CW25 - 2025-05-15

Hack Day Idea: - HP1-CW2CC

Hackday Idea Proposer
Matt Gillman

This document should be used to capture the information for a Hack Day Idea.

Hack Day Idea title (provisional)

Python to C++ converter

Context / Research Domain

Potentially unlimited!!

Problem

I have seen data which indicates Python is a lot more resource-hungry than C++. However, I have not performed a proper literature review to establish how true this may be. Regardless, C++ is certainly a lot faster than Python (again, it would be good to have some indicative benchmark figures here). Many programmers know Python, but not C++.

I therefore think time and resources could (potentially) be achieved with this tool.

Example article about this: https://www.efinancialcareers.com/news/2023/06/which-programming-language-uses-the-most-energy

which suggests Python is evil.

Solution

This is initially a project just for fun. How easy is it to generate C++ "Hello, world!" from the Python equivalent? And go on from there.

C++ has massively evolved over the last couple of decades. It is difficult to keep up-to-date with the constant changes in the language. Therefore I suggest a version be chosen, perhaps C++11, 14 or 17, to convert to.

Commented [1]: Tuppence worth - C++20 would be the best target for lots of reasons, but at minimum C++11 since you can sidestep the need for type inference by requiring variable initialisation and declaring variables using the auto keyword

Commented [2]: Thanks

The idea is that C++ source code would be generated from the Python code. This would then need to be compiled and run.

*Please note that this is **not** extending Python to use C++ modules, or embedding Python within C++ (or vice versa).*

To do this properly, we would need to define the architecture, libraries, etc. Maybe that will happen one day. But this hack day idea is just to have some fun, and maybe to capture ideas about what the architecture would require, etc. Perhaps to be captured in a blog post or web page.

A quick (non-exhaustive) Google search suggests the existing converters all use AI. We will NOT be using AI on this project.

Perhaps call it cpppy or pycpp? Assuming the desired name is available!

I have not undertaken any research into what the demand may (or may not) be for this idea. It's just for fun (at this stage!)

Commented [3]: It's possible to build a transpiler atop python's abstract syntax tree (AST), a project I work on has done similar for a very limited CUDA subset.

https://github.com/FLAMEGPU/FLAMEGPU2/blob/master/swig/python/codegen/codegen.py

(A bit unwieldy to maintain though)

Commented [4]: Looks like it's part of SWIG

Commented [5]: No, it's just used within our SWIG interfaced C++ library.
ast is core python
https://docs.python.org/3/library/ast.html

Converting Python to C++ would result in that program needing fewer HPC or other resources. And less likely to be kicked off a job queue because it's taken too long

A couple of less technical opportunities on the team:

- *Research benchmarking of C++ vs Python*
- *Research demand for the tool*
- *writing/contributing to documentation*
- *Project management*

Diagrams / Illustrations

You can include diagrams in this section. Please ensure you have the right to use the image(s), and include an attribution if applicable.

Couple of related things that I've built:

- https://github.com/sparkslabs/microbit-prototype/tree/master/compiler - A python to C++ compiler for the original microbit prototype that we took into schools
- https://github.com/sparkslabs/pyxie - A second iteration I'm done of this - which aims to solve some of the side-steps the former made.