

CS2006 P1

Overview:

For this practical we were asked to implement a basic support of the OpenMath encoding in Python3.

Requirements Fulfilled:

The below show all the requirements that the team fulfilled:

Basic:

Support for:

- integers
- IEEE floats
- strings
- logic1.true, logic1.false
- nums1.rational
- complex1.complex Cartesian
- interval1.integer interval
- linalg2.matrix, linalg2.matrixrow

Extensions:

Easy:

- Implement pretty-printing for OpenMath objects.
- Develop and implement a content dictionary to encode Python dictionaries. You may assume that they will be mapped to records in other programming languages. Your report should include a short specification describing the meaning and use of OpenMath symbol(s) to represent Python dictionaries. Your implementation should work for key:value pairs where the value is any object that may be encoded in OpenMath using your project. It should also give a meaningful error message if conversion of the value to OpenMath is not supported.

Medium:

- Implement support for arithmetical operations from arith1 CD. • Medium: Implement support for integer1.factorial.
- Write a script that reads an OpenMath object, performs some meaningful calculation and then writes the result in OpenMath.

Medium to Hard:

- Implement some other non-trivial OpenMath symbol(s) of your choice.
- Implement functionality to output OpenMath objects using prescribed content dictionaries. For example, one could try to output matrices using linalg2.matrix and linalg2.matrixrow instead of using lists.

Hard:

- Implement support for OpenMath attributions
- Implement support for OpenMath errors
- Implement client-server communication between two Python instances which will use OpenMath to send remote procedure calls and get back the results. Use exception handling to catch and transmit an error to the client using OpenMath errors.

Design:

This python project has the following files that is used to make it:

- OMTypes.py
- demo.py
- error.py
- omparse.py
- omput.py
- openmath.py

OMTypes.py defines the types needed to implement the basic requirements such as that of `nums1.rational`, `complex1.complex_cartesian`, `interval1.integer_interval` and `linalg2.matrix`, `linalg2.matrixrow`.

Demo.py is where the tests are run from.

Error.py defines the error exceptions

omparse.py is where the parser for the OpenMath lies. This is where the support for the other functionalities are written as well. It converts the OpenMath into objects.

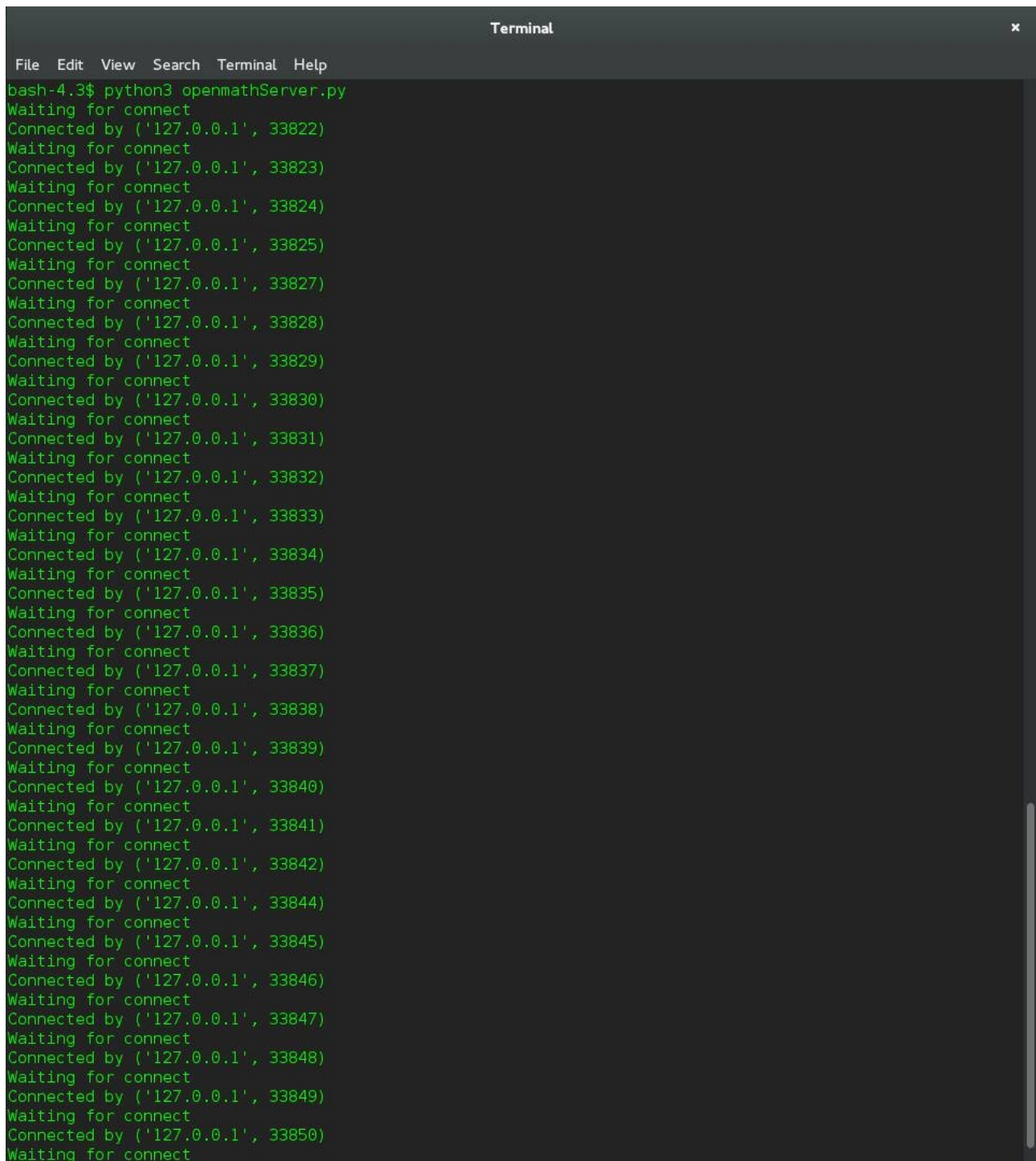
Omput.py contains the functions that produce the OpenMath encodings.

Openmath.py contains some higher level functions to parse the strings and files containing OpenMath objects. It also contains the sockets to connect to a host and send the encoded objects.

Testing:

Most of the testing has been done through the demo.py file. The output of that file is shown in the testing folder and contains the demo-output file. This file has the out put to the tests. The below are screen shots of those tests not included in the demo-output file.

The screen shot below shows the client waiting to connect to the server.

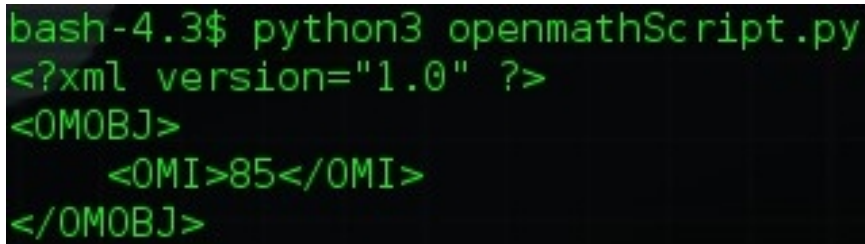
A terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the execution of a Python script named openmathServer.py. The script enters a loop where it repeatedly prints "Waiting for connect" and then "Connected by ('127.0.0.1', 33822)" through "Connected by ('127.0.0.1', 33850)".

```
bash-4.3$ python3 openmathServer.py
Waiting for connect
Connected by ('127.0.0.1', 33822)
Waiting for connect
Connected by ('127.0.0.1', 33823)
Waiting for connect
Connected by ('127.0.0.1', 33824)
Waiting for connect
Connected by ('127.0.0.1', 33825)
Waiting for connect
Connected by ('127.0.0.1', 33827)
Waiting for connect
Connected by ('127.0.0.1', 33828)
Waiting for connect
Connected by ('127.0.0.1', 33829)
Waiting for connect
Connected by ('127.0.0.1', 33830)
Waiting for connect
Connected by ('127.0.0.1', 33831)
Waiting for connect
Connected by ('127.0.0.1', 33832)
Waiting for connect
Connected by ('127.0.0.1', 33833)
Waiting for connect
Connected by ('127.0.0.1', 33834)
Waiting for connect
Connected by ('127.0.0.1', 33835)
Waiting for connect
Connected by ('127.0.0.1', 33836)
Waiting for connect
Connected by ('127.0.0.1', 33837)
Waiting for connect
Connected by ('127.0.0.1', 33838)
Waiting for connect
Connected by ('127.0.0.1', 33839)
Waiting for connect
Connected by ('127.0.0.1', 33840)
Waiting for connect
Connected by ('127.0.0.1', 33841)
Waiting for connect
Connected by ('127.0.0.1', 33842)
Waiting for connect
Connected by ('127.0.0.1', 33844)
Waiting for connect
Connected by ('127.0.0.1', 33845)
Waiting for connect
Connected by ('127.0.0.1', 33846)
Waiting for connect
Connected by ('127.0.0.1', 33847)
Waiting for connect
Connected by ('127.0.0.1', 33848)
Waiting for connect
Connected by ('127.0.0.1', 33849)
Waiting for connect
Connected by ('127.0.0.1', 33850)
Waiting for connect
```

The screen shot below shows the demo.py file running.

```
Terminal
File Edit View Search Terminal Help
bash-4.3$ python3 demo.py
--Integer Print--
42
--List Print--
[41, 42, 43]
--Nested List Print--
[4.1, 42, 43, [41, 42, 43]]
--Integer File--
42
--List File--
[41, True, 43]
--Float File--
[0.0, 1.0, 0.5, -1.0, 19487171.0, 5.1315811823070673e-08, -19487171.0, -5.1315811823070673e-08]
--Matrix File--
1 2 3 42 5 6 0 -1 -100
--Complex File--
2/3 5/4j
--Rational File--
[1, <OMTypes.Rational object at 0x7f49149a3da0>]
--unary_minus.xml File--
-10
--lcm.xml File--
10.0
--plus.xml File--
9
--minus.xml File--
1
--gcd.xml File--
2
--times.xml File--
40
--divide.xml File--
2.0
--power.xml File--
625
--abs.xml File--
42
--root.xml File--
4.013799001275562
--plus.xml File--
9
--sum.xml File--
153
--product.xml File--
6658606584104736522240000000
--String Creation And Parsing--
Hello world
--OMPrint Integer--
<?xml version="1.0" ?>
<OMOBJ>
  <OMI>42</OMI>
</OMOBJ>
--OMPrint List--
<?xml version="1.0" ?>
<OMOBJ>
```

The screen shot below shows the python script being run and printing out the correct answer.



```
bash-4.3$ python3 openmathScript.py
<?xml version="1.0" ?>
<OMOBJ>
  <OMI>85</OMI>
</OMOBJ>
```

Implementation:

This part shows the implementations that I had done and my contributions to the group work at hand.

Basic:

Support for:

- Strings
 - I followed the examples of what was already done for integers and simply modified the necessary files in order to support the basic OpenMath object.
- logic1.true, logic1.false with 150002227
 - Implemented OMBool in omput.py in order to output XML. Also implemented the ability for input in the omparse.py file
- nums1.rational with 150002227
 - Implemented the functionality to parse XML objects for nums1.rational in omparse.py. We also implemented our own type for rational as python does not have a rational type built in. The use of types is so that we can do both input and output.
- complex1.complex_cartesian with 150002227
 - Implemented OMComplex in omput.py in order to output XML. Also made a complex type due to python not having a built in one. Also implemented the ability for it to parse the XML files
- interval1.integer_interval with 150002227
 - Implemented the ability for it to parse the XML files in omparse.py. Also created a type for this. When I parse the file I used range to return a interval instead in order for the arith1.sum to work.
- linalg2.matrix, linalg2.matrixrow with 150002227
 - Created our own types for these as well. In omput.py Ommatrixrow and Ommatrix have been implemented in order to output XML. In omparse.py two functions have also been defined in order to parse them.

Extensions:

- pretty-printing with 150002227
 - For pretty printing the `xml.dom.minidom` was found and imported. The functions provided by it were used in order to print the OpenMath objects in the proper form.
- Implemented support for the arithmetical operations from arith1 CD
 - This took some time to implement as arith1 CD has got so many object types each one slightly different. I started from the top of the arith1 CD page and worked my way down when implementing support for it. Basically each function would take in a list and would then be evaluated upon depending in the XML or input.

Modified Files:

- OMTypes.py - created by 150002227
- demo.py - original source, modified by all
- error.py – created by 140009941
- omparse.py – original source, modified by all
- omput.py – original source, modified by all
- openmath.py – original source, modified by 150002227 and I
- openmathServer.py – created by 150002227
- openmathScript.py - created by 150002227

Evaluation:

There are a few things I would change in this practical if I had more time, or if I were to do so again. I would have liked to have done more work and implemented the socket module. This part of the implementation sounds very interesting.

I think that communication has improved greatly amongst the team compared to that of the last practical. The team was more proactive in communicating ideas and the implementations they wanted to write.

There were a problem I encountered with when trying to implement the arith1 CD sum. This is due to the fact that we would have to implement a lambda function. I did get stuck in trying to implement the lambda function and so in the end I resorted to using the interval function created for the basic requirements instead.

Conclusion:

In conclusion I think that our implementation of the specification has been very successful. I am now more comfortable with the use of python. I'm also comfortable with distributing the code of the project across multiple files and of manipulating python data types and working with XML parsers. I also feel that I have a greater understanding of the OpenMath dictionaries and how the XML is used.