ON APPLICATION OF OPEN-SOURCE EDUCATION MODELS TO NUCLEAR ENGINEERING

SHRIWISE, P.[1] ROMANO, P.K.[1]

Argonne National Laboratory
Lemont, IL USA

[1]Argonne National Laboratory, 9700 S Cass Ave, Lemont, IL 60439, United States

Email contact of corresponding author: pshriwise@anl.gov

To support the pace of innovation and next generation reactor design, research institutions have found the ability to collaborate effectively critical to their success in making progress. These needs have been well-met by the use of open-source programming models, resulting in the development of quality open-source codes. With many entities on the verge of licensing next generation reactor designs, an indication for demand of new nuclear engineers for these facilities is apparent. In addition to fostering collaboration, open-source codes can also provide a way to attract, educate, and train the next generation of engineers on the core concept of reactor physics and design for the peaceful production of nuclear energy to meet the growing energy demands across the globe.

Models and Tools for Education and Training

The number of organizations that provide workshops for training in software development skills has become an industry of its own across the world as software development skills continue to proliferate to new industries. Many of these programs are paid models, involving proprietary learning tools and curricula. However, non-profit organizations have applied the same open-source models used to create the tools they teach to lesson planning and curriculum development. One such organization is The Carpentries, which provides software skill development workshops dubbed software carpentry [1,2]. These workshops feature accessible lessons on critical software development tools and concepts using a hands-on code-along teaching model -- primarily focusing on interactions with the shell, version control, and a high-level interpreted programming language (typically Python or R). These modules and lessons are developed in a modular fashion with the ability to swap out different learning objectives and/or languages so workshops can be catered to a specific institution or learning group. Great care is taken to ensure that the installation process of different tools goes smoothly for learners and that learners and instructors will be interacting with the same inputs and outputs.

The Carpentries uses the same open-source community based approaches to lesson development and maintenance as those used to develop the tools used in lessons. All lesson content is hosted on collaborative open-source platforms where additions, improvements, and modifications to lessons go through public peer review before being incorporated into the canonical lesson set. Separate repositories are created for new workshops to support any customizations the instructors of that workshop may want to apply, with the workshop website and lesson content automatically generated from the repository contents. Any new issues or improvements discovered can then easily be propagated back to the main repository via code change requests. Importantly, these workshops also encourage an open, inclusive environment focused on empowering learners as outlined in their code of conduct. These

ideals should be a high priority in any engineering discipline where gender inequity and lack of diversity is increasingly apparent.

As open-source codes become more mature, they become more and more amenable to successful open-source models like the ones established by The Carpentries. Many codes now feature interfaces that allow for the same level of interactivity and on-the-fly experimentation shown to benefit learners at Software Carpentry workshops [2].

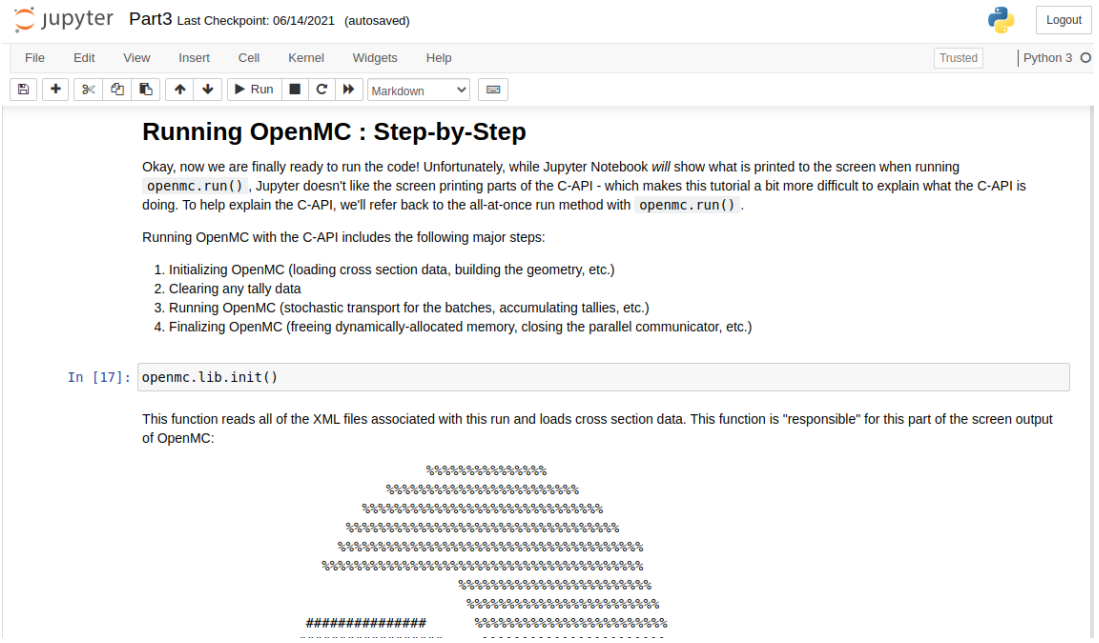

FIG. 1 Image of a jupyter notebook used at an OpenMC workshop. Here, OpenMC is run using the Python interface by directly accessing the compiled library object.

While the concepts and background may be more specific to one domain, the application of these approaches established by The Carpentries are applicable in our field as well. OpenMC workshops, for example, have been slowly approaching the format of the Software Carpentry workshops over time. These workshops now feature lessons performed entirely in Jupyter notebooks [3], an interactive web application used to create text, display equations, and run live code blocks using OpenMC's PythonAPI [4]. As shown in Fig. 1, the lesson narrative and relevant equations are folded directly into the notebook. This narrative is walked through step-by-step by the instructor with opportunities for learner exercises and experimentation along the way. Though the content and focus of these lessons is quite different from those of the software development skills taught by Software Carpentry, these workshops serve as proof of principle that the same practices for instruction and learning environment can be applied.

CHALLENGES

While many open-source codes are amenable to key features for effective learning highlighted by the learning models above, several barriers are present for instructors, code developers, and learners. One incongruous aspect in applying models from software development workshops is that the skills taught are broadly applicable. Prerequisites for those workshops are minimal, requiring only basic knowledge of computer systems whereas workshops geared toward nuclear engineering rely on domain-specific knowledge of physical

concepts and simulation methods. The variety of course descriptions across different institutions and need for knowledge of the language used in the workshop make defining prerequisites more difficult. It is possible to address some of these issues by gathering information prior to the workshop on learner experience regarding the lesson topics, language of choice, and code-specific experience. This information can be applied to the modular lesson structures described above to tailor the workshop to the learners needs.

For developers, one challenge can be provision of an interface conducive to interactive learning. Codes oriented toward large problems are typically compiled, which makes interactive learning difficult as the compilation step and interrupts the rhythm of lessons. At the same time, compiled codes are typically oriented toward high performance computing and the types of problems we intend to train nuclear engineers to solve. In turn, compiled codes tend to be those most often employed in research and analysis in the field. Education using codes applied in research is motivational for learners as they develop transferable skills that will serve them beyond the workshop. However, it is notoriously difficult to fund software engineering endeavors for scientific codes, so the likelihood that development teams have the bandwidth to support the addition of interfaces for interpreted languages purely for education and training purposes is low.

Benefits

The accessibility of open-source codes provide several benefits in the context of education and training. By nature, these codes remove numerous barriers for instructors and workshop locations. Additionally, these tools also allow instructors to engage with students using modern software practices and easily employ other open-source tools that make engagement with the material attractive.

A logistical challenge for instructors is often installation. While the use of open-source codes for learning guarantees that they are accessible, it is not always the case that these codes are easy to install. Configuration and installation of codes can require esoteric changes to a learner's computing environment and knowledge of various build systems. The use of open-source codes allows instructors to provide contained computing environments to learners via services like Docker [5] or cloud-based services to remove any installation issues they may encounter before getting to experience working with the code. This runs counter to the concept that learners with software installed locally on their machine are more likely to use the software after the workshops as it is readily available to them. However, this philosophy is more commonly applied in the context of tools which are very lightweight, installed with native package managers, or are supplied natively with mainstream operating systems. To remedy this, it is beneficial to provide follow-up installation sessions for those interested in having the software on their local machine so they may continue to experiment and learn on their own.

Similar to contributions made to open-source codes themselves, learners who have positive experiences at these workshops are commonly encouraged to return as helpers (in The Carpentries nomenclature) and, eventually, instructors. Open-source management of learning material reduces barriers for engagement for those interested in contributing new content.

Conclusions

In summary, leveraging the history and and lessons learned from other organizations on open-source models for teaching and learning in an inclusive environment could be of great

benefit to the nuclear engineering community at large. A renewed focus in the nuclear engineering community on modern software practices and open-source codes (interface development, documentation, user-friendly design, etc.) sets the stage for exploring application of these learning models in our field to broadly engage and train the next generation of nuclear engineers.

References

[1] The Carpentries https://carpentries.org

[2] WILSON, G. Software Carpentry: Lessons Learned. F1000Research 3 (2016).

[3] KLUYVER, T. RAGAN-KELLEY, B. PÉREZ, F. GRANGER, B. BUSSONNIER, G. FREDERIC, J. KELLEY, K. HAMRICK, et. al. Jupyter Notebooks - a Publishing Format for Reproducible Computational Workflows. IOS Press (2016) 87-90.

[4] ROMANO, P.K., HORELIK, N.E., HERMAN, B.R., NELSON, A.G., FORGET, B., SMITH, K. OpenMC: A state-of-the-art Monte Carlo code for research and development. Ann. Nucl. Energy 82 (2015) 90–97.

[5] MERKEL, D. Docker: Lightweight Linux Containers for Consistent Development and Deployment. Linux Journal **239** (2014) 2.