Experimental PyROOT: Forward Compatibility & Switch to Default

Enric Tejedor, Massimiliano Galli

ROOT Planning Meeting, 12-02-20

ROOT

Data Analysis Framework

https://root.cern

Forward compatibility in 6.20

- A few forward compatible changes have been introduced in the old PyROOT for 6.20
- In some cases, deprecation warnings are issued
- Goal: provide means to adapt now the PyROOT scripts/tests to the changes in the new PyROOT

Template instantiation (functions)

- New PyROOT requires **square bracket** syntax for function template instantiation
  - Square bracket syntax is now supported in old PyROOT too
  - Parenthesis syntax still usable, but throws warning

OLD

```
 > ROOT.gInterpreter.Declare("""
 template<typename T> T foo(T arg) { return arg; }""")

 > ROOT.foo['int']  # new syntax, instantiation
 <ROOT.TemplateProxy object at 0x7f706045bcc8>

 > ROOT.foo('int')  # old syntax, warning + instantiation
 __main__:1: FutureWarning: Instantiating a function template with
parentheses ( f(type1, ..., typeN) ) is deprecated [...]
<ROOT.TemplateProxy object at 0x7f1eaf85ac78>
```

None and pointer types

- The conversion between None and C++ pointer types is not allowed anymore in new PyROOT
  - Still usable in old PyROOT, but throws a warning

```
 > ROOT.gInterpreter.Declare("""
 class A {};
 void foo(A* a) {}""")

 > ROOT.foo(ROOT.nullptr)  # ok

 > ROOT.foo(None)          # ok, but throws warning
 __main__:1: FutureWarning: The conversion from None to null
pointer is deprecated and will not be allowed anymore in a
future version of ROOT. Instead, use ROOT.nullptr or 0
```

Passing basic types by reference

- In new PyROOT, **ctypes** must be used to pass basic types by reference
- Old PyROOT has **ROOT.Long** and **ROOT.Double**
  - Now use of Long and Double issues a warning
  - ctypes supported (c_int, c_long, c_double)

```
 > ROOT.gInterpreter.Declare("void foo(int& i) { ++i; }")

 > i = ROOT.Long(1)
 > ROOT.foo(i)
 __main__:1: FutureWarning: ROOT.Long is deprecated and will disappear
in a future version of ROOT. Instead, use ctypes.c_int for pass-by-ref
of ints

 > i = ctypes.c_int(1)
 > ROOT.foo(i); i
 c_int(2)
```

Likewise for ROOT.Double

Name changes

- Several name changes for cppyy APIs & proxy object attributes
  - New names backported to old PyROOT

cppyy.gbl.MakeNullPointer(klass) cppyy.gbl.BindObject cppyy.AsCObject	←	cppyy.bind_object(0, klass) cppyy.bind_object libcppyy.as_cobject
cppyy.add_pythonization cppyy.compose_method cppyy.gbl.nullptr	←	cppyy.py.add_pythonization cppyy.py.compose_method cppyy.nullptr
buffer.SetSize(N) + warning	←	buffer.reshape((N,))
obj.__cppname__ obj._get_smart_ptr callable._creates callable._mempolicy callable._threaded	←	obj.__cpp_name__ obj.__smartptr__ callable.__creates__ callable.__mempolicy__ callable.__release_gil__

Switch to default

- Experimental (new) PyROOT is not yet the default
- Need to plan **when** to switch it to default
- Proposal:
  - Before end of February (the sooner the better)
  - Leave it as default in **master** for the moment
- After the switch, we would have:
  - 6.20: old PyROOT with forward compatible changes
  - master: new PyROOT