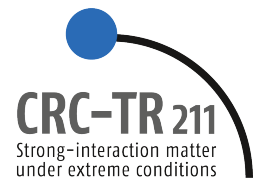


From 6 to 10 October 2025

# Z02-powered HGS-HIRe power week: Clean software development

→ aimed at Ph.D. students and young postdocs



Dr. Alessandro Sciarra<sup>†</sup>  
Trainer

Dr. David A. Clarke<sup>‡</sup>  
Trainer

## Abstract

Modern computational physics is numerically more and more demanding and the complexity of both algorithms and architectures has constantly grown over the last decades. This has naturally led to the development of large codebases (10k+ lines of code) in several research groups over multiple generations of PhD students and postdocs. Unless properly handled, the standard scenario is that software maintenance and even reading cost naturally explodes and within few years (if not months) it becomes terribly expensive if not impossible to make changes or implement new features.

Academic curricula only started to adapt to this rapid evolving landscape in the last years and this change will take time. The only possibility for young scientists is then to learn while dealing with the large codebases in their new group e.g. during their PhD and the pressure to focus on physics and obtain results often reduces the time to invest in learning alone to the bare minimum. Furthermore, technicalities often come first and, for instance, the used language syntax or the hardware architecture are needed to be studied at the beginning in order to be able to understand the existing code and to undertake some programming task. Effort to keep code quality high is often postponed, making one of the biggest mistake for the long-term software sustainability: *«To go fast, you go well»* – R.C. Martin.

In this pedagogical and exciting training, general principles will be explored in order to let participants easily improve their knowledge and toolkit and, in turn, their daily coding quality. Using many examples, the idea of clean code and clean testing will be introduced, several aspects discussed and participants will immediately apply what learnt. Most of the time will be spent in training sessions letting participants not only exercise on the discussed topics, but also practice techniques like pair programming and test/behaviour driven development.

As much as time will allow it, different topics naturally belonging to software development will be touched (e.g. use of a versioning control system, semantic versioning, branching patterns, profiling and optimising).

After this training, which is offered requiring as few as possible prerequisites and in the spirit of life-long-learning, participants will be able to write code from a totally new perspective and, at least, will be aware of what should be done to work in a sustainable way and what it would instead happen when choosing the quick-and-dirty way.

---

<sup>†</sup>sciarra@itp.uni-frankfurt.de

<sup>‡</sup>clarke.davida@gmail.com

# Course information

## Thematic focus

The training is tailored to PhD students doing computational science and, hence, have to deal with medium to large software on a regular or even daily basis. The main focus is *clean* software development. The ideas of clean code and clean testing will be introduced in the very beginning of the course and will be explored more and more throughout the week, enriching the participants section after section.

## Prerequisites

It will be assumed that you have been (or in the best case will be) working with software *for a while*. Basically, the only prerequisite is some experience in coding on which you will reflect during the course. The programming language used is not (too) relevant, because most of ideas and principles discussed during the week apply to all of them. However, you should have in your toolkit some program to do rudimentary plotting as well as know some scripting language to deal with data files in a basic way. Python serves both aspects very well, but you will not be forced to use it.

If you are totally new to Git, you should take some time and read through some introduction to it. You can pick your favourite tutorial online or read through the first two parts of [this trilogy](#) 🍷 (the third part will be discussed in the training).

## Requirements

Every participant is assumed to fulfil the following requirements.

1. **Bring a laptop** to work on during the training<sup>1</sup>.
2. **Choose a piece of your own software and have it ready**. This has to be in a language known to the participant and, ideally, it is either a small program or a stand-alone-working part extracted from a larger project. As rule of thumb, such a small program should be a few hundreds of lines long, but not longer of ~1000 lines.
3. **Have a computing environment ready to be used**. This might be your own laptop or a remote machine (for instance the own university computer e.g. reached over ssh). In particular, every participant should be able to **edit, possibly compile and run their own chosen piece of software**.
4. **Have Git installed in your computing environment**. It is likely that Git is already installed on your operating system. In order to check it, try to type `git version` in your terminal and see if the command is recognised. In the unlikely case Git is not installed, you can follow [this nice guide](#) 🍷 in order to install it.
5. **Have a GitHub account**. You might already have one, that's fine. If not, **create one** 🍷 for the power week (you might need it in your future at some point anyhow). During the training, we will collaborate in a private repository, so do not worry about making private work public.

## Registration

Registration is simply done by email to [info@hgs-hire.de](mailto:info@hgs-hire.de). Please, specify in the registration email

1. the programming languages you typically use in your work and
2. your GitHub user name.

## Further information

This power week has been jointly organized together with the CRC-TR211 collaboration. In case of a large request, a waiting list will be established and organisers will consider to enlarge the participants number, as far as possible.

## Timetable

In the following page you find the week timetable.

---

<sup>1</sup>If you do not have one, please contact the trainer.

Monday 6	Tuesday 7	Wednesday 8	Thursday 9	Friday 10
9:00				
10:00	Names, comments, ... Documentation Clean testing (II) Class, functions, IOSP	Git in real life	Data validation and data cleaning	Code review
11:00	Break	Break	New project (I)	Review project from another group
12:00	Rename! Check comments Add tests for function Apply IOSP	Clean testing (IV) TDD live demo	New project (II)	Break
13:00	Lunch	Lunch	Lunch	Finalise review
14:00	Welcome, week setup and Clean code	Practice TDD (I)	Statistical analysis and plotting	Sum up reviews
15:00	Splitting & Pick-up Clean testing (I)	Break	New project (III)	Lunch
16:00	Break	Break	Break	Feedback and closing
17:00	Testing framework and write Functional tests	Practice TDD (II) GitHub PR	New project (IV)	Departure from location
18:00	Sum up and closing	Sum up and closing	Sum up and closing	
19:00	Dinner	Dinner	Dinner	
20:00	Informal discussion Office hour	Informal discussion Office hour	Informal discussion Office hour	