RNTuple Workshop 2024
Python API

Vincenzo Eduardo Padulano (CERN, EP-SFT)
for the ROOT Team

02.12.2024

Agenda

- Understanding differences between C++ and Python idiomatic behaviours
- Providing a minimal Python API
- Open questions

Context and goals

- The ROOT team provides a **public** RNTuple **API** in **C++**, which was the object of the **HEP-CCE review**
- A public **Python** API was so far not available, **today** we present a first **minimal** implementation
- By and large, using **same** classes and methods of the **C++** API
- **Usability** is favoured over performance for the Python API
- For **performant** reading & writing of RNTuple in Python, we encourage using **RDataFrame**
  - Which also allows introducing more Pythonic features

C++ and Python APIs

- RNTuple employs modern C++ constructs and patterns
- Not everything applies 1:1 to an idiomatic Python usage
- e.g. move semantics, access to pointees
- In order to provide an idiomatic Python behaviour, the interface needs to be adjusted accordingly

A minimal writing API

Recent API changes introduced by #17104

```
import ROOT

RNTupleWriter = ROOT.Experimental.RNTupleWriter
RNTupleModel = ROOT.Experimental.RNTupleModel

model = RNTupleModel.Create()
model.MakeField["int"]("f")

with RNTupleWriter.Recreate(model, "ntpl", "ntuple_example.root") as writer:
    entry = writer.CreateEntry()
    entry["f"] = 42
    writer.Fill(entry)
```

A minimal writing API

Recent API changes introduced by #17104

```
import ROOT

RNTupleWriter = ROOT.Experiment
RNTupleModel = ROOT.Experimenta

model = RNTupleModel.Create()
model.MakeField["int"]("f")

with RNTupleWriter.Recreate(model, "ntpl", "ntuple_example.root") as writer:
    entry = writer.CreateEntry()
    entry["f"] = 42
    writer.Fill(entry)
```

No default entry, MakeField returns None
The usage of std::shared_pt does not apply well

A minimal writing API

Recent API changes introduced by #17104

```
import ROOT

RNTupleWriter = ROOT.Experiment
RNTupleModel = ROOT.Experimenta

model = RNTupleModel.Create()
model.MakeField["int"]("f")

with RNTupleWriter.Recreate(model, "ntpl", "ntuple_example.root") as writer:
    entry = writer.CreateEntry()
    entry["f"] = 42
    writer.Fill(entry)
```

RNTupleWriter works as a context manager
dataset is automatically committed at context exit

A minimal writing API

Recent API changes introduced by #17104

```
import ROOT

RNTupleWriter = ROOT.Experiment
RNTupleModel = ROOT.Experimenta

model = RNTupleModel.Create()
model.MakeField["int"]("f")

with RNTupleWriter.Recreate(model, "ntpl", "ntuple_example.root") as writer:
    entry = writer.CreateEntry()
    entry["f"] = 42
    writer.Fill(entry)
```

Entry **must** be requested for writing contents
filled with a dictionary syntax

A minimal writing API

Recent API changes introduced by #17104

```
with RNTupleWriter.Recreate(model, "ntpl", "ntuple_example.root") as writer:
    entry = writer.CreateEntry()
    entry["f"] = 42
    writer.Fill(entry)
print(writer)
print(model)
print(entry)
```

```
<cppyy.gbl.ROOT.Experimental.RNTupleWriter object at 0x558a0a5082a0 held by std::unique_ptr<ROOT::Experimental::RNTupleWriter,default_delete<ROOT:
:Experimental::RNTupleWriter> > at 0x558a0772c2a0>
<cppyy.gbl.ROOT.Experimental.RNTupleModel object at 0x(nil) held by std::unique_ptr<ROOT::Experimental::RNTupleModel,default_delete<ROOT::Experime
ntal::RNTupleModel> > at 0x558a07721950>
<cppyy.gbl.ROOT.Experimental.REntry object at 0x558a0a47be50 held by std::unique_ptr<ROOT::Experimental::REntry,default_delete<ROOT::Experimental:
:REntry> > at 0x558a0a3be7b0>
```

After context state:
writer exists, but further modifications will fail
model is a nullptr, Python object will throw exception on use
entry is usable as a read-only dictionary

A minimal reading API

Similar concepts can be applied to the RNTupleReader API:

- Forbid default entry
- Must create entry and use `LoadEntry(index, entry)`
- Or use the RNTupleView

Not supported yet, but a concrete idea of the API

```
import ROOT

RNTupleReader = ROOT.Experimental.RNTupleReader

with RNTupleReader.Open("ntpl", "ntuple_example.root") as ntuple:
    view = ntuple.GetView["int"]("f")
    for i in ntuple.GetEntryRange():
        val = view(i)
```

Some caveats

Some parts of the C++ API need further attention

- When returning references, such as from
  - `const RNTupleModel &RNTupleWriter::GetModel()`
  - This can potentially lead to dangling references
- When a function takes a std::unique_ptr
  - Potentially many ways to deal with this
  - Our solution: automatic move, original variable becomes nullptr

Possible extensions

Open questions remaining:

- API for writing many entries at once (e.g. fill a field with values from a numpy array)?
- Should we explore support for writing Python objects?