```
class PyPipeline
{
  xrdcl::Pipeline *pipeline; // = nullptr;
}
```

```
PyObject *self, ...

PyPipeline PyOpen(...)
{
  xrdcl::Pipeline *p = new ...
  p->pipeline = Open(...);
  new xrdcl::Pipeline(Open(...));
```

```
PyPipeline PyOpen(...)
{ // parsing
  PyPipeline *p = new PyPipeline();
  p->pipeline = new xrdcl::Pipeline(Open(...));
  return p; // <- python bind. for factory
}           //          method

    PyRead(...)
  { ... }

    PyClose(...)
  { ... }
```

Python

--pipe-- -> |
-> >>

Python Bindings
C++
cpp.Pipeline Open(url, flags)
return cpp.Open(...)

C++
Pipeline Open(...)
{
1. translate args
2. Pipeline p = Open(...);
3. translate p into py wrapper
4. return wrapper
}

~~cpp~~ Python Bind

--pipe--(cpp.Pipeline p1, cpp.Pipeline p2)
return cpp.pipe(p1, p2)

C++

Pipeline pipe(Pipeline p1, ...)
{ 1. translate args
2. Pipeline ret = ~~p1~~ cpp.p1 | cpp.p2;
3. wrap up ret into py & return
}

Python Bind

```
status WaitFor(*cpp.Pipeline p)
    return cpp.WaitFor;
```

C++

```
XRootDStatus PyWaitFor*(cpp.Pipeline p)
{
    1.
    2. auto st = WaitFor(p)
    3. convert to py & return.
}
```

---

1. PyPipeline class → expose in python

2. PyOpen
   PyRead
   PyClose
   ;
   others
   } function → — " ——

3. overloading

4. PyWaitFor fun