

# General Information and Environment Setup

- To work on the exercises, a **Jupyter** Hub server is provided which can be accessed from any device via a web browser under the link

<https://jupytermachine.etp.kit.edu>.

Please use your KIT account credentials to log in, which must be registered for access to the Physik-Pool beforehand. In case you do not have access yet, please register your account under

<https://comp.physik.kit.edu/Account/>.

On this page you can also find a link for the prolongation of an existing account.

- Once logged on to the **jupytermachine**, you can spawn a server instance to work on. There are several choices available.
- Depending on the particular exercise, you need to choose a specific server image from the following list:
  - **Datenanalyse**: suitable for general purposes
  - **Geant4**: image for the exercise covering the detector simulation with GEANT4
- To change an image loaded already at log in, you can stop the server by going to **File** -> **Hub Control Panel** in the upper panel, and clicking on the **Stop My Server** button. Then you should wait until the button **Start My Server** appears, click on it, and then click on the button **Launch Server** when it appears. After that, you will arrive at the menu, where you can choose again a server image to be loaded.
- In the **Geant4** image, the button to click for changing server is on top right: **Control Panel**.
- From the chosen server instance, you will be able to access your home directory of the Physik-Pool in the File browser on the left side of your Jupyter Lab window.
- The exercises are provided as Python 3 Jupyter Notebooks in the following repository:

<https://gitlab.etp.kit.edu/gdepietro/geant4-exercise>.
- Please clone this repository to your Jupyter server instance by going to **Git** -> **Clone a Repository** in the upper panel. A pop-up window should appear, where you can enter <https://gitlab.etp.kit.edu/gdepietro/geant4-exercise>, and press **CLONE** after that. The repository will be created in the directory you are currently in, which is your user directory in the Physik-Pool by default. The directory **geant4-exercise** containing the exercises should appear in the file browser on the left.
- To clone the repository you might need to use the **Datenanalyse** server image and then switch back to the **Geant4** after having successfully cloned the repository.
- To change to the directory, you can click on the folder and will see different subfolders for the exercises appear.

# Working with Jupyter: Python, Markdown, and LaTeX

Jupyter allows to unify all ingredients you need to obtain and document your results in a nice format, including executable code, text formatting and formula formatting. In the context of this exercise course, you will make use the following languages:

- **Python:** Interpreted programming language. You will make use of already familiar libraries like `numpy` and `matplotlib`, but also learn about other packages like `pandas` which allows to process efficiently a big amount of data, stored e.g. in `.csv` format, and `iminuit` suitable for statistical inference and parameter optimization.
- **Markdown:** Web text formatting language to display text in a browser, usually stored in `.md` files (e.g. `README.md` in github repositories)
- **LaTeX:** Text formatting language very convenient for formatting mathematical expressions. In Jupyter notebooks, you can display a LaTeX formula via `$$your-latex-formula$$`.

More material and links to refresh and extend the knowledge on python and Jupyter can be found on the webpage of Prof. Dr. G. Quast:

<https://web.etp.kit.edu/~quast/jupyter/jupyterTutorial.html>.

The webpage <https://web.etp.kit.edu/~quast/> also contains hints how to install the necessary software on your own linux system, or how to run your own docker container or a virtual machine with all the software installed.

## About GEANT

In experimental particle physics, Monte Carlo (MC) methods (simulations) are used in order to design detectors, understand their behaviour, and compare experimental data to theory. MC production proceeds in two steps: event generation and detector response simulation. In the first step, sets of outgoing particles produced in the collisions of incoming particles in the accelerator called *events*, are generated. Then, the detector response to these particles is simulated. The most widely used tool for this purpose in particle physics is GEANT4<sup>1</sup>.

GEANT4 is a toolkit for the simulation of the passage of particles through matter. Its areas of application include particle, nuclear, and accelerator physics, as well as studies in medical and space science. GEANT4 includes tools for handling geometry, tracking, detector response, run management, visualisation, and a user interface:

---

<sup>1</sup><https://geant4.web.cern.ch>

- “Geometry” is a description of the physical layout of the experiment. This includes the detectors, absorbers, etc. and considerations how the layout will affect the trajectories of particles in the experiment.
- “Tracking” is the simulation of the passage of a particle through matter. This takes into account possible interactions and decay processes.
- “Detector response” is a record of when a particle passes through a given volume of a detector and approximates how a real detector would respond.
- “Run management” includes recording the details of each run (a set of events) as well as setting up the experiment in different configurations between runs.
- The GEANT4 package offers a number of options for visualisation, including OpenGL, and a user interface, based on tclsh.

For the simulation of the events we will use GEANT4. We will steer GEANT4 using Python via the interface provided with the `geant4_simulation`. This package is a wrapper for the Geant4 python bindings `Geant4Py`<sup>2</sup>. It provides the necessary code to perform simple detector simulations with GEANT4. This includes the description of various geometries needed for the exercises, some additional classes that allow the tracking of particles and energy deposits in the detector and a simple interface to steer the simulation. The parts of the interface necessary to work on the exercises are briefly described in the corresponding section of the Jupyter Notebook provided for the exercise.

The exercises, as well as the complete source code and some introductory code showing the usage of the Python package `geant4_simulation`, are provided in the form of Jupyter Notebooks in the repository:

<https://gitlab.etp.kit.edu/gdepietro/geant4-exercise>

Log on to the [jupytertermachine](#) and choose the **Geant4** image.

In this exercise, a sampling calorimeter will be simulated with GEANT4. The exercise will be done inside the two Jupyter Notebooks `Geant4_Part1.ipynb` and `Geant4_Part2.ipynb` where all descriptions given below are contained as well. So from this point on you only need the Jupyter Notebooks to work on the exercise.

---

<sup>2</sup><https://geant4-userdoc.web.cern.ch/UsersGuides/ForApplicationDeveloper/html/LanguageBindings/languageBindings.html>

## Commands of the GEANT interface

In this section, some of the commonly used code snippets of the GEANT4 interface are given. These are given in an introductory fashion in the Jupyter notebook as well. The following Python commands will create a particle simulation and an event display with GEANT4. You will also find them in the Jupyter Notebook.

- `g4 = g4sim.ApplicationManager()`

Create a new instance of the `ApplicationManager` class.

- `g4.set_geometry(<filename>)`

Set the geometry to the one described in the file:

`./geant4_simulation/geometry/<filename>.py`.

- `g4.set_physics_list('QGSP_BERT')`

Set the 'physics lists' we will need in the corresponding exercise. The physics list specifies all particles and interactions considered in the simulation. It is mandatory for all simulations, you do not need to modify it and the relevant physics lists for each exercise are given in the exercise.

- `g4.initialize()`

Initialise the GEANT4 kernel.

- `g4.set_particle(<PDG code>)`

Set the type of the primary particle. The `<PDG code>` is an integer number defined by the Particle Data Group to specify different particle types<sup>3</sup>. For example, `<PDG code>=22` is a photon.

- `g4.set_energy(<energy>)`

Set the energy of the primary particle in units of GeV.

- `g4.start_run()`

Start the simulation. You can also provide an argument which will define the number of simulated events, the default is 1. Also by default, the output of the simulation is printed as text, if you want a graphical representation you need to use `visualize=True`.

---

<sup>3</sup><https://pdg.lbl.gov/2024/reviews/rpp2024-rev-monte-carlo-numbering.pdf>