ESCAPE

European Science Cluster of Astronomy &
Particle physics ESFRI research Infrastructures

OSSR Final Workshop

km3py and other open source software contributions from **KM3NeT/FAU**

Tamás Gál

ESCAPE - The European Science Cluster of Astronomy & Particle Physics ESFRI Research Infrastructures has received funding from the European Union’s Horizon 2020 research and innovation programme under the Grant Agreement n° 824064.

Software Packages

- **Focus on data access, micro-services, pipeline management and general purpose software**
- **km3py**
  - **km3io (onboarding started)**: native Python package to access KM3NeT data formats based on CERN/ROOT
  - **km3pipe**: general purpose pipeline framework with KM3NeT related modules, I/O helpers and provenance tracking (spin-off package **thepipe** without KM3NeT specific modules available)
  - **km3astro**: bridge to AstroPy -- KM3NeT specific coordinate transformations (UTM -> Sky), plotting utilities and experiment-specific conventions/definitions related to astronomy
  - **km3services**: microservices prototype infrastructure
  - **openkm3**: Package to use KM3NeT open science products from the KM3NeT Open Data Center
- **Julia based packages -- no onboarding presentation yet, but planned contributions to OSSR**
  - **UnROOT.jl**: Pure Julia package to read the ROOT dataformat including KM3NeT definitions
  - **Corpuscles.jl**: Utility library to access particle properties and identification codes summarised and defined by the Particle Data Group (PDG) collaboration

Funded by the European Union's
Horizon 2020 - Grant N° 824064

KM3NeT Dataformats

- **Internal formats:**
  - custom binary formats for DAQ communication
  - ROOT format to store
    - raw hit data
    - intermediate files in processing chains (calibration, reconstruction, monitoring)
    - high-level data (reconstructed events, summary files)
  - HDF5 conversions available for a subset of data structures.
    Mainly used in
    - machine learning
    - high-level analysis
- **Open data includes:**
  - ROOT (reconstructed events and also hit level data)
  - HDF5 and FITS (reconstructed events and summary files)

ROOT
Data Analysis Framework

HDF

Funded by the European Union's
Horizon 2020 - Grant N° 824064

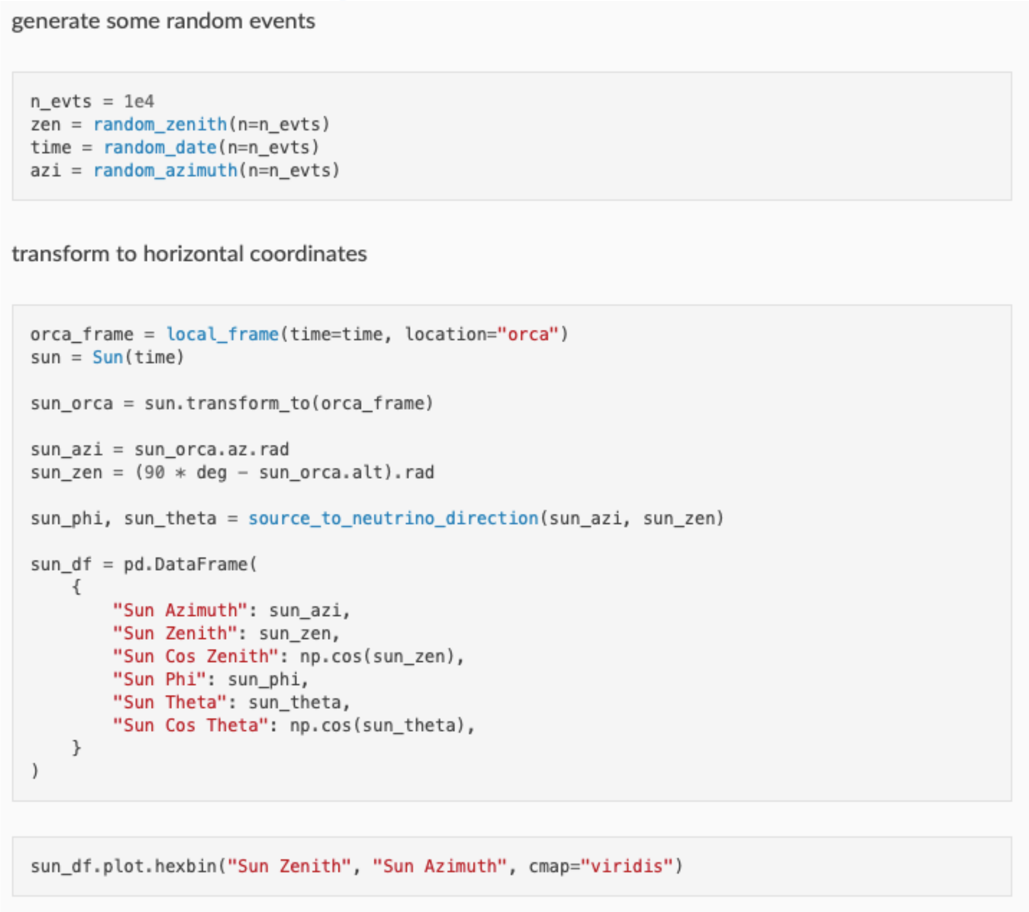
km3io and km3pipe

- **km3io**: provides access to high- and low-level dataformats of KM3NeT
  - without the need of installing large dependencies (ROOT) or KM3NeT specific internal (closed source) software
  - requires technical training to understand the details of data structures but in principle, full data access is granted to parse even internal data formats
  - Depends on Python/AwkwardArray/Numba and is comparable with our C++ framework performance-wise
  - Onboarding is in progress
- **km3pipe**: general purpose pipeline to stitch together data analysis codes
  - Includes access to high-level dataformats (HDF5)
  - Spin-off package of the pipeline-core available as a lightweight package (**thepipe**) including all the pipeline-related features like performance and provenance tracking
- Both packages were already used successfully in a joint analysis of CTA and KM3NeT

Funded by the European Union's
Horizon 2020 - Grant N° 824064

km3astro

A Python library with KM3NeT specific astro-definitions and utilities



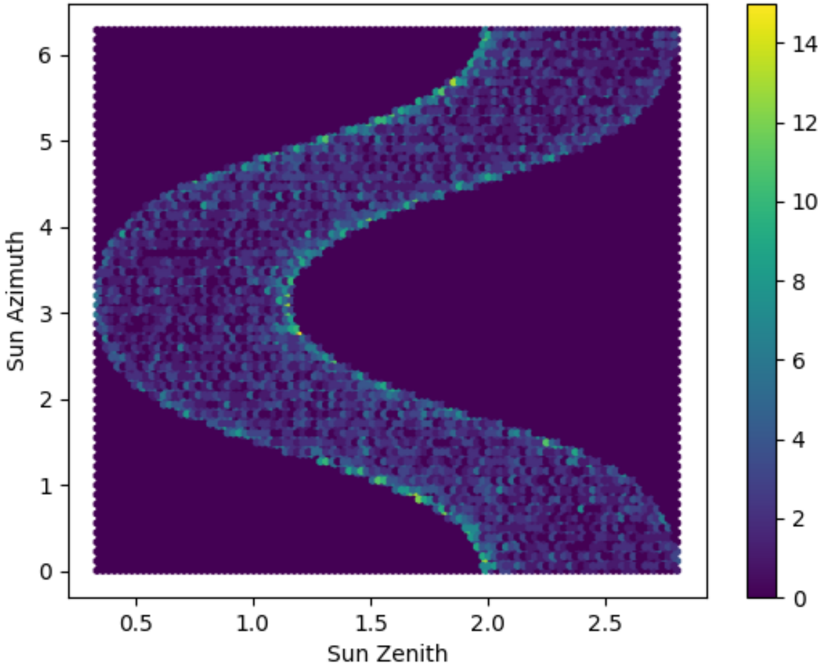
generate some random events

```
n_evts = 1e4
zen = random_zenith(n=n_evts)
time = random_date(n=n_evts)
azi = random_azimuth(n=n_evts)
```

transform to horizontal coordinates

```
orca_frame = local_frame(time=time, location="orca")
sun = Sun(time)

sun_orca = sun.transform_to(orca_frame)

sun_azi = sun_orca.az.rad
sun_zen = (90 * deg - sun_orca.alt).rad

sun_phi, sun_theta = source_to_neutrino_direction(sun_azi, sun_zen)

sun_df = pd.DataFrame(
    {
        "Sun Azimuth": sun_azi,
        "Sun Zenith": sun_zen,
        "Sun Cos Zenith": np.cos(sun_zen),
        "Sun Phi": sun_phi,
        "Sun Theta": sun_theta,
        "Sun Cos Theta": np.cos(sun_theta),
    }
)
```

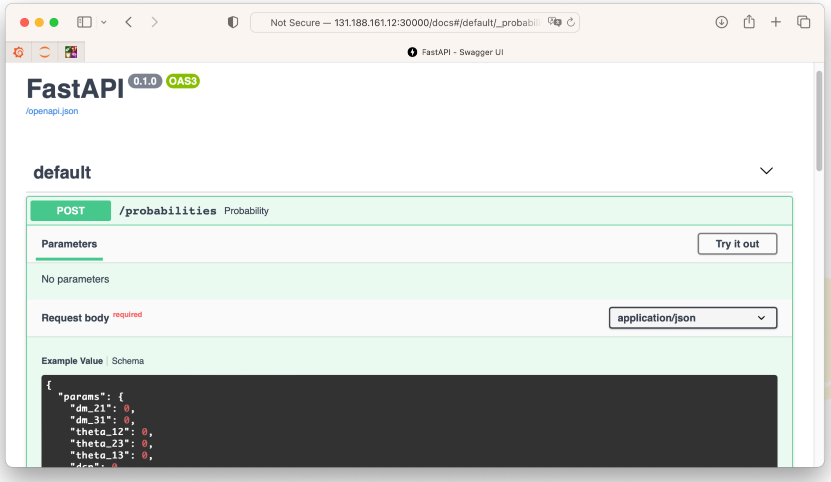
```
sun_df.plot.hexbin("Sun Zenith", "Sun Azimuth", cmap="viridis")
```



Funded by the European Union's
Horizon 2020 - Grant N° 824064

km3services

- Micro-services running as Docker containers e.g. in a Docker swarm or locally
- REST API to interact with the micro-services and send/receive data to/from the contained software
- Already used with success in KM3NeTprocessing pipelines
- Public demo available: calculating neutrino oscillation probabilities using OscProb (https://github.com/joaoabcoelho/OscProb)
  Docker image running with OscProb and all dependencies (ROOT, Eigen, ...) in a KM3NeT Docker swarm hosted at ECAP



Not Secure — 131.188.161.12:30000/docs#/default/_probabil

FastAPI - Swagger UI

FastAPI 0.1.0 OAS3

/openapi.json

default

POST /probabilities Probability

Parameters

Try it out

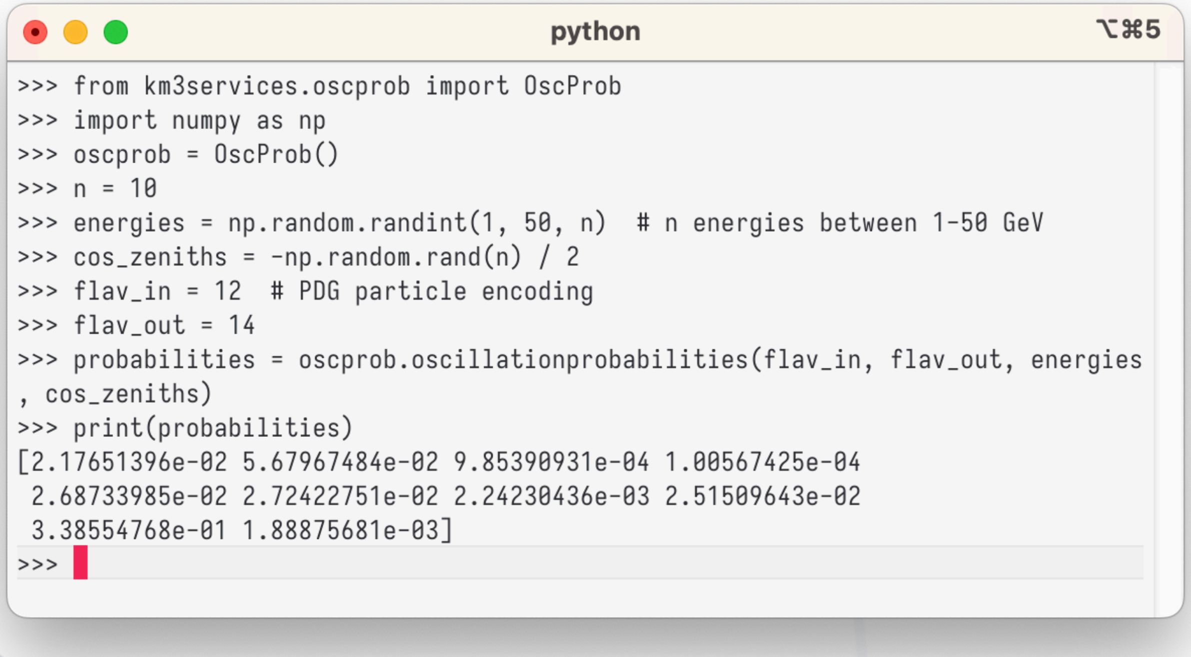
No parameters

Request body required

application/json

Example Value | Schema

```
{
  "params": {
    "dm_21": 0,
    "dm_31": 0,
    "theta_12": 0,
    "theta_23": 0,
    "theta_13": 0,
```

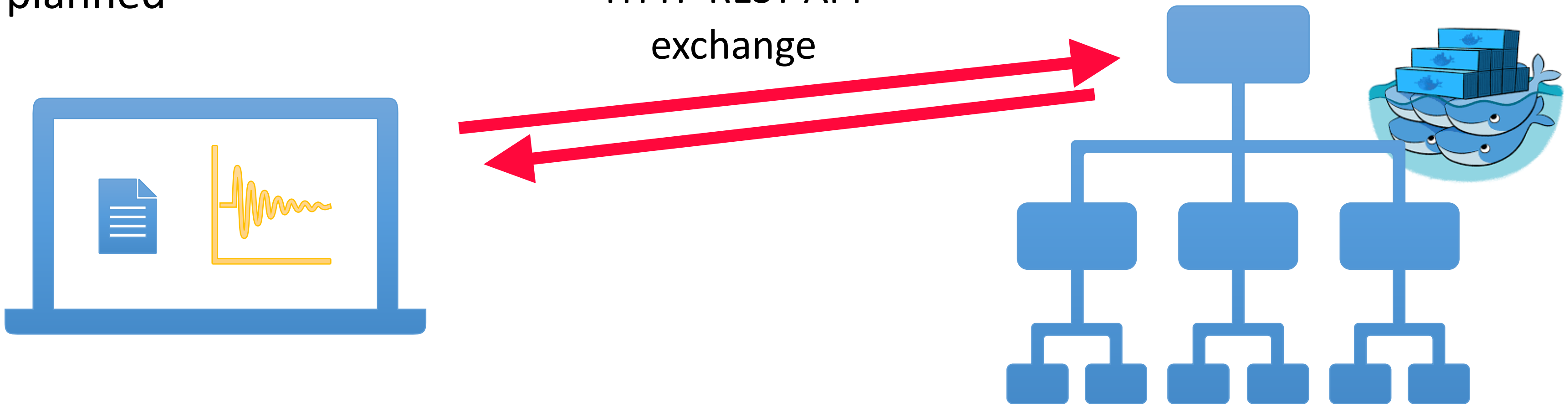


```
>>> from km3services.oscprob import OscProb
>>> import numpy as np
>>> oscprob = OscProb()
>>> n = 10
>>> energies = np.random.randint(1, 50, n)  # n energies between 1-50 GeV
>>> cos_zeniths = -np.random.rand(n) / 2
>>> flav_in = 12  # PDG particle encoding
>>> flav_out = 14
>>> probabilities = oscprob.oscillationprobabilities(flav_in, flav_out, energies
, cos_zeniths)
>>> print(probabilities)
[2.17651396e-02 5.67967484e-02 9.85390931e-04 1.00567425e-04
 2.68733985e-02 2.72422751e-02 2.24230436e-03 2.51509643e-02
 3.38554768e-01 1.88875681e-03]
>>>
```

Funded by the European Union's
Horizon 2020 - Grant N° 824064

km3services

- **pip install km3services** installs a lightweight package which gives access to heavyweight software
- Wraps/unwraps data in/to numpy arrays
- Onboarding planned



HTTP REST API exchange

km3services
entrypoint/load balancer

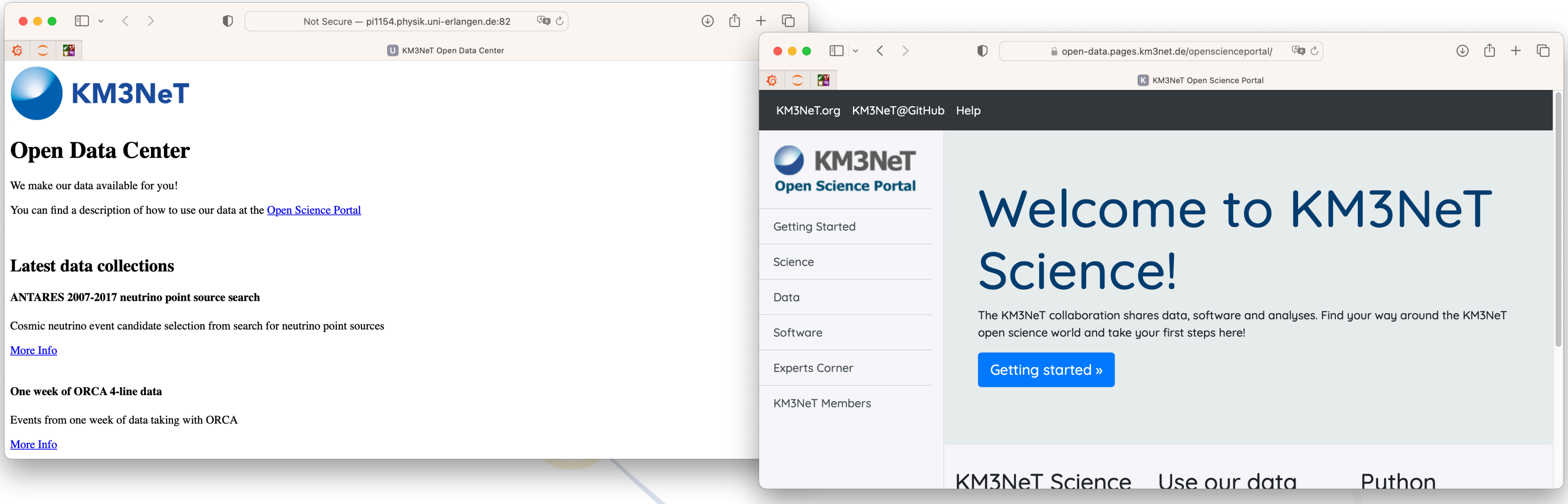
parallel instances of services to distribute load

The implementation is hidden and the service **feels like a regular Python package**

Funded by the European Union's
Horizon 2020 - Grant N° 824064

openkm3

- Python package (from **Jutta Schnabel**) for use of KM3NeT open science products from the **KM3NeT Open Data Center**
- uses numpy, pandas and pyvo as service packages to interpret the various data fromats
- **pip install git+https://git.km3net.de/open-data/openkm3**



Open Data Center

We make our data available for you!

You can find a description of how to use our data at the Open Science Portal

Latest data collections

ANTARES 2007-2017 neutrino point source search

Cosmic neutrino event candidate selection from search for neutrino point sources

More Info

One week of ORCA 4-line data

Events from one week of data taking with ORCA

More Info

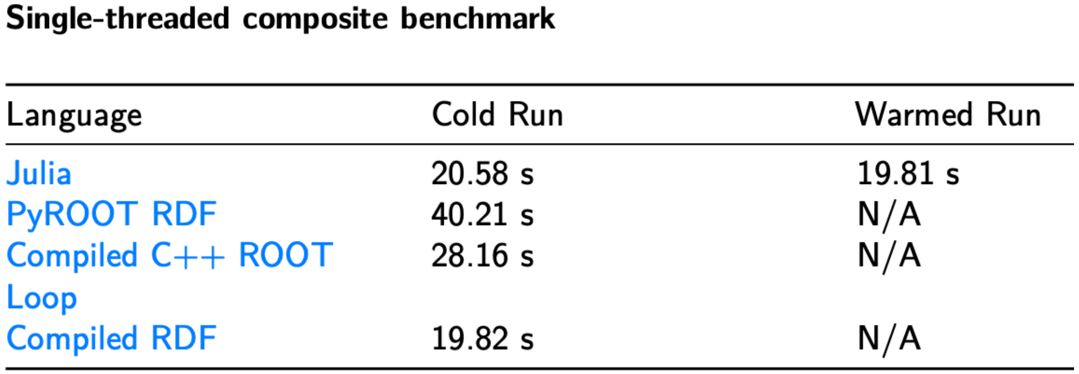
Welcome to KM3NeT Science!

The KM3NeT collaboration shares data, software and analyses. Find your way around the KM3NeT open science world and take your first steps here!

Funded by the European Union's
Horizon 2020 - Grant N° 824064

UnROOT.jl

- Similar to **km3io**, **UnROOT.jl** was created to grant open and lightweight, yet high-performance access to ROOT files
- It was not only focussing on **KM3NeT** related ROOT files but as a general purpose, pure Julia-based I/O library to parse (read) ROOT data
- JOSS paper released on 18th August 2022 (DOI: 10.21105/joss.04452) in collaboration with **Jerry (Jiahong) Ling** from the **Harvard University** and **Nick Amin** from the **University of California, Santa Barbara** who jumped in helped with the development (built-in support for the NANOAOD ROOT specification used in **CMS**)
- Onboarding planned

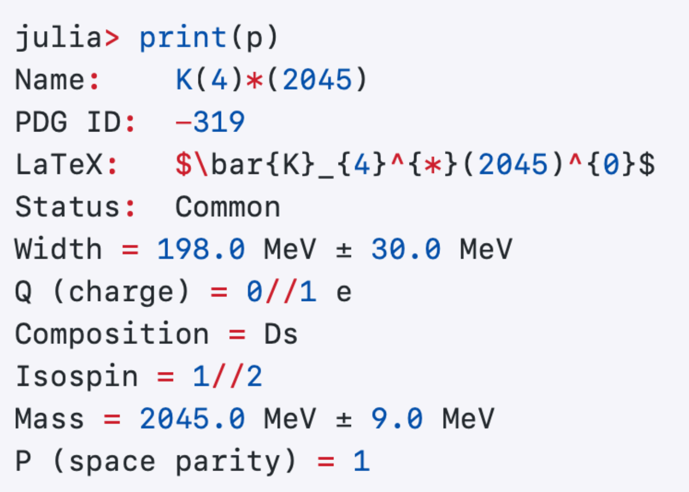


Single-threaded composite benchmark

Language	Cold Run	Warmed Run
Julia	20.58 s	19.81 s
PyROOT RDF	40.21 s	N/A
Compiled C++ ROOT Loop	28.16 s	N/A
Compiled RDF	19.82 s	N/A

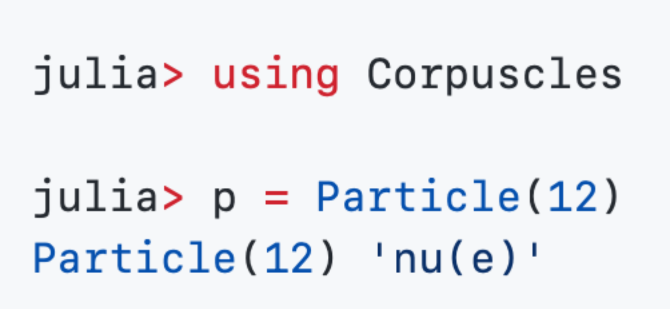
Corpuscles.jl

- access to particle properties and identification codes summarised and defined by the Particle Data Group (PDG) collaboration
- general purpose library to be integrated in low- or high-level analyses
- onboarding planned

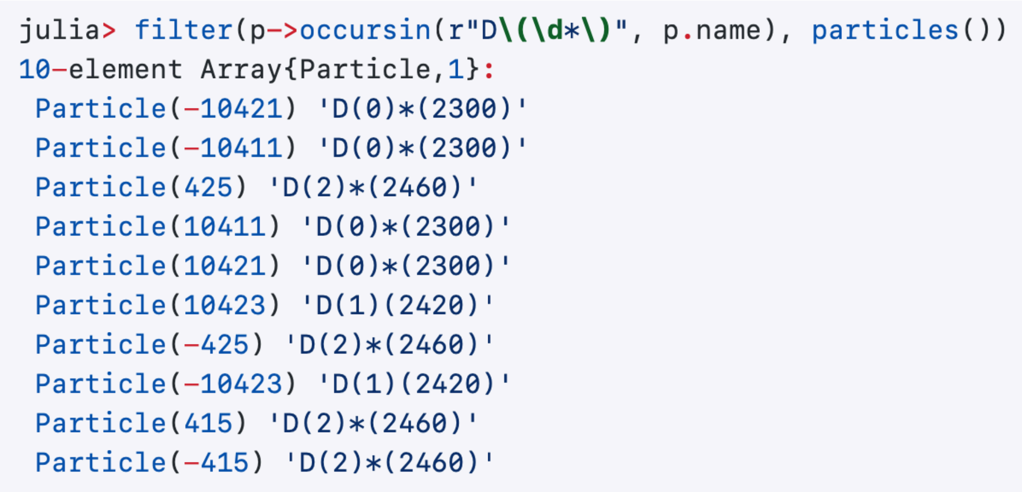


```
julia> print(p)
Name:    K(4)*(2045)
PDG ID:  -319
LaTeX:   $\bar{K}_{4}^{*}(2045)^{0}$
Status:  Common
Width = 198.0 MeV ± 30.0 MeV
Q (charge) = 0//1 e
Composition = Ds
Isospin = 1//2
Mass = 2045.0 MeV ± 9.0 MeV
P (space parity) = 1
```

Corpuscles.jl



```
julia> using Corpuscles

julia> p = Particle(12)
Particle(12) 'nu(e)'
```



```
julia> filter(p->occursin(r"D\(\d*\)", p.name), particles())
10-element Array{Particle,1}:
 Particle(-10421) 'D(0)*(2300)'
 Particle(-10411) 'D(0)*(2300)'
 Particle(425) 'D(2)*(2460)'
 Particle(10411) 'D(0)*(2300)'
 Particle(10421) 'D(0)*(2300)'
 Particle(10423) 'D(1)(2420)'
 Particle(-425) 'D(2)*(2460)'
 Particle(-10423) 'D(1)(2420)'
 Particle(415) 'D(2)*(2460)'
 Particle(-415) 'D(2)*(2460)'
```

Funded by the European Union's
Horizon 2020 - Grant N° 824064