**Introduction**

**Aim**

- Remove ROOT dependency of cppyy to reduce code bloat and make it faster.
- Create cppyy style libinterop library and add it to llvm mainline

**Progress**

Estimated 50% of the first aim has been completed
import cppy works but C++ functions called from Python are not working.

**Milestones**

import cppy

under the hood

- Initialise gbl
- Initialise gbl.std
- Declare functions for user access to the interpreter like cppdef, cppexec
- Handle the anomalies such as accommodating for std::shared_ptr and std::unique_ptr
Python Statement Example

cppyy.gbl.cling.runtime.gCling.AddIncludePath

```
meta_getattro("cling", parent=<>)
CreateScopeProxy("cling", parent=<>)
```

```
meta_getattro("gCling", parent=<runtime>)
LookupName("gCling", parent=<runtime>)
CPPDataMember("gCling")
```

```
meta_getattro("__dict__", parent=<cling::Interpreter>)
CreateScopeProxy("__dict__", parent=<cling::Interpreter>)
CreateScopeProxy("Interpreter", parent=<cling>)
BuildScopeProxyDict(<Interpreter>, ...)```

Handled by Python
Details of Progress

- **Major CPyCppyy Functions that have been converted to use cling:**
  - CreateScopeProxy
  - BuildScopeProxyDict
  - CPPDataMember::Set
  - meta_getattro
  - meta_setattro
  - CreateConverter

- The work done in CPyCppyy package can be seen [here](#).
Details of Progress (contd.)

- LLVM and clang symbols are exposed to `cppyy-backend`:
  - `clingwrapper.cxx` is built alongside metacling
  - The shared library is then copied to the python directory and renamed to `libcppyy_backend.so`
  - The python directory is added to the `PYTHON_PATH`

- Functions in `cppyy-backend` that have been clingized can be seen [here](#).
Future Work

- Fix the build system to let cppyy build standard headers and run AddIncludePath
- Clingize the TemplateProxy methods in CPyCppyy
- Move CallFunc interface from ROOT to cling
- Clingize the remaining parts of cppyy
- Remove ROOT from the build system completely
- Work on libInterop
Thankyou