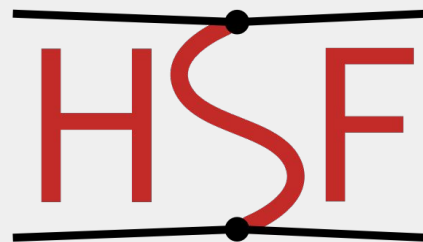


HSF Training WG

Sudhir Malik (University of Puerto Rico Mayaguez),
Kilian Lieret (Ludwig Maximilian University Munich, from June 2022),
Wouter Deconinck (University of Manitoba, from Jan 2022),
Michel Hernandez Villanueva (DESY),
Meirin Oan Evans (University of Sussex, outgoing)

EICUG Software Working Group Meeting
Feb 16, 2022



HEP Software Foundation

Meet Your Conveners



Sudhir Malik

CMS, U. Puerto Rico,
Mayaguez



Kilian Lieret

Belle II, Ludwig Maximilian U.,
Munich



Wouter Deconinck

EIC, U. Manitoba



Michel Hernandez

Villanueva
Belle II, DESY

Goals

The HSF Training Working Group aims to help the research community to provide training in the computing skills needed for researchers to produce high quality and sustainable software. The group works with experiment training groups, HEP initiatives (such as **IRIS-HEP** and **FIRST-HEP**) and organisations like **Software Carpentry** to coordinate training activities.

The group aims to develop a training program that can be pursued by researchers to achieve the level of required knowledge. This ranges from basic core software skills needed by everyone to the advanced training required by specialists in software and computing.

Partner organizations



THE
CARPENTRIES



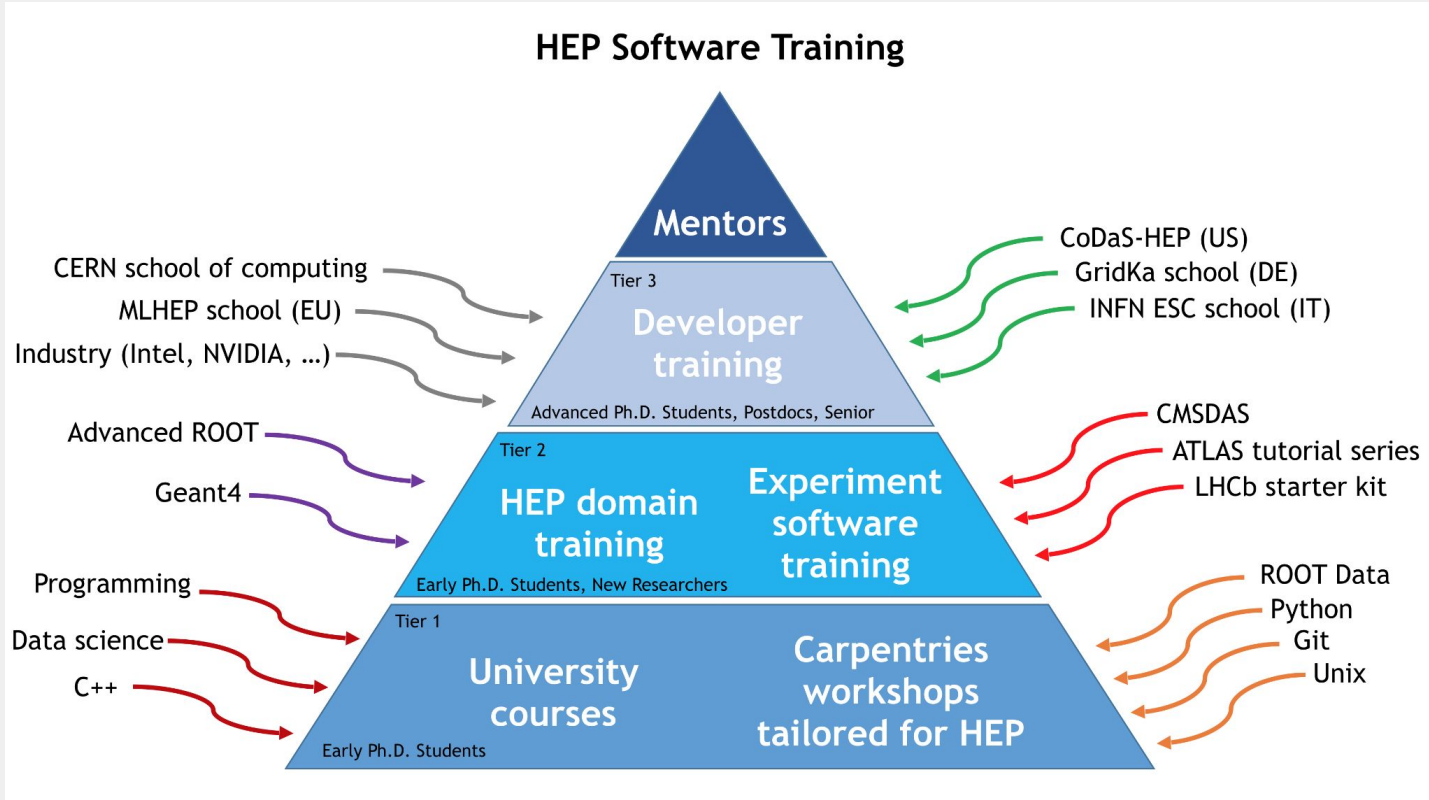
Mission

The software skills required fall into two groups.

Nearly all researchers need **basic programming skills** (Python, C++), **basic software engineering skills** (Unix, git/GitHub/GitLab, continuous integration, performance evaluation), and **skills in the core data tools in HEP** (e.g., the ROOT data format and analysis framework).

More **advanced training** is then needed (with domain examples!) in parallel programming, efficient software implementations, performance optimization, and machine learning and data science tools. A workforce trained in this range of software skills is the critical ingredient from which the solutions to the computing challenges can grow.

Software Training Pyramid



Training Curriculum: The Basics

Basics

The UNIX Shell

A guide through the basics of the file systems and the shell.

Start learning now!

Contribute!

Version controlling with git

Track code changes, undo mistakes, collaborate. This module is a must.

Start learning now!

Contribute!

Programming with python

Get started with an incredibly popular programming language.

Start learning now!

Contribute!

SSH

Introduction to the **Secure Shell (SSH)**

⚠ Status: Early development

Start learning now!

Contribute!

Machine learning

Get behind the buzzword and teach machines to work for you intelligently!

Start learning now!

Watch the videos!

Contribute!

Software Development and Deployment

Version controlling with git

Track code changes, undo mistakes, collaborate. This module is a must.

Start learning now!

Contribute!

CI/CD (gitlab)

Continuous integration and deployment with **gitlab**.

Start learning now!

Watch the videos!

Contribute!

CI/CD (github)

Continuous integration and deployment with **github actions**.

Start learning now!

Watch the videos!

Contribute!

Docker

Introduction to the **docker** container image system.

Start learning now!

Watch the videos!

Contribute!

Unit testing

Unit testing in python.

⚠ Status: Beta testing

Start learning now!

Contribute!

C++ corner

HEP C++ Course

A full introduction to C++ based on a series of slides and exercises.

Start learning now!

Watch the videos!

Contribute!

Basic Modern C++

A brand new C++ course that is currently in development.

⚠ Status: Early development

Start learning now!

Contribute!

Build systems: **cmake**

Building code is hard. **CMake** makes it easier.

Start learning now!

Contribute!

Training Curriculum: Advanced and In Development

HEP specific tools

uproot

Reading and writing ROOT files without having to install ROOT.

✳ Status: Beta testing

📖 Start learning now!

🔗 Contribute!

Miscellaneous

A simple analysis

A simple analysis using CMS open data.

📖 Start learning now!

📺 Watch the videos!

🔗 Contribute!

Planned or in early development

ROOT

The most famous data analysis framework used in HEP.

⚠ Status: Early development

Advanced git

Working with branches and more.

⚠ Status: Early development

🔗 Contribute!

Distributed file systems and grid computing

⚠ Status: Early development

Parallel programming

⚠ Status: Early development

alpaka

alpaka is a header-only C++ abstraction library for accelerator development.

⚠ Status: Early development

🔗 Contribute!

Workflows & reproducibility

E.g. yadage and reana

⚠ Status: Early development

Documentation

sphinx, doxygen, etc.

⚠ Status: Early development

Event generation and MC

pythia, sherpa, madgraph, etc.

⚠ Status: Early development

Matplotlib for HEP

Make science prettier with beautiful plots!

⚠ Status: Early development

📖 Start learning now!

🔗 Contribute!

Typical Carpentries Training Layout

	Setup	Download files required for the lesson
00:00	1. Introduction	What is C++? When is C++ the right language? How do I get started in C++?
00:10	2. Core Syntax and Types	What are the basic syntactical elements of C++ What are C++ types and which basic types exist
00:30	3. Arrays and Vectors	How can I create collections of things in C++ How can I get and set values in a collection? If I don't know the size of the collection in advance, what do I do?
01:10	4. Basic C++ operators	What is specific in basic C++ operators ?
01:20	5. Compound datatypes	How do we combine existing types into new types that are greater than the sum of their parts? What are classes, and how do we define and use them?
01:20	6. Functions	How to define a function ? What are the different ways to pass input arguments ? What are the different ways to get back the results ?
01:30	7. References	How do we use references in C++?
01:40	8. Control Instructions	How do I execute certain lines of code but not others? How do I reuse code and execute it many times?
02:10	9. Headers and Interfaces	What is an interface? Why separate some of the code into header files?
02:20	10. Templates	How to factorize the code of similar functions and classes, where only few types and sizes are changing ?
02:40	11. Type inference	Should I repeat again and again the obvious type of everything ?
02:50	12. Classes	What are classes, how do they differ from structs, and how do we build them?
02:50	13. Sum types	How do sum types differ from product types?

Upcoming Workshops

- **15 Mar - 17 Mar 2022** - 4th HEP C++ Course and Hands-on Training - The Essentials
- **28 Mar - 30 Mar 2022** - Software Carpentry (Virtual) 
- **1 May - 7 May 2022** - Thematic CERN School of Computing on “*Scientific Software for Heterogeneous Architectures*” - **Deadline:** 23 Jan 2022
- **19 Jun - 25 Jun 2022** - Thematic CERN School of Computing on “*Security of Research Computing Infrastructures*” - **Deadline:** 13 Mar 2022
- **28 Aug - 10 Sep 2022** - CERN School of Computing 2022 - **Deadline:** 17 Apr 2022

Achievements & Lessons from 2021

- **Virtual events have been beneficial for training.**
 - A large number of participants from all over the world, no travel costs
 - Over 300 participants trained, 50 mentors/instructors/facilitators
 - 3 [Training Events](#) (not including C++ training)
 - First Software Training paper published in Springer [Computing and Software for Big Science](#)
 - Two presentations about Software Training - [vCHEP2021](#) and [ACAT2021](#)
- **Successful organization of Software Carpentries Workshops**
 - Feedback collected. Surveys are a highly valuable input.
 - Events have been improved with each iteration.
 - The sustainability of the event depends on keeping a pool of instructors.
- **The C++ training events have been driven by the momentum of the community**
 - [2 C++ events](#), 150 participants, 35 mentors/instructors/facilitators
 - Monthly meetings for material development
- **A community built around training**
 - Active community of members from HEP and Nuclear Physics.
 - Time dedicated on voluntary basis, great dedication and enthusiasm

Plans for 2022

- **Sustainability of the training model.**
 - Software Carpentries training 3-4 times per year
 - Future training events depend on keeping a pool of instructors.
 - Ensure the reward of participating as mentor. Exposure outside experiments.
 - Contact participants from previous workshops and invite them to contribute to coming events.
 - Advertise the opportunities to grow professionally.
- **Scale up efforts to collect feedback before and after training events**
 - After basic courses, collect more information on what people are interested in. Use it for prioritize development of training modules.
 - Take the results of surveys and prepare a publication with the outcome.
- **Strength the communication with other WGs for training topics**
 - HSF Simulation group & reconstruction group?
- **Conference and Publications**
 - Look for conference to advertise training efforts
 - Publish papers on data collected during surveys

Ongoing Activities

- **Instructor training**

- With support from IRIS-HEP: training of up to 20 new HSF training instructors
- Up to 4 members of EIC (JLab & BNL) community will be included

- **Development of new trainings**

- matplotlib (in early testing)
- singularity (in early development)
- spack (planned, with HSF Dev Tools & Software Packaging)

Stay Informed or Get Involved

- **Enroll on our mailing list to find out about training events**
 - <https://groups.google.com/g/hsf-training-wg>
- **Attend weekly working group meetings**
 - Monday 16:00 CET weekly
 - <https://indico.cern.ch/category/10294/>
- **Contribute to training materials**
 - <https://github.com/hsf-training>
 - Pull requests to training materials are enthusiastically invited