

OpenMath / **py-scscp**

Unwatch ▾

4

★ Unstar

2

🍴 Fork

2

<> Code

🔔 Issues 1

🔗 Pull requests 0

📁 Projects 0

📖 Wiki

📶 Pulse

📊 Graphs

Branch: master ▾

py-scscp / README.rst

Find file

Copy path

 defeo Moved examples in own dir, added example for SageMath

5ba9f51 11 hours ago

1 contributor

111 lines (75 sloc) 2.98 KB

Raw

Blame

History



SCSCP

build passing

SCSCP 1.3 implementation for Python.

Description

The Symbolic Computation Software Composability Protocol (SCSCP) is a network protocol for software systems to exchange mathematical objects. Think RPC (Remote Procedure Call) for CAS (Computer Algebra Systems).

Installation

```
pip install scscp
```

Usage

This package provides a command-line SCSCP client and a base class that an SCSCP server may extend. An example implementation of an SCSCP server is also provided.

Server

The module `scscp.server` provides a class `SCSCPServer` that an SCSCP server may extend. Lower level classes are also available, for more details see the API docs.

This source distribution also contains an example server `examples/demo_server.py`, capable of performing very basic arithmetic operations. To run the demo server, simply run:

```
python examples/demo_server.py
```

Client

The module `scscp.client` provides a class `SCSCPClient` that an SCSCP client may extend. Lower level classes are also available, for more details see the API docs.

The package also contains a synchronous command-line client `scscp.SCSCPCLI` to query SCSCP servers. To connect to a server running on, e.g., localhost, type

```
>>> from scscp import SCSCPCLI
>>> c = SCSCPCLI('localhost')
```

The client automatically queries the server for the available functions, and populates the `heads` attribute:

```
>>> c.heads
{'arith1': ['minus', 'abs', 'power', 'divide', 'unary_minus', 'plus', 'times'], 'scscp2': ['get_allowed_heads', 'get_
```

Functions on the server can be queried via the syntax `c.heads.<cd>.<func>(args)` where `<cd>` is the name of the OpenMath *content dictionary*, `<func>` is the name of the function, and `args` is the list of arguments.

Integers, floats, complex numbers, booleans, strings, lists and binary data are automatically converted to and from Python native types.

```
>>> c.heads.arith1.power([2, 100])
1267650600228229401496703205376
```

The client also understands OpenMath data via the [openmath](#) package, which can be used to express more complex data.

```
>>> from openmath import openmath as om
>>> c.heads.arith1.power([om.OMInteger(2), om.OMInteger(100)])
1267650600228229401496703205376
```

To disconnect the client, simply use the `quit()` method.

```
>>> c.quit()
```

Contributing

The source code of this project can be found on [GitHub](#). Please use GitHub issues and pull requests to contribute to this project.

Credits

This work is supported by [OpenDreamKit](#).

License

This work is licensed under the MIT License, for details see the LICENSE file.

