Table of Contents

Getting started with BenderScript

This *"pseudo-hands-on"* tutorial is an introduction to BenderScript a powerful Bender-based enhancement of GaudiPython

Dedicated tutorials on Bender and GaudiPython could be useful here

BenderScript could be consicered as a kind of *steroid-enhanced and doped* GaudiPython session.

The described funcntinoality corresponds to Bender version `v28r0`

Prerequisites

"First steps in LHCb" starter kit could be considered as a right prerequisite to BenderScript.

IMPORTANT Vladimir Romanovsky is working now on integration of BenderScript with Ganga

Pros and Contras

BenderScript shares all pros and contras with GaudiPython

- It definitely provides very simple and efficient way for exploring the data, in particular very simple way for investigation of the content for the input data and TES. One can easily loop over events, data containers, make simple calcualtions, apply large part of LoKi functors, use some part of DaVinci machinery, etc
- There is large part of tasks that are very difficult in BenderScrip:
    - large part of DaVinci tools is not working. It includes all tools and functionality related to e.g. actions with *the associated best primary vertex*. These manipulations are very complicated or some of them (e.g. PV-refit) is practically impossible to perform in a correct way
    - Many *tools* do not work without significant efforts
    - The results of *tools* often are very difficult to save to be used/reused e.g. for subsequent processing (e.g. save manually created particles and vertices to TES and to output file)
    - Some LoKi functors do not operate (mainly due to the issues from the previous item), e.g **BPV*'** or ***DV** or **DTF_*** - those are not easy to use from plain command line. There are some alternatives , but not for all functors.

Start of (interactive) BenderScript session

To start interactive BenderScript session `bender` command should be used. The command has many command line arguments, that can be inspected using `-h` key:

```
1000> bender -h
```

BenderScript is smart enough and many of the keys be easily deduced from the input data file name. Input data files could be specified via command-line, e.g.

```
1000> bender /lhcb/LHCb/Collision12/PSIX.MDST/00037249/0000/00037249_00000090_1.psix.mdst
```

In this case following keys will be setup

- `-s/--simulation : False`
- `-m/--micro : True`
- `-r/--rootintes : /Event/PSIX`

Choice of the interactive "shell" for BenderScript

On-default BenderScritp starts IPython session, but using command line arguments it could be substituted with on of the alternatives:

- embedded python using `-e/--embed` option
- plain python, using `-y/--plain` option

Alternative could be more appropriate for some tricky cases with python imports, serialization and usage of `subprocess` (depending on combination of versions)

Batch

As another alternative to interactive session, there exist also batch option activated using `-b/--batch` key. In this regime, BenderScript executes all specified (python) input files and exits.

Input data

Input data for BenderScript could be specified in two ways

- via command line arguments
- via *options* files (aka "Configurabales"), imported using `importOptions` directive. E.g. it could be files generated by `lhcb_bkk`

Both way could be combined.

Important:

- *currently* (but it could be changed in future) BenderScript **requires** the presence of at least one input data file either through command line argument of from *options* files (aka *"Configurables"*)

Input data via command line

As input file name many possible semantics are supported:

- plain file name e.g. in local directory, or afs, or somewhere else

```
1000> bender  ./local_file.psix.mdst
```

- `eos` file (unless `--no-castor` command line option is activated)

```
1000> bender /eos/lhcb/grid/prod/lhcb/LHCb/Collision12/PSIX.MDST/00037249/0000/00037249_00000090
```

- `castor` file (unless `--no-castor` command line option is activated)
- fully (or partly) specified input file description

```
1000> bender "DATAFILE='PFN:root://eoslhcb.cern.ch//eos/lhcb/grid/prod/lhcb/LHCb/Collision12/PSI
1001> bender 'PFN:root://eoslhcb.cern.ch//eos/lhcb/grid/prod/lhcb/LHCb/Collision12/PSIX.MDST/000
1002> bender root://eoslhcb.cern.ch//eos/lhcb/grid/prod/lhcb/LHCb/Collision12/PSIX.MDST/00037249
```

- `LFN` ((if `-g/--gris` command line option **and** GRID proxy is active) **or** (no `--no-castor` option is set))

```
1000> bender /lhcb/LHCb/Collision12/PSIX.MDST/00037249/0000/00037249_00000090_1.psix.mdst
```

If neither GRID access nor eos/castro access is available, for resolution of LFN one needs to specify file catalogs, either via options file or command line options `-x/--xml`

Input data via `importOptions`

Input data can be also specified via configuration files using `-i/--import` command line option:

```
1000> bender -i $STRIPPINGSELECTIONSROOT/tests/data/Reco15a_Run164668.py
```

Actions at (i)python prompt

`ls` command

```
In [1]: ls()
 /Event                                                                 DataObject
 /Event/Rec                                                             DataObject
 /Event/Rec/Header                                                      LHCb::RecHeader
 /Event/Rec/Status
 /Event/Rec/Summary
 /Event/Trigger
 /Event/Calo
 /Event/Muon
 /Event/Rich
 /Event/Other
 /Event/Strip
 /Event/PID
 /Event/Dimuon
 /Event/PSIX
Out[1]: SUCCESS
```

The command runs recursively though the *loaded* leaves of the whole Transient Event Store (TES) tree, but it does not force loading of objects. When object is loaded some basic information is printed about it. One can specify a path in TES as an argument for command `ls()`

```
In [2]: ls('/Event/PSIX/Phys')
 /Event/PSIX/Phys                                                       DataObject
 /Event/PSIX/Phys/SelPsiKForPsiX
 /Event/PSIX/Phys/SelPsiPiForPsiX
 /Event/PSIX/Phys/SelPsi2KForPsiX
```

```
 /Event/PSIX/Phys/SelPsi2KPiForPsiX
 /Event/PSIX/Phys/SelPsi2PiForPsiX
 /Event/PSIX/Phys/SelPsi3KForPsiX
 /Event/PSIX/Phys/SelPsi3KPiForPsiX
...
```

The loading of the objects can be forced via `forceload=True` key:

```
In [4]: ls('/Event/PSIX/Phys',forceload = True )
 /Event/PSIX/Phys                                                    DataObject
 /Event/PSIX/Phys/SelPsiKForPsiX                                     DataObject
 /Event/PSIX/Phys/SelPsiKForPsiX/Particles                (empty)    <class 'cppyy.KeyedContai
 /Event/PSIX/Phys/SelPsiPiForPsiX                                    DataObject
 /Event/PSIX/Phys/SelPsiPiForPsiX/Particles               (empty)    <class 'cppyy.KeyedContai
 /Event/PSIX/Phys/SelPsi2KForPsiX                                    DataObject
 /Event/PSIX/Phys/SelPsi2KForPsiX/Particles               (empty)    <class 'cppyy.KeyedContai
 /Event/PSIX/Phys/SelPsi2KPiForPsiX                                  DataObject
 /Event/PSIX/Phys/SelPsi2KPiForPsiX/Particles             (empty)    <class 'cppyy.KeyedContai
 /Event/PSIX/Phys/SelPsi2PiForPsiX                                   DataObject
 /Event/PSIX/Phys/SelPsi2PiForPsiX/Particles              (empty)    <class 'cppyy.KeyedContai
 /Event/PSIX/Phys/SelPsi3KForPsiX                                    DataObject
 /Event/PSIX/Phys/SelPsi3KForPsiX/Particles               (empty)    <class 'cppyy.KeyedContai
 /Event/PSIX/Phys/SelPsi3KPiForPsiX                                  DataObject
 /Event/PSIX/Phys/SelPsi3KPiForPsiX/Particles             (empty)    <class 'cppyy.KeyedContai
...
```

`run` command

Command `run` is used to run over specified number of events. Prrogram runs over events:

```
In [11]: run(1)
Out[11]: SUCCESS
In [12]: run(100)
Out[12]: SUCCESS
In [13]: run(10000)
DaVinciInitAlg       INFO Memory has changed from 2110344 to 2501336 KB (390992KB, 18.5274%) in 1
EventSelector     SUCCESS Reading Event record 1001. Record number within stream 1: 1001
EventSelector     SUCCESS Reading Event record 2001. Record number within stream 1: 2001
ONLINE_2012.Dat...   INFO Connecting to database
DDDB.DataBaseOp...   INFO Connecting to database
DQFLAGS.DataBas...   INFO Connecting to database
EventSelector     SUCCESS Reading Event record 3001. Record number within stream 1: 3001
EventSelector     SUCCESS Reading Event record 4001. Record number within stream 1: 4001
EventSelector     SUCCESS Reading Event record 5001. Record number within stream 1: 5001
EventSelector     SUCCESS Reading Event record 6001. Record number within stream 1: 6001
EventSelector     SUCCESS Reading Event record 7001. Record number within stream 1: 7001
EventSelector     SUCCESS Reading Event record 8001. Record number within stream 1: 8001
EventSelector     SUCCESS Reading Event record 9001. Record number within stream 1: 9001
EventSelector     SUCCESS Reading Event record 10001. Record number within stream 1: 10001
Out[13]: SUCCESS
```

`skip` command

Command `skip` is used to *skip* processing of certain events. Actually it disables all known algorithms, and invokes command `run` and then re-enables all algorithms disabled earlier

```
In [1]: run(1)
Out[1]: SUCCESS
In [2]: skip(1000)
EventSelector     SUCCESS Reading Event record 1001. Record number within stream 1: 1001
Out[2]: SUCCESS
```

```
In [3]: skip(1000)
EventSelector     SUCCESS Reading Event record 2001. Record number within stream 1: 2001
Out[3]: SUCCESS
```

`rewind` command

Command `rewind` allows to "rewind" input data at the start:

```
In [4]: rewind()
EventSelector        INFO Stream:EventSelector.DataStreamTool_1 Def:DATAFILE='PFN:root://eoslhcb.
In [5]: run(1)
EventSelector     SUCCESS Reading Event record 1. Record number within stream 1: 1
Out[5]: SUCCESS
```

warnings

- sometimes it does not rewind properly if several input files have been opened.
- *Currently* (but it could be changed in future) command `run(1)` is **required** after command `rewind`

.

`get` command

Command `get` allows to get data from certain TES location:

```
In [9]: r = get('/Event/PSIX/Phys/SelB2ChicPiPiForPsiX0/Particles')
In [10]: print r
 0 |->B_s0                          M/PT/E/PX/PY/PZ: 4.8287/ 2.4833/ 95.65/ -2.36/-0.7732/ 95.5  [
                                        EndVertex  X/Y/Z:0.3624/0.0754/ 50.2  [mm]  Chi2/nDoF  18.
 1    |->chi_c1(1P)               M/PT/E/PX/PY/PZ: 3.7328/ 1.6591/ 61.8 /-1.385/-0.9127/ 61.67 [
                                        EndVertex  X/Y/Z:0.3749/ 0.041/ 50.11 [mm]  Chi2/nDoF 0.55
 2       |->J/psi(1S)              M/PT/E/PX/PY/PZ: 3.109 / 1.8157/ 57.85/-1.703/-0.6294/ 57.74 [
                                        EndVertex  X/Y/Z:0.3749/ 0.041/ 50.11 [mm]  Chi2/nDoF 0.55
 3          |->mu+                 M/PT/E/PX/PY/PZ: 0.1057/ 1.5739/ 18.12/-1.103/ 1.123/ 18.05 [G
 3          |->mu-                 M/PT/E/PX/PY/PZ: 0.1057/ 1.8517/ 39.74/-0.5996/-1.752/ 39.7  [
 2       |->gamma                  M/PT/E/PX/PY/PZ:   0   / 0.424 / 3.948/0.3165/-0.2822/ 3.925 [
 1    |->pi-                       M/PT/E/PX/PY/PZ: 0.1396/ 0.2581/ 16.29/-0.2547/-0.0418/ 16.29
 1    |->pi+                       M/PT/E/PX/PY/PZ: 0.1396/ 0.7438/ 17.56/-0.7207/0.1838/ 17.54 [
```

Data can be transformed of filtered "on-flights", e.g. get the first element of container pf particles:

```
In [17]: o = get('/Event/PSIX/Phys/SelPionForPsiX0/Particles', lambda s : s.containedObjects()[0]
In [18]: print o
 0 |->pi-                          M/PT/E/PX/PY/PZ: 0.1396/ 0.2581/ 16.29/-0.2547/-0.0418/ 16.29
```

apply some filtering on elements of container:

```
In [21]: o = get('/Event/PSIX/Phys/SelPionForPsiX0/Particles', PT > 0.5 * GeV  )
In [22]: o
Out[22]:
 0 |->pi+                          M/PT/E/PX/PY/PZ: 0.1396/ 0.7438/ 17.56/-0.7207/0.1838/ 17.54 [
```

`seek` family of commands

`seekForData`

This function allows to loop over events lookinng some data at given location. If data corresponds to container of elemements, containenr is required to be non-empty

```
In [33]: data,nevt = seekForData('/Event/PSIX/Phys/SelPsi3KPiForPsiX/Particles')
In [34]: print nevt
18
In [35]: print data
 0 |->B-                          M/PT/E/PX/PY/PZ: 5.3023/ 1.8854/ 93.63/ 1.674/0.8679/ 93.46 [G
                                      EndVertex  X/Y/Z:0.7403/0.1503/-9.709 [mm]  Chi2/nDoF  20.
 1    |->J/psi(1S)               M/PT/E/PX/PY/PZ: 3.115 / 0.7981/ 57.15/0.6018/-0.5243/ 57.06 [
                                      EndVertex  X/Y/Z: 0.746/0.1629/-10.07 [mm]  Chi2/nDoF  3.3
 2       |->mu+                  M/PT/E/PX/PY/PZ: 0.1057/ 1.8768/ 41.69/ 1.815/-0.4762/ 41.65 [
 2       |->mu-                  M/PT/E/PX/PY/PZ: 0.1057/ 1.2144/ 15.46/-1.214/-0.0473/ 15.41 [
 1    |->K-                      M/PT/E/PX/PY/PZ: 0.4937/ 0.4866/ 16.11/-0.003/0.4866/ 16.09 [G
 1    |->pi-                     M/PT/E/PX/PY/PZ: 0.1396/ 0.7754/ 13.45/0.3801/0.6759/ 13.42 [G
 1    |->pi+                     M/PT/E/PX/PY/PZ: 0.1396/ 0.7367/ 6.933/0.7024/0.2223/ 6.893 [G
 0 |->B-                          M/PT/E/PX/PY/PZ: 5.2865/ 0.8967/ 85.17/0.8484/0.2904/  85   [G
                                      EndVertex  X/Y/Z: 0.724/0.1516/ -9.94 [mm]  Chi2/nDoF  31.
 1    |->J/psi(1S)               M/PT/E/PX/PY/PZ: 3.115 / 0.7981/ 57.15/0.6018/-0.5243/ 57.06 [
                                      EndVertex  X/Y/Z: 0.746/0.1629/-10.07 [mm]  Chi2/nDoF  3.3
 2       |->mu+                  M/PT/E/PX/PY/PZ: 0.1057/ 1.8768/ 41.69/ 1.815/-0.4762/ 41.65 [
 2       |->mu-                  M/PT/E/PX/PY/PZ: 0.1057/ 1.2144/ 15.46/-1.214/-0.0473/ 15.41 [
 1    |->K-                      M/PT/E/PX/PY/PZ: 0.4937/ 0.4866/ 16.11/-0.003/0.4866/ 16.09 [G
 1    |->pi-                     M/PT/E/PX/PY/PZ: 0.1396/ 0.4535/ 4.983/-0.4422/0.1007/ 4.96  [
 1    |->pi+                     M/PT/E/PX/PY/PZ: 0.1396/ 0.7367/ 6.933/0.7024/0.2223/ 6.893 [G
```

It returns a tuple of two elements:

1. `data` : the data found
2. `nevt` number of events looped till success

Maximal looping length is controlled by the second argument (default is 1000 events)

`seekStripDecision`

`seekForODIN`

`seekForEvtRun`

`seekAlgDecision`

`seekforVoidDecision`

`writeEvent`: writing "interesting" events into separate output file

"Interesting" events can be copied into separate outptu file using command `writeEvent`

```
> writeEvent()
```

This command is activated if the output file name is provided for BenderScript using command line option `-o/--output`

```
bender ....   -o interesting_events.dst
```

Examples of basic actions:

Using the basic functions, described above one can easily code very simple *analysis*, e.g. one can loop over certain selected candiates for certain event and prints the decays of candidates found:

```
In [49]: def my_fun() :
     data,nevt = seekForData('/Event/PSIX/Phys/SelPsi3KPiForPsiX/Particles' , 1000 )
     print '#evnt %d, #B %d' % ( nevt , len(data) )
```

```
     for b in data : print b.decay()
     writeEvent()
     run(1)
   ....:
In [50]:
In [50]: for i in range(3) : my_fun()
#evnt 17, #B 1
 ( B- ->  ( J/psi(1S) ->  mu+  mu-  )  K+  pi-  pi-  )
#evnt 7, #B 1
 ( B- ->  ( J/psi(1S) ->  mu+  mu-  )  K+  pi-  pi-  )
#evnt 0, #B 4
 ( B+ ->  ( J/psi(1S) ->  mu+  mu-  )  K+  pi+  pi-  )
 ( B+ ->  ( J/psi(1S) ->  mu+  mu-  )  K+  pi+  pi-  )
 ( B+ ->  ( J/psi(1S) ->  mu+  mu-  )  K+  pi+  pi-  )
 ( B- ->  ( J/psi(1S) ->  mu+  mu-  )  K+  pi-  pi-  )
```

Objects in BenderScript

Most popular objects, in particular particles (both reconstructed, `LHCb::Particle` and simulated, `MC::Particle` and `HepMC::Particle`), vertices, tracks, protoparticles, calorimeter clusters and hypos, as well as various containers (including corresponding `SmartTefVectors`) are *decorated* for more friendly (interactive) operations. One can e.g. compare "bare" GaudiPython with BenderScript:

```
> python   ## start plain python session
Python 2.7.10 (default, Nov 27 2015, 20:46:47)
[GCC 4.9.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import cppyy
>>> cpp = cppyy.makeNamespace('')
>>> LHCb = cpp.LHCb
>>> p = LHCb.Particle( LHCb.ParticleID( 11))
>>> p
<ROOT.LHCb::Particle object at 0xbdaae10>
>>>
```

In BenderScript printout of objects is more informative (see above), and even for "empty" newly created particle is is more detailed:

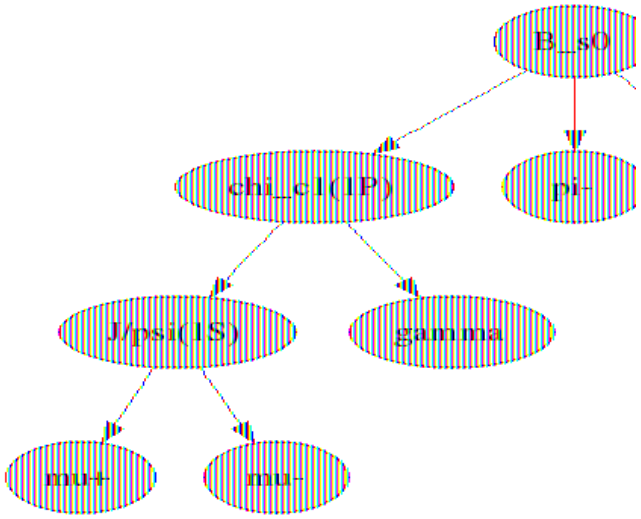
```
In [1]: p = LHCb.Particle(LHCb.ParticleID(11))
In [2]: p
Out[2]:
 0 |->e-                          M/PT/E/PX/PY/PZ:   1   /   0   /  -1  /   0  /   0  /   0   [G
```

One can compare the output of `help(LHCb.Paricle)` and `dir(LHCb.Particle)` commands in GaudiPython and BenderScript and easily see a lot of newly added functions. E.g. for "bare" GaudiPython `len(dir(LHCb.Particle))` reports **181**, while in BenderScript context it reports **237**. Important: all newly added mehtods are documented and command `help` provide good desription:

```
In [2]: data, nevt = seekForData('/Event/PSIX/Phys/SelB2ChicPiPiForPsiX0/Particles')
In [3]: help(data.view)
Help on method view in module LoKiPhys.graph:

view(particle, command=None, format='png') method of cppyy.KeyedContainer<LHCb::Particle,Containe
    Prepare the graph and vizualize it
    p = ...
    p.view ( format  = 'png' )
    p.view ( format  = 'png' , commmand = 'eog' )
```

Execution of this command will result in the following graph:



In a similar way many other classes are decorated to be more friendly for usage with Bender and BenderScript.

"Decoration" also include simple intuitive looping over containers (E.g. automaticlaly resolving `SmartRef`s), easy selection of various daughter particles. tracks, etc:

```
In [15]: B
Out[15]:
 0 |->B_s0                        M/PT/E/PX/PY/PZ: 4.8287/ 2.4833/ 95.65/ -2.36/-0.7732/ 95.5  [
                                     EndVertex  X/Y/Z:0.3624/0.0754/ 50.2  [mm]  Chi2/nDoF  18.
 1    |->chi_c1(1P)               M/PT/E/PX/PY/PZ: 3.7328/ 1.6591/ 61.8 /-1.385/-0.9127/ 61.67 [
                                     EndVertex  X/Y/Z:0.3749/ 0.041/ 50.11 [mm]  Chi2/nDoF 0.55
 2       |->J/psi(1S)             M/PT/E/PX/PY/PZ: 3.109 / 1.8157/ 57.85/-1.703/-0.6294/ 57.74 [
                                     EndVertex  X/Y/Z:0.3749/ 0.041/ 50.11 [mm]  Chi2/nDoF 0.55
 3          |->mu+                M/PT/E/PX/PY/PZ: 0.1057/ 1.5739/ 18.12/-1.103/ 1.123/ 18.05 [G
 3          |->mu-                M/PT/E/PX/PY/PZ: 0.1057/ 1.8517/ 39.74/-0.5996/-1.752/ 39.7  [
 2       |->gamma                  M/PT/E/PX/PY/PZ:   0   / 0.424 / 3.948/0.3165/-0.2822/ 3.925 [
 1    |->pi-                       M/PT/E/PX/PY/PZ: 0.1396/ 0.2581/ 16.29/-0.2547/-0.0418/ 16.29
 1    |->pi+                       M/PT/E/PX/PY/PZ: 0.1396/ 0.7438/ 17.56/-0.7207/0.1838/ 17.54 [

In [16]: for c in B.children() : print c.name()
chi_c1(1P)
pi-
pi+

In [19]: for c in B.children( LoKi.Child.Selector( PT > 1 * GeV ))  : print c.name()
B_s0
chi_c1(1P)
J/psi(1S)
mu+
mu-

In [23]: for t in B.tracks() : print t.pt() / GeV
1.57387760282
1.85170771471
0.258078675215
0.743781922842

In [30]: for t in B.tracks( 'mu+' == ABSID ) : print t.pt() / GeV
1.57387760282
1.85170771471
```

Histograms&Tuples: `nTuple` and `book`

Surely one can rely on native ROOT histograms and TTrees, but for some cases it could have sense to rely on Gaudi's abilities to maintain histograms and tuples/trees. Gaudi's machinery requires the files for histograms and n-tuples/trees to be separated and to be declared at configuration time. it could be done e.g. in one of configuration files

```
from Configurables import DaVinci
dv = DaVinci ( HistogramFile = 'MyHistos.root' ,
               TupleFile = 'MyTuples.root' )
```

These configuration files can be read by bender using `-i/--import` option:

```
> bender .....  -i MyConf.py
```

Alternatively the names of files for histograms and n-tuples/trees can be specified via command line otpions `--histofile` and `-t/--tuplefile`:

```
> bender .....  --histofile MyHistos.py -t MyTuples.root
```

If the files for histogram and n-tuples are defined, the histos and tuples can be used inn very simple way:

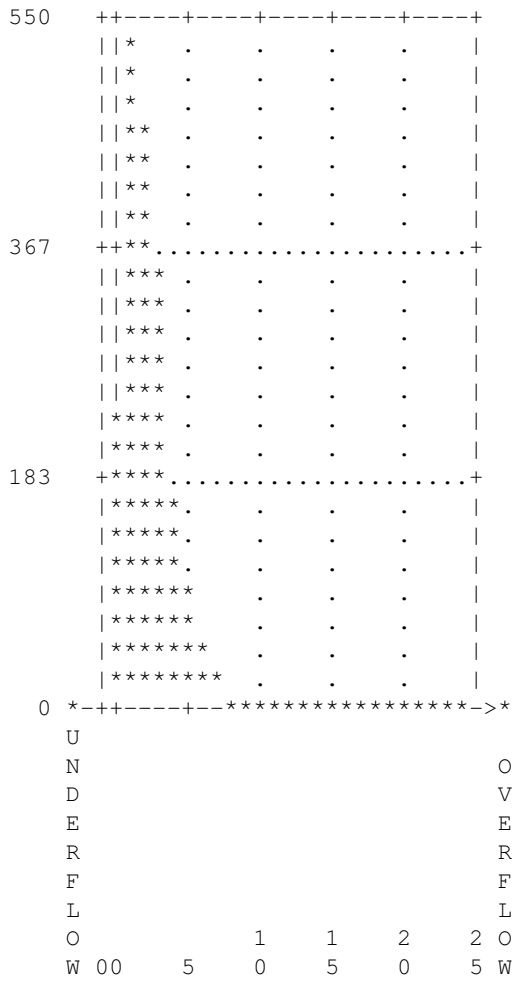
```
In [2]: t = nTuple('QQQ','MyTuple')
# Bender.Fixes_Gaudi        INFO    Booked n-tuple Tuples::Tuple(/NTUPLES/FILE1/QQQ/MyTuple)
In [3]: data,nevt = seekForData('/Event/PSIX/Phys/SelPsi3KPiForPsiX/Particles' , 1000 )
In [4]: for b in data :
   ...:     t.column_float ( 'pt' , PT ( b ) )
   ...:     t.column_int   ( 'id' , int ( ID ( b ) ) )
   ...:     t.column_float ( 'mass' , M ( b ) )
   ...:     t.write()
In [5]: run(1)
Out[5]: SUCCESS
In [6]: for b in data :
    t.column_float ( 'pt' , PT ( b ) )
    t.column_int   ( 'id' , int ( ID ( b ) ) )
    t.column_float ( 'mass' , M ( b ) )
    t.write()
```

In a similar way one deals with histograms:

```
In [9]: h1 = book ( 'some_path_here' , 'title' , 100 , 0 , 100 )
In [10]: data,nevt = seekForData('/Event/PSIX/Phys/SelPsi3KPiForPsiX/Particles' , 1000 )
In [11]: for b in data : h1.fill  ( PT ( b ) / GeV )
```

The given histogram can be "dumped" as *pseudographic*, e.g. for quick inspection:

```
In [2]: print h.dump(30,10)
 Histo TES   : "the_path"
 Histo Title : "title"
 Mean        :      2.8911 +- 0.04689
 Rms         :      2.1031 +- 0.08553
 Skewness    :      2.4874 +- 0.05453
 Kurtosis    :      11.309 +- 0.1088
 Entries     :
 |    All    |  In Range | Underflow |  Overflow | #Equivalent |   Integral  |    Total    |
 |    2012   |    2012   |     0     |     0     |     2012    |     2012    |     2012    |
 Annotation
 | Title                    : title                                          |
 | title                    : title                                          |
 | id                       : the_path                                       |
```


550
367
183
0
UNDERFLOW
OVERFLOW
00
5
10
15
20
25

Execution of scripts

BenderScript can execute several kinds of python scripts

1. self-contained scripts
2. scripts that are executed in context (controlled with `-w/--with-context` command line option)

The first one is very trivial case. To code such scripts one needs to use many `import` operations. The second kind of scripts is executed in a context that is identical to the context available in interactive BenderScript shell - all functions, symbols, etc are available. In this way the script can be incredibly compact and one can just copy the lines from `,bender_history` file directly into the script

e.g. one can code following python file, lets call it `bscript.py`

```
t = nTuple('QQQ','Tuple')
def action() :
    data,nevt = seekForData('/Event/PSIX/Phys/SelPsi3KPiForPsiX/Particles')
    print '#B %s / #evt %d' % ( len(data) , nevt )
    for b in data :
        t.column_float ( 'pt'   , PT  ( b ) / GeV )
        t.column_int   ( 'id'   , int ( ID ( b ) ) )
        t.column_float ( 'mass' , M   ( b ) / GeV )
        t.write()
    run(1)
## event loop
for i in range(10) :
    action()
```

And then this file can be executed with BenderScript

```
> bender  /lhcb/LHCb/Collision12/PSIX.MDST/00037249/0000/00037249_00000090_1.psix.mdst  bscript.p
```

Operations with *Tools*, *Services* and *Algorithms*

Services

Many *services* are available directly from `AppMgr` instance:

- Algorithm Context Service: `cntxSvc`
- Detector Data Service: `detSvc`
- Event Data Service: `evtSvc`
- Event Selector Service: `evtSel`
- File Record Service: `filerecordsvc`
- Histogram Service: `histSvc`
- Incident Service: `incSvc`
- N-Tuple Service: `ntupleSvc`
- (LHCb) Particle Property Service: `ppSvc`
- Tool service: `toolSvc`
- ...

```
In [4]: gaudi.ppSvc()
Out[4]: <PartProp.Service.iParticlePropertySvc at 0x7ff644866110>
In [5]: gaudi.evtSvc()
Out[5]: <GaudiPython.Bindings.iDataSvc at 0x7ff644866310>
In [6]: gaudi.toolSvc()
Out[6]: <GaudiPython.Bindings.iToolSvc at 0x7ff644866210>
```

Some of them accessible direclty form BenderScript prompt, e.g. `evtSvc`, `detSvc`, `toolSvc`, and some others

```
In [7]: toolSvc
Out[7]: <GaudiPython.Bindings.iToolSvc at 0x7ff659e4d050>
In [8]: detSvc
Out[8]: <GaudiPython.Bindings.iDataSvc at 0x7ff65acfffd0>
In [9]: evtSvc
Out[9]: <GaudiPython.Bindings.iDataSvc at 0x7ff65acffc50>
```

Many of them have much improved with respect to bare GaudiPython interface. All *new* and *improved* methods are well documented, use `help` command to get more information.

To get certain generic service by name one can use `AppMgr.service`method:

```
In [2]: mysvc = gaudi.service('MessageSvc')
In [3]: mysvc
Out[3]: <GaudiPython.Bindings.iService at 0x7f82e4ed7050>
```

Tools

Acquiring tools

Getting certain tools from *Tool Service* is easy:

```
In [2]: tool = toolSvc.tool('DaVinci::ParticleTransporter', interface  = cpp.IParticleTransporter
In [3]: tool
Out[3]: iAlgTool('DaVinci::ParticleTransporter',interface=IParticleTransporter)
```

Warning concerning usage of *private* tools from python prompt:

- *aqcuiring* of *private* tools form the command prompt has not much sense, and could be very fragile

- *using* of *private* tools via python prompt is very fragile and unsafe

Using tools

```
In [4]: tool = toolSvc.tool('DaVinci::ParticleTransporter', interface  = cpp.IParticleTransporter
In [5]: data, nevt = seekForData('/Event/PSIX/Phys/SelB2ChicPiPiForPsiX0/Particles')
In [6]: B = data[0]
In [7]: print B
 0 |->B_s0                          M/PT/E/PX/PY/PZ: 4.7175/ 0.9365/ 51.63/0.5916/-0.726/ 51.4  [G
                                        EndVertex  X/Y/Z:0.6053/0.1641/-25.71 [mm]  Chi2/nDoF  18.
 1    |->chi_c1(1P)               M/PT/E/PX/PY/PZ: 3.5344/ 0.9029/ 33.37/0.4004/-0.8092/ 33.17 [
                                        EndVertex  X/Y/Z:0.5967/0.1681/-25.71 [mm]  Chi2/nDoF 0.03
 2       |->J/psi(1S)              M/PT/E/PX/PY/PZ: 3.1004/ 0.4001/ 29.79/ 0.183/-0.3558/ 29.63 [
                                        EndVertex  X/Y/Z:0.5967/0.1681/-25.71 [mm]  Chi2/nDoF 0.03
 3          |->mu+                 M/PT/E/PX/PY/PZ: 0.1057/ 1.7593/ 16.71/0.8563/-1.537/ 16.61 [G
 3          |->mu-                 M/PT/E/PX/PY/PZ: 0.1057/ 1.3595/ 13.08/-0.6735/ 1.181/ 13.01 [
 2       |->gamma                  M/PT/E/PX/PY/PZ:   0   / 0.5039/ 3.577/ 0.218/-0.4543/ 3.541 [
 1    |->pi-                       M/PT/E/PX/PY/PZ: 0.1396/ 0.3579/ 9.828/-0.3188/0.1625/ 9.82  [
 1    |->pi+                       M/PT/E/PX/PY/PZ: 0.1396/ 0.5121/ 8.431/0.5051/-0.0849/ 8.415 [
In [8]: B.referencePoint()
Out[8]: ( 0.605291, 0.164137, -25.7105)
In [9]: transported_B = LHCb.Particle()

In [11]: tool.transport( B , B.referencePoint().Z() + 5*mm , transported_B )
Out[11]: SUCCESS
In [12]: transported_B
Out[12]:
 0 |->B_s0                          M/PT/E/PX/PY/PZ: 4.7175/ 0.9365/ 51.63/0.5916/-0.726/ 51.4  [G
                                        EndVertex  X/Y/Z:0.6053/0.1641/-25.71 [mm]  Chi2/nDoF  18.
 1    |->chi_c1(1P)               M/PT/E/PX/PY/PZ: 3.5344/ 0.9029/ 33.37/0.4004/-0.8092/ 33.17 [
                                        EndVertex  X/Y/Z:0.5967/0.1681/-25.71 [mm]  Chi2/nDoF 0.03
 2       |->J/psi(1S)              M/PT/E/PX/PY/PZ: 3.1004/ 0.4001/ 29.79/ 0.183/-0.3558/ 29.63 [
                                        EndVertex  X/Y/Z:0.5967/0.1681/-25.71 [mm]  Chi2/nDoF 0.03
 3          |->mu+                 M/PT/E/PX/PY/PZ: 0.1057/ 1.7593/ 16.71/0.8563/-1.537/ 16.61 [G
 3          |->mu-                 M/PT/E/PX/PY/PZ: 0.1057/ 1.3595/ 13.08/-0.6735/ 1.181/ 13.01 [
 2       |->gamma                  M/PT/E/PX/PY/PZ:   0   / 0.5039/ 3.577/ 0.218/-0.4543/ 3.541 [
 1    |->pi-                       M/PT/E/PX/PY/PZ: 0.1396/ 0.3579/ 9.828/-0.3188/0.1625/ 9.82  [
 1    |->pi+                       M/PT/E/PX/PY/PZ: 0.1396/ 0.5121/ 8.431/0.5051/-0.0849/ 8.415 [

In [13]: transported_B.referencePoint()
Out[13]: ( 0.662837, 0.0935148, -20.7105)
```

Manipulaitons with tools

Tools and their properties

Get and (re)set properties:

```
In [12]: tool.OutputLevel
Out[12]: 1
In [13]: tool.OutputLevel = 3
In [14]: tool.OutputLevel
Out[14]: 3
```

Warning redefinition of certain properties for certain tools is taken into account only before tool is initialized.

Get list of all declared properties fro the given tool:

```
In [15]: tool.properties()
Out[15]:
{'AuditFinalize': <GaudiPython.Bindings.PropertyEntry at 0x7f3d40905c90>,
```

```
 'AuditInitialize': <GaudiPython.Bindings.PropertyEntry at 0x7f3d40905e90>,
 'AuditStart': <GaudiPython.Bindings.PropertyEntry at 0x7f3d40905bd0>,
 'AuditStop': <GaudiPython.Bindings.PropertyEntry at 0x7f3d40905a10>,
 'AuditTools': <GaudiPython.Bindings.PropertyEntry at 0x7f3d40905e10>,
 'Context': <GaudiPython.Bindings.PropertyEntry at 0x7f3d40905ed0>,
 'ContextService': <GaudiPython.Bindings.PropertyEntry at 0x7f3d4090f510>,
 'CounterList': <GaudiPython.Bindings.PropertyEntry at 0x7f3d4090f210>,
 'EfficiencyRowFormat': <GaudiPython.Bindings.PropertyEntry at 0x7f3d40905cd0>,
 'ErrorsPrint': <GaudiPython.Bindings.PropertyEntry at 0x7f3d40905d10>,
 'Extrapolator1': <GaudiPython.Bindings.PropertyEntry at 0x7f3d4090f190>,
 'Extrapolator2': <GaudiPython.Bindings.PropertyEntry at 0x7f3d4090f050>,
 'GlobalTimeOffset': <GaudiPython.Bindings.PropertyEntry at 0x7f3d40905dd0>,
 'MeasureCPUPerformance': <GaudiPython.Bindings.PropertyEntry at 0x7f3d4090f250>,
 'MonitorService': <GaudiPython.Bindings.PropertyEntry at 0x7f3d418fdad0>,
 'OutputLevel': <GaudiPython.Bindings.PropertyEntry at 0x7f3d40905f90>,
 'Particle2State': <GaudiPython.Bindings.PropertyEntry at 0x7f3d4090f150>,
 'PropertiesPrint': <GaudiPython.Bindings.PropertyEntry at 0x7f3d40905f50>,
 'RegularRowFormat': <GaudiPython.Bindings.PropertyEntry at 0x7f3d40905fd0>,
 'RootInTES': <GaudiPython.Bindings.PropertyEntry at 0x7f3d40905d50>,
 'StatEntityList': <GaudiPython.Bindings.PropertyEntry at 0x7f3d4090f110>,
 'StatPrint': <GaudiPython.Bindings.PropertyEntry at 0x7f3d40905c50>,
 'StatTableHeader': <GaudiPython.Bindings.PropertyEntry at 0x7f3d40905f10>,
 'TrackStateProvider': <GaudiPython.Bindings.PropertyEntry at 0x7f3d4090f2d0>,
 'TrajectoryRegion': <GaudiPython.Bindings.PropertyEntry at 0x7f3d4090f490>,
 'TypePrint': <GaudiPython.Bindings.PropertyEntry at 0x7f3d40905b90>,
 'UseEfficiencyRowFormat': <GaudiPython.Bindings.PropertyEntry at 0x7f3d4090f390>}
In [16]: tool.properties().keys()
Out[16]:
['StatTableHeader',
 'ErrorsPrint',
 'MonitorService',
 'RootInTES',
 'Particle2State',
 'AuditFinalize',
 'RegularRowFormat',
 'UseEfficiencyRowFormat',
 'ContextService',
 'AuditTools',
 'StatEntityList',
 'AuditInitialize',
 'OutputLevel',
 'TrajectoryRegion',
 'TypePrint',
 'StatPrint',
 'AuditStop',
 'Context',
 'PropertiesPrint',
 'GlobalTimeOffset',
 'Extrapolator1',
 'Extrapolator2',
 'MeasureCPUPerformance',
 'TrackStateProvider',
 'AuditStart',
 'EfficiencyRowFormat',
 'CounterList']
```

For tools, inherited from **`GaudiTool`** (that accounts a bulk of tools used in LHCb) some additional functionality is available:

```
In [18]: tool.PropertiesPrint = True
ToolSvc.DaVinci...SUCCESS List of ALL properties of DaVinci::ParticleTransporter/ToolSvc.DaVinci:
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'MeasureCPUPerformance':False
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'TrajectoryRegion':( -300.00000 , 1000.0000
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'Extrapolator2':TrackRungeKuttaExtrapolator
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'Extrapolator1':TrackParabolicExtrapolator:
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'Particle2State':Particle2State:PUBLIC
```

```
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'TrackStateProvider':TrackStateProvider:PUB
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'ContextService':AlgContextSvc
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'StatEntityList':[  ]
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'CounterList':[ '.*' ]
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'UseEfficiencyRowFormat':True
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'EfficiencyRowFormat': |*%|-48.48s|%|50t||%
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'RegularRowFormat': | %|-48.48s|%|50t||%|10
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'StatTableHeader': |    Counter
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'GlobalTimeOffset':0.0000000
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'RootInTES':
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'Context':
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'TypePrint':True
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'StatPrint':True
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'PropertiesPrint':True
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'ErrorsPrint':True
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'AuditFinalize':False
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'AuditStop':False
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'AuditStart':False
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'AuditInitialize':False
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'AuditTools':False
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'OutputLevel':3
ToolSvc.DaVinci...SUCCESS Property ['Name': Value] =  'MonitorService':MonitorSvc
```

(and similar for **`tool.StatPrint = True`**)

Algorithms

One can get list of (already) created algorithms from **`AlgorithmManager`**

```
In [22]: gaudi.algorithms()
Out[22]:
['DaVinciEventSeq',
 'DaVinciInitAlg',
 'FilteredEventSeq',
 'LumiSeq',
 'DaVinciEventInitSeq',
 'DaVinciAnalysisSeq',
 'KillDAQ',
 'DaVinciUserSequence',
 'MonitoringSequence',
 'EventAccount']
```

The algorithm can be picked up by name:

```
In [6]: alg = gaudi.algorithm('DaVinciInitAlg')
In [7]: alg
Out[7]: <GaudiPython.Bindings.iAlgorithm at 0x7fc422714a10>
```

All functionality described above is also applicable for *algorithms*, e.g.

```
In [8]: alg.PropertiesPrint = True
DaVinciInitAlg    SUCCESS List of ALL properties of DaVinciInit/DaVinciInitAlg  #properties = 46
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'PrintEvent':False
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'MinMemoryDelta':16
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'MemoryPurgeLimit':3400000
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'Increment':1000
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'EvtCounter':EvtCounter
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'PrintEventTime':False
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'PrintFreq':1
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'PreloadGeometry':False
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'SingleSeed':False
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'SkipFactor':0
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'RequireObjects':[  ]
```

```
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'VetoObjects':[  ]
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'StatEntityList':[  ]
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'CounterList':[ '.*' ]
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'UseEfficiencyRowFormat':True
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'EfficiencyRowFormat': |*%|-48.48s|%|50t||%
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'RegularRowFormat': | %|-48.48s|%|50t||%|10
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'StatTableHeader': |    Counter
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'GlobalTimeOffset':0.0000000
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'RootInTES':
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'Context':
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'TypePrint':True
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'StatPrint':True
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'PropertiesPrint':True
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'ErrorsPrint':True
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'NeededResources':[  ]
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'Cardinality':1
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'IsClonable':False
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'RegisterForContextService':True
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'MonitorService':MonitorSvc
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'AuditStop':True
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'AuditStart':True
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'AuditEndRun':True
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'AuditBeginRun':True
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'AuditFinalize':True
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'AuditExecute':True
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'AuditRestart':True
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'AuditReinitialize':True
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'AuditInitialize':True
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'AuditAlgorithms':True
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'DataOutputs':
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'DataInputs':
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'ErrorCounter':0
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'ErrorMax':1
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'Enable':True
DaVinciInitAlg    SUCCESS Property ['Name': Value] =  'OutputLevel':3
```

histograms and counters

```
[ibelyaev@lbcsrv01]~/cmtuser/BenderDev_HEAD% bender -t MyTuples.root --histo MyHistos.root /lhcb/
```

Lets *implicitly* call few algorithms (via *Data-on-Demand Service*)

```
In [3]: for i in range(0,20) :
    run(1)
    print len(get('/Event/Phys/StdAllLoosePions/Particles'))
```

... and then inspect the algorithm:

```
In [4]: pions = gaudi.algorithm('StdAllLoosePions')
In [5]: pions.Counters()
Out[5]: {'#accept': #=20      Sum=20          Mean=     1.000 +- 0.0000     Min/Max=     1.000/1.
In [7]: for c in pions.Counters() : print c
#accept
In [8]: print pions.Counters('#accept')
#=20      Sum=20          Mean=     1.000 +- 0.0000     Min/Max=     1.000/1.000
```

In a similar way if *algorithm* (or *tool*) internally creates histograms using **`GaudiHisto`** approach) they can be extracted and inspected using the method **`Histos()`**

-- VanyaBelyaev - 2016-01-29

This topic: LHCb > BenderScriptTutorial
Topic revision: r8 - 2016-02-06 - VanyaBelyaev