Contents

USING THE PYTHON WRAPPER FOR ARIA

A "wrapper" library for Python make it possible to use the ARIA libary with the Python programming language. In general, the Python API mirrors the C++ API, and most features of ARIA are available – it cann connect to any robot, or the MobileSim simulator.

A Python wrapper for ArNetworking is also available; see the ArNetworking directory.

On Ubuntu and Debian Linux, the Python wrapper is provided by installing an optional package (`libaria-python`). For Windows, it is included in the ARIA installation package. It can also be built from the ARIA source package (see "REBUILDING" below.)

The Aria Python wrapper currently requires Python 2.7. Multiple versions of Python may be installed on Debian and Ubuntu Linux; if so, commands such as `python2.7` or `python3` can be used to invoke the different versions of Python. See below for instructions on rebuilding the wrapper to run in Python 3.

NOTE: You must use the versions of Python mentioned above, since the wrapper library code is generally only compatible with the runtime libraries provided by these versions. To use a different version of Python, you must rebuild the wrapper library (see below).

For Windows, download Python 2.7 from `http://www.python.org`

On Ubuntu, you can install Python 2.7 with the following command:

```
sudo apt-get install python2.7
```

(NOTE: Python 2.7 and later allows native C++ DLLs to be named with a ".pyd" file extension, rather than .dll/.so as was allowed in previous versions. So if Python is unable to import the AriaPy DLL module, rename `_AriaPy.dll` or `_AriaPy.so` to `_AriaPy.pyd` in the `python/` subdirectory.)

Documentation about the Python language and libraries is at http://docs.python.org

If ARIA has been installed on Ubuntu Linux using the libaria-python package, then the ARIA Python modules are installed on the system dist-packages, which will allow Python to locate them.

If ARIA has been installed on Windows using the ARIA installer, then Windows registry keys and environment variables have been set to allow Python to locate the ARIA wrapper Python modules.

If you have compiled ARIA from source code, you must configure your environment so that Python can locate the ARIA Python modules: On Linux, set a `PYTHONPATH` environment variable to `/usr/local/Aria/python`. On Windows, set it to: `C:\Program Files\MobileRobots\Aria\python`, and also append the following to `PATH`: `C:\Program Files\MobileRobots\Aria\python;C:\Program Files\MobileRobots\Aria\bin;C:\Python27;C:\Python27\Scripts`

Environment variables are set in Windows in the Advanced settings section of the System control panel. To open the System Control Panel, right click on the start menu and choose System. Then choose Advanced Settings then click the Environment Variables button.

On Windows 7, the System control panel is also available through the Control Panel entry in the Start menu. On Windows 10, it is also available by opening the system Settings, clicking System, then choosing System Info, or by searching for System control panel.

If using the Windows command terminal to run scripts, you can also temporarily set the `PATH` and `PYTHONPATH` environment variables in that command shell: set PATH=%PATH%;C:\Program Files\MobileRobots\Aria\bin;C:\Program Files\MobileRobots\Aria\python set PYTHONPATH=%PYTHONPATH%:C:\Program Files\MobileRobots\Aria\python

For an example using the MobileSim simulator, start the simulator, then open a commmand/terminal shell, enter the pythonExamples directory and run simple.py:

Linux: cd /usr/local/Aria/pythonExamples python simple.py

Windows: cd "\Program Files" cd MobileRobots\Aria\pythonExamples python simple.py

To connect to the robot, run these scripts on the robot's onboard computer, or use appropriate ARIA command line arguments to connect to a robot via

serial cable (`-robotPort`) or wifi (`-remoteHost`). See the next section for more details on connecting to a robot via WiBox wifi interface. See robot manual, `CommandLineOptions.txt`, and online documentation (http://robots.mobilerobots.com) for more information on ARIA command line
options.

To begin making changes and developing your own program based on `simple.py` or any of the other Python examples in this directory, copy the script to a location in your home directory.

MAKING A WIRELESS CONNECTION VIA WIBOX SERIAL-WIFI INTERFACE

Most AmigoBot and some Pioneer3 robots are equipped with a wifi interface to allow you to run software on your computer without having to connect it to the robot with a cable.
details.

To make a connection through this interface, the interface must be configured for your wireless network (See http://robots.mobilerobots.com/wiki/WiBox for instructions), and you must specify the IP address of the interface at the beginning of the script.

You can automatically search the network for WiBox devices, prompt you to choose one, and then connect to the robot, in one function:

```
from AriaPy import *
robot = Aria.connectToRobot()
```

(Alternatively, you can also call `chooseWiBox()` to search for wifi interfaces and prompt, and add appropriate program command-line arguments. Then use `ArArgumentParser` and `ArRobotConnector` to connect.)

RUNNING PYTHON SCRIPT WITH COMMAND-LINE ARGUMENTS

Various options in ARIA can be configured via program command-line arguments (See <../CommandLineOptions.txt> or run with `--help` for a list).

These can be used to explicitly set up device connection parameters, and other options.

For example:

Linux:

```
cd /usr/local/Aria/pythonExamples
python simple.py -remoteHost 10.0.126.32
python simple.py -robotPort /dev/ttyUSB0
```

Windows:

```
cd "\\Program Files"
cd MobileRobots\\Aria\\pythonExamples
python simple.py -remoteHost 10.0.126.32
python simple.py -robotPort COM1
```

(Replace values with the correct address of your wifi interface or serial port to which robot is connected.)

On Windows, command-line arguments can also be specified in a Shortcut to the script. Right click on the Shortcut and choose Properties.

NOTES

The Python wrapper API is not as well tested as Aria itself. If you encounter problems, please notify the aria-users mailing list. Furthermore, some methods have been omitted or renamed, and you have to do a few things differently (see next section).

ARIA LIBRARY DOCUMENTATION

You can use the ARIA C++ API Reference documentation to discover available classes and functions in the library. This reference documentation will show the C++ syntax of course, but will include notes about how usage differs in Python or if any classes or functions are not available in Python.

USING IDLE IDE ON WINDOWS

On Windows, Python 2.7 includes a simple tool for editing and running Python scripts called IDLE. To use IDLE, install Python and set up your environment variables as described above. Right-click on any Python script, and choose "Edit with IDLE" if available. If the "Edit with IDLE" menu item is not available, choose "Open With...", "Choose Default Program...", i click "Browse...", and navigate to the directory `C:\Python27\Lib\idlelib`, click on `idle.bat` and click "Open" to choose. To run a script in IDLE, , choose "Run Module" from the "Run" menu or press F5. You can also enter Python code directly into the Python console window.

See https://docs.python.org/2/library/idle.html for more information.

If you need to add command line arguments (e.g. to configure a device connection), you can append them to `sys.argv` at the beginning of your script (before passing `sys.argv` to ArArgumentParser).

USING JUPYTER NOTEBOOK

Jupyter http://jupyter.org/ is a system that allows you to use a web interface to interactively edit and run code in a "notebook" format that allows nicely formatted explanation and parts of code to be assembled in one document. You can run the Jupyter "kernel" and launch a web browser to begin a Jupyter project locally, or if your robot has an onboard computer, you can run the kernel on the onboard computer and users can work on your notebook and code via a web browser interface on a separate computer.

On Ubuntu Linux, you can install Jupyter with the following commands:

```
sudo apt-get install python3-pip
pip3 install jupyter
```

See http://jupyter.readthedocs.io/en/latest/install.html for full installation instructions and see http://jupyter-notebook.readthedocs.io/en/latest/ for documentation on editing and running code from the notebook interface.

Use the `--no-browser` argument to avoid opening the browser automatically (you can then access it from another browser at port 8888).

Note: Jupyter does not use the PYTHONPATH environment variable to find modules, instead you must install have installed ARIA from Ubuntu/Debian packages or using the Windows installer. If Jupyter is not finding the AriaPy wrapper, copy or link `AriaPy.py` and `_AriaPy.pyd` or `_AriaPy.so` into the Python `site-packages` directory.

For an example, make a copy of `/usr/local/Aria/pythonExamples/intro.ipynb`, run jupyter notebook, then open `intro.ipynb`:

```
cp /usr/local/Aria/pythonExamples/intro.ipynb .
jupyter notebook
```

USING ACTIONS IN PYTHON

Read this if you are implemeting a custom ArAction subclass. This is an optional feature of ARIA. If you are just getting started and do not (yet) want to do this, you can skip these instructions.

Writing classes in Python that subclass classes from Aria is not as straightforward as making subclasses of native Python classes. Subclasses of classes in the wrapper are not direct subclasses of the C++ classes, they are only subclasses of the Python wrapper classes, which implement the logic to redirect calls between the C++ and wrapper classes. The practical consequences of this include the following:

1. You cannot call a virtual method in the parent class which the subclass overrides – it will always be directed to the subclass's implementation. For example, you can override `ArAction::setRobot()`, which is a virtual method. But then, any call to `setRobot()` on the subclass will always be directed to the subclass's implementation; calling `self.ArAction.setRobot()` from you subclass's `setRobot()` override would result in an infinite recursion, so Swig throws an exception instead. There is no workaround for this other than extending each parent class in wrapper.i ad-hoc to include an additional method with a new name to invoke the method defined in the parent. We have done this for a few cases (such as `ArAction::setRobot()`), but If this ever becomes neccesary for any others, please let us know and we will add the extension.
2. Swig currently does not make protected methods available to subclasses, though this may be fixed in the future.

OTHER DIFFERENCES

See the C++ API docs for various classes for notes about how they might be used differently in Python. For example, ArConfigArg and ArFunctor are used slightly differently (since those classes use features unique to C++).

REBUILDING THE PYTHON WRAPPER

You must rebuild the Python wrapper if you make any interface changes to ARIA and want them to be available in Python. You may also need to rebuild the wrapper if you want to use another version of Python than was originally used to build the Python wrapper.

If you want to rebuild the Python wrapper you need to install SWIG, you can get it from http://www.swig.org/download.html (use version 1.3.29 or newer.) You'll also need Python installed (see above) or install it using apt-get: sudo apt-get install swig1.3

On Linux, you should also to install the Python development package in addition to the main Python runtime package. e.g. on Debian, install python-dev. On Windows, if the `python24_d.lib` library is missing, you can download the

Python 2.4 source code and compile the "python" project in the PCbuild subdirectory. The development package for Python 3 on Ubuntu is python3-dev.

On Linux, set the environment variable `PYTHON_INCLUDE` to the full path of Python's include directory. On Linux, the default is `/usr/include/python2.7` for Python 2.7. Set it to another directory in /usr/include for another version of Pyhon (e.g. `/usr/include/python3.4` for Python 3.4. Check that this directory exists.)

For Windows 32-bit builds, set `PYTHON_INCLUDE` to the include subdirectory of the directory where you installed Python (default is `C:\Python27\include` for Python 2.7). Set `PYTHON_LIB` to the full path to the Python library (e.g. `C:\Python27\libs\python27.lib`) and set `PYTHON_LIBDIR` to the directory containing all of the Python libraries (e.g. `C:\Python27\libs`).

For a Windows 64-bit build, set `PYTHON_INCLUDE_X64` to the include subdirectory of the directory where you installed Python (e.g. `C:\Python27_x64\include` for Python 2.7 64-bit installed in `C:\Python27_x64 directory`). Set `PYTHON_LIB_X64` to the full path to the Python library (e.g. `C:\Python27_x64\libs\python27.lib`) and set `PYTHON_LIBDIR_X64` to the directory containing all of the Python libraries (e.g. `C:\Python27_x64\libs`). Two sets of environment variables are used on Windows to allow you to install both 32-bit and 64-bit Python and build the wrapper for either architecture on the same machine.

Set Windows environment variables in the Advanced section of the System control panel.

On Linux Run `make python` in the Aria directory, and again run `make python` in the ArNetworking subdirectory if you also want the wrapper library for ArNetworking. On Windows, open the AriaPy-vc2012.sln file in the `python` directory with Visual Studio 2012 and build AriaPy and, if desired, ArNetworkingPy in Release configuration.

To build the wrapper, first clean any previously built code with `make cleanPython` on Linux or use Clean All or rebuild the AriaPy Visual C++ projects. Then run `make python` in Linux or build the AriaPy Visual C++ project on Windows.