

# HSF Curriculum page

Claire David

February 8, 2021



# General remarks on the curriculum page

**Main impression:** great ongoing effort, very promising platform for HEP!

**Caveat:** the layout is not ideal. Some remarks and suggestions in these slides to make it better.

HEP Software Foundation

Working Groups ▾

Activities ▾

Communication ▾

Projects & Support ▾

About ▾

## Towards a HEP Software Training curriculum

### Idea

Training in software and computing are essential ingredients for the success of any HEP experiment. As most experiments have similar basic prerequisites (Unix shell, Python, C++, ...) we want to join our efforts and create one introductory software training curriculum that serves HEP newcomers the software skills needed as they enter the field, and in parallel, instill best practices for writing software.

The curriculum is comprised of a set of standardized *modules*, so that students can focus on what is most relevant to them.

### The modules

- Want to study? Click on the book  or video  button
- Want to contribute? Click on the wrench 



# Title and intro

## “Curriculum”

This hints to a full program to validate, a whole checklist of skills to achieve.

Of course there are basic prerequisites that most of newcomers should go through, but some may not find several advanced modules relevant for their research. Thus the coining of *curriculum* could be daunting.

The mindset of this page is to have everything modular. You come and help yourself.

Thus I suggest to rename it for instance: “HSF training center”, or “HSF training modules”

This would better translate the cool feature of this supermarket of tutorials. And it is still valid even if this effort is ongoing.

# Ordering / categories

## Classification by level

I find several drawbacks in having lists of modules split by level:

- most folks entering HEP have some programming experience, let it be limited. What is exactly meant by “beginner” compared to “intermediate”?
- some ‘flavors’ in HEP activities make students to work on advanced tools, e.g MC event generators, quite early. If they are listed in “advanced”, this can be scary for them.
- some tutorials could benefit from having a mix of levels, with the first part very basic and the later sections geared toward experienced users. This is not possible if the modules are strictly segmented by level.

My suggestion? See first an idea of a “module legend” (next slide) that will show how these caveats can be addressed (actually suppressed).




# The module legend

## Taking advantage of the modularity

I suggest to present the modules in a square.

There could be a dummy example at the top of the page (or not, as long as there is a legend).

### How to read symbols:

-  alpha version: features to be added
-  beta version: done, but under tests
-  stable release, bug disinfected

### The Tutorial Title

Some explanations of the module with some links to [wikipedia](#) articles if relevant.

Status: 

Prerequisite(s): none

Material:  

Developer zone: 

[Credits](#)

here we solve the issue of level-splitting with an extra advantage: the user knows what is needed to follow a given module.

no need for a legend, the symbols are explained by the context

goes to author list.  
They need credits.

# A possible layout

## Robust over time

Let's imagine this page in two or three years: way more content! How to present it nicely without falling into the 'scrolling-down-paradigm'? Users should quickly find what they want. Suggestion: a gallery display, organized into categories such as (some examples):

Basics	← bare minimum: shell, ssh, python, ROOT.
Collaborative tools	← git, CI/CD, ...
C++ corner	← all the rest is.. python. Almost all.
Machine Learning	← basics, GPU interfacing,
HEP tools	← matplotlib for our nice plots, MC event generators, etc...
Documentation	← Sphinx, doxygen, etc...




# A little demo



# HEP Software Training Center

## Basics

### How to read symbols:



-  alpha version: features to be added
-  beta version: done, but under tests
-  stable release, bug disinfected

### The Unix Shell

Introduction to the unix command line/shell

Status: 

Prerequisite(s): none

Material:  

Developer zone: 

[Credits](#)

### SSH

Introduction to the Secure Shell (SSH)

Status: 

Prerequisite(s): none

Material:  

Developer zone: 

[Credits](#)

### Build systems: **cmake**

Introduction to the build systems cmake.

Status: 

Prerequisite(s): none

Material:  

Developer zone: 

[Credits](#)

### A simple analysis

A simple analysis workflow using CMS open data.

Status: 

Prerequisite(s): none

Material:  

Developer zone: 

[Credits](#)



## Collaborative tools & good practices

### Version control: git

Getting fluent in version controlling the code while allowing collaborators to improve it.

Status: 

Prerequisite(s): none


Material:  

Developer zone: 

[Credits](#)


### CI/CD with gitlab

Continuous integration and deployment with gitlab.

Status: 

Prerequisite(s): Version control: git

Material:  

Developer zone: 

[Credits](#)

### CI/CD with github

Continuous integration and deployment with github.

Status: 

Prerequisite(s): Version control: git

Material:  

Developer zone: 

[Credits](#)

### Unit testing

Unit testing in python.

Status: 

Prerequisite(s): Programming with Python

Material:  

Developer zone: 

[Credits](#)



# Happy to exchange more!

**These slides are suggestions to improve the HSF curriculum, as of January 26, 2021.**

Happy to brainstorm more on that.

Good luck with everything and ...

Thanks for reading!