The Scikit-HEP project

Eduardo Rodrigues, on behalf of the Scikit-HEP Developers
University of Cincinnati

LHCb Analysis & Software Week, CERN, 6 April 2017

Python shaping the daily life of a HEP analyst

Python usage in the last few years

□ Mostly for simple scripting tasks

- Small well-defined analysis tasks
- Configuration of applications / programs

□ In daily tasks such as plotting, code tests

□ As an analysis framework

⇨ ***This is where we start to strongly link with the scientific computing community …***

There are many reasons for the success

□ Python is simple, readable, good-looking, and very well documented

□ Almost all one needs is already available out there …

□ … since the community is huge

A tale of 2 worlds

HEP, ROOT-based

- ❑ ROOT for almost everything
- ❑ Toolkit for modeling / fitting: RooFit
- ❑ Statistics: RooStats
- ❑ Machine learning: TMVA

Scientific Computing in Python

- ❑ The father of them all: SciPy
- ❑ Data manipulation: NumPy, Pandas
- ❑ Plotting: matplotlib, seaborn, Bokeh
- ❑ Machine learning: scikit-learn, TensorFlow
- ❑ Etc.

- ❑ + dedicated projects built atop the above: Astropy, biopython, etc.

⇨ *Are we missing something, i.e. what should we be learning from this ?*

Why Scikit-HEP ?

The facts

❑ ROOT is at the heart of HEP software, and likely to remain

- Usage well beyond analysis, eg. I/O

❑ Python is here to stay, at least as far as analysis work is concerned

❑ And the scientific toolkit in Python is excellent & wide-ranging

The evident conclusion(s)

❑ No need to be exclusive, we can exploit this all !

❑ How to bridge between ROOT and the Python scientific ecosystem?

❑ Various initiatives exist out there,
but only tackling specific tasks/issues

❑ Scope / need for a more general(ised) effort

- Others did it: Astropy, biopython

Interoperability

Collaboration

Reproducibility

The Scikit-HEP project

The idea, in just one sentence

The Scikit-HEP project (http://scikit-hep.org/) is a community-driven and community-oriented project with the aim of providing Particle Physics at large with a Python package containing core and common tools.

What it is NOT …

❑ **A replacement for ROOT**

❑ **A replacement for the Python ecosystem based on NumPy, scikit-learn & co.**

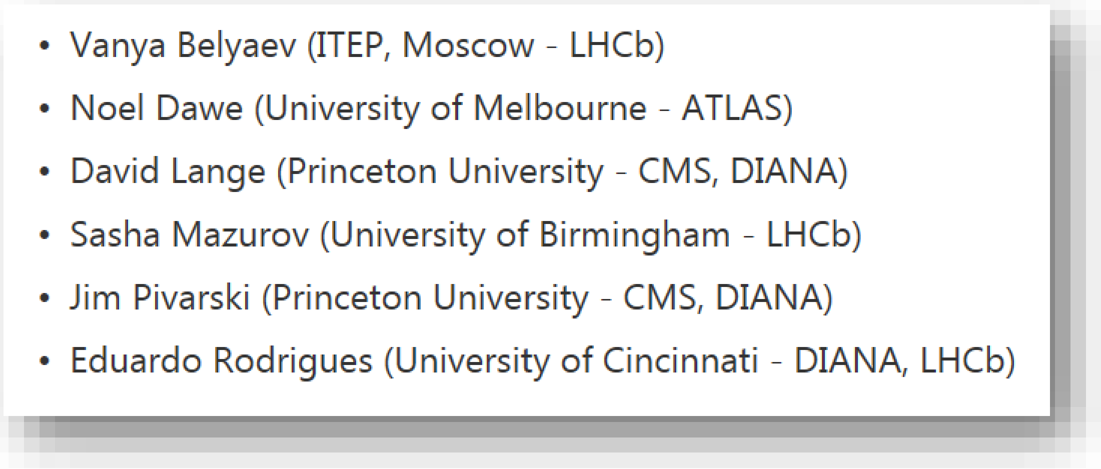
… and what IT IS

❑ **Bridge/glue between the ROOT-based and the Python scientific ecosystem**
- **Expand typical toolkit of HEP physicists**
- **Common definitions and APIs to ease "cross-talk"**

❑ **Project similar to the Astropy project – learn from good examples ;-)**

The Scikit-HEP project – team

❑ **Project started with a team with varied experience and expertise**



- Vanya Belyaev (ITEP, Moscow - LHCb)
- Noel Dawe (University of Melbourne - ATLAS)
- David Lange (Princeton University - CMS, DIANA)
- Sasha Mazurov (University of Birmingham - LHCb)
- Jim Pivarski (Princeton University - CMS, DIANA)
- Eduardo Rodrigues (University of Cincinnati - DIANA, LHCb)

+ Alex Pearce(LHCb) for the website design

❑ **Building the core from existing packages & experience, as a starting point**

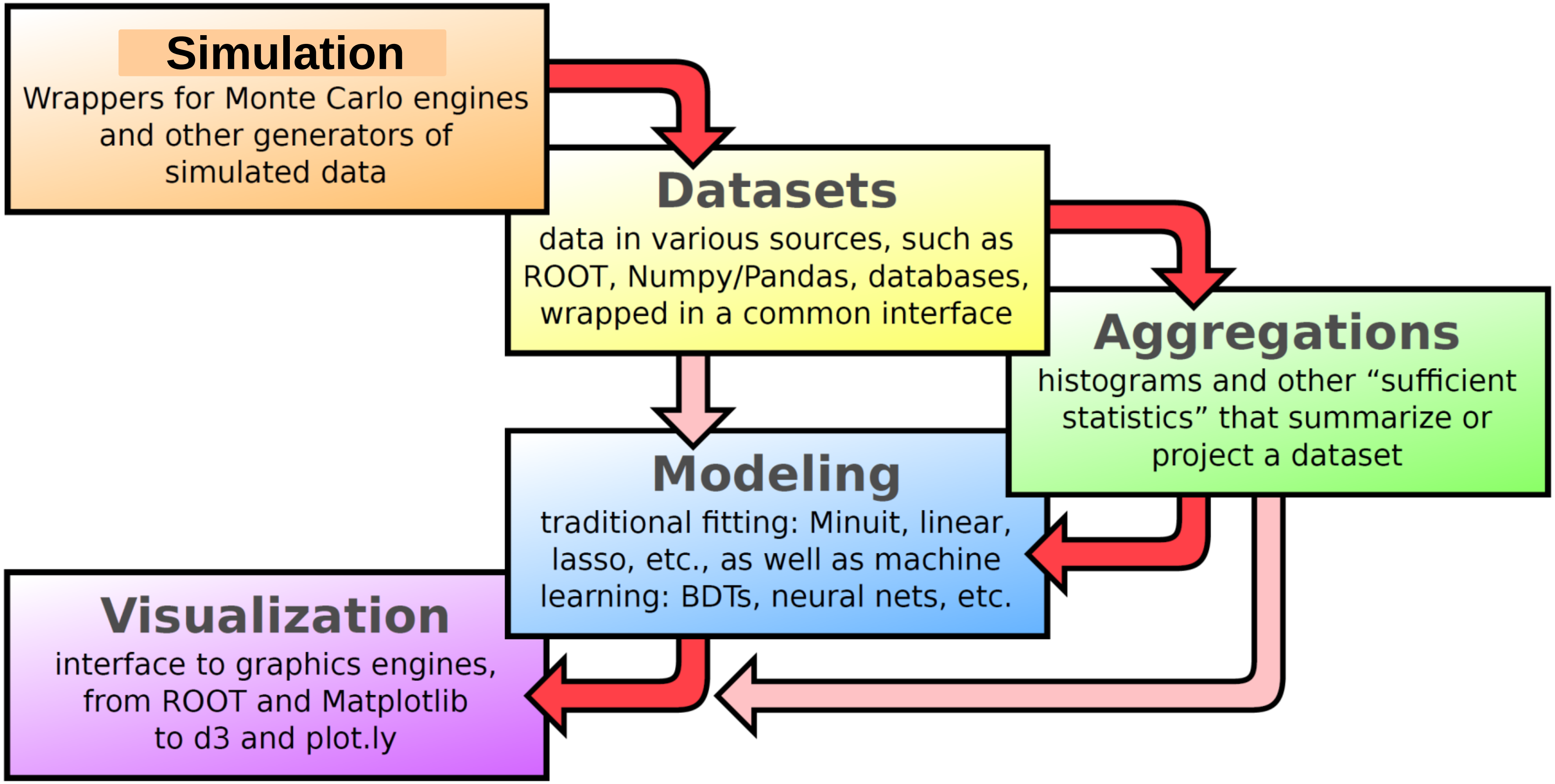
- Ostap (Vanya)

- rootpy and root_numpy (Noel et al.)

❑ **Bring in other packages & ideas**

- Either to core package or as an “affiliated package” with common API, rules and standards

The Scikit-HEP project – 5 « pillars »



Simulation
Wrappers for Monte Carlo engines and other generators of simulated data

Datasets
data in various sources, such as ROOT, Numpy/Pandas, databases, wrapped in a common interface

Aggregations
histograms and other “sufficient statistics” that summarize or project a dataset

Modeling
traditional fitting: Minuit, linear, lasso, etc., as well as machine learning: BDTs, neural nets, etc.

Visualization
interface to graphics engines, from ROOT and Matplotlib to d3 and plot.ly

➡ **They cover all grand topics … !**

The Scikit-HEP software suite, in short

- ***A set of sub-packages / modules***
- ***Corresponding tests***

- ***A set of command-line scripts for well-defined tasks***

- ***And a set of affiliated packages***

N.B.: all under development/design!

The Scikit-HEP software package (non-exhaustive!)

❑ Dataset

- Common interface for data in various sources
- Dealing with ROOT TTree and Numpy arrays for a start (profit from root_numpy project!)

❑ Aggregation

- Summarise or project a dataset
- Typically data aggregation = histogram
- Make use of the Histogrammar project?

❑ Modeling

- Data models and fitting utilities
- Will need careful design to talk smoothly to the Python scientific ecosystem at large

❑ Visualization

- Interface to graphics engines such as ROOT and matplotlib, among others
- Build from rootpy project!

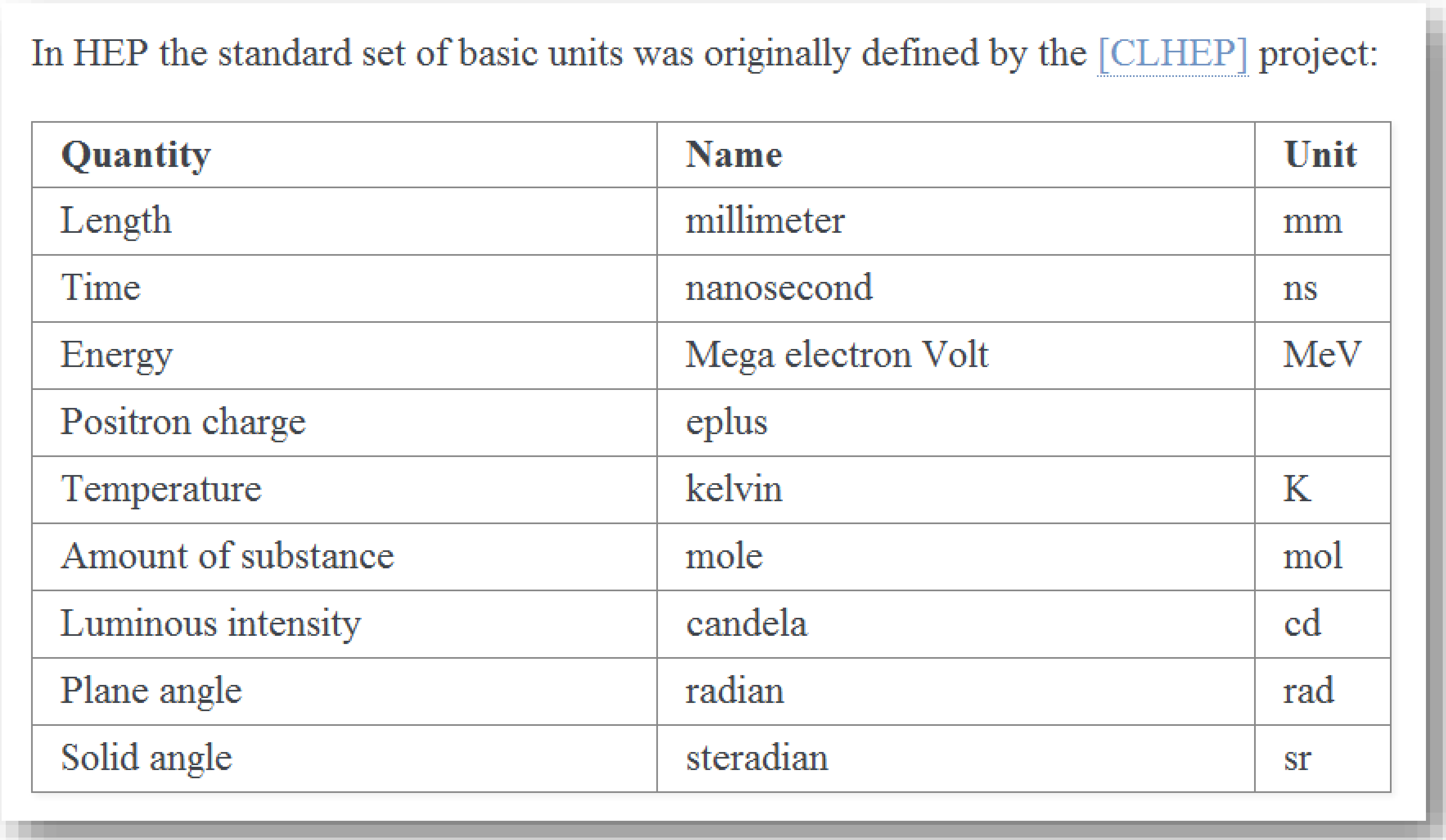
❑ Simulation

- utilities, wrappers for Monte Carlo engines
  and other generators of simulated data

❑ Modules for units and constants

❑ Maths and statistics tools

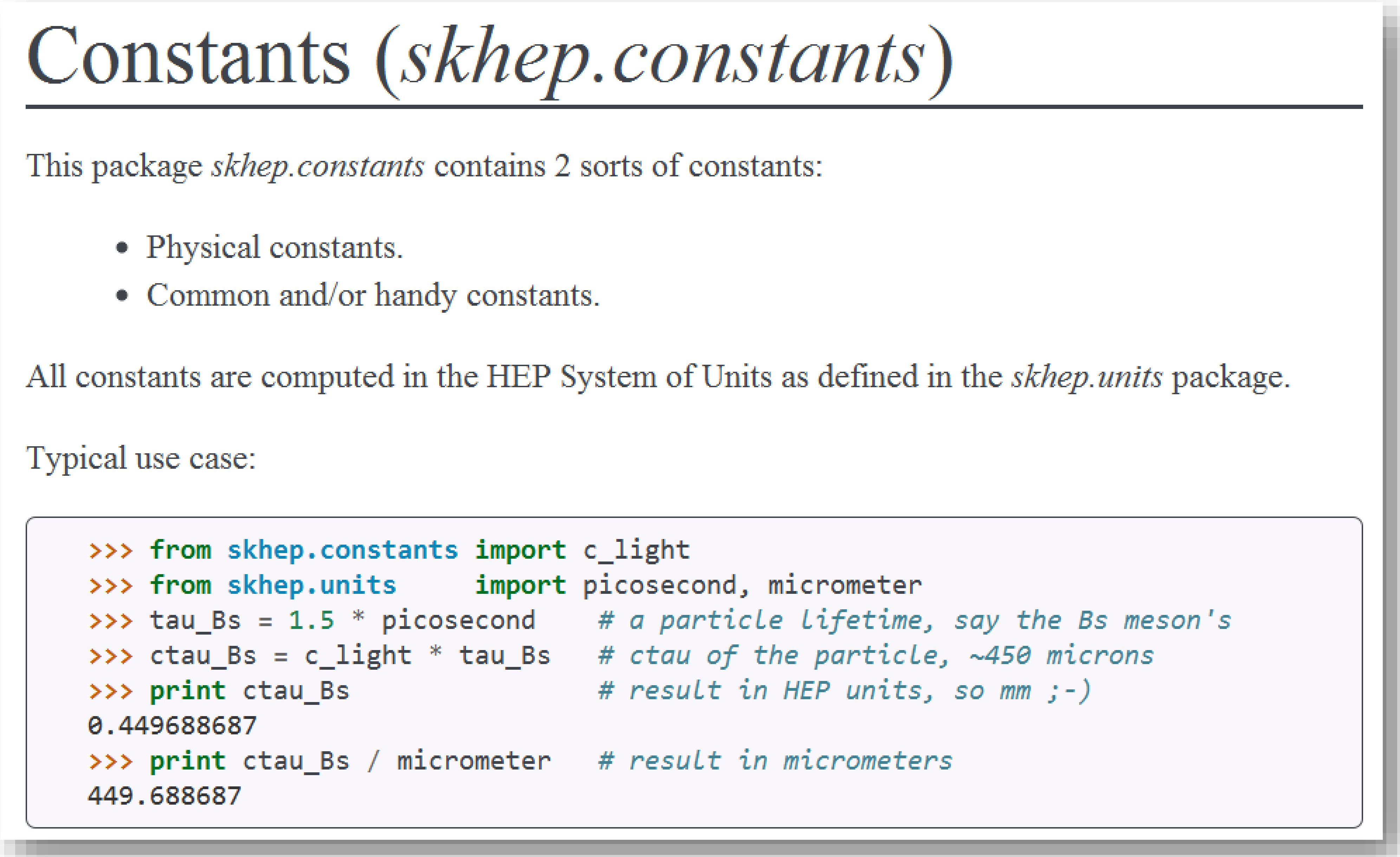
Module examples – HEP units



In HEP the standard set of basic units was originally defined by the [CLHEP] project:

Quantity	Name	Unit
Length	millimeter	mm
Time	nanosecond	ns
Energy	Mega electron Volt	MeV
Positron charge	eplus	
Temperature	kelvin	K
Amount of substance	mole	mol
Luminous intensity	candela	cd
Plane angle	radian	rad
Solid angle	steradian	sr

Module examples – constants



Constants (*skhep.constants*)

This package *skhep.constants* contains 2 sorts of constants:

- Physical constants.
- Common and/or handy constants.

All constants are computed in the HEP System of Units as defined in the *skhep.units* package.

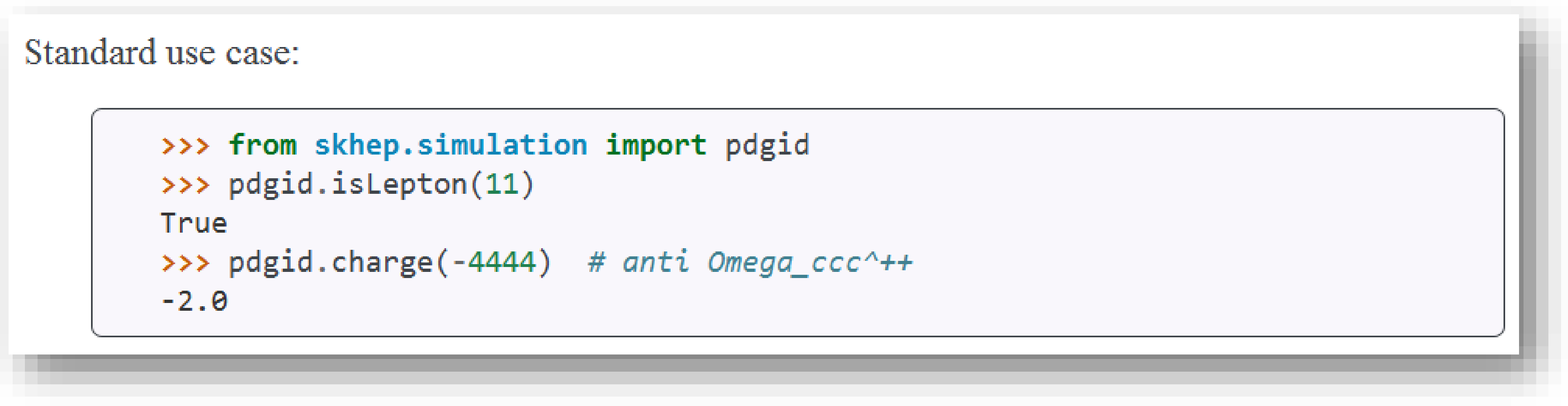
Typical use case:

```
>>> from skhep.constants import c_light
>>> from skhep.units     import picosecond, micrometer
>>> tau_Bs = 1.5 * picosecond    # a particle lifetime, say the Bs meson's
>>> ctau_Bs = c_light * tau_Bs   # ctau of the particle, ~450 microns
>>> print ctau_Bs                # result in HEP units, so mm ;-)
0.449688687
>>> print ctau_Bs / micrometer   # result in micrometers
449.688687
```

Module examples – simulation

❑ **Trivial wrapper for the HepPID C++ library, using PyPDT**

❑**(More is coming on this front)**



Standard use case:

```
>>> from skhep.simulation import pdgid
>>> pdgid.isLepton(11)
True
>>> pdgid.charge(-4444)  # anti Omega_ccc^++
-2.0
```

Building a community

- ❑ The project has been defined as community-driven and community-oriented
  ⇒ the concept of a community is central !

- ❑ We welcome contributions and contributors from all horizons !

- ❑ We have a site page for a forum of project ideas …
- ❑ You are most welcome to bring your own ideas too !

- ❑ We are and will be engaging with (future) collaborators in various experiments
  - E.g. LHC, neutrino community, simulation community, Belle-II, FCC,SHiP

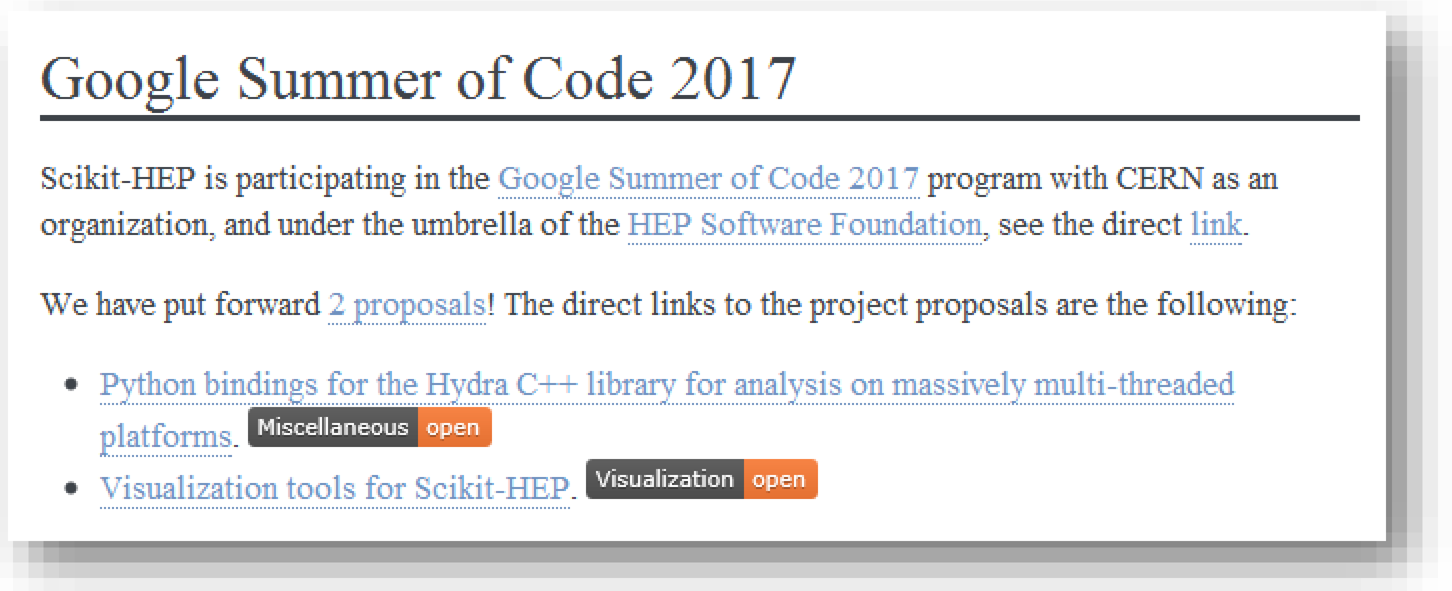
Example EoIs ***(a.k.a. expressions of interest)***

- ❑ Andy Buckley (ATLAS): simulation tools expert, author of PyPDT
- ❑ DUNE software developers Robert Sulej & Dorota Stefan

Building a community – project ideas

- ❑ As said, dedicated page exists on the website
- ❑ For now just a handful of examples actually displayed

- ❑ Even a couple of proposals for the GSoC!



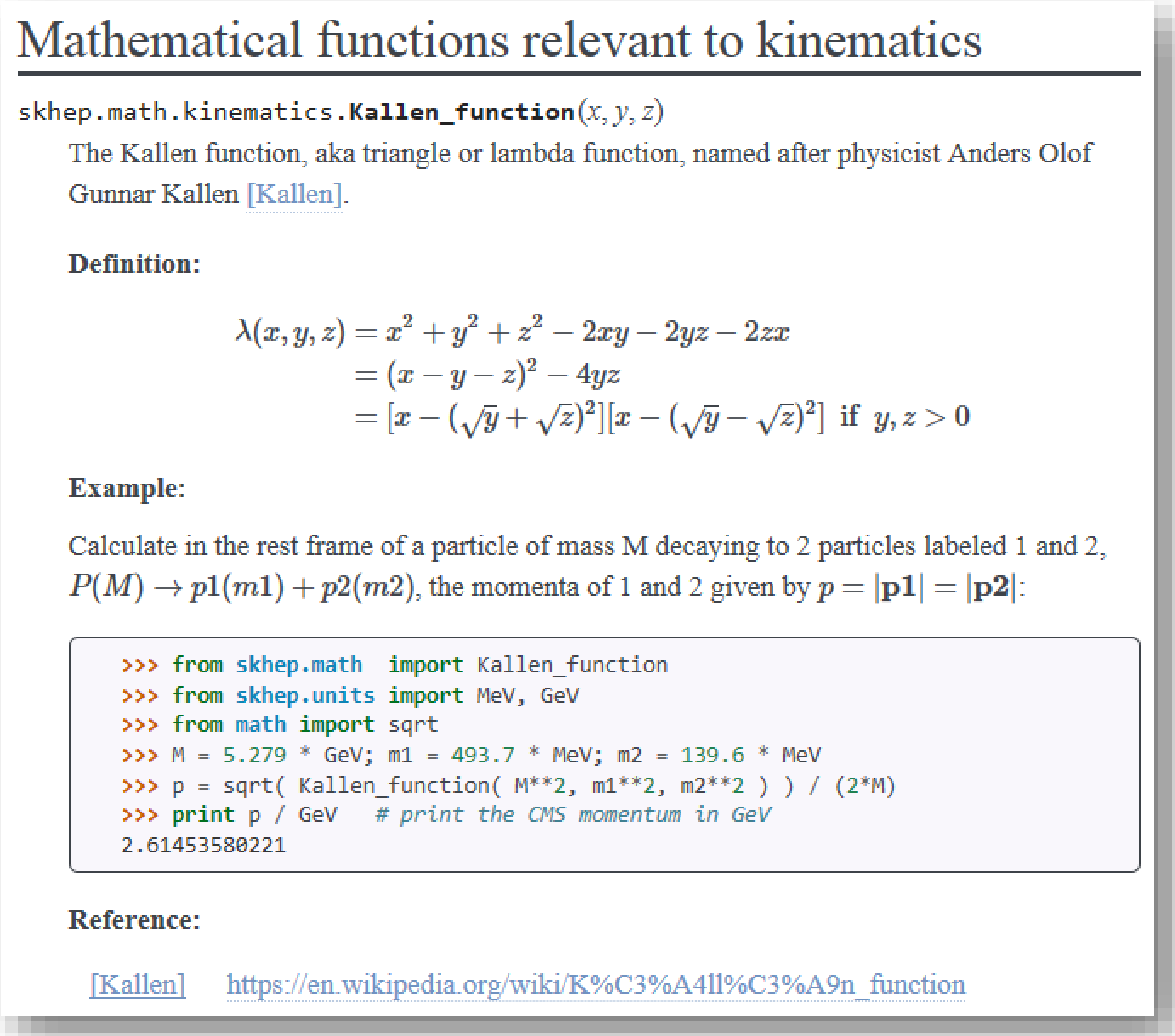
Google Summer of Code 2017

Scikit-HEP is participating in the Google Summer of Code 2017 program with CERN as an organization, and under the umbrella of the HEP Software Foundation, see the direct link.

We have put forward 2 proposals! The direct links to the project proposals are the following:

- Python bindings for the Hydra C++ library for analysis on massively multi-threaded platforms. Miscellaneous open
- Visualization tools for Scikit-HEP. Visualization open

Documentation – online version



Mathematical functions relevant to kinematics

`skhep.math.kinematics.`**`Kallen_function`**(x, y, z)

The Kallen function, aka triangle or lambda function, named after physicist Anders Olof Gunnar Kallen [Kallen].

Definition:

$$\begin{aligned}\lambda(x,y,z) &= x^2 + y^2 + z^2 - 2xy - 2yz - 2zx \\ &= (x-y-z)^2 - 4yz \\ &= [x - (\sqrt{y} + \sqrt{z})^2][x - (\sqrt{y} - \sqrt{z})^2] \ \text{ if } \ y, z > 0\end{aligned}$$

Example:

Calculate in the rest frame of a particle of mass M decaying to 2 particles labeled 1 and 2, $P(M) \to p1(m1) + p2(m2)$, the momenta of 1 and 2 given by $p = |\mathbf{p1}| = |\mathbf{p2}|$:

```
>>> from skhep.math  import Kallen_function
>>> from skhep.units import MeV, GeV
>>> from math import sqrt
>>> M = 5.279 * GeV; m1 = 493.7 * MeV; m2 = 139.6 * MeV
>>> p = sqrt( Kallen_function( M**2, m1**2, m2**2 ) ) / (2*M)
>>> print p / GeV   # print the CMS momentum in GeV
2.61453580221
```

Reference:

[Kallen] https://en.wikipedia.org/wiki/K%C3%A4ll%C3%A9n_function

Affiliated packages

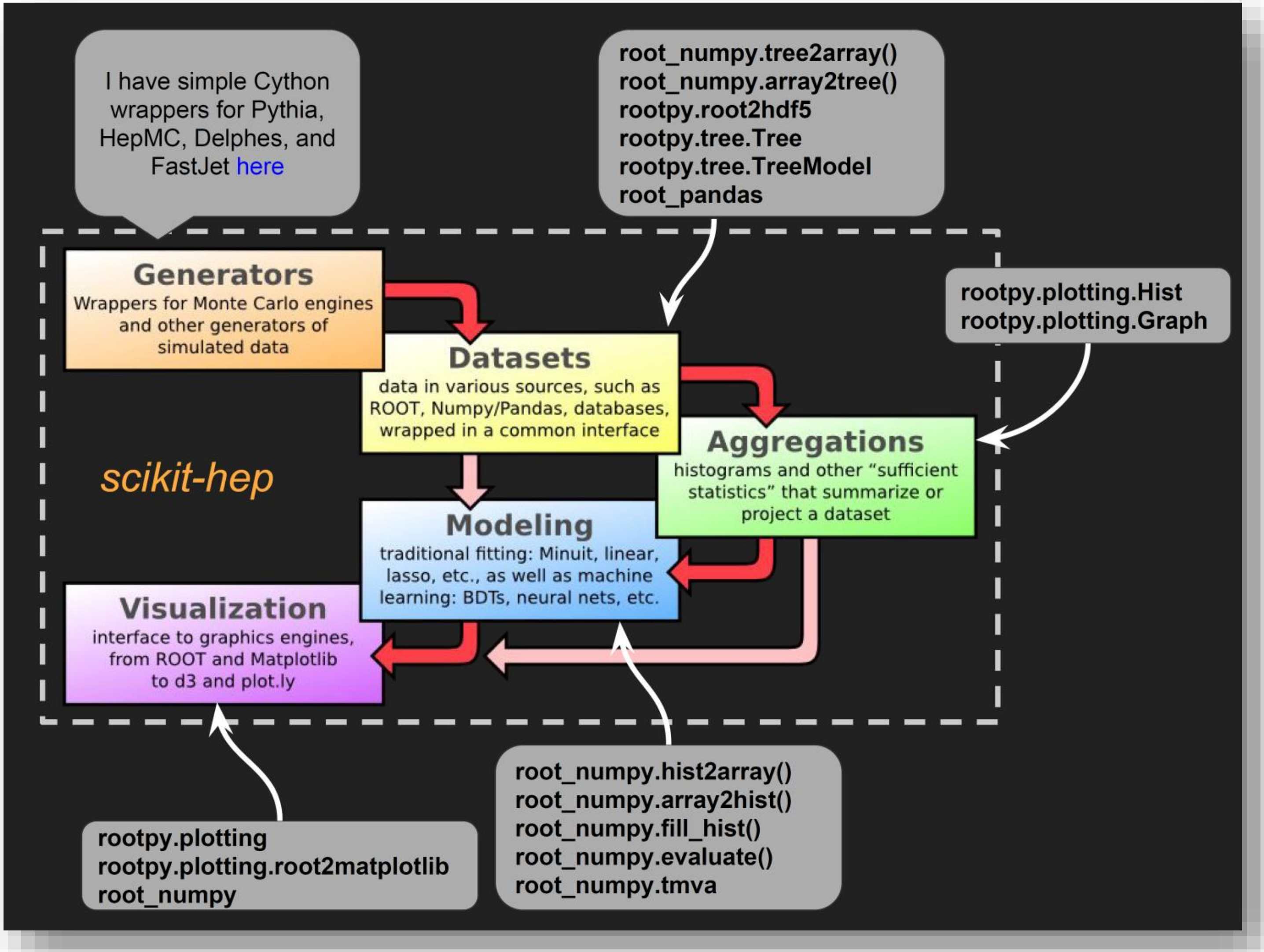
❑ Take good concept from Astropy of an *affiliated package*:
Python package not part of the Scikit-HEP core but related to, and seen as part of, the Scikit-HEP community and project

❑ Allows expansion of toolkit avoiding a gigantic do-everything package

❑ Bring-in functionality specific to certain topics/areas not of the widest community interest

❑ Potential examples are
- root_numpy – agreed already that it will be under the Scikit-HEP organisation
- Hydra, specifically a Python API to this
  header-only C++ library for data analysis in massively parallel platforms
- hep_ml, a ML library with miscellaneous tools for HEP
- Many more.

Affiliated packages – example how root_numpy can fit in



I have simple Cython wrappers for Pythia, HepMC, Delphes, and FastJet here

root_numpy.tree2array()
root_numpy.array2tree()
rootpy.root2hdf5
rootpy.tree.Tree
rootpy.tree.TreeModel
root_pandas

rootpy.plotting.Hist
rootpy.plotting.Graph

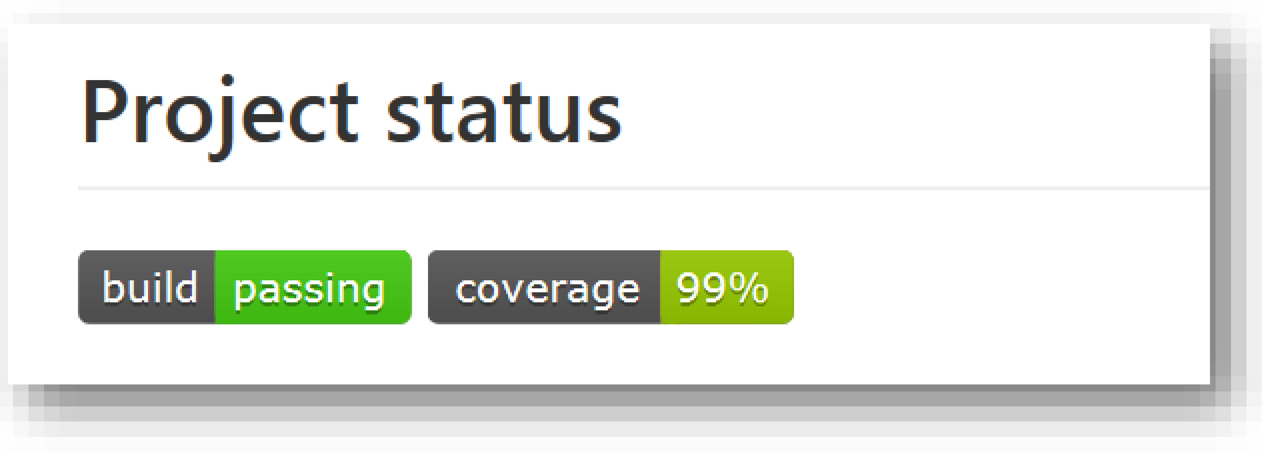
Generators
Wrappers for Monte Carlo engines and other generators of simulated data
Datasets
data in various sources, such as ROOT, Numpy/Pandas, databases, wrapped in a common interface
Aggregations
histograms and other “sufficient statistics” that summarize or project a dataset
scikit-hep
Modeling
traditional fitting: Minuit, linear, lasso, etc., as well as machine learning: BDTs, neural nets, etc.
Visualization
interface to graphics engines, from ROOT and Matplotlib to d3 and plot.ly

rootpy.plotting
rootpy.plotting.root2matplotlib
root_numpy

root_numpy.hist2array()
root_numpy.array2hist()
root_numpy.fill_hist()
root_numpy.evaluate()
root_numpy.tmva

Miscellaneous – continuous integration

- ❑ **Important aspect to be taken into account**
- ❑ **Status of code displayed on the GitHub site**
  - **Code built to be compatible with Python 2.6, 2.7 and 3.4**
  - **Test coverage with *Coveralls.io***



Project status

Miscellaneous – distribution & deployment

- ❑ "pip vs conda" discussion ongoing …

At present

- ❑ pip does the job well for typical Python projects
- ❑ Suitable for now since *scikit-hep* does not yet depend on ROOT

In the near future

- ❑ Dependence on ROOT will eventually need special treatment, at least in principle
- ❑ Will need a bit more discussion

Planning

Next few months

❑ Development releases will happen soon-ish

❑ Main goals:
- Ease the feedback from users
- Test the distribution/deployment set up

❑ Engage (further) with Particle Physics community at large
- E.g. present project to experiments

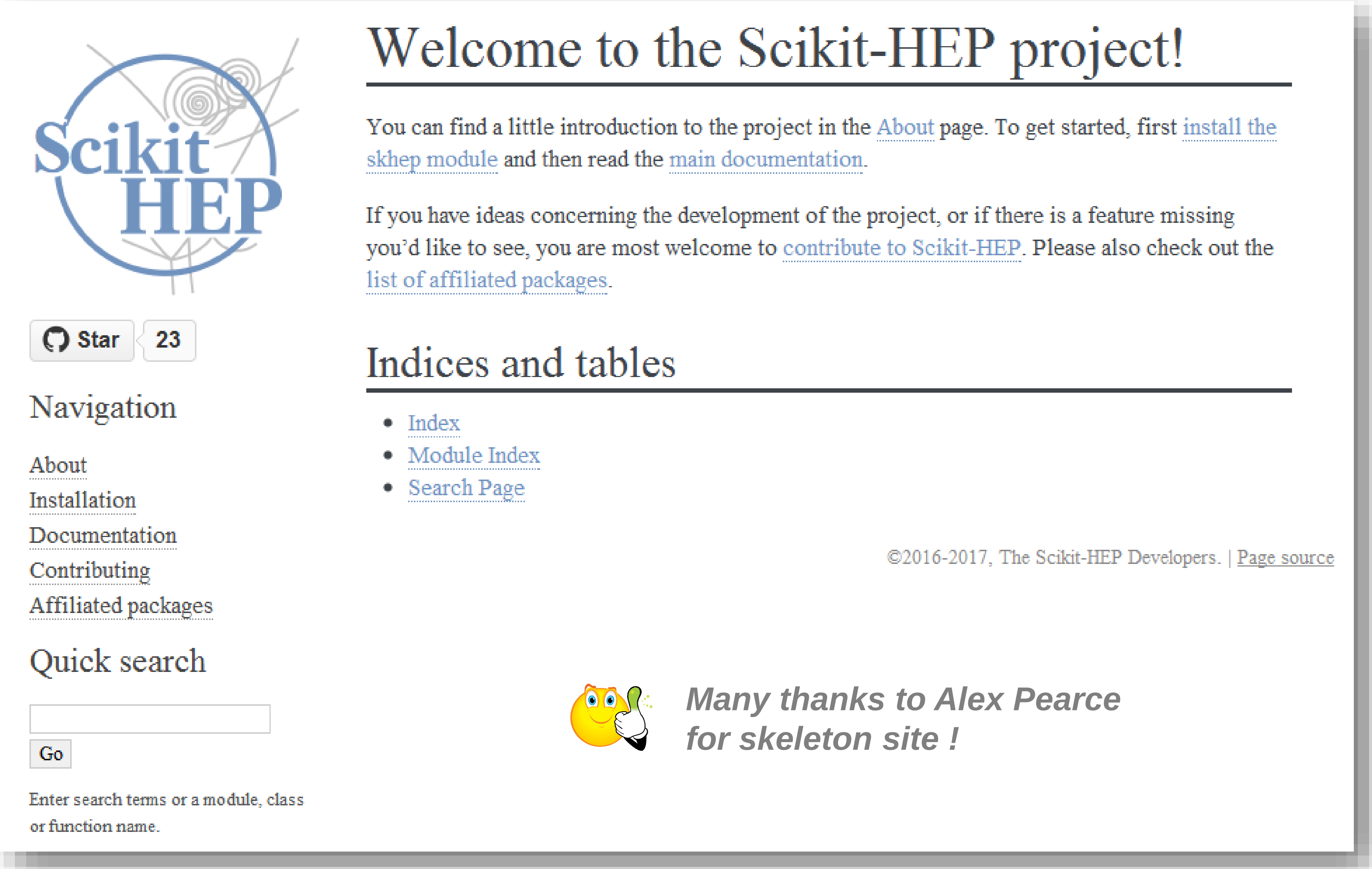
Towards the end of 2017

❑ First release of scikit-hep package

❑ Continue engaging with community

❑ Training on the software package

Website scikit-hep.org



Enter search terms or a module, class or function name.

Welcome to the Scikit-HEP project!

You can find a little introduction to the project in the About page. To get started, first install the skhep module and then read the main documentation.

If you have ideas concerning the development of the project, or if there is a feature missing you'd like to see, you are most welcome to contribute to Scikit-HEP. Please also check out the list of affiliated packages.

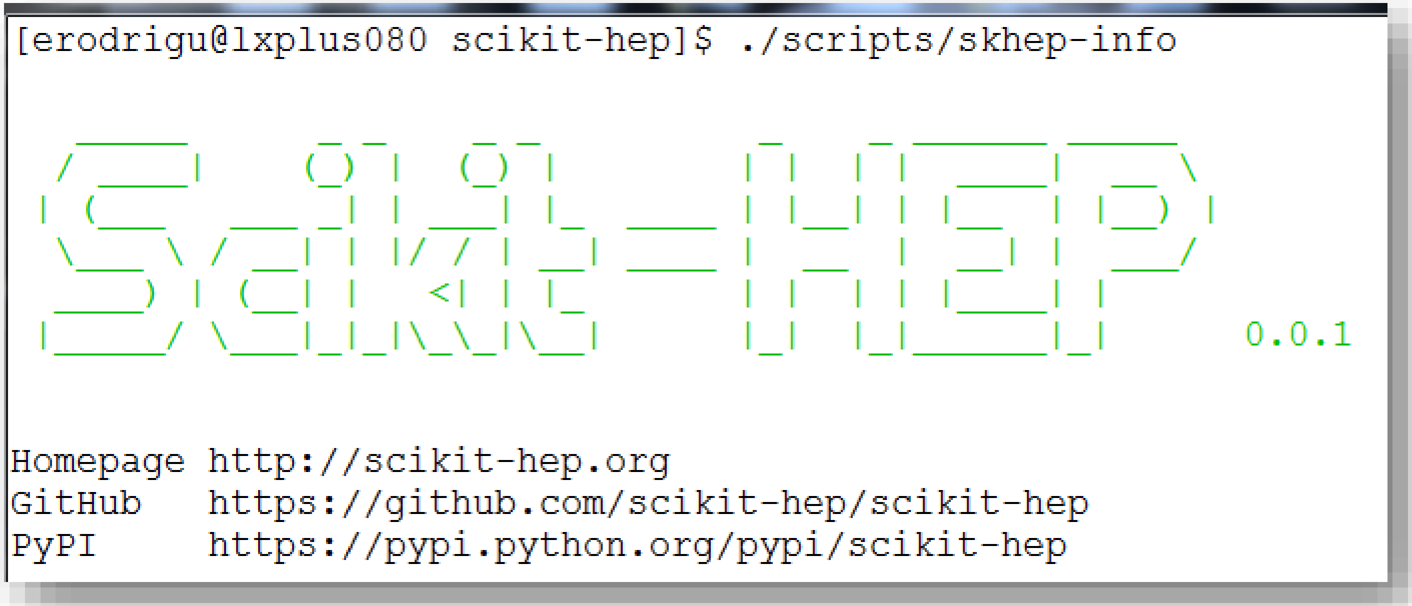
Indices and tables

- Index
- Module Index
- Search Page

©2016-2017, The Scikit-HEP Developers. | Page source

Many thanks to Alex Pearce for skeleton site !

Want to know more … ?



```
[erodrigu@lxplus080 scikit-hep]$ ./scripts/skhep-info

   _____      _ _    _ _          _    _ ______ _____
  / ____|    (_) |  (_) |        | |  | |  ____|  __ \
 | (___   ___ _| | ___| |_  ____ | |__| | |__  | |__) |
  \___ \ / __| | |/ / | __| ____ |  __  |  __| |  ___/
  ____) | (__| |   <| | |_       | |  | | |____| |
 |_____/ \___|_|_|\_\_|\__|      |_|  |_|______|_|      0.0.1


Homepage http://scikit-hep.org
GitHub   https://github.com/scikit-hep/scikit-hep
PyPI     https://pypi.python.org/pypi/scikit-hep
```

Thank you