HelpDev

Release 0.7

Daniel Cosmo Pizetta

Sep 02, 2020

CONTENTS

CHAPTER

ONE

HELPDEV

Helping users and developers to get information about the environment to report bugs or even test your system without spending a day on it. It can get information about hardware, OS, paths, Python distribution and packages, including Qt-things. Operates in Linux, Windows and Mac. Working on Python 2.7+ and Python 3.4+.

If you want to get information at runtime of your application, you need to call using the same environment (and process) in which your application is running. This module can be imported and integrated into your application, providing a report about the current environment.

Some information can be depedent or independent of your Python environment, and some others can be dependent your running application. So, there are some acronymn used to refer to them:

- PEI: Python environment independent;
- PED: Python environment DEPENDENT;
- PEAD: Python environment and application DEPENDENT.

Caution:

- This script is not supposed to get personal information using the option `--all`, but you must check the information before using the output.
- Using the option `--all-for-sure` it is added information about paths and variables that can show personal information. So, be sure when using this option when publishing in the web.
- I'm not responsible for bad use or problems with the information given by this script, but if pointed in the Issues, I can help fixing it.

1.1 Installing, updating and uninstalling

To install and/or update, do

```
$ pip install -U helpdev
```

To remove

```
$ pip uninstall helpdev
```

1.2 Running

You just need to run in the terminal the line(s) below.

To get a minimalist output

```
$ helpdev
```

To filter a set of packages to get info, which lists all that starts with sphinx, qtpy and pyqt5

```
$ helpdev --packages="sphinx.*,qtpy,PYQT5"
```

To get a complete output without personal information

```
$ helpdev --all
```

To get a complete output WITH PERSONAL INFORMATION

```
$ helpdev --all-for-sure
```

To get some help information

```
$ helpdev --help
```

1.3 Examples from v0.6

1.3.1 Help

```
$ helpdev --help
```

```
usage: helpdev
    [--hardware] [--os] [--thread] [--network [NETWORK]]
    [--distributions] [--python] [--conda]
    [--qt] [--qt-bindings] [--qt-abstractions]
    [--packages [PACKAGES]]
    [--packages-pip] [--packages-pip-e]
    [--packages-conda] [--packages-conda-e]
    [--numbers] [--float] [--int]
    [--personal] [--path] [--scope]
    [--all]
    [--all-for-sure]
    [--version]
    [--help]

HelpDev - Extracts information about the Python environment easily.

optional arguments:

--hardware            CPU, memory and architecture (PEI)
--os                  Operating system (PEI)
--thread              Threads specification in the system (PEI)
--network [NETWORK]   Network information, DNS and load for usual sites
                       (PEI). NETWORK timeout defaults to 5s. 0 is disabled
```

(continues on next page)

(continued from previous page)

```
--distributions       All options for distributions below (PED)
--python              Python distribution (PED)
--conda               Conda/Anaconda Python distribution (PED)

--qt                  All options for Qt below (PEAD)
--qt-bindings         Available Qt bindings (PyQt/Pyside) (PEAD)
--qt-abstractions     Available Qt abstractions (QtPy/Qt.Py/PyQtGraph) (PEAD)

--packages [PACKAGES] All options for packages below, except '-e' (PED)
                       Filter PACKAGE(s) to report. Accepts regex, separator is ','
--packages-pip        PIP installed packages + PIP check (PED)
--packages-pip-e      PIP locally installed packages + PIP check (PED)
--packages-conda      CONDA installed packages (PED)
--packages-conda-e    CONDA locally installed packages (PED)

--numbers             All options for numbers below (PEI)
--float               Float representation in the system (PEI)
--int                 Integer representation in the system (PEI)

--personal            All options for personal information below (PEAD)
--path                Show Python current paths i.e. 'sys.path' (PEAD)
--scope               Show Python current scope i.e. 'dir()' (PEAD)

--all                 Run all options above, except 'personal' (PEAD)
--all-for-sure        Run all options above, INCLUDING 'PERSONAL' (PEAD)

--version, -v         Show program's version number and exit
--help, -h            Show this help message and exit
```

1.3.2 With –packages filter

This filtering feature provides a clean list of packages to report. It accepts regular expressions. Each expression must be separated by comma.

The basic regular expression checks the start until the end of the package name and they are case insenstitive.

```
# gets all that starts with 'sphinx', 'qtpy' and 'PYQT5' (not case sensitive)
$ helpdev --packages="sphinx.*,qtpy,PYQT5"
```

```
* PYTHON PACKAGES----------------------------------------
    - PyQt5..................... 5.12.1
    - QtPy...................... 1.7.0
    - Sphinx.................... 2.0.1
    - sphinx-rtd-theme.......... 0.4.3
    - sphinxcontrib-applehelp... 1.0.1
    - sphinxcontrib-bibtex...... 0.4.2
    - sphinxcontrib-devhelp..... 1.0.1
    - sphinxcontrib-excel....... 0.0.1
    - sphinxcontrib-fulltoc..... 1.2.0
    - sphinxcontrib-htmlhelp.... 1.0.2
    - sphinxcontrib-jsmath...... 1.0.1
    - sphinxcontrib-plantuml.... 0.15
    - sphinxcontrib-qthelp...... 1.0.2
    - sphinxcontrib-serializinghtml. 1.1.3
```

(continues on next page)

(continued from previous page)

```
* CONDA PACKAGES----------------------------------------
    - pyqt5..................... 5.12.1
    - qtpy...................... 1.7.0
    - sphinx.................... 2.0.1
    - sphinx-rtd-theme.......... 0.4.3
    - sphinxcontrib-applehelp... 1.0.1
    - sphinxcontrib-bibtex...... 0.4.2
    - sphinxcontrib-devhelp..... 1.0.1
    - sphinxcontrib-excel....... 0.0.1
    - sphinxcontrib-fulltoc..... 1.2.0
    - sphinxcontrib-htmlhelp.... 1.0.2
    - sphinxcontrib-jsmath...... 1.0.1
    - sphinxcontrib-plantuml.... 0.15
    - sphinxcontrib-qthelp...... 1.0.2
    - sphinxcontrib-serializinghtml. 1.1.3
```

This code is based on many other scripts from:

- zhreshold
- QDarkStyle
- QtPy

CHAPTER

TWO

REFERENCE

HelpDev - Extracts information about the Python environment easily.

Authors:

- Daniel Cosmo Pizetta <daniel.pizetta@usp.br>

Since: 2019/04/16

License: MIT

helpdev.**QT_ABSTRACTIONS = ['qtpy', 'pyqtgraph', 'Qt']**
values of all Qt abstraction layers to import.

Type list

helpdev.**QT_BINDINGS = ['PyQt4', 'PyQt5', 'PySide', 'PySide2']**
values of all Qt bindings to import.

Type list

helpdev.**check_conda**()
Check Conda Python distribution information.

It is Python/Conda environment dependent.

Returns Dictionary filled with respective information.

Return type dict(str)

helpdev.**check_conda_packages**(*edit_mode=False*, *packages=None*)
Check conda inslalled packages information filtering for packages.

It is Python/Conda environment dependent.

Returns Dictionary filled with respective information.

Return type dict(str)

helpdev.**check_float**()
Check float limits information.

Get information from `sys` library.

Returns Dictionary filled with respective information.

Return type dict(str)

helpdev.**check_hardware**()
Check hardware information.

It uses subprocess commands for each system along with `psutil` library. So you need to install psutil library.

Returns Dictionary filled with respective information.

Return type dict(str)

helpdev.**check_installed**(*import_list*)
Return a list of installed packages from import_list.

Note that the strings in the list must match the import name, e.g. pyqt5 will not work as the import name is PyQt5.

Parameters **import_list** (*list(str)*) – List of of import names to check installation.

Returns Filtered list of installed packages.

Return type list(str)

helpdev.**check_int**()
Check int limits information.

Get information from `sys` library.

Returns Dictionary filled with respective information.

Return type dict(str)

helpdev.**check_network**(*timeout*)
Check network connection for URLS list with timeout.

Parameters **timeout** (*int*) – timout in seconds.

Returns Dictionary filled with respective information.

Return type dict(str)

helpdev.**check_numbers**()
Check numbers related float and integer information.

helpdev.**check_os**()
Check operating system information.

Returns Dictionary filled with respective information.

Return type dict(str)

helpdev.**check_path**()
Check Python path from `sys` library.

Returns Dictionary filled with respective information.

Return type dict(str)

helpdev.**check_python**()
Check Python information.

It is Python environment dependent. Get information from `platform` and `sys` libraries.

Returns Dictionary filled with respective information.

Return type dict(str)

helpdev.**check_python_packages**(*edit_mode=False*, *packages=None*)
Check PIP installed packages filtering for packages.

Returns Dictionary filled with respective information.

Return type dict(str)

helpdev.**check_qt**()
Check Qt related bindings and abstractions information.

helpdev.**check_qt_abstractions**()
Check all Qt abstractions related information.

Returns Dictionary filled with respective information.

Return type dict(str)

helpdev.**check_qt_bindings**()
Check all Qt bindings related information.

Returns Dictionary filled with respective information.

Return type dict(str)

helpdev.**check_scope**()
Check Python scope or dir().

Returns Dictionary filled with respective information.

Return type dict(str)

helpdev.**check_thread**()
Check threads information.

Get information from `sys` library.

Returns Dictionary filled with respective information.

Return type dict(str)

helpdev.**customize**(*package*)
Get the custom filter from the package imported.

This is a way to promote a standard format to get a customized information from a specific package. This function try to import the package and run the public method `get_custom_helpdev(help_dev_version='')` from that package which gives a customized filter to provide the results.

For example, this line

```
info_dict = helpdev.customize('spyder')
```

will try this

```
from spyder import get_custom_helpdev
custom_filter = get_custom_helpdev(helpdev.__version__)
```

Then, it will apply that filter.

Parameters **package** (*str*) – Import name to check installation.

Returns Customized information from imported packages.

Return type dict(str)

helpdev.**filter_packages**(*dict_packages*, *expression*)
Filter the dict_packages with expression regex.

In the expression, each item separated by comma is splitted and then surrounded parenthesis (i.e. group), then joined with OR (`|`). The expressions are finally started to match the begin until the end ignoring case (`i`). See the example below

```
expression = "sphinx.*,qtpy,PYQT5"
```

Then it will be processed resulting in

```
expression = "(?i:^((sphinx.*)|(qtpy)|(PYQT5))$)"
```

If the expression of all of them not match with the package name, this package is removed from the dict (a copy of it).

Parameters

- **dict_packages** (*dict*) – Dictionary with package_name:version_number.
- **expression** (*str*) – Regular expression separated by commas.

Returns Filtered dict with that matches the expression.

Return type dict(rst)

helpdev.**installed_qt_abstractions**()
Return a list of qt abstraction layers available.

Returns Dictionary filled with respective information.

Return type dict(str)

helpdev.**installed_qt_bindings**()
Return a list of qt bindings available.

Returns List filled with respective information.

Return type list(str)

helpdev.**print_output**(*info_dict*)
Print output in a nested list format.

helpdev.**qt_abstraction_information**(*import_name*)
Get abstraction layer version and binding (default or current if in use).

Note:

- The name of the installed package can differ from the import name. This is an weird thing from PIP/CONDA, e.g, the abstraction 'qt.py' is imported as 'Qt'.
- Since each package is build as it is, sometimes we are not able to define its information, e.g, Qt.py is installed but no binding is. This will cause an error that, for now, it is impossible to us to show any other information about it, e.g, version. We need to deal with a better way.
- This function should be called with pre-defined list of installed packages passed throuw import_name, do not use it to try import.

Parameters **import_name** (*str*) – Import name of abstraction for Qt.

Raises **ImportError** – When the import is not found.

Returns

(abstraction version, environment variable, binding variable, import name, status)

Return type tuple

helpdev.**qt_binding_information**(*import_name*)

Get binding information of version and Qt version.

Note: The name of the installed package can differ from the import name. This is an weird thing from PIP/CONDA, e.g, the binding 'pyqt5' in PIP is 'pyqt' in Conda and both are imported as 'PyQt5'.

Parameters **import_name** (*str*) – Import name of binding for Qt.

Raises **ImportError** – When the import is not found.

Returns

(binding version, qt version)

Return type tuple

CHAPTER

THREE

TUTORIALS

3.1 Customizing for your application

You can use this tool to provide a easy way to the users get necessary information about their environment when reporting bugs. Even if developers can use it to *easilly* get all the information necessary.

1. Add `helpdev` to your list of requirements in `setup.py`

```
# ...

install_requires=['helpdev'...]

# ...
```

2. Import `helpdev` functions that are important for you

```
# ...

import helpdev

report_dict = {}

# get basic information updating the dictionary
report_dict.update(helpdev.check_hardware())
report_dict.update(helpdev.check_os())
report_dict.update(helpdev.check_python())
report_dict.update(helpdev.check_qt_bindings())
report_dict.update(helpdev.check_qt_abstractions())

# list of important packages for your app
# note that you can use regex (spyder.*)
packages = "spyder.*,ipython,cython,jedi,matplotlib,numpy,pandas,"
           "psutil,pycodestyle,pyflakes,pygments,pylint,qtconsole,"
           "rope,sphinx,sympy"

# get filtered information for those packages
report_dict.update(helpdev.check_python_packages(packages))

# ...
```

3. You can use, then, the dictionary-like information or print it

```
# ...
```

(continues on next page)

(continued from previous page)

```
# printing the output in the terminal
helpdev.print_output(report_dict)

# ...
```

4. The output for this example is, then

```
* HARDWARE--------------------------------------------------------------------
    - Machine..................... x86_64
    - Processor................... Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz
    - Total Memory................ 16689 MB
    - Free Memory................. 534 MB
    - Total Swap.................. 19999 MB
    - Free Swap................... 19999 MB
* OPERATING SYSTEM------------------------------------------------------------
    - System...................... Linux
    - Release..................... 4.15.0-48-generic
    - Platform.................... Linux-4.15.0-48-generic-x86_64-with-debian-
↪buster-sid
    - Version..................... #51-Ubuntu SMP Wed Apr 3 08:28:49 UTC 2019
* PYTHON DISTRIBUTION---------------------------------------------------------
    - Version..................... 3.6.8
    - C Compiler.................. GCC 7.3.0
    - C API Version............... 1013
    - Implementation.............. cpython
    - Implementation Version...... 3.6.8
* QT BINDINGS-----------------------------------------------------------------
    - PyQt5 Version............... 5.12.2
    - PyQt5 Qt Version............ 5.12.3
* QT ABSTRACTIONS-------------------------------------------------------------
    - qtpy Version................ 1.7.1
    - qtpy Binding................ pyqt5
    - qtpy Binding Variable....... os.environ['QT_API']
    - qtpy Import Name............ qtpy
    - qtpy Status................. OK
    - pyqtgraph Version........... 0.10.0
    - pyqtgraph Binding........... Not set or inexistent
    - pyqtgraph Binding Variable.... os.environ['PYQTGRAPH_QT_LIB']
    - pyqtgraph Import Name....... pyqtgraph
    - pyqtgraph Status............ OK
    - Qt Version.................. 1.1.0
    - Qt Binding.................. PyQt5
    - Qt Binding Variable......... Qt.__binding__
    - Qt Import Name.............. Qt
    - Qt Status................... OK
* PYTHON PACKAGES-------------------------------------------------------------
    - ipython..................... 7.5.0
    - jedi........................ 0.13.3
    - numpy....................... 1.16.3
    - psutil...................... 5.6.2
    - pycodestyle................. 2.5.0
    - pyflakes.................... 2.1.1
    - Pygments.................... 2.3.1
    - qtconsole................... 4.4.4
    - rope........................ 0.14.0
    - Sphinx...................... 2.0.1
    - spyder...................... 3.3.4
```

(continues on next page)

(continued from previous page)

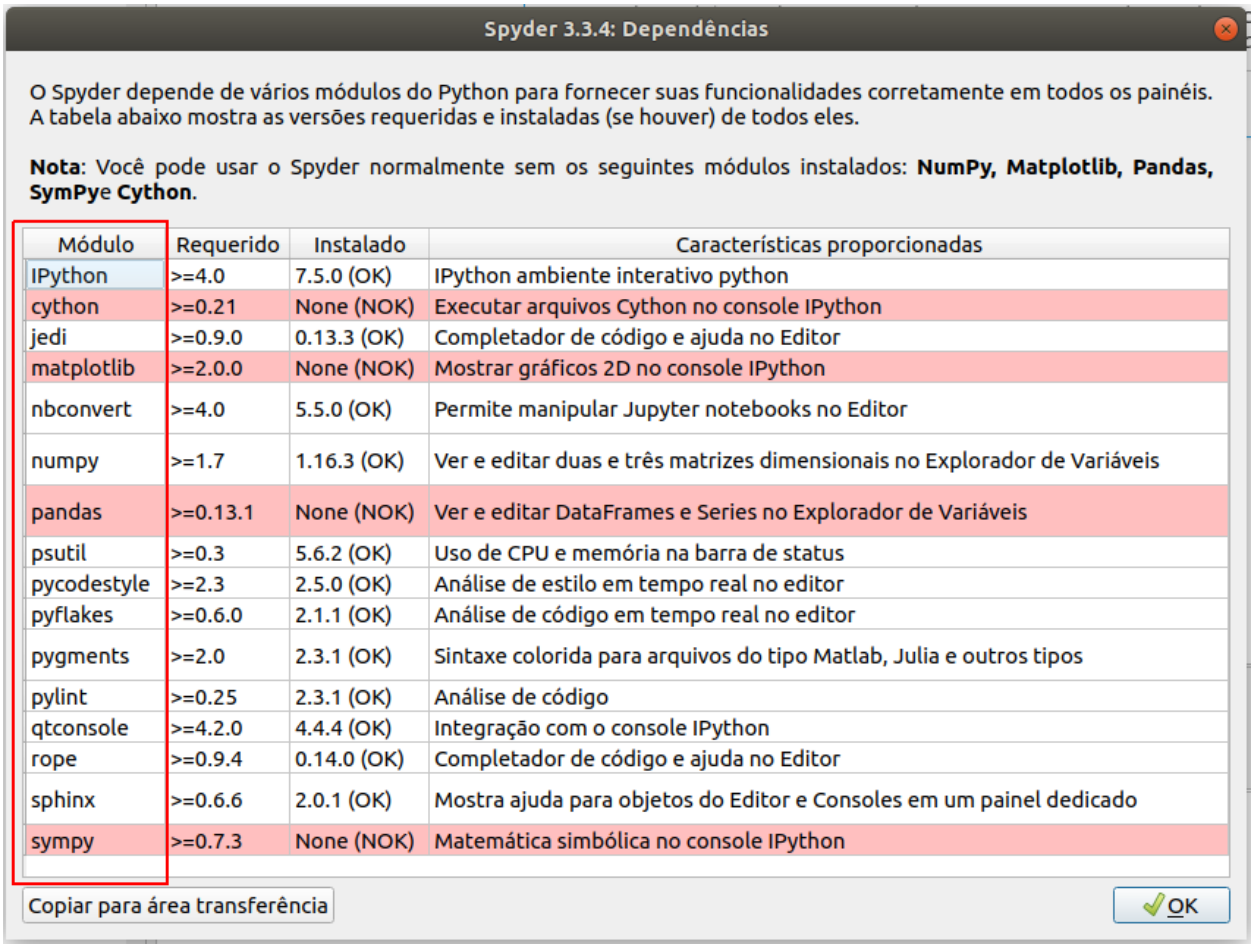
```
    - spyder-kernels.............. 0.4.4
```

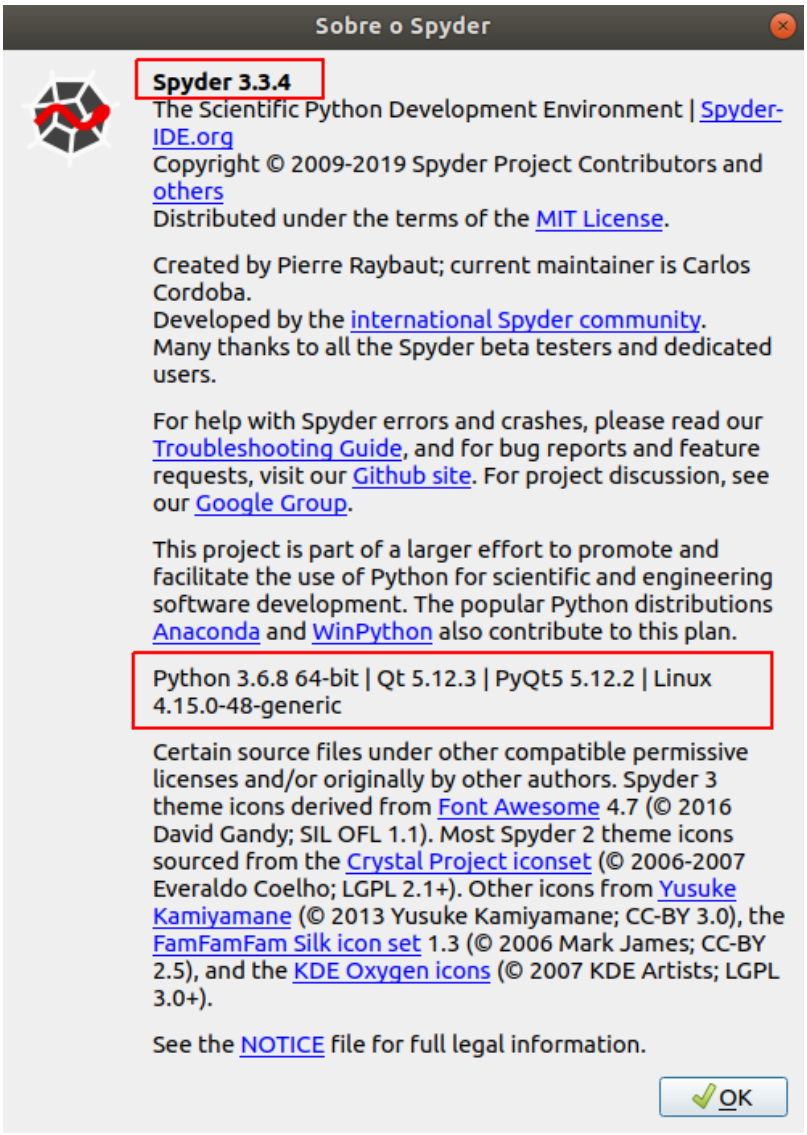
You can obtain the same output using the command-line options as follows

```
helpdev --hardware --os --python --qt

helpdev --packages="spyder.*,ipython,cython,jedi,matplotlib,numpy,
                    pandas,psutil,pycodestyle,pyflakes,pygments,
                    pylint,qtconsole,rope,sphinx,sympy"
```

In this example, the packages list was created from the Spyder dependencies list and *about* which can be seem below:




Sobre o Spyder
Spyder 3.3.4
The Scientific Python Development Environment | Spyder-IDE.org
Copyright © 2009-2019 Spyder Project Contributors and others
Distributed under the terms of the MIT License.
Created by Pierre Raybaut; current maintainer is Carlos Cordoba.
Developed by the international Spyder community.
Many thanks to all the Spyder beta testers and dedicated users.
For help with Spyder errors and crashes, please read our Troubleshooting Guide, and for bug reports and feature requests, visit our Github site. For project discussion, see our Google Group.
This project is part of a larger effort to promote and facilitate the use of Python for scientific and engineering software development. The popular Python distributions Anaconda and WinPython also contribute to this plan.
Python 3.6.8 64-bit | Qt 5.12.3 | PyQt5 5.12.2 | Linux 4.15.0-48-generic
Certain source files under other compatible permissive licenses and/or originally by other authors. Spyder 3 theme icons derived from Font Awesome 4.7 (© 2016 David Gandy; SIL OFL 1.1). Most Spyder 2 theme icons sourced from the Crystal Project iconset (© 2006-2007 Everaldo Coelho; LGPL 2.1+). Other icons from Yusuke Kamiyamane (© 2013 Yusuke Kamiyamane; CC-BY 3.0), the FamFamFam Silk icon set 1.3 (© 2006 Mark James; CC-BY 2.5), and the KDE Oxygen icons (© 2007 KDE Artists; LGPL 3.0+).
See the NOTICE file for full legal information.
OK

CHAPTER

FOUR

CHANGES

4.1 v0.7

- Add `customize` function, part of #1
- Change _filter to filter_packages and make it public
- Improve filter_package explanation
- Improve docs
- Add check_qt() function
- Add check_numbers() function
- Enhance help from command line parser
- Enhance command line order
- Correct mispelling errors
- Remove importlib_metadata for Pytho 3.8+, incoporated into std lib, MR#2
- Add MANIFEST.in, part of #9
- Improve internal code and linting
- Remove support for Python 2.7 and 3.4, closes #8

4.2 v0.6.10

- Fix gitlab-ci, again
- Whole pipeline working on gitlab, even upload to pypi

4.3 v0.6.9

- Fix gitlab-ci

4.4 v0.6.8

- Change stage names
- Split test in more envs
- Improve tutorial

4.5 v0.6.7

- Lint code and docs
- Add more docs
- Add pylint configuration
- Improve tests and linters
- Add docs generate with sphinx
- Add tutorial for personalization - Spyder example
- Add build, test and release on gitlab-ci
- Add release requirements
- Fix tox commands to build, test and release

4.6 v0.6.6

- Fix Python 2 compatibility of FileNotFoundError
- Lint code
- Add tox and gitlab CI - working, passing

4.7 v0.6.5

- Make it compatible with Python 2.7, no errors are issued
- Remove shell=True to improve security
- Improve command calls
- Test url to check starts with http
- Add tests using tox for py27, py34, py36, py37
- Add requirement files for develop, doc, stable, test and update envs
- Fix packages-conda adding –no-pip option (seems not working, conda issue)

4.8 v0.6.4

- Remove print from code

4.9 v0.6.3

- Fix problems when none binding is installed for abstractions, fixes #5

4.10 v0.6.2

- Add function to print output

4.11 v0.6.1

- Fix readme links

4.12 v0.6

- Add filter to target packages
- Fix conda list not to not list packages installed with pip

4.13 v0.5

- Enhance internal docs
- Add `--distributions` to list Python and Conda distributions
- Remove `--network` from minimalist output
- Enhance README including examples from files

4.14 v0.4.2

- Enhance readme and update changes

4.15 v0.4.1

- Fix network command

4.16 v0.4

- Add commands for bindings and abstractions
- Add binding variable and import name

4.17 v0.3

- Fix readme format

4.18 v0.2

- Fix `conda` command not found error
- Fix memory info and add swap info
- Add thread information
- Add more options and change some options names
- Improve docs
- Classifiers and year update

4.19 v0.1

- First working version

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

h

INDEX