

Table of Contents

The bender tutorials	1.1
Contributing	1.1.1
Getting started	2.1
Examples of formatting	2.1.1
The first two (useless) code	2.1.2
Another top level section	3.1
A subsection	3.1.1
A subsubsection	3.1.1.1
Download PDF	4.1

The Bender tutorials

build passing

These are tutorials for Bender application: "[User-friendly python analysis environment for LHCb](#)".

It is the first attempt to convert existing [TWiki-based tutorials](#) to GitHub platform, inspired by the great success of [LHCb StarterKit lessons](#).

Bender is [LHCb Python-based Physics Analysis Environment](#). It combines the physics content of [DaVinci-project](#) with the interactive python abilities provided by [GaudiPython](#). It also could be considered as "[Interactive LoKi](#)". The major functionality comes from ROOT/Reflex dictionaries for the basic C++ classes and the interfaces.

These dictionaries are used primary for POOL persistency and effectively reused for interactivity. The main purpose of top-level scripts is the coherent orchestration of the Reflex dictionaries and the proper decoration of the available interfaces.

Bender dependencies are sketched here:[dependencies](#)

Doxigen documentation for Bender is accessible [here](#).

It is assumed that users are already has *some* knowledge of LHCb software, in particular [DaVinci](#) and are familiar with [LHCb Starterkit](#).

You can also add relative links within the website like this one to the [first section](#)!

Contributing

[bender-tutorials](#) is an open source project, and we welcome contributions of all kinds:

- New lessons;
- Fixes to existing material;
- Bug reports; and
- Reviews of proposed changes.

By contributing, you are agreeing that we may redistribute your work under [these licenses](#). You also agree to abide by our [contributor code of conduct](#).

Getting Started

1. We use the [fork and pull](#) model to manage changes. More information about [forking a repository](#) and [making a Pull Request](#).
2. To build the lessons please install the [dependencies](#).
3. For our lessons, you should branch from and submit pull requests against the `master` branch.
4. When editing lesson pages, you need only commit changes to the Markdown source files.
5. If you're looking for things to work on, please see [the list of issues for this repository](#). Comments on issues and reviews of pull requests are equally welcome.

Dependencies

To build the lessons locally, install the following:

1. [Gitbook](#)

Install the Gitbook plugins:

```
$ gitbook install
```

Then (from the `bender-tutorials` directory) build the pages and start a web server to host them:

```
$ gitbook serve
```

You can see your local version by using a web-browser to navigate to `http://localhost:4000` or wherever it says it's serving the book.

Getting started

Click on the "[Examples of formatting](#)" section on the left

The title

Learning Objectives

- The starterkit lessons all start with objectives about the lesson
- Objective 2 with some *formatted text like* this

Basic formatting

You can make **bold**, *italic* and ~~strikethrough~~ text. Add relative links like [this one](#) and absolute links in a [couple](#) of [different](#) ways.

Have bulleted lists:

- Point 1
- Point 2
 - Sub point
 - Sub point
 - Sub point
- Point 2

Use numbered lists:

1. First
2. Second
 - i. Second first
 - i. Second first first
 - ii. Second second
3. Third

LaTeX

You can use inline LaTeX maths such as talking about the decay $D^{*+} \rightarrow D^0 \rightarrow K^{\{-}\pi^{\{+}}$.

Code highlighting

And have small lines of code inline like saying `print("Hello world")` or have multiple lines with syntax highlighting for python:

```
import sys

def stderr_print(string):
    sys.stderr.write(string)

stderr_print("Hello world")
```

bash:

```
lb-run Bender/latest $SHELL
dst_dump -f -n 100 my_file.dst 2>&1 | tee log.log
```

and more!

Callouts

Prerequisites

- Prerequisite 1
- Prerequisite 2

Objectives

- Objective 1
- Objective 2

Challenge

Set a challenge here, and the solution will remain hidden until it's clicked

- How to print?

Solution

The answer is:

```
print("Hello world")
```

Extra details that are hidden by default

Some extra details

Keypoints

- Summary point 1

- Summary point 2

Quotes

This was said by someone

Tables

Simple tables are possible

First Header	Second Header
Content from cell 1	Content from cell 2
Content in the first column	Content in the second column

Images



Section types

This is a section

Subsections

And a subsection

Subsubsections

And a subsubsection

The first two *useless*, but *illustrative* examples

Any valid Bender module must have essential parts

- function `run` with the predefined signature
- function `configure` with the predefined signature

For the most trivial ("*do-nothing*") scenario function `run` is

```
def run ( nEvents ) :  
    # some fictive event loop  
    for i in range( 0 , min( nEvents , 10 ) ) : print ' I run event %i ' % i  
    return 0
```

In a similar way, the simplest "*do-nothing*"-version of `configure` -function is

```
def configure ( datafiles , catalogs = [] , castor = False , params = {} ) :  
    print 'I am configuration step!'  
    return 0
```

Such that the whole script looks as:

Sections

Click on the subsection on the left to see the subsubsection

A subsection

Learning Objectives

- Objective 1
- Another objective

A subsection

Learning Objectives

- Objective 1
- Another objective