Python Agent

Paolo Bosetti

2025-06-02

The `python_agent` repo on GitHub provides a MADS agent with an embedded python3 interpreter for developing MADS sgents in Python

Table of contents

Contents

The Python3 MADS agent is available on https://github.com/MADS-net/python_agent.

Installing

You need to have `python3` and `python3-dev` installed. Then proceed as follows depending on your platform.

UNIX

```
python3 -m venv .venv
source .venv/bin/activate
pip install numpy
# also install other necessary Python libs

cmake -Bbuild -DCMAKE_INSTALL_PREFIX="$(mads -p)"
cmake --build build -j6
sudo cmake --install build
```

The above is tested on MacOS and Ubuntu 22.04.

Windows

Run the following from project root:

```
python -m venv .venv
.venv\Scripts\activate
pip install numpy
# also install other necessary Python libs
```

Then:

```
cmake -Bbuild -DCMAKE_INSTALL_PREFIX="$(mads -p)"
cmake --build build --config Release
sudo cmake --install build
```

> ⚠ Enable sudo on Windows
>
> For `sudo` to work on Windows, you need to enable it on *Settings > System > For Developers* and set *Enable sudo* to On.

Executing

The new agent is installed as `mads-python`, so you can just type `mads python -h` (or `mads-python -h` on Windows) to know more:

```
> mads python -h
python ver. 1.2.6

Usage:
  python [OPTION...]

  -p, --period arg          Sampling period (default 100 ms)
  -m, --module arg          Python module to load
  -n, --name arg            Agent name (default to 'python')
  -i, --agent-id arg        Agent ID to be added to JSON frames
  -s, --settings arg        Settings file path/URI
  -S, --save-settings arg   Save settings to ini file
  -v, --version             Print version
  -h, --help                Print usage
```

Typically, to launch an agent named `python_source`, which gets its settings from a `python_source` section in `mads.ini`, and uses the Python module named `source` defined in the `source.py` file and that runs every 100 ms, the command is:

```
mads python -n python_source -m source -p100
```

where:

- `-n python_source` sets the agent name to `python_source`, and gets its settings from the same section in the `mads.ini` file
- `-m source` sets the Python module to `source.py`, which is searched for in the Python modules search paths, see below
- `-p100` sets the sampling period to 100 ms

Python modules search paths

The Python modules are searched for in the following folders:

- `./python`
- `./scripts`
- `../python`
- `../scripts`
- `../../python`
- `../../scripts`
- `INSTALL_PREFIX + /python`
- `INSTALL_PREFIX + /scripts`

plus any path listed in the `mads.ini` file under the `search_path` key (an array or a single string).

The `mads.ini` section

The following fields are typically used:

```ini
[python_source]
period = 200
venv = "/path/to/.venv"
python_module = "my_source"
search_paths = ["/path/to/python/folder"
```

> ⚠ Warning
>
> The section name must match the `-m` option argument when you launch the agent, so in the case aboxe you must use `-m python_source`.

Module Types

Python modules can be of type `source`, `filter`, or `sink`. The module type is defined by setting a top level variable like this, typically at the beginning of the script, just after the various `imports`:

```python
agent_type = "sink"
```

All the modules **must** implement a `setup()` function, which is expected to use the dictionary available in the module variable `params` (a dictionary) to do initial setup (opening ports or files, etc.)

Source modules **must** implement a `get_output()` function, that produces the JSON string that will be published.

Filter modules **must** implement a `process()` function, that is supposed to operate on the last received data dictionary (available as `data`, a module variable) and produce a JSON string that will be published.

Sink modules **must** implement a `deal_with_data()` function, that operates on the `data` dictionary, a module variable.

Examples

ℹ Note

To be completed