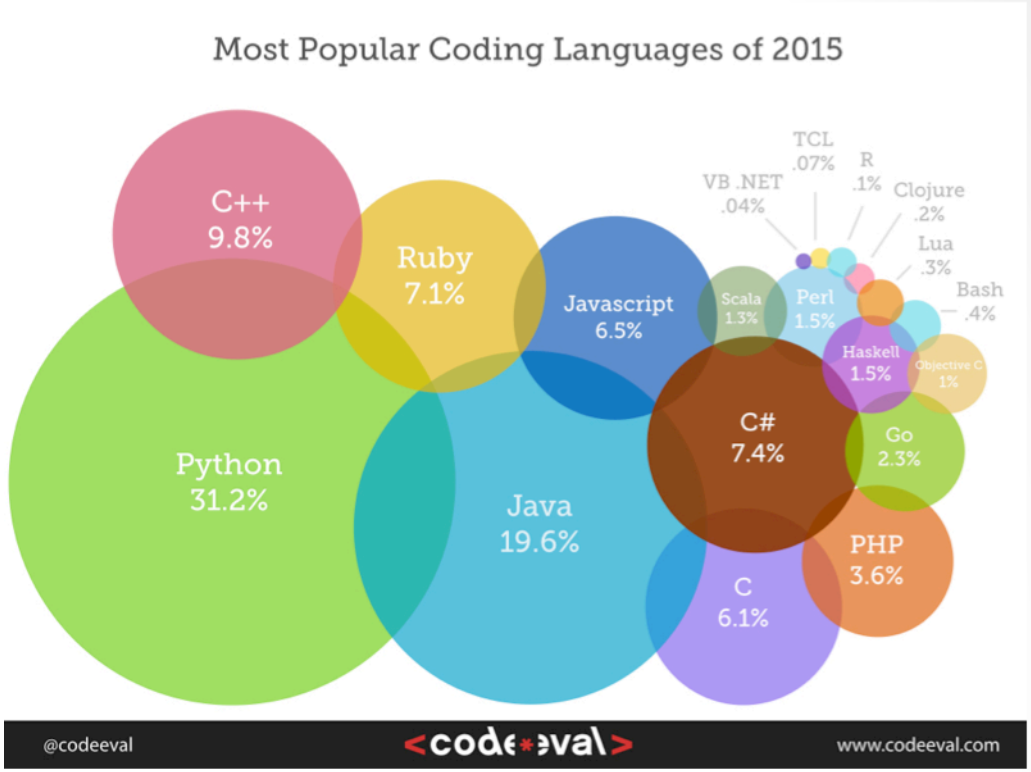
FCC analysis software

Colin Bernet, Clement Helsens

Why python

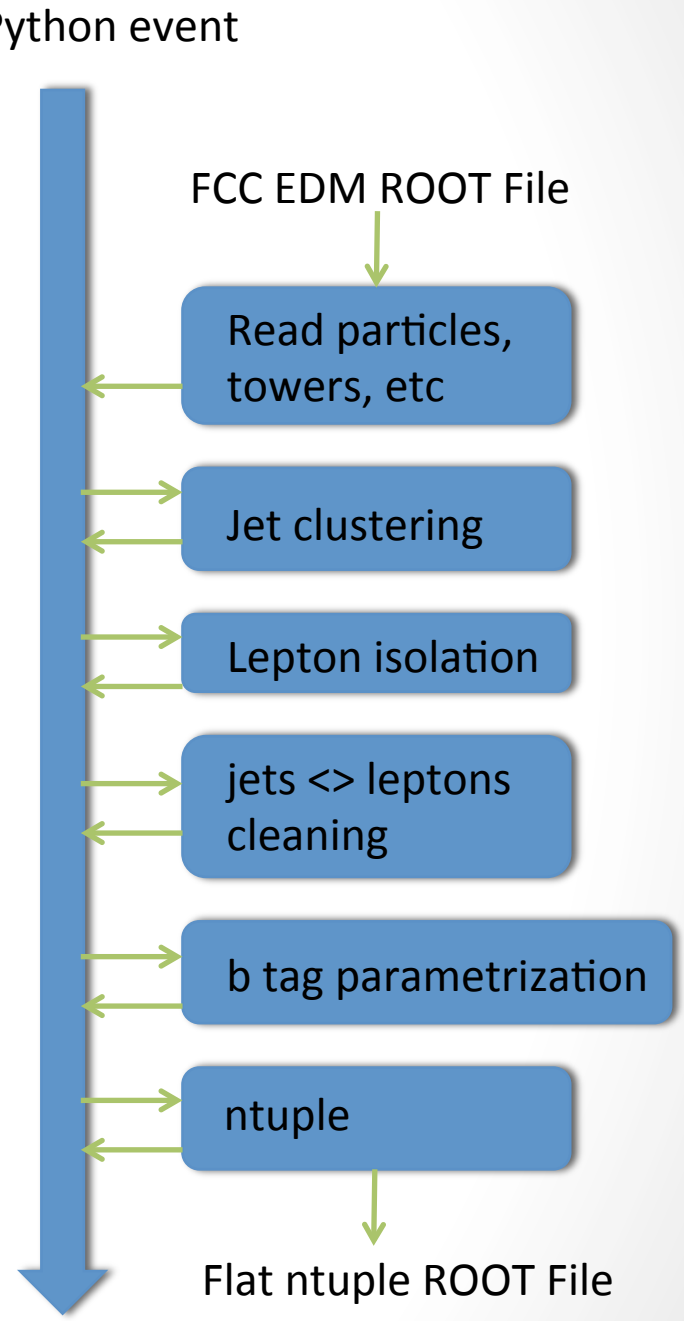


- Super easy to learn
- Light & short code
- Good performance
  - usually wraps C or C++ modules
- « Batteries included »
  - massive and easy-to-use standard library

- Dynamic typing
  - good for multi-channel analyses
  - code highly reusable
- Dynamic object modification
  - Can attach new attributes (or methods) to an existing object
- Productivity x 5-10 w/r C++
- Forget about ROOT CINT
- A lot of fun!

Heppy

- Advanced framework for your PyROOT macros
- Design ~ Athena, CMSSW, Gaudi, Marlin
- Goals:
  - high-level reco & selection
  - write out flat ntuple or histograms for statistical analysis



Heppy usage

- 50 users
- CMS analyses
  - Higgs:
    - H→ZZ→4l
    - H→tau tau
    - ttH→multileptons
    - W/Z H→bb
  - W mass
  - Susy
    - fully hadronic
    - 1 lepton
    - multilepton

- Can read any event format
  - CMS EDM
  - FCC EDM
  - Plain ROOT (pheno studies)
  - Soon: ATLAS, LCIO for ILC/CLIC

- Write transparent code for several experiments

https://github.com/HEP-FCC/heppy

Example

```python
from heppy.framework.analyzer import Analyzer
from heppy.particles.tlv.resonance import Resonance2 as Resonance

import pprint
import itertools

mass = {23: 91, 25: 125}

class ResonanceBuilder(Analyzer):

    def process(self, event):
        legs = getattr(event, self.cfg_ana.leg_collection)
        resonances = []
        for leg1, leg2 in itertools.combinations(legs,2):
            resonances.append( Resonance(leg1, leg2, self.cfg_ana.pdgid) )
        # sorting according to distance to nominal mass
        nominal_mass = mass[self.cfg_ana.pdgid]
        resonances.sort(key=lambda x: abs(x.m()-nominal_mass))
        setattr(event, self.cfg_ana.output, resonances)
        # getting legs of best resonance
        legs = []
        if len(resonances):
            legs = resonances[0].legs
        setattr(event, '_'.join([self.cfg_ana.output, 'legs']), legs)
```

Typical analyzer code…

…and configuration

```python
# Building Zeds
# help(ResonanceBuilder) for more information
from heppy.analyzers.ResonanceBuilder import ResonanceBuilder
zeds = cfg.Analyzer(
    ResonanceBuilder,
    output = 'zeds',
    leg_collection = 'sel_iso_leptons',
    pdgid = 23
)
```

A first FCC-hh use case

- Aim:
  - Provide an analysis skeleton
- Use ttbar as a complete example:
  - Use jets > 30GeV
  - Use isolated electrons/muons > 30GeV
  - Overlap removal
    - electron/jet -> priority to electron
    - Muon/jet -> priority to muon
  - Use b-tagging from Delphes (but will write a parameterization example)
  - Select events
  - Calculate m3 and mTW
  - Produce a tree

definition of a sequence of analyzers, the analyzers will process each event in this order

```
sequence = cfg.Sequence( [
    source,
    jets_30,
    muons,
    electrons,
    iso_muons,
    iso_electrons,
    match_jet_electrons,
    sel_jets_electron,
    match_muon_jets,
    sel_muons_jet,
    btagging,
    selection,
    m3,
    mtw,
    gen_tree
    ] )
```

B-tagging

```
from heppy.analyzers.Btagging import Btagging
btagging = cfg.Analyzer(
    Btagging,
    'b_jets_30',
    output = 'b_jets_30',
    input_objects = 'sel_jets_noelectron_30',
    filter_func = lambda jet : jet.tags['bf']>0.
)
```

Define a collection of tag jets

B-tagging value added as a jet.tags when reading Delphes, but could also create our own algorithm

```
from heppy.framework.analyzer import Analyzer

class Btagging(Analyzer):

    def process(self, event):
        jets = getattr(event, self.cfg_ana.input_objects)
        bjets = [jet for jet in jets if self.cfg_ana.filter_func(jet)]

        for jet in jets:
            jet.tags['b'] = self.cfg_ana.filter_func(jet)

        setattr(event, self.cfg_ana.output, bjets)
```

Add a new tag that is now a bool (result of the tag function

Top had mass

Build m3 with jets > 35GeV

```
from heppy.analyzers.M3Builder import M3Builder
m3 = cfg.Analyzer(
    M3Builder,
    instance_label = 'm3',
    jets = 'sel_jets_noelectron_30',
    filter_func = lambda x : x.pt()>35.
)
```

```
from heppy.framework.analyzer import Analyzer
from heppy.particles.tlv.resonance import Resonance

import pprint
import itertools

class M3Builder(Analyzer):

    def process(self, event):
        jets = getattr(event, self.cfg_ana.jets)
        jets = [jet for jet in jets if self.cfg_ana.filter_func(jet)]

        m3 = None
        pt3max=0
        seljets=None
        if len(jets)>=3:
            for l in list(itertools.permutations(jets,3)):
                pt3=(l[0].p4()+l[1].p4()+l[2].p4()).Pt()
                if pt3>pt3max:
                    ptmax=pt3
                    seljets=l

            top_pdgid = 6
            m3 = Resonance(seljets, top_pdgid)
        setattr(event, self.instance_label, m3)
```

Combination with the vectorial sum of highest p_T

Could also veto combinations without exactly 1 b-tagged jet

Build a resonance out of the 3 selected jets

Selection cut/flow

From the collections defined before
- Select events with >= 4 jets
- Select events with exactly 1 lepton (electron or muon)
- Select events with >= 1 b-tag
- Select events with MET > 20GeV

```
class Selection(Analyzer):

    def beginLoop(self, setup):
        super(Selection, self).beginLoop(setup)
        self.counters.addCounter('cut_flow')
        self.counters['cut_flow'].register('All events')
        self.counters['cut_flow'].register('At least 4 jets')
        self.counters['cut_flow'].register('At least 1 b-jet')
        self.counters['cut_flow'].register('Exactly 1 lepton')
        self.counters['cut_flow'].register('MET > 20GeV')

    def process(self, event):
        self.counters['cut_flow'].inc('All events')

        #select events with at least 4 jets
        if len(event.sel_jets_noelectron_30)<3:
            return False
        self.counters['cut_flow'].inc('At least 4 jets')


        #select events with at least 1 b-jet
        if len(event.b_jets_30)<1:
            return False
        self.counters['cut_flow'].inc('At least 1 b-jet')

        #select events with exactly 1 lepton
        if (len(event.sel_iso_electrons) + len(event.sel_iso_muons_nojets_30) != 1):
            return False
        self.counters['cut_flow'].inc('Exactly 1 lepton')

        #select events with MET>20GeV
        if event.met.pt()>20.:
            self.counters['cut_flow'].inc('MET > 20GeV')
```

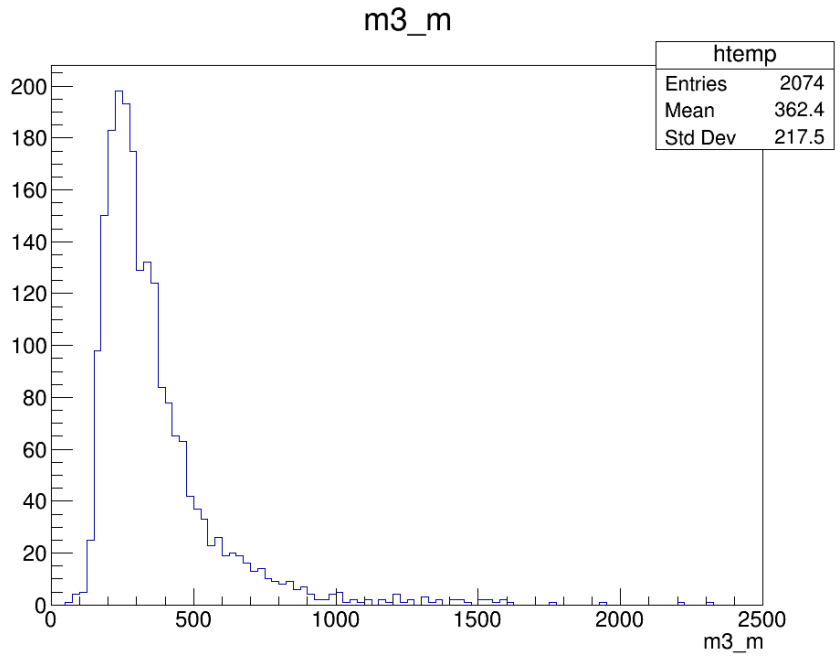
```
from heppy.analyzers.examples.ttbar.selection import Selection
selection = cfg.Analyzer(
    Selection,
    instance_label='cuts'
)
```

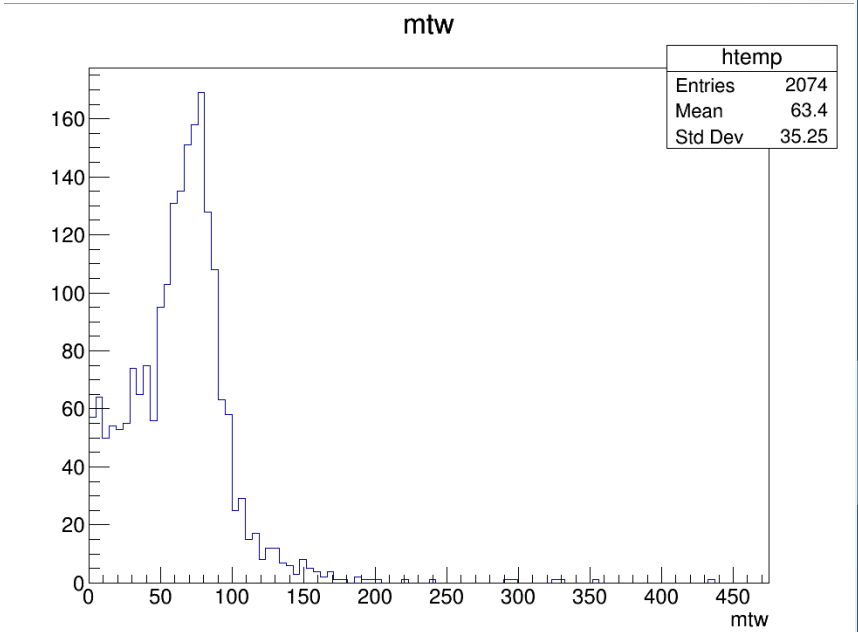
Cut flow

```
Counter cut_flow :          N evt    Eff(cut-1)    Eff
        All events          50000      1.00      1.0000
        At least 4 jets     27682      0.55      0.5536
        At least 1 b-jet    21894      0.79      0.4379
        Exactly 1 lepton     2074      0.09      0.0415
        MET > 20GeV          1840      0.89      0.0368
```

Some plots

- M3-> represents the top hadronic mass
- m_T(W) -> represents the transverse mass of the leptonic W (semi leptonic event selection)





Status/next steps

- A lot of work has happened last month
- We are close to release a recipe but
  - LCG environment problem on lxplus
  - Problem in PyROOT in v6.04 need migration to 6.06
  - Simplified documentation being finalized

  } solved yesterday B. Hegner
- We are finalizing a nice and simple tutorial
- New Delphes 3.3.2 to be used
  - Need to adapt the card already available before producing events
- Stay tuned, we will circulate an email once we are confident the analysis software can be used by users