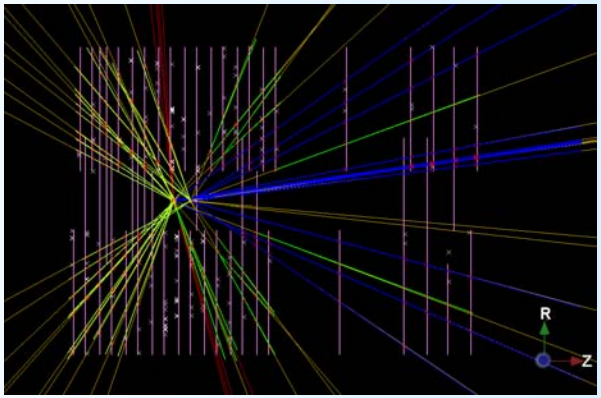
Tracking in Python

E. Rodrigues, NIKHEF



How to get started

Disclaimer

What this tutorial is not

- A course on Python
- A course on GaudiPython
- A technical presentation

What this tutorial is

- An incentive to start exploiting Python in our everyday work
- A presentation a la “learning by examples”
- An informal tutorial

Getting started …

C++ to Python Mapping

:: (the global namespace)	gaudimodule.gbl
Namespace::Class	gaudimodule.gbl.Namespace.Class
object = new Class(…)	object = Class(…)
enum::item	enum.item
Null pointer	None

Examples will be given throughout the next transparencies …

Python Packages available (1/2)

Purely descriptive – examples will follow …

GaudiPython

- **Top-level framework for working with Gaudi applications in Python**
- **Interface of Gaudi to Python**
- **Contains a series of modules:**
  - ***e.g.: gaudimodule:*** **main module, allows instantiation of ApplicationManager**
    - ***GaudiAlgs*** **: for using GaudiAlgorithm, GaudiTupleAlg, etc.**
    - ***units*** **: standards HEP units as in Geant4**

PyRoot

- **Exposes ROOT to Python**
- **RooFit is now also available!**
  - ***ROOT*** **module**

Python Packages available (2/2)

Tr/TrackPython

- Introduced to expose main tracking tools to Python
- Access to extrapolators, fitter, projectors, clone finder, etc.
- More will be added …
- Note: simple module that is likely to evolve as GaudiPython evolves too …
- *gtracktools* module

Event/LinkerInstances

- Facilitates access to main linker classes LinkedTo & LinkedFrom in Event/LinkerEvent
- Easy manipulation of links to MC-truth in Python
- *eventassoc* module

Setting the Environment

Software management

- LHCb software managed via CMT
- CMT sets up the consistent environment for a given application

In the CMT requirements file

```
use GaudiPython v*
```

For using GaudiPython

```
use TrackPython      v*     Tr

use EventInstances   v*     Event
```

For using some tracking tools and the association tables

Some first manipulations

Reading a Gaudi file

Minimum required to read a file

```
>>> import gaudimodule
>>> appMgr = gaudimodule.AppMgr( outputlevel=3,
                                 joboptions='$EVENTSYSROOT/options/PoolDicts.opts' )
>>> SEL = appMgr.evtSel()
>>> SEL.open( 'PFN:rfio:/castor/cern.ch/user/c/cattanem/Boole/v11r5/0601-11144100.digi' )
>>> appMgr.run( 1 )
 >>> EVT = appMgr.evtSvc()
 >>> EVT.dump()
```

Reading only needs dictionaries

OUTPUT OF EVT.dump()

```
/Event
/Event/Gen
/Event/MC
/Event/MC/Header
# some lines skipped ...
/Event/Link/Rec
/Event/Link/Rec/Track
/Event/Link/Rec/Track/Forward
/Event/Link/Rec/Track/RZVelo
/Event/Link/Rec/Track/Velo
# some lines skipped ...
/Event/Rec
/Event/Rec/Header
/Event/Rec/Status
/Event/Rec/Track
/Event/Rec/Track/Forward
/Event/Rec/Track/RZVelo
/Event/Rec/Track/Velo
# some lines skipped ...
```

Content of the TES

(not all content is shown by default …)

Running a Gaudi Application

Example: to run Brunel

```
>>> import gaudimodule
>>> appMgr = gaudimodule.AppMgr( outputlevel=3, joboptions='../options/v200601.opts' )
>>> appMgr.run( 1 )
# feel like having a look at the TES?
>>> EVT = appMgr.evtSvc()
>>> EVT.dump()
```

Making use of interactivity …

```
>>> SEL = appMgr.evtSel()
>>> dir(SEL)
['__class__', '__delattr__', '__dict__', '__doc__', '__getattr__', '__getattribute__', '__hash__', '__init__',
'__module__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__str__', '__weakref__',
'_ip', '_isvc', '_name', '_optsvc', '_svcloc', 'finalize', 'g', 'getInterface', 'initialize', 'isnumber', 'isvector',
'name', 'open', 'properties', 'reinitialize', 'retrieveInterface', 'rewind', 'typecnv']
# say you need more printout of the algorithms/tools names …
>>> MSG = appMgr.service( 'MessageSvc' )
>>> MSG.Format = "% F%60W%S%7W%R%T %0W%M";
>>> SEL.rewind()
# off you go again …
>>> appMgr.run( 1 )
```

Just one possibility …

Retrieving objects from the TES

From now on let us assume we've always run at least 1 event …

```
>>> import gaudimodule
>>> appMgr = gaudimodule.AppMgr( outputlevel=3, joboptions='../options/v200601.opts' )
>>> appMgr.run( 1 )
>>> EVT = appMgr.evtSvc()
```

One might need to load some dictionaries for inspecting objects

```
appMgr.loaddict( 'MCEventDict' )
appMgr.loaddict( 'TrackEventDict' )
```

This will become unnecessary in the future …

```
>>> tracks = EVT[ 'Rec/Track/Forward' ]
>>> print tracks.size()
11
>>> mcps = EVT[ 'MC/Particles' ]
>>> for mcp im mcps:
>>>     print mcp.particleID.pid(), mcp.p()
```

Looping over the container

Tracking in Python

LHCb Note 2006-014 for further explanations

Event Model Classes (1/2)

Our classes are defined in the LHCb namespace

```
>>> Track = gaudimodule.gbl.LHCb.Track
>>> Track
<class '__main__.LHCb::Track'>
>>> defaultTrack = gaudimodule.gbl.LHCb.Track()
>>> defaultTrack
{ chi2PerDoF :  0
nDoF :  0 flags : 0
lhcbIDs :
states :
measurements :
nodes :
 }
```

Get hold of the class definition

Track class instantiation

Track data members

```
appMgr.loaddict( 'TrackFitEventDict' )
appMgr.loaddict( 'LHCbKernelDict' )
```

```
>>> aLineTraj = gaudimodule.gbl.LHCb.LineTraj
>>> aStateTraj = gaudimodule.gbl.LHCb.StateTraj
>>> otMeas = gaudimodule.gbl.LHCb.OTMeasurement()
```

Event Model Classes (2/2)

Playing with the enums …

```
>>> State = gaudimodule.gbl.LHCb.State
>>> State.LocationUnknown
0
>>> State.AtT
5
>>> State.BegRich2
8
>>> Measurement = gaudimodule.gbl.LHCb.Measurement
>>> Measurement.Unknown
0
>>> Measurement.VeloR
1
>>> Measurement.TT
3
```

Neat way of using the enums:
As close as possible to the C++ syntax
e.g.: State::AtT -> State.AtT

Tracking Classes (1/3)

If you ever thought the Track class cannot answer many questions …

```
>>> track = tracks[0]

>>> dir(track)

['AlreadyUsed', 'Backward', 'CnvForward', 'CnvKsTrack', 'CnvMatch', 'CnvSeed', 'CnvVelo',
'CnvVeloBack', 'CnvVeloTT', 'Downstream', 'FitFailed', 'FitUnknown', 'Fitted', 'HistoryUnknown',
'IPSelected', 'Invalid', 'IsA', 'Kalman', 'Long', 'PIDSelected', 'PatForward',  'PatKShort', 'PatRecIDs',
'PatRecMeas', 'PatVelo', 'PatVeloTT', 'ShowMembers', 'StatusUnknown', 'Streamer',
'StreamerNVirtual', 'TrackIdealPR', 'TrackKShort', 'TrackMatching', 'TrackSeeding', 'TrackVeloTT',
'TrgForward', 'TsaTrack', 'Ttrack', 'TypeUnknown', 'Unique', 'Upstream', 'Velo', 'VeloR', '__class__',
'__delattr__',

# a few lines skipped ...

'__weakref__', 'addToAncestors', 'addToLhcbIDs', 'addToMeasurements', 'addToNodes',
'addToStates', 'ancestors', 'charge', 'checkFlag', 'checkHistory', 'checkHistoryFit', 'checkStatus',
'checkType', 'chi2', 'chi2PerDoF', 'clID', 'classID', 'clearAncestors', 'clearStates', 'clone',
'cloneWithKey', 'closestState', 'copy', 'fillStream', 'firstState', 'flag', 'flags', 'hasKey', 'hasStateAt',
'history', 'historyFit', 'index', 'isMeasurementOnTrack', 'isOnTrack', 'key', 'lhcbIDs', 'measurement',
'measurements', 'momentum', 'nDoF', 'nLHCbIDs', 'nMeasurements', 'nMeasurementsRemoved',
'nStates', 'nodes', 'p', 'parent', 'posMomCovariance', 'position', 'positionAndMomentum', 'pt',
'removeFromAncestors', 'removeFromLhcbIDs', 'removeFromMeasurements', 'removeFromNodes',
'removeFromStates', 'reset', 'serialize', 'setChi2PerDoF', 'setFlag', 'setFlags', 'setHistory',
'setHistoryFit', 'setLhcbIDs', 'setNDoF', 'setParent', 'setSpecific', 'setStatus', 'setType', 'slopes',
'specific', 'stateAt', 'states', 'status', 'type']
```

Tracking Classes (2/3)

Operations on the enums

```
>>> track.checkType( Track.Long ), track.type() == Track.Long
(1, True)
>>> track.checkHistory( Track.PatForward )
1
>>> track.checkStatus( Track.Fitted )
0
```

Basic properties

```
>>> track.charge()
1
>>> track.nDoF()
12
```

Track contents

```
>>> track.nStates()
2L
>>> state = track.states()[0]
>>> state.x(), state.y(), state.z()
(-0.0321825934759312, 0.11126547797391403, -14.582833595894137)
```

Tracking Classes (3/3)

The Track can also be modified …

```
>>> newtrack = track.clone()
>>> newtrack.setFlag( Track.Clone, True )
>>> newtrack.checkFlag( Track.Clone )
1
```

What's on the track?

```
>>> print track.nLHCbIDs()
32
>>> print track.nMeasurements()
0
>>> ids = track.lhcbIDs()
>>> ids
<ROOT.vector<LHCb::LHCbID> object at 0xf4eab24>
```

Extrapolators (1/2)

```
# Helper dict. of package with gtracktools module
appMgr.loaddict( 'TrackPythonDict' )
```

```
>>> from gtracktools import setToolSvc, extrapolator
>>> setToolSvc( appMgr )
>>> linprop = extrapolator( 'TrackLinearExtrapolator' )
>>> dir(linprop)
# some lines skipped ...
'__setattr__', '__str__', '__weakref__', 'addRef', 'finalize',
'initialize', 'interfaceID', 'momentum', 'name', 'p', 'parent',
'position', 'positionAndMomentum', 'propagate', 'pt',
'queryInterface', 'release', 'slopes', 'transportMatrix', 'type']
>>>
>>> help(linprop.propagate)
# one gets several lines of documentation
```

Needs to be called only once (may become unnecessary …)

Get hold of the Linear extrapolator

Extrapolators (2/2)

```
# get hold of the Track to be extrapolated
>>> track = tracks[3]
# instantiate a State, to retrieve the result of the extrapolation
>>> state = gaudimodule.gbl.LHCb.State()
>>> state.x(), state.y(), state.z()
(0.0, 0.0, 0.0)
# instantiate the parabolic extrapolator
>>> parprop = extrapolator( 'TrackParabolicExtrapolator' )
# define the z-position to extrapolate to
>>> znew = 100.
>>> parprop.propagate( track, znew, state )
SUCCESS
>>> state.x(), state.y(), state.z()
(-5.859069220236659, -1.5738179596044015, 100.0)
```

```
# "state" variable contains some random State
>>> state.x(), state.y(), state.z()
(491.95847296136429, -39.394353509484759, 9520.0)
>>> newstate = state.clone()
>>> linprop.propagate( newstate, 5000. )
SUCCESS
>>> newstate.x(), newstate.y(), newstate.z()
(72.562676254077573, -21.114433639356392, 5000.0)
```

Checking on Clone Tracks

```
>>> from gtracktools import cloneFinder
>>> CLONEFINDER = cloneFinder( 'TrackCloneFinder' )
```

GETTING HOLD OF THE CLONES FINDER

```
>>> track0 = tracks[0]
>>> track1 = tracks[1]
>>> CLONEFINDER.areClones( track0, track1 ) == True
False
>>> track1 = track0.clone()
>>> CLONEFINDER.areClones( track0, track1 ) == True
True
```

Pick up some 2 tracks

Are they clones?

Fitting a Tracking "from A to Z"

```
>>> from gtracktools import fitter
>>> MASTERFITTER = fitter( 'TrackMasterFitter' )
# one gets some printout at this level ...
```

GETTING HOLD OF THE MASTER FITTER

```
# Fitting tracks with the default options cannot get simpler.
# The "complete job", i.e. fitting and setting of the appropriate flags, only takes a few lines:
>>> track = tracks[0]
>>> sc = MASTERFITTER.fit( track )
>>> if sc == gaudimodule.SUCCESS:
...    track.setStatus( Track.Fitted )
... else:
...    track.setStatus( Track.FitFailed )
...    track.setFlag( Track.Invalid, True )
```

← *Pick up some track and fit it*

← *Done! Set Status flag*

Linker Tables of Tracks (1/2)

Finding the Linker table Track - MCParticle

```
>>> location = 'Rec/Track/Velo'
>>> tracks = EVT[ location ]
>>> from eventassoc import linkedTo
>>> Track = gaudimodule.gbl.LHCb.Track
>>> MCParticle = gaudimodule.gbl.LHCb.MCParticle
>>> LT = linkedTo( MCParticle, Track, location )
>>> LT
<ROOT.LinkedTo<LHCb::MCParticle,LHCb::Track> object at 0xf485460>
>>> LT.notFound() == False
True
```

Get some tracks (e.g. VELO tracks)

Class definitions

Retrieve Linker table

```
>>> track = tracks[7]
>>> mcp = LT.first( track )
>>> mcp.key()
849
>>> mcp
{ momentum :    (-673.62,-416.03,11761.6,11789)
particleID :    { pid : 211
 }
 }
>>> mcp = LT.next()
>>> mcp == None
True
```

Get link with heighest weight

Get next linked MCParticle, if any

Linker Tables of Tracks (2/2)

Going « back and forth» with links …

```
>>> from eventassoc import linkedFrom
>>> LF = linkedFrom( Track,MCParticle,location )
>>> LF
<ROOT.LinkedFrom<LHCb::Track,LHCb::MCParticle> object at 0xf4d6210>
>>> LF.notFound() == False
True
>>> track.key()
7
>>> mcp = LT.first( track )
>>> mcp.key()
849
>>> trk = LF.first( mcp )
>>> trk.key()
7
```

With the looks of the Associators …

```
>>> range = LT.range( track )
>>> range
<ROOT.vector<LHCb::MCParticle*> object at 0xfe27e90>
>>> range.size()
1L
>>> mcp = range[0]
>>> mcp.key()
849
```

Using what is available …

EMACS@LXPLUS064.CERN

File Edit Options Buffers Tools

```
import gaudimodule
from GaudiAlgs import GaudiAlgo

from units import GeV

# ==============================================================================
# Define the histograms
# ==============================================================================
from ROOT import gROOT, TH1F
gROOT.SetBatch()

histo = TH1F( 'long_p', 'P (GeV) for  Long tracks', 100, 0., 100. )

##############################################################################
class LongTrackPAlgo( GaudiAlgo ):
    """Algorithm to plot the momentum of Long tracks from PatForward"""
##############################################################################

# ==============================================================================
    def __init__ ( self , name = 'LongTrackP' ) :
        GaudiAlgo.__init__( self , name )
        print self.name(), '==> Initialize'

# ==============================================================================
    def execute( self ):
        print self.name(), '==> Execute'
        tracks = self.get( 'Rec/Track/Forward' )
        [ histo.Fill( t.p() / GeV ) for t in tracks ]
        return gaudimodule.SUCCESS

# ==============================================================================
    def finalize( self ):
        #print some statistics on tracks properties
        print 'Long tracks P (MeV) : mean', histo.GetMean(),\
              'rms', histo.GetRMS()

        # write out the histogram
        from ROOT import TFile
        f = TFile( 'LongTrackPAlgo.root', 'recreate' )
        histo.Write()
        f.Close()

        return GaudiAlgo.finalize()
```

--:-- LongTrackPAlgo.py (Python ADVANCE Select)--L1--C0--Top----------

C-c-

Python version of GaudiAlgorithm

Standard HEP units

Algorithm definition

Algorithm's main methods

Save the ROOT histo to a file

Second part of the file

All of this will execute if the whole previous file is run as python -i myJob.py (with this bit appended):

```
>>> appMgr = gaudimodule.AppMgr( outputlevel=3, joboptions='../options/v200601.opts' )
>>> appMgr.loaddict( 'TrackEventDict' )
>>> appMgr.loaddict( 'TrackPythonDict' )
>>> EVT  = appMgr.evtSvc()
>>> appMgr.addAlgorithm( LongTrackPAlgo( 'LongTrackP' ) )
>>> appMgr.run( 250 )
>>> from ROOT import TCanvas
>>> histo.Draw()
# not necessary unless you really want to finish everything:
#appMgr.finalize()
# run some more events!
>>> appMgr.run( 100 )
# continue playing interactively …
```

```
# with the « -i » option the user gets back control on the prompt …
>>> f = ROOT.TFile( LongTrackPAlgo.root', 'READ' )
# play with the contents …
```