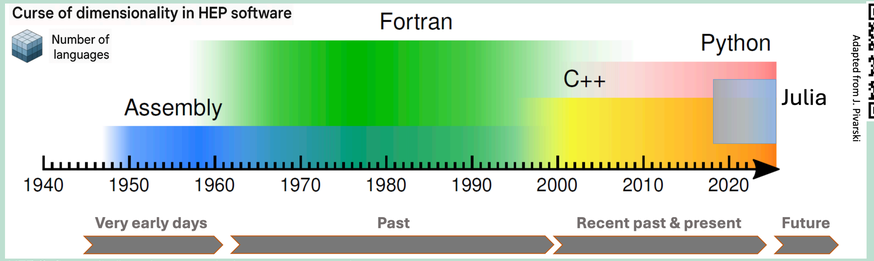
Advancing High Energy Physics Data Analysis with Julia: A Case for JuliaHEP

Ianna Osborne, Princeton University

For the past 25 years, the High-Energy Physics (HEP) community has steadily adopted Python as its primary language for data analysis, supported by compiled-backend libraries like NumPy (*Harris et al., 2020)* and Awkward Array, along with other Python tools like Uproot for I/O. However, the Julia programming language offers a compelling alternative, addressing the two-language problem with C++-comparable performance and Python-like ease of use. JuliaHEP, is an initiative to leverage Julia for HEP data analysis, outlining its advantages, ongoing developments, and integration with existing Python-based tools.

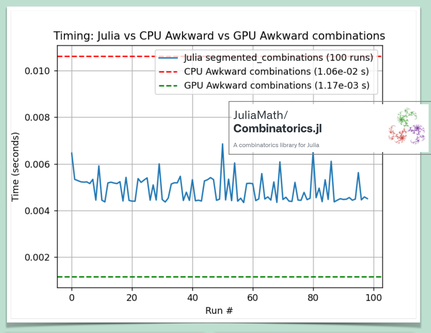


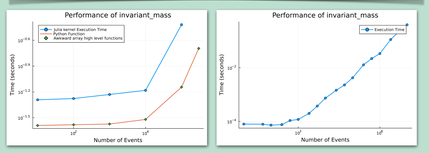
Adapted from J. Piwarski

HEP Software: it all revolves around a language, or several

Python provides ease of use and a rich scientific ecosystem, but it struggles with performance for large-scale analyses.
C++, on the other hand, offers speed but comes with increased complexity and slower development cycles.
Julia, a modern language designed for scientific computing, promises the best of both worlds: high-level expressiveness with near-native execution speed.

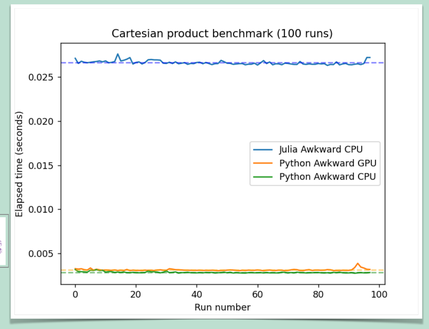
Performance benefits for HEP computations





Awkward Array, a key library for handling complex and jagged HEP data, is integrated into Julia workflows.
By using Julia native type that provides similar capabilities, we compared performance and usability against the traditional Python stack.

Julia integrates well with Python-based tools such as Awkward Array, and vice versa.



Comparing Awkward Array combinatorics in Julia and Python with C++ and GPU backends, highlight cases where Julia can offer significant speedup with less boilerplate code.

JuliaHEP: an emerging set of tools and libraries designed to facilitate HEP data analysis in Julia

Julia's strengths: just-in-time (JIT) compilation, multiple dispatch, and GPU acceleration.

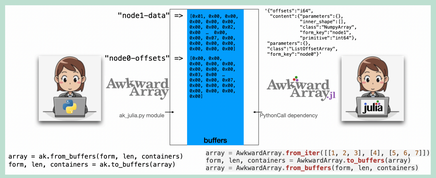
https://github.com/JuliaHEP

- Speed of code execution
- Designed for interactive use
- Composability, leading to highly reusable code that is easy to maintain
- Package management: built in state-of-the-art package manager
- Ease of integration with other languages

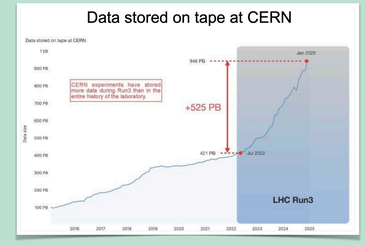
Julia Introduction to Python HEP Community

Physicists can gradually adopt Julia without abandoning Python-based tools, thanks to interlanguage operability.

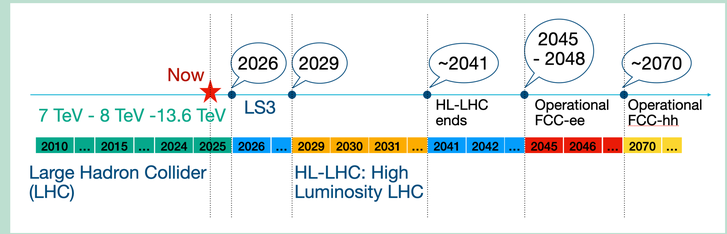
Physicists are using Awkward Array in Python and data format conversion is the hardest part of language boundary-hopping.



Sharing Awkward Array data structures between Python and Julia to encourage the Python users to run their analysis both in an eco-system of their choice and in Julia



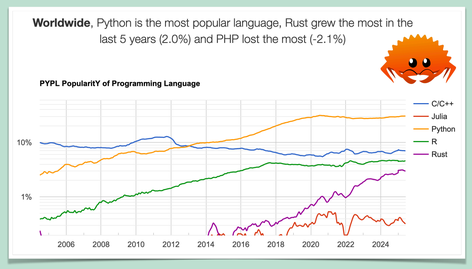
the WLCG Workshop 2025



Case studies from CMS and other experimental collaborations illustrate Julia's real-world applicability in tasks like event processing, data analysis, and simulation (see upcoming JuliaHEP 2025 workshop).

Scikit-HEP and PyHEP: Scientific Python HEP Community

Julia's potential for parallel computing and its built-in support for GPUs can provide an efficient pathway for scaling HEP computations.



SciPy 2025