ThriftUsagePython

Assuming you're using the definitions in "tutorial.thrift":

Generate the Code

Make sure the thrift compiler is installed somewhere in your PATH. Then generate new style classes (inherit from object, better introspection) for the tutorial:

```
thrift -gen py:new_style tutorial.thrift
thrift -gen py:new_style shared.thrift
```

To generate old style classes:

```
thrift -gen py tutorial.thrift
thrift -gen py shared.thrift
```

This makes a `gen-py/` subdirectory for the generated code.

Import the Classes

First make sure you have added the gen-py/ subdirectory to your `sys.path` with:

```
import sys
sys.path.append('gen-py')
```

Then import the classes:

```
import tutorial.Calculator
from tutorial.ttypes import *
from thrift.protocol import TBinaryProtocol
from thrift.transport import TTransport
```

Create Objects, Using Constants

Make a Work thrift object, and set its 'op' field to the ENUM `Operation.ADD` defined in `tutorial.thrift` and set two numeric fields to some values.

```
work = Work()
work.num1 = 7
work.num2 = 9
work.op = Operation.ADD
```

Serialize to/from a string

Create a `TMemoryBuffer` object to contain the serialized bytes, a `TBinaryProtocol` object to perform the serialization, and use the thrift `Work` object's `write` method to produce the serialized bytes.

Then, take the serialized bytes and pass them into a new `TMemoryBuffer`, pass that into another `TBinaryProtocol`, create a new/empty thrift `Work` object, and use it's `read` method to deserialize the bytes, setting all the fields.

```
transportOut = TTransport.TMemoryBuffer()
protocolOut = TBinaryProtocol.TBinaryProtocol(transportOut)
work.write(protocolOut)
bytes = transportOut.getvalue() # the string 'bytes' can be written out to disk
                                #  to be read in at a different time

transportIn = TTransport.TMemoryBuffer(bytes)
protocolIn = TBinaryProtocol.TBinaryProtocol(transportIn)
moreWork = Work()
moreWork.read(protocolIn)
```