PyCon au 26

Brisbane
August 26-30

Academic Proceedings

Published 16 September 2026

Copyright

Publication Date
September 2026

Published by PyCon AU.
Auspice by Linux Australia Inc.

Digital Object Identifier: **https://doi.org/10.5281/zenodo.22784195**

Contents

Preface

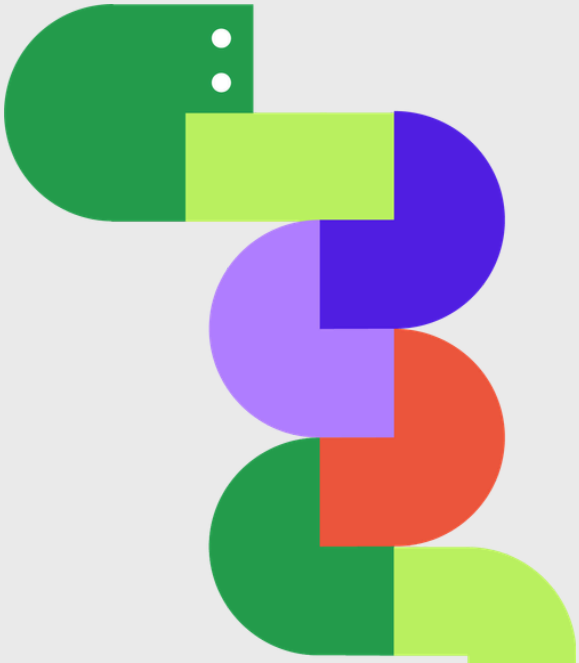
Abstracts

PyCon au 26

Brisbane
August 26-30

Sofitel Brisbane Central

For 16+ years, PyCon AU has been welcoming people from across Australia and the world, to connect, share and learn.



Welcome to PyCon AU

PyCon AU is the national conference for the Python programming community in Australia. The conference welcomes a wide cross-section of technology professionals, students and enthusiast developers spanning almost every sector and industry in the country.

Of note, this includes many researchers and academics who use Python to further our understanding of our societies, our histories, and the world around us. The conference regularly features practitioners across many fields including climate science, physics, bioinformatics, aerospace engineering and archaeology (to name a few).

The careful and rigorous selection process, chaired in 2026 by Rhydwyn McGuire, has produced a world-leading program that captures a broad cross-section of how Python and related technologies are developed, adopted, and taught, around both Australia and Aotearoa New Zealand.

Perhaps most impressively, this is achieved by an all-volunteer team whose drive and passion make the conference and proceedings possible.

I am excited to see an increasing number of speakers elect to publish their works in the academic proceedings this year, shepherded by editors Alan Rubin and Richard Littauer.

Jack Skinner
Conference Director
PyCon AU 2026

Program Selection

PyCon AU 2026 held an open Call For Proposals during February and March and received more submissions than any prior PyCon AU. I am grateful for everyone who submitted as PyCon AU would not be possible without the diversity of speakers.

Each proposal was anonymously reviewed by a program committee of over 50 reviewers. Anonymity guidelines were in place to address unconscious bias and to ensure early-in-career speakers were not penalised.

The review committee consisted of industry and academic leaders recognised for their expertise across a diverse range of backgrounds and fields. Specialist Track organisers further invited industry specialists to augment the committee's existing broad expertise.

The conference also recognised the need to build a pipeline of reviewers for years to come, and took conscious steps to invite new reviewers who represent the industry and perspectives of people who use Python *in Australia*. New perspectives are paramount to building a sustainable conference and industry.

The resulting program represents a thoughtful and rigorous review process and captures a cross-section of expertise and stories from the Python ecosystem in Australia and New Zealand.

Rhydwyn McGuire
Program Chair
PyCon AU 2026

From The Editors

Python is a fundamental part of the academic open source and research software ecosystem. We were delighted to see the new dedicated Research Software Engineering specialist track at this year's conference, co-organised by Paul Leopard and Jasmeen Kaur.

To further engage and support our community, speakers at PyCon AU 2026 were also offered the opportunity to have their abstracts published in these proceedings, no matter what specialist track their talk was delivered in.

These abstracts run the gamut from AI to teaching, from working with privacy, to launching rockets. Each abstract was vetted by peer reviewers, both as part of conference proposal selection and again for this work. We hope that these continue to be cited and used for future research.

These proceedings could not have happened without the full support of the organising committee, in particular Jack Skinner and Rhydwyn McGuire.

We are so grateful for their work!

Each abstract also has its own Zenodo page, which includes links to the recordings of the talks on YouTube.

See you next year!

Sincerely,
Alan Rubin & Richard Littauer
Proceedings Editors

Team

PyCon AU is organised by an all-volunteer team, auspiced by Linux Australia Inc.

Directors

Jack Skinner
Conference Director

Nic Crouch
Co-Director & Treasurer

Sophie Quinn-Graham
Co-Director

Program

Rhydwyn McGuire
Program Chair

Academic Proceedings

Alan Rubin
Co-Editor

Richard Littauer
Co-Editor

Why and How to Cite Software (and Get Your Software Cited Too)

Stephanie Chong

Software is widely used and produced in the course of research. Yet, published works (e.g. journal papers and conference papers) do not always cite the software that was essential for the research, and even when software is cited, the citation is often informal (see e.g., Howison & Bullard, 2016). Citing software has numerous benefits, including facilitating the reproducibility and transparency of research, as well as providing attribution and credit for the developers (Chue Hong et al., 2019a, 2019b; Katz et al., 2021; Smith et al., 2016). This talk will discuss the benefits of citing software, how to cite software and practical suggestions for how to make your software easier for others to cite.

This talk will include real examples and practical tips. Topics covered will include:

- The benefits of citing software
- How to cite software
- Finding the citation information for software
- Practical suggestions for making your software easier for others to cite

This talk is aimed at anyone who uses software in their research, creates software that is used in research, or does both.

Resources

Presentation recording (YouTube): https://www.youtube.com/watch?v=rR3Nn7fpRGY

Conference page: https://2026.pycon.org.au/schedule/8PXFQM/

https://doi.org/10.5281/zenodo.22784195

References

Abramatic, J.-F., Di Cosmo, R., & Zacchiroli, S. (2018). Building the universal archive of source code. *Communications of the ACM*, *61*(10), 29–31. https://doi.org/10.1145/3183558

American Psychological Association. (2020). *Publication manual of the American Psychological Association* (7th ed.). https://doi.org/10.1037/0000165-000

Australian Research Data Commons. (2026, August 14). *Making software citable.* Retrieved August 19, 2026, from https://ardc.edu.au/resource/making-software-citable/

Borenstein, M., Hedges, L., Higgins, J., & Rothstein, H. (2014). *Comprehensive meta-analysis* (Version 3.3.070) [Computer software]. Biostat. https://meta-analysis.com/

Brady, R., Spring, A., Huang, A., Banihirwe, A.& Bell, R. (2023). *climpred: Verification of weather and climate forecasts* (Version 2.4.0) [Computer software]. Zenodo. https://doi.org/10.5281/zenodo.10094242

Chue Hong, N. P., Allen, A., Gonzalez-Beltran, A., de Waard, A., Smith, A. M., Robinson, C., Jones, C., Bouquin, D., Katz, D. S., Kennedy, D., Ryder, G., Hausman, J., Hwang, L., Jones, M. B., Harrison, M., Crosas, M., Wu, M., Löwe, P., Haines, R., … Pollard, T. (2019a). *Software Citation Checklist for Authors* (Version 0.9.0). Zenodo. https://doi.org/10.5281/ZENODO.3479199

Chue Hong, N. P., Allen, A., Gonzalez-Beltran, de Waard, A., Smith, A. M., Robinson, C., Jones, C., Bouquin, D., Katz, D. S., Kennedy, D., Ryder, G., Hausman, J., Hwang, L., Jones, M. B., Harrison, M., Crosas, M., Wu, M., Löwe, P., Haines, R., … Pollard, T. (2019b). *Software Citation Checklist for Developers* (Version 0.9.0). Zenodo. https://doi.org/10.5281/ZENODO.3482769

Di Cosmo, R. & Zacchiroli, S. (2017, September 25-29). *Software Heritage: Why and how to preserve software source code* [Paper presentation]. iPRES 2017: 14th International Conference on Digital Preservation, Kyoto, Japan.

Digital Scholar. (2026). *Zotero* (Version 10.0.0) [Computer software]. https://www.zotero.org/

Druskat, S., Spaaks, J. H., Chue Hong, N., Haines, R., Baker, J., Bliven, S., Willighagen, E., Pérez-Suárez, D., & Konovalov, O. (2021). *Citation File Format* (Version 1.2.0) [Schema]. https://doi.org/10.5281/zenodo.5171937

The EndNote Team. (2026). *EndNote* (Version 2025.3.1) [Computer software]. Clarivate. https://endnote.com

European Organization For Nuclear Research & OpenAIRE. (2013). *Zenodo.* CERN. https://doi.org/10.25495/7GXK-RD71

Google Scholar. (n.d.). *Google Scholar entry for "Data structures for statistical computing in Python".* Retrieved August 26, 2026, from https://scholar.google.com/scholar?hl=en&as_sdt=0,5&q=Data+structures+for+statistical+computing+in+Python.&btnG=

Google Scholar. (n.d.). *Google Scholar entry for "pandas-dev/pandas: Pandas 1.0. 5".* Retrieved August 26, 2026, from https://scholar.google.com/citations?view_op=view_citation&hl=en&user=EvBzFF8AAAAJ&citation_for_view=EvBzFF8AAAAJ:2osOgNQ5qMEC

Google Scholar. (n.d.). *Google Scholar entry for "scores: A Python package for verifying and evaluating models and predictions with xarray".* Retrieved August 26, 2026, from https://scholar.google.com/citations?view_op=view_citation&hl=en&user=pqb0CE8AAAAJ&citation_for_view=pqb0CE8AAAAJ:YOwf2qJgpHMC

Google Scholar. (n.d.). *Google Scholar entry for "scores: Metrics for the verification, evaluation and optimisation of forecasts, predictions or models".* Retrieved August 26, 2026, from https://scholar.google.com/citations?view_op=view_citation&hl=en&user=pqb0CE8AAAAJ&citation_for_view=pqb0CE8AAAAJ:4JMBOYKVnBMC

Hahnel, M. (2022). Figshare – A place where open academic research outputs live. In J. Schöpfel & V. Rebouillat (Eds.), *Research Data Sharing and Valorization: Developments, Tendencies, Models* (pp. 175-196). John Wiley & Sons. https://doi.org/10.1002/9781394163410.ch10

Howison, J., & Bullard, J. (2016). Software in the scientific literature: Problems with seeing, finding, and using software mentioned in the biology literature. *Journal of the Association for Information Science and Technology*, *67*(9), 2137–2155. https://doi.org/10.1002/asi.23538

Katz, D. S., & Chue Hong, N. P. (2024). Special issue on software citation, indexing, and discoverability. *PeerJ Computer Science, 10*, Article e1951. https://doi.org/10.7717/peerj-cs.1951

Katz, D. S., Chue Hong, N. P., Clark, T., Muench, A., Stall, S., Bouquin, D., Cannon, M., Edmunds, S., Faez, T., Feeney, P., Fenner, M., Friedman, M., Grenier, G.,

Harrison, M., Heber, J., Leary, A., MacCallum, C., Murray, H., Pastrana, E., … Yeston, J. (2021). Recognizing the value of software: A software citation guide. *F1000Research*, *9*, 1257. https://doi.org/10.12688/f1000research.26932.2

Leeuwenburg, T., Loveday, N., Ebert, E. E., Cook, H., Khanarmuei, M., Taggart, R. J., Ramanathan, N., Carroll, M., Chong, S., Griffiths, A., & Sharples, J. (2024). scores: A Python package for verifying and evaluating models and predictions with xarray. *Journal of Open Source Software, 9*(99), 6889. https://doi.org/10.21105/joss.06889

Leeuwenburg, T., Loveday, N., Ebert, E. E., Cook, H., Khanarmuei, M., Taggart, R. J., Ramanathan, N., Carroll, M., Chong, S., Griffiths, A., Sharples, J., Shrestha, D., Bishop, S., Fisher, A. J., Jinghan, F., Bluett, L., & Squire, D. T. (2025). *scores: Metrics for the verification, evaluation and optimisation of forecasts, predictions or models* (Version 2.2.0) [Computer software]. Zenodo. https://doi.org/10.5281/ZENODO.16431993

Leeuwenburg, T., Loveday, N., Ramanathan, N., Chong, S., Taggart, R. J., Shrestha, D., Khanarmuei, M., Cook, H., Bluett, L., Ebert, E. E., Carroll, M., Trotta, B., Sharples, J., Bishop, S., Squire, D. T., Griffiths, A., Pagano, T. C., Fisher, A. J., Mandelbaum, T., … Wu, X. (2026a). *scores: Metrics for the verification, evaluation and optimisation of forecasts, predictions or models* (Version 2.5.0) [Computer software]. Zenodo. https://doi.org/10.5281/ZENODO.18638494

Leeuwenburg, T., Loveday, N., Ramanathan, N., Chong, S., Taggart, R. J., Shrestha, D., Khanarmuei, M., Cook, H., Bluett, L., Ebert, E. E., Carroll, M., Trotta, B., Sharples, J., Bishop, S., Squire, D. T., Griffiths, A., Pagano, T. C., Fisher, A. J., Mandelbaum, T., … Morrison, O. M. (2026b). *scores: Metrics for the verification, evaluation and optimisation of forecasts, predictions or models* (Version 2.6.0) [Computer software]. Zenodo. https://doi.org/10.5281/ZENODO.21403944

McKinney, W. (2010). Data structures for statistical computing in Python. *Proceedings of the 9th Python in Science Conference*, 56–61. https://doi.org/10.25080/majora-92bf1922-00a

The pandas development team. (2026). *pandas-dev/pandas: Pandas* (Version 3.0.5) [Computer software]. Zenodo. https://doi.org/10.5281/ZENODO.21500199

Reback, J., jbrockmendel, McKinney, W., Van Den Bossche, J., Augspurger, T., Cloud, P., Hawkins, S., gfyoung, Sinhrks, Roeschke, M., Klein, A., Terji Petersen, Tratner, J., She, C., Ayd, W., Hoefler, P., Naveh, S., Garcia, M., Schendel, J., …

Kaiqi Dong. (2021). *pandas-dev/pandas: Pandas 1.3.1* (Version 1.3.1) [Computer software]. Zenodo. https://doi.org/10.5281/ZENODO.5136416

Reback, J., McKinney, W., jbrockmendel, Van den Bossche, J., Augspurger, T., Cloud, P., gfyoung, Sinhrks, Klein, A., Hawkins, S., Roeschke, M., Tratner, J., She, C., Ayd, W., Terji Petersen, MomIsBestFriend, Garcia, M., Schendel, J., Hayden, A., … Winkel, M. (2020). *pandas-dev/pandas: Pandas 1.0.5* (Version 1.0.5) [Computer software]. Zenodo. https://doi.org/10.5281/ZENODO.3898987

Smith, A. M., Katz, D. S., Niemeyer, K. E., & FORCE11 Software Citation Working Group. (2016). Software citation principles. *PeerJ Computer Science*, *2*, Article e86. https://doi.org/10.7717/peerj-cs.86

How Being Terrified of Code Helped Me Get Girls to Love It

Gwyneth De Guia[1,2]

[1]Girls' Programming Network, Brisbane, QLD, Australia

[2]Robogals, Brisbane, QLD, Australia

Girls don't leave STEM once; they leave three times. First, in primary school when "I'm bad at maths" becomes a personality (EngineeringUK, 2023). Second, in high school when they drop senior tech for "something more cool" (González-Pérez et al., 2022). Third, in first-year engineering when the boyish culture and difficulty finally wear them down (Raabe et al., 2019). I'm a mechatronics engineer who is terrified of coding and still somehow convinces girls to give it a chance. By being openly honest about my fear, connecting code to real hobbies, and normalising failure out loud, I turn anxiety into curiosity - and keep more girls in the room.

This talk is for Python educators, mentors, and community organisers who care about women in tech but keep watching girls disappear at every stage. I'll share what I've seen as an educator and volunteer across schools and outreach programs: three crucial dropout points for girls - primary school, senior subject choices, and first-year engineering - and how my "I'm terrified of coding" honesty weirdly helps.

We'll look at why coding felt impossible for me: no safe space to fail, classes that only praised people who "got it first go," and an unspoken rule that struggling meant you didn't belong. I'll show how I now use that experience to design environments where girls are allowed to make mistakes, laugh about it, and try again.

You'll hear stories of turning K-pop obsessions, arts and craft, and even sport into coding gateways, and how linking Python to real hobbies flips girls from "this isn't for me" to "wait, that's kind of cool." We'll break down the three dropout moments with concrete tactics you can apply: how to support curious primary-schoolers, how to keep tech-strong teens from drifting away, and how to make first-year/early-career spaces less hostile and more human. Attendees will leave with practical ideas to make their classrooms, workshops, and meetups places where girls don't just show up - they stay.

https://doi.org/10.5281/zenodo.22784195

Resources

Presentation recording (YouTube): https://www.youtube.com/watch?v=rwc9EmYxpcE

Conference page: https://2026.pycon.org.au/schedule/CGG8LN/

References

EngineeringUK. (2023). Rapid evidence review: Interventions to increase girls' aspirations for engineering and technology careers. EngineeringUK. https://www.tomorrowsengineers.org.uk/media/5ofhrq3x/rapid-evidence-review-girls-stem-aspirations-final.pdf

González-Pérez, S., Martínez-Martínez, M., Rey-Paredes, V., & Cifre, E. (2022). I am done with this! Women dropping out of engineering majors. Frontiers in Psychology, 13. https://doi.org/10.3389/fpsyg.2022.918439

Raabe, I. J., Boda, Z., & Stadtfeld, C. (2019). The Social Pipeline: How Friend Influence and Peer Exposure Widen the STEM Gender Gap. Sociology of Education, 92(2), 105–123. https://doi.org/10.1177/0038040718824095

https://doi.org/10.5281/zenodo.22784195

Your Research Code Works... Until It Doesn't: Rethinking Reproducibility

Zara Hassan[1]

[1]Australian National University, Acton, ACT, Australia

Reproducibility remains a persistent challenge in research software, even when code and data are openly shared [1]. In real-world research projects, reproducibility can break in subtle ways through missing dependencies, unclear workflows, undocumented steps, or computational environments that no longer work. These issues rarely appear all at once; rather, they accumulate gradually as research software evolves.

This talk introduces **Reproducibility Debt (RpD)** as a way of understanding how reproducibility issues accumulate over time [1]–[3]. Reproducibility Debt is not simply the result of poor software development practices; it can also emerge from practical constraints such as time pressure, experimentation, changing requirements, and shifting research goals. While reproducibility is often framed primarily as a technical problem addressed through tools such as Docker, Conda, or improved documentation, research on RpD highlights a broader, multifaceted challenge involving interacting technical, organisational, and human factors [2], [3].

The talk presents a structured view of factors contributing to RpD and demonstrates how reproducibility can be approached as something to identify, assess, and manage throughout the research software lifecycle rather than something to fix at the end [4]. It also introduces a lightweight approach for identifying and tracking reproducibility risks based on these contributing factors. Through this perspective, research software teams can make reproducibility risks more visible, reason about trade-offs, and make informed decisions about when and how those risks should be addressed.

https://doi.org/10.5281/zenodo.22784195

Resources

Presentation recording (YouTube): https://www.youtube.com/watch?v=OmCEDTX24Tk

Conference page: https://2026.pycon.org.au/schedule/ZPGW8C/

References

[1] Z. Hassan, C. Treude, M. Norrish, G. Williams, and A. Potanin, "Reproducibility Debt: Challenges and Future Pathways," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering (FSE 2024)*, Porto de Galinhas, Brazil, 2024, pp. 462–466.

[2] Z. Hassan, C. Treude, M. Norrish, G. Williams, and A. Potanin, "Characterising reproducibility debt in scientific software: A systematic literature review," *Journal of Systems and Software*, vol. 222, Art. no. 112327, 2025.

[3] Z. Hassan, C. Treude, G. Williams, M. Norrish, and A. Potanin, "Reproducibility Debt in Scientific Software," in *Companion Proceedings of the 2025 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity (SPLASH Companion '25)*, Singapore, 2025, pp. 50–51.

[4] Z. Hassan, C. Treude, G. Williams, M. Norrish, and A. Potanin, "Managing Reproducibility Debt in Scientific Software: A Practical Framework," in *Proceedings of the 2026 1st International Workshop on Software Engineering and Research Software*, 2026, pp. 23–24.

https://doi.org/10.5281/zenodo.22784195

The 20-Year Port: from PhD project to polished professional Python packaging via Research Software Engineering and AI coding agents

Paul Leopardi[1,2]

[1]ACCESS-NRI, Australian National University, Acton, ACT, Australia

[2]University of New South Wales, Sydney, NSW, Australia

The application of Research Software Engineering principles can complement AI coding agents in preserving both academic rigour and thorough testing throughout the process of porting research code, resulting in a reliable, well-tested and well-documented library.

In 2006, my mathematics PhD project [1] resulted in a highly specialized geometrical MATLAB toolbox that over the years has been used in diverse scientific and engineering applications. In 2013, at a SciPy conference, one of the attendees asked when the toolbox would be ported to Python.

On and off over a period of 20 years, I took that toolbox on a journey from an open source package within a proprietary mathematical software ecosystem to its current state as a polished, production-grade Python library. In 2024, motivated by my deepening understanding of the diversity of its applications, I first modernized the toolbox itself, including automating the testing of help examples. The rapid rise of Large Language Models as coding agents [2] then prompted me to undertake the long-delayed Python port, using coding agents to translate the MATLAB to Python [3,4], to convert the help text into docstrings and doctests, and to organize and improve the documentation.

https://doi.org/10.5281/zenodo.22784195

Resources

Presentation recording (YouTube): https://www.youtube.com/watch?v=EinnQT0ptRY

Conference page: https://2026.pycon.org.au/schedule/8LEEB7/

References

[1] P. Leopardi, "A partition of the unit sphere into regions of equal area and small diameter," *Electronic Transactions on Numerical Analysis*, vol. 25, pp. 309–327, 2006.

[2] Palo Alto Networks, "2024 State of Cloud Native Security Report," Palo Alto Networks, Tech. Rep., 2024.

[3] B. Smith, "From Matlab to Lambda: Transforming Structural Engineering Research Tools," PyCon AU 2025, 2025. https://2025.pycon.org.au/program/8DZTCP/

[4] M. Croucher and T. Takeuchi, "MATLAB + Agentic AI: The Workflow That Actually Works," 2026. https://blogs.mathworks.com/matlab/2026/01/26/matlab-agentic-ai-the-workflow-that-actually-works/

https://doi.org/10.5281/zenodo.22784195

Privacy data encryption & deletion while preserving analytical integrity

XiaoHan Li[1]

[1]Xebia Data, Amsterdam, Noord-Holland, The Netherlands

Privacy regulations around the world impose strict requirements on how organisations handle personal data (such as the Australian Privacy Act [1], the EU's GDPR [2], US's HIPAA [3] & CCPA [4], etc). Among the hardest to implement in a data pipeline are: protecting sensitive data from breaches, and erasing an individual's personal data on request. Finding and deleting every copy of sensitive data is cumbersome, error-prone, and breaks referential and analytical integrity. Meanwhile, sensitive data sitting in plaintext is exposed the moment a breach occurs.

The author proposes practical implementation solutions to integrate the crypto-shredding pattern in analytical data platform's pipeline architecture, enabled by the open-source Python library GDPR-officer [5] created by the author.

- Sensitive fields are encrypted with a unique encryption key per data subject (e.g. customer), before data is loaded into the platform, restricting access and protecting it during leaks.
- Analytical sensitive fields are generalised to enable analytics use, while protecting the original value.
- Encryption keys are stored separately outside the data platform, ensuring sensitive data stays unreadable even when the data platform is breached.
- Data erasure becomes a single operation: delete the encryption key, and the respective customer's data is rendered permanently unreadable across all tables, without modifying or scanning any data, preserving analytics and referential integrity.
- Decrypts sensitive data under authorisation for any customer whose key is maintained.

The author introduces the architecture, the cryptographic foundations of the Python library, and practical integration into Python ingestion pipelines as well as managed ingestion pipelines. She further proposes a governance framework for encryption key management, rotation, retention, and role-based policies. A live demo of the library is shown during PyCon AU 2026.

https://doi.org/10.5281/zenodo.22784195

Resources

Presentation recording (YouTube): https://www.youtube.com/watch?v=1Akx-O3elxQ

Conference page: https://2026.pycon.org.au/schedule/PZTFZF/

References

1. OAIC. (2019, July 22). Chapter 11: APP 11 security of personal information. Office of the Australian Information Commissioner. https://www.oaic.gov.au/privacy/australian-privacy-principles/australian-privacy-principles-guidelines/chapter-11-app-11-security-of-personal-information

2. European Union. (2016). General Data Protection Regulation. https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679

3. Federal Register. (2025, January 6). HIPAA Security Rule to Strengthen the Cybersecurity of Electronic Protected Health Information. Federal Register. https://www.federalregister.gov/documents/2025/01/06/2024-30983/hipaa-security-rule-to-strengthen-the-cybersecurity-of-electronic-protected-health-information

4. State of California Department of Justice. (2024). California Consumer Privacy Act (CCPA). State of California - Department of Justice - Office of the Attorney General. https://oag.ca.gov/privacy/ccpa

5. XiaoHan Li. (2026). GitHub - xiaohan-data/gdpr-officer. GitHub. https://github.com/xiaohan-data/gdpr-officer

https://doi.org/10.5281/zenodo.22784195

Research Software Stories: Four Years of Lessons in Sustainable Science

Paula Andrea Martinez[1]

[1]Australian Research Data Commons, University of Queensland, Brisbane, QLD, Australia

Python powers the world's most critical research, yet the software behind the science often remains hidden and under supported. This talk distils four years of insights from Research Software Engineering Stories to explore the real world motivators of the people building Australia's most impactful research tools. Attendees will be motivated to transition from fragile scripts to sustainable and citable reusable software infrastructure that endure long after the code is working for a specific use case.

As Python continues to dominate the research landscape across fields from radio astronomy to biosecurity. The fundamental challenge has shifted from "how do we code?" to "how do we sustain code?". This presentation deep dives into the real motivators of the people building and sustaining research software by distilling insights from four years of interviews with leading Australian Research Software Engineers.

By exploring case studies of successful transitions where individual researcher scripts evolved into community validated Python packages like DStools in Astronomy or whereabouts in spatial data we identify the common threads of success. The session highlights technical best practices for high impact software including architectural patterns and modular design for long term maintenance.

Attendees will walk away motivated to transform their Python scripts into tools that other scientists can easily use and cite and all participants are invited to engage with and expand the series. For developers and engineers this talk provides a clearer understanding of the Australian RSE landscape and the career pathways available within research intensive environments. Finally we highlight the various awards and networking opportunities specifically designed to support and professionalise the work of Research Software Engineers in science.

https://doi.org/10.5281/zenodo.22784195

Resources

Presentation recording (YouTube): https://www.youtube.com/watch?v=OLujgkCsZek

Conference page: https://2026.pycon.org.au/schedule/J9DDEA/

https://doi.org/10.5281/zenodo.22784195

Launching Rockets with Python

Freddy Mcloughlan[1], Amber Taylor[1], Jennifer L Palmer[1], Timothy Wiley[1]

[1]RMIT University, Melbourne, VIC, Australia

We're not joking. In 2025 and 2026 our university rocketry team launched several high-powered 12ft tall competition rockets with a custom Python-based process manager framework that controlled all of our rocket telemetry and ground control systems. We called it the Ground Control Station, or GCS for short.

Freddy led the creation of the GCS software to compete at the 2025 International Rocket Engineering Competition (IREC) with the intention of winning the Rocket Telemetry Visualisation Sub-Category. Amber joined the project in mid-2025 and upgraded the GCS for IREC 2026 where it won first place in Live Telemetry Visualisation!

We have our own custom flight computer with its own custom firmware and dual LoRa/LAN networking setup. We built our visualisation system from the ground up for reliability, scalability and deployability. All in our Python process manager.

The Ground Control Station (GCS) is a computer control system for GSE control, avionics communication, and data visualisation. The core of the GCS is a single computer running Student Researched And Designed (SRAD) software with SRAD LoRa radio hardware peripherals. All OSI layers in our networking stack above the physical protocol are SRAD for use with our Australis (avionics) ecosystem. The software converts raw serial input from physical radio interfaces into human-readable output for efficient system monitoring by the GCS operator and visualisations for observers. We use a WebSocket and a protocol buffer based IPC API to communicate with our GCS services. Our web frontend is fully SRAD aside from industry-standard libraries. The GCS operator can see if any system is performing sub-optimally via alert and warning readouts, so they can make an informed GO/NO-GO call quickly. Spectators and other team members have access to several different views detailing all telemetry from both the GSE and avionics systems

https://doi.org/10.5281/zenodo.22784195

This project was built using the following tools, languages and systems.

- Radio communication:
  - LoRa with both COTS and SRAD hardware
- Multithreaded data ingestion server
  - Written in C++
  - Built with ZeroMQ for IPC communication
  - IPC Data serialisation with Google's Protocol Buffers
- Multithreaded CLI based process manager
  - Written in Python
  - Includes a device emulator for internal system tests that attaches from the hardware layer to create a fake unix device file at /dev/

Cool fact: Our GCS runs at less than 1% CPU utilization on a Raspberry Pi 5 during regular use.

Authorship Statement

FM and AT designed and implemented the project. JLP and TW supervised the work.

Resources

Presentation recording (YouTube): https://www.youtube.com/watch?v=ztSzPvTBilU

Conference page: https://2026.pycon.org.au/schedule/9KBBEQ/

https://doi.org/10.5281/zenodo.22784195

Ship Reliable RAG: Evals That Grow with Your Retrieval Complexity

Ravi Vats

RAG applications come in many shapes. Some start as simple keyword searches over historical records. Others layer on semantic embeddings, hybrid retrieval (semantic + keyword/lexical searches), reranking, and increasingly sophisticated prompting. But regardless of where your system sits on that spectrum, the question is the same: how do you know it actually works?

This talk presents a level-by-level framework for building and evaluating RAG systems in Python, using a real AI-assisted ticket resolution application as the running example. Starting from the simplest possible retrieval setup (lexical matching with BM25 over historical tickets), the talk introduces evaluations early, explains why they matter, and then walks through growing both the RAG application and its eval suite in lockstep through four levels of complexity: lexical retrieval, semantic embedding-based retrieval, hybrid retrieval with log filtering using Drain3, and hybrid retrieval with reranking and context window tuning. At each level, the talk introduces only the evaluation metrics that become necessary at that stage, so the audience builds intuition for which evals matter and when.

Along the way, the talk covers practical topics including how to compose simple metrics like faithfulness, context precision, and recall into an experiment harness for tuning reranking strategies (comparing Weighted RRF, plain RRF, Z-score normalization, and Min-Max normalization), and for detecting context rot. At the end, the talk covers how to avoid common pitfalls in LLM-based evaluation such as positional bias and numerical scoring scales.

This talk is for Python developers and data scientists who are building or planning to build reliable and robust RAG applications and want a practical mental model for when to introduce which evaluations as their system grows in complexity.

The talk begins by introducing the use case: an AI-assisted IT support ticket resolution system that takes a ticket's title, description, comments, and log files as input and produces a one-shot resolution.

This application serves as the running example throughout, but the framework applies to any domain where RAG is used to ground LLM responses in a knowledge base (e.g. customer support chatbots, travel agency chatbots, internal knowledge assistants).

https://doi.org/10.5281/zenodo.22784195

Before diving into the levels, the talk covers a short primer on why evaluations matter for LLM applications. This includes the three stages at which evals become relevant: during feature development (20 to 100 targeted examples per workflow), just before production deployment (regression testing), and on live traffic after launch (monitoring resolution rates, knowledge drift, and user feedback). The goal is to establish that evals grow into an ongoing practice that evolves alongside the system, rather than a task you finish once.

Level 1: Lexical retrieval. The simplest retrieval approach: using BM25 keyword matching to find historically resolved tickets that resemble the incoming query. This is a viable first "vertical slice" that can be deployed for basic functionality. Evals at this stage focus on search query quality and hit rate. The audience learns how even a simple retrieval system benefits from having a ground truth test set.

Level 2: Semantic retrieval. Once keyword matching starts missing tickets that are worded differently but mean the same thing, embedding-based vector search becomes the natural next step to capture meaning beyond exact keyword matches. This stage introduces retrieval-specific evaluation metrics: precision@k, recall@k, and contextual recall. The talk shows how these metrics help catch failure cases that lexical search alone would miss or failure cases that semantic search alone would not be able to handle.

Level 3: Hybrid (semantic + lexical) retrieval with log filtering. When neither approach alone is enough, combining semantic and lexical search gives the best of both, plus using the Drain3 library for intelligent log filtering and preprocessing before ingestion. With two retrieval sources feeding the context window, the evaluation scope needs to expand. Document ingestion evals (BLEU/ROUGE for LLM-preprocessed text, chunking tuning via retrieval metrics) become relevant. Retrieval evals like embedding spread and generation evals like faithfulness, completeness, tonality, and safety/refusal checking are introduced here, because more retrieved context from multiple sources gives the LLM more room to produce inconsistent or ungrounded responses.

Level 4: Hybrid retrieval with reranking and context window tuning. As more sources feed results into a single answer, the order they arrive in starts to matter, so the final level adds reranking to merge and prioritize results from multiple retrieval sources, and explores context window limits. This is where the core idea of the talk comes together: composing simple eval metrics (faithfulness, context precision, recall, answer relevancy, safety) into a systematic experiment harness, then sweeping across reranking strategies and weight configurations to find optimal settings. The talk walks through a concrete comparison of four approaches (Weighted RRF with tuned weights, plain RRF, Z-score normalization, Min-Max normalization) using a RAGAS-powered

https://doi.org/10.5281/zenodo.22784195

eval suite and a Streamlit dashboard. It also covers context rot evaluation to determine the maximum context size before retrieval quality degrades. Augmentation metrics like NDCG are introduced at this stage.

The talk closes with key learnings from production: designing for testability from day one, preferring binary or categorical grading scales over numerical Likert scales for LLM-based evaluation, and accounting for known LLM biases (positional, verbosity, self-bias) when building eval pipelines.

Resources

Presentation recording (YouTube): https://www.youtube.com/watch?v=7b3wm5wyqfo

Conference page: https://2026.pycon.org.au/schedule/XWLM8L/

https://doi.org/10.5281/zenodo.22784195