

# IceTray-free I3File Access

Alex Olivas

# Plotting Cascades

```
#!/usr/bin/env python

import argparse
parser = argparse.ArgumentParser(description = "Plot cascade energies.")
parser.add_argument('-i', '--infile', dest = 'INFILE')
args = parser.parse_args()

from icecube import dataio

infile = dataio.I3File(args.INFILE)
```

# Plotting Cascades

```
#!/usr/bin/env python
```

```
import argparse
parser = argparse.ArgumentParser(description = "Plot cascade energies.")
parser.add_argument('-i', '--infile', dest = 'INFILE')
args = parser.parse_args()
```

```
from icecube import dataio
```

```
infile = dataio.I3File(args.INFILE)
```

**Not necessary, but nice.**

**Be nice to your collaborators.**

**If that's not good enough, be nice to your future self.**

# Plotting Cascades

```
#!/usr/bin/env python

import argparse
parser = argparse.ArgumentParser(description = "Plot cascade energies.")
parser.add_argument('-i', '--infile', dest = 'INFILE')
args = parser.parse_args()
```

```
from icecube import dataio

infile = dataio.I3File(args.INFILE)
```

← Import dataio because that's where the I3File lives.

# Plotting Cascades


```
#!/usr/bin/env python

import argparse
parser = argparse.ArgumentParser(description = "Plot cascade energies.")
parser.add_argument('-i', '--infile', dest = 'INFILE')
args = parser.parse_args()

from icecube import dataio
infile = dataio.I3File(args.INFILE)

for frame in infile:
    print frame
```

Loop over the frames and print them.



# Plotting Cascades

```
#!/usr/bin/env python
```

```
import argparse
parser = argparse.ArgumentParser(description = "Plot cascade energies.")
parser.add_argument('-i', '--infile', dest = 'INFILE')
args = parser.parse_args()

from icecube import dataio, dataclasses
infile = dataio.I3File(args.INFILE)

cascade_energies = list()
for frame in infile:
    if "I3MCTree" in frame:
        for particle in frame["I3MCTree"]:
            if particle.is_cascade:
                e = particle.energy
                cascade_energies.append(e)
```

Take a look at I3MCTree docs

<http://software.icecube.wisc.edu/documentation/projects/dataclasses/i3mctree.html>

# Plotting Cascades

```
#!/usr/bin/env python
```

```
import argparse
parser = argparse.ArgumentParser(description = "Plot cascade energies.")
parser.add_argument('-i', '--infile', dest = 'INFILE')
args = parser.parse_args()

from icecube import dataio, dataclasses
infile = dataio.I3File(args.INFILE)

cascade_energies = list()
for frame in infile:
    if "I3MCTree" in frame:
        for particle in frame["I3MCTree"]:
            if particle.is_cascade:
                e = particle.energy
                cascade_energies.append(e)
```

Take a look at I3Particle docs.

<http://software.icecube.wisc.edu/documentation/projects/dataclasses/particle.html>

# Plotting Cascades

```
#!/usr/bin/env python
```

```
import argparse
parser = argparse.ArgumentParser(description = "Plot cascade energies.")
parser.add_argument('-i', '--infile', dest = 'INFILE')
args = parser.parse_args()
```

```
from icecube import dataio, dataclasses
import pylab
```

```
infile = dataio.I3File(args.INFILE)
```

```
cascade_energies = list()
for frame in infile:
    if "I3MCTree" in frame:
        for particle in frame["I3MCTree"]:
            if particle.is_cascade:
                e = particle.energy
                cascade_energies.append(e)
```

```
xmin = 0 # What units?
xmax = 100 # What units?
pylab.hist(cascade_energies, range=(xmin,xmax), log=True, histtype='step')
pylab.title("Cascade Energies in Who-Knows-What Units")
pylab.xlabel("E(???)")
pylab.show()
```

- 1) import pylab, matplotlib, PyROOT, etc...
- 2) Make the plot.



```
#!/usr/bin/env python
```

```
import argparse  
parser = argparse.ArgumentParser(description = "Plot cascade energies.")  
parser.add_argument('-i', '--infile', dest = 'INFILE')  
args = parser.parse_args()
```

```
from icecube import dataio, dataclasses  
import pylab
```

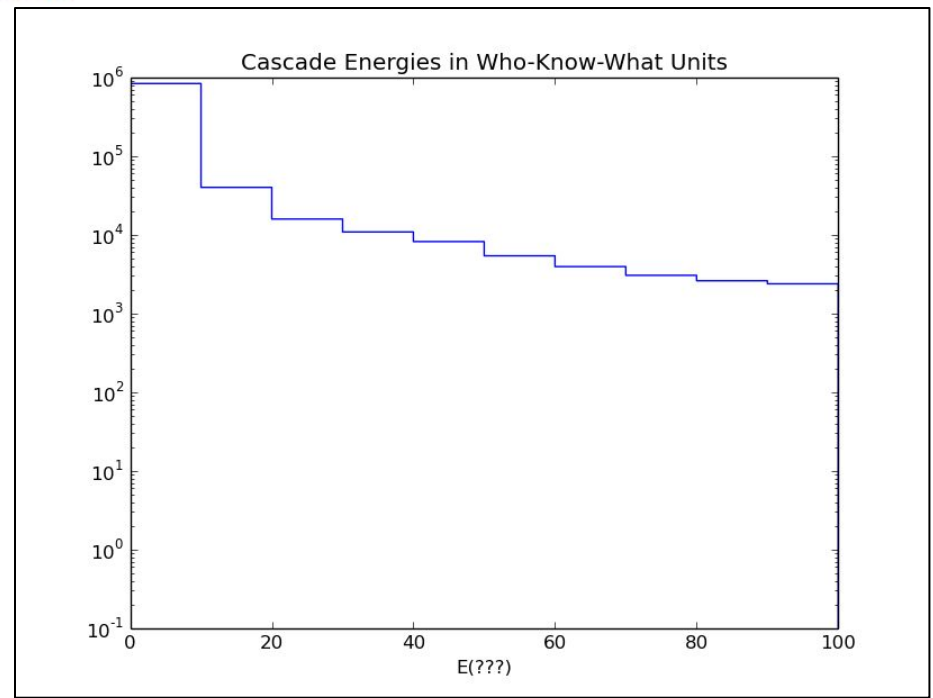
```
infile = dataio.I3File(args.INFILE)
```

```
cascade_energies = list()  
for frame in infile:  
    if "I3MCTree" in frame:  
        for particle in frame["I3MCTree"]:  
            if particle.is_cascade:  
                e = particle.energy  
                cascade_energies.append(e)
```

```
xmin = 0 # What units?  
xmax = 100 # What units?
```

```
pylab.hist(cascade_energies, range=(xmin,xmax), log=True, histtype='step')  
pylab.title("Cascade Energies in Who-Knows-What Units")  
pylab.xlabel("E(???)")  
pylab.show()
```

# Plotting Cascades



# I3Units

```
#!/usr/bin/env python
```

```
import argparse
parser = argparse.ArgumentParser(description = "Plot cascade energies.")
parser.add_argument('-i', '--infile', dest = 'INFILE')
args = parser.parse_args()

from icecube import dataio, dataclasses
from icecube.icetray import I3Units
import pylab

infile = dataio.I3File(args.INFILE)

cascade_energies = list()
for frame in infile:
    if "I3MCTree" in frame:
        for particle in frame["I3MCTree"]:
            if particle.is_cascade:
                e = particle.energy/I3Units.GeV
                cascade_energies.append(e)

xmin = 0 * I3Units.GeV
xmax = 100 * I3Units.GeV
pylab.hist(cascade_energies, range=(xmin,xmax), log=True, histtype='step')
pylab.title("Cascade Energies")
pylab.xlabel("E(GeV)")
pylab.show()
```

**import I3Units.**



# I3Units

```
#!/usr/bin/env python
```

```
import argparse
parser = argparse.ArgumentParser(description = "Plot cascade energies.")
parser.add_argument('-i', '--infile', dest = 'INFILE')
args = parser.parse_args()

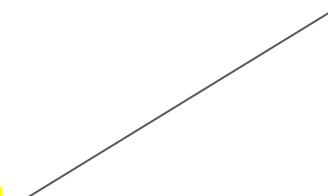
from icecube import dataio, dataclasses
from icecube.icetray import I3Units
import pylab

infile = dataio.I3File(args.INFILE)

cascade_energies = list()
for frame in infile:
    if "I3MCTree" in frame:
        for particle in frame["I3MCTree"]:
            if particle.is_cascade:
                e = particle.energy/I3Units.GeV
                cascade_energies.append(e)

xmin = 0 * I3Units.GeV
xmax = 100 * I3Units.GeV
pylab.hist(cascade_energies, range=(xmin,xmax), log=True, histtype='step')
pylab.title("Cascade Energies")
pylab.xlabel("E(GeV)")
pylab.show()
```

**Convert an energy  
to whatever units I  
specify.**



```
e = particle.energy/I3Units.GeV
```

# I3Units

```
#!/usr/bin/env python
```

```
import argparse
parser = argparse.ArgumentParser(description = "Plot cascade energies.")
parser.add_argument('-i', '--infile', dest = 'INFILE')
args = parser.parse_args()

from icecube import dataio, dataclasses
from icecube.icetray import I3Units
import pylab

infile = dataio.I3File(args.INFILE)

cascade_energies = list()
for frame in infile:
    if "I3MCTree" in frame:
        for particle in frame["I3MCTree"]:
            if particle.is_cascade:
                e = particle.energy/I3Units.GeV
                cascade_energies.append(e)

xmin = 0 * I3Units.GeV
xmax = 100 * I3Units.GeV

pylab.hist(cascade_energies, range=(xmin,xmax), log=True, histtype='step')
pylab.title("Cascade Energies")
pylab.xlabel("E(GeV)")
pylab.show()
```

**I'm going to follow  
IceCube convention  
and specify units for  
any bald numbers.**

<http://software.icecube.wisc.edu/documentation/projects/icetray/i3units.html>