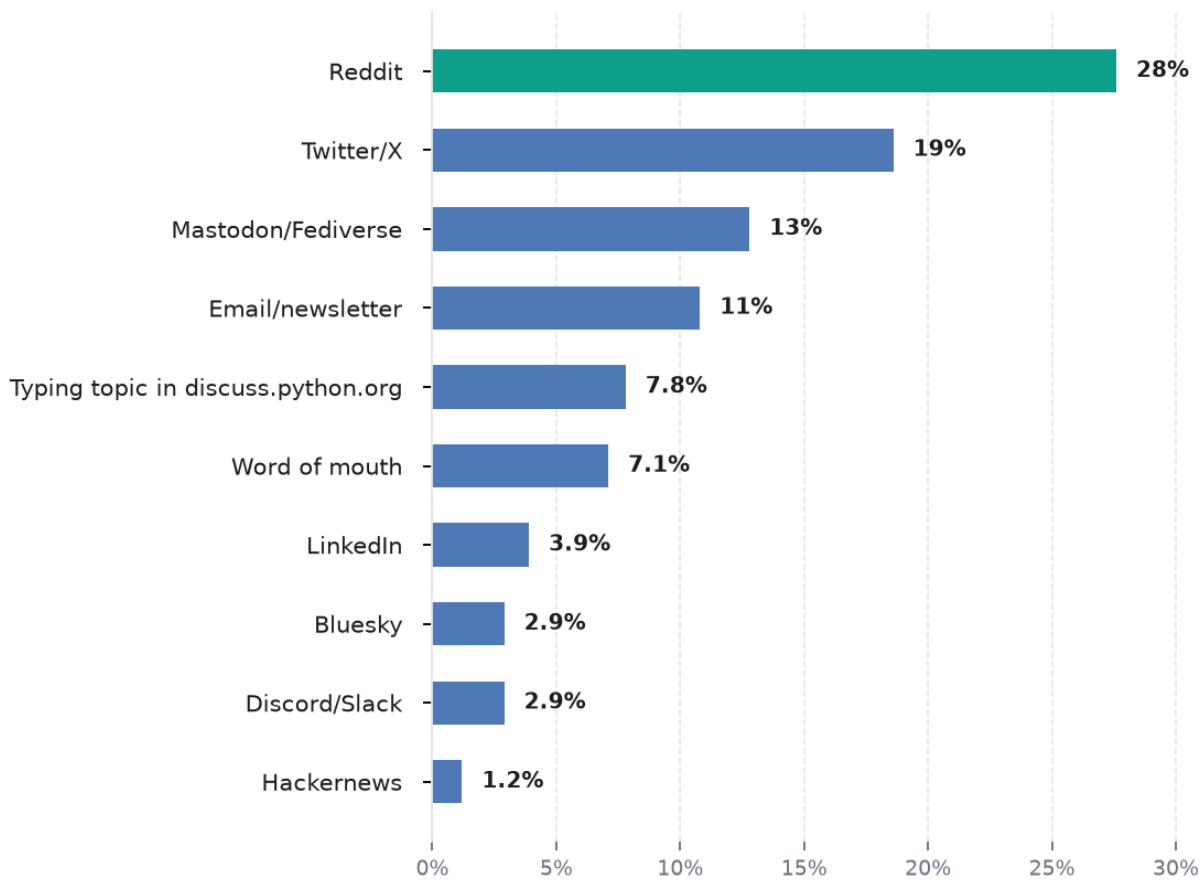


Top answer: **Better Callable / function type syntax (easier ways to annotate function signatures)** — 45.7%. *Other*: 165 (11.1%). Examples: "Python should move towards Haskell 98"; "Extending or omitting types Typescript style"; "The "missing feature" is that we would need a type hierarchy that is ...".

8 About this survey

How did you hear about this survey?

n = 1,469 (select all that apply)



Top answer: **Reddit** — 27.6%. *Other*: 132 (9.0%). Examples: “Work”; “Pycoders Weekly newsletter”; “Banner on Pyrefly's website”.

Methodology

1,794 people answered the survey between Aug 5 – Sep 2, 2026. They are a self-selected group of people who care about typing: 85.5% already use type hints often or always. These results describe the people who answered.

The survey used two paths. The 49 respondents who never use type hints answered a short set of questions about why, skipped the tooling block (Q4–Q17), and rejoined at the end. The 708 respondents who use types only some of the time were not asked why, so the survey has no direct data on partial use. Later questions have fewer answers (1,469 reached the last one), mostly because less experienced, occasional users dropped out.

Every chart states its own respondent count. “Select all that apply” answers show which tools each person uses; they do not identify a main tool or a completed switch. Blank answers count as skipped. “Other” boxes hold free text and are counted separately. Appendix A lists full counts for every answer.

The survey export covers 1,794 responses.

Appendix A Response counts

Every answer choice with raw counts and shares. Single-choice rows follow survey order; “select all that apply” rows are ranked as in the charts. Q7 cells show count (share).

Q1 · How many years of experience do you have using Python (typed or untyped)? (n = 1,794)

Answer	Responses	Share
0-2 years	88	4.9%
3-5 years	335	18.7%
5-10 years	639	35.6%
10+ years	732	40.8%

Q2 · How often do you use type hints (static types) in your Python projects? (n = 1,794)

Answer	Responses	Share
Never	49	2.7%
Rarely	68	3.8%
Sometimes	144	8.0%
Often	496	27.6%
Always	1,037	57.8%

Q3 · If you DO NOT use static types, what are the reasons? (n = 42)

Answer	Responses	Share
Not a valuable use of developer time	20	47.6%
Not required for my projects	18	42.9%
I’m opposed to static typing in Python	16	38.1%
Lack of familiarity	10	23.8%
They’re too complex	7	16.7%
I haven’t found a typechecker that suits my project’s needs	3	7.1%
Lack of OSS community/support for typechecking tools	2	4.8%
Typechecker results are unreliable	2	4.8%
Lack of documentation	1	2.4%
Type hints interfere with or confuse AI code generation tools	1	2.4%
Pushback from coworkers/co-maintainers	0	0.0%
Key frameworks/libraries that I need to use do not support typing	0	0.0%
Other (please specify) — free text	17	40.5%

Q4 · How many years of experience do you have using type hints in Python? (n = 1,571)

Answer	Responses	Share
0-2 years	396	25.2%
3-6 years	954	60.7%
7+ years	221	14.1%

Q5 · If you DO use static types, what are the reasons? (n = 1,571)

Answer	Responses	Share
It improves readability/documentation of my code	1,471	93.6%
It helps me catch bugs earlier	1,312	83.5%
Makes refactoring/maintenance easier	1,119	71.2%
It helps with runtime validation (e.g., Pydantic, Beartype)	876	55.8%
Makes AI-generated code more accurate/reliable	604	38.4%
The project I contribute to enforces typing (e.g. with a typechecker)	384	24.4%
I am expected to write typed Python in my role	238	15.1%
Helps with performance optimization	149	9.5%
Other (please specify) — free text	103	6.6%

Q6 · Compared to a year ago, how has the use of type hints in your projects changed? (n = 1,571)

Answer	Responses	Share
Type hints are used significantly less than a year ago	6	0.4%
Type hints are used somewhat less than a year ago	9	0.6%
About the same	864	55.0%
Type hints are used somewhat more than a year ago	353	22.5%
Type hints are used significantly more than a year ago	339	21.6%

Q7 · Usefulness of typed-Python benefits (n = 1,551-1,567)

Benefit	1	2	3	4	5	Not sure
Documentation	14 (0.9%)	29 (1.9%)	158 (10.1%)	457 (29.2%)	894 (57.1%)	14 (0.9%)
Better IDE support (e.g. autocomplete)	43 (2.8%)	47 (3.0%)	137 (8.8%)	343 (22.0%)	924 (59.2%)	67 (4.3%)
Preventing bugs	28 (1.8%)	65 (4.1%)	201 (12.8%)	467 (29.8%)	773 (49.3%)	33 (2.1%)
Faster code review	58 (3.7%)	134 (8.6%)	310 (19.9%)	525 (33.7%)	433 (27.8%)	96 (6.2%)
Better AI/agent coding	138 (8.9%)	87 (5.6%)	254 (16.4%)	315 (20.3%)	295 (19.0%)	462 (29.8%)
Runtime guarantees	173 (11.1%)	275 (17.7%)	395 (25.4%)	339 (21.8%)	256 (16.4%)	120 (7.7%)

Q8 · What type checking tools do your projects use? (n = 1,517 — Ruff is a linter, Pydantic a runtime validator)

Answer	Responses	Share
Ruff (type-aware linting rules)	862	56.8%
Pydantic	748	49.3%
Mypy	700	46.1%
Pyright / Pylance (VSCode default)	698	46.0%
ty	562	37.0%
Pyrefly	264	17.4%
BasedPyright	238	15.7%
ZubanLS	45	3.0%
I don't use type checker / I'm not sure what I use	39	2.6%
Beartype	25	1.6%
Pyre	23	1.5%

Answer	Responses	Share
Typeguard	16	1.1%
Basilisk	8	0.5%
Pytype	5	0.3%
Pyanalyze / Pycroscope	5	0.3%
Pylyzer	2	0.1%
Other (please specify) — free text	37	2.4%

Q9 · Why did you choose / switch to your current type checker(s)? (n = 1,517)

Answer	Responses	Share
Speed / performance	857	56.5%
Better IDE integration	559	36.8%
Ease of installation & setup	474	31.2%
Better error messages	447	29.5%
More correct / fewer false positives	444	29.3%
Typing spec & conformance test compliance	365	24.1%
Rich feature & configurability	306	20.2%
Active maintenance / short release cadence	294	19.4%
Peer or community recommendation	285	18.8%
Consistent result across IDE and CI	281	18.5%
Better build system integration	162	10.7%
Team or organization requirement	143	9.4%
I'm not sure	99	6.5%
Other (please specify) — free text	123	8.1%

Q10 · Which editors/IDEs do you mainly use for Python development? (n = 1,517)

Answer	Responses	Share
VS Code	899	59.3%
PyCharm	362	23.9%
(Neo)vim/VIM (with a language server protocol)	283	18.7%
Zed	219	14.4%
(Neo)vim/VIM (without a language server protocol)	123	8.1%
Cursor	73	4.8%
Sublime Text	61	4.0%
Emacs (with a language server protocol)	59	3.9%
Helix	59	3.9%
IntelliJ	40	2.6%
Emacs (without a language server protocol)	25	1.6%
Antigravity	17	1.1%
I don't normally use an IDE	10	0.7%
Windsurf	7	0.5%
Kiro	5	0.3%
Trae	2	0.1%
Other (please specify) — free text	75	4.9%

Q11 · Which IDE plugins/extensions do you use for typing? (n = 1,517)

Answer	Responses	Share
VSCode default (pylance/pyright)	712	46.9%

Answer	Responses	Share
ty	458	30.2%
Mypy	310	20.4%
PyCharm default	284	18.7%
Pyrefly	217	14.3%
Basedpyright	214	14.1%
I don't use IDE plugin/extension for type checking	137	9.0%
ZubanLS	30	2.0%
Not sure	27	1.8%
Basilisk	4	0.3%
Other (please specify) — free text	32	2.1%

Q12 · How do you run your type checker? (n = 1,517)

Answer	Responses	Share
In my IDE / editor (real-time feedback)	1,257	82.9%
From the command line (on-demand)	908	59.9%
Part of my continuous integration (CI)	835	55.0%
Pre-commit hooks	575	37.9%
AI coding tool invokes it	370	24.4%
I'm not sure	24	1.6%
Other (please specify) — free text	14	0.9%

Q13 · How do AI coding tools interact with your use of type annotations? (n = 1,517)

Answer	Responses	Share
AI tools write type annotations for me	644	42.5%
I don't use AI coding tools or no meaningful typing interactions	567	37.4%
I write types to get better AI-generated code	494	32.6%
I use AI to fix type errors flagged by my type checker	449	29.6%
I ask AI to explain type errors or complex annotations	331	21.8%
Other (please specify) — free text	34	2.2%

Q14 · Where do you hear about new type system features? (n = 1,480)

Answer	Responses	Share
Python version release notes	832	56.2%
PEPs	533	36.0%
Type checker release notes	473	32.0%
Reddit	432	29.2%
Fellow Pythonistas	271	18.3%
Coworkers	237	16.0%
Hacker News	228	15.4%
Newsletter	215	14.5%
Twitter/X	205	13.9%
discuss.python.org	203	13.7%
I don't actively track new typing features	179	12.1%
Mastodon/Fediverse	157	10.6%
AI assistants (ChatGPT, Copilot, Claude, etc.)	116	7.8%
Bluesky	62	4.2%
Discord	61	4.1%

Answer	Responses	Share
Other (please specify) — free text	96	6.5%

Q15 · How quickly do you adopt new type system features? (n = 1,480)

Answer	Responses	Share
As soon as my type checker supports them (even via <code>`typing_extensions`</code>)	504	34.1%
When my project's minimum Python version includes them natively	400	27.0%
When it's supported by most major type checkers	206	13.9%
I don't actively track new typing features	370	25.0%

Q16 · How did you learn to use Python typing? (n = 1,480)

Answer	Responses	Share
Official documentation	1,084	73.2%
Reading other people's code	806	54.5%
Online tutorials	613	41.4%
Blog posts	548	37.0%
Code reviews	322	21.8%
YouTube videos	287	19.4%
AI assistants (ChatGPT, Copilot, Claude, etc.)	242	16.4%
Mentorship from colleagues	215	14.5%
discuss.python.org	123	8.3%
Books	81	5.5%
Podcasts	50	3.4%
Courses	49	3.3%
Discord / slack communities	36	2.4%
I don't learn typing in Python	11	0.7%
Other (please specify) — free text	72	4.9%

Q17 · Where do you get help using Python typing? (n = 1,480)

Answer	Responses	Share
Python Typing docs	975	65.9%
Python documentation	921	62.2%
Type checker specific documentation	895	60.5%
LLM chat tools - ChatGPT, Gemini, Claude, etc	617	41.7%
Stack Overflow	339	22.9%
LLM in-editor suggestions - Copilot, Codeium, Tabnine, Cody, etc.	267	18.0%
Reddit	138	9.3%
YouTube	132	8.9%
Github @python/typing	106	7.2%
Github (elsewhere)	106	7.2%
I don't normally get help with Python typing	74	5.0%
Hacker News	28	1.9%
Other (please specify) — free text	70	4.7%

Q18 · What are the hardest parts about using the Python type system? (n = 1,486)

Answer	Responses	Share
Third-party libraries lacking type stubs/annotations	968	65.1%
The type system can't describe everything my code does	663	44.6%
Inconsistencies between different type checkers	495	33.3%
Retrofitting types onto legacy/existing code is too hard	428	28.8%
Type system is too complex / verbose	262	17.6%
Error messages from type checkers are unclear	224	15.1%
Type checker toolings are not good enough for me (speed, reliability, availability, etc.)	192	12.9%
None — it works well for me	84	5.7%
Other (please specify) — free text	156	10.5%

Q19 · What features do you think are missing from the Python type system? (n = 1,486)

Answer	Responses	Share
Better Callable / function type syntax (easier ways to annotate function signatures)	679	45.7%
Standard library should have complete inline type annotations	655	44.1%
Intersection types (a value can be declared to satisfy multiple types at once)	570	38.4%
Runtime type enforcement (types checked at execution, not just by a static tool)	516	34.7%
Better numpy / tensor / array typing (shape-aware and dtype-aware annotations)	483	32.5%
Higher-kinded types (generic types that take other generic types as parameters)	433	29.1%
Type Manipulators (PEP 827) (transforming types programmatically, like mapped types in TypeScript)	365	24.6%
Nothing — it covers my needs	130	8.7%
Other (please specify) — free text	165	11.1%

Q20 · How did you hear about this survey? (n = 1,469)

Answer	Responses	Share
Reddit	406	27.6%
Twitter/X	273	18.6%
Mastodon/Fediverse	188	12.8%
Email/newsletter	158	10.8%
Typing topic in discuss.python.org	114	7.8%
Word of mouth	105	7.1%
LinkedIn	58	3.9%
Bluesky	43	2.9%
Discord/Slack	43	2.9%
Hackernews	17	1.2%
Other (please specify) — free text	132	9.0%