pyangext Documentation

Release 0.0.1

(none)

June 02, 2016

Contents

pyang + sensible **extensions**

CHAPTER 1

Contents

1.1 pyangext

Sensible extensions for pyang

1.1.1 What's this all about?

YANG is a data modeling language born in the context of configuration and management of network devices (like routers and other internet-related stuff). It is envisioned to work with XML data encoding and remote procedure calls (so 2000s ...), but it is extremelly flexible and can be used for a multitude of purposes. In turn, pyang is a python project that provides parsing, validation, transformation and code generation functionalities. Despite of being extensible, the pyang code is a little bit complex, and the documentation scarce. This makes the task of building plugins difficult.

pyangext aims to provide a common foundation for plugins, wrapping pyang features, and making easier to use it programatically outside pyang own code-base.

If you are one of the pyang authors...

You guys have done an amazing job, please don't feel upset about this documentation. I'm trying to make it interesting and a little bit funny. The ultimate goal is to have an amazing pyang environment, and if you would like to merge pyangext inside pyang, please let me know.

1.1.2 Getting Started

If you are not a plugin writer

Well, pyangext will not exactly change your life... but you can have a little benefit from it, so let's **install all the things!**

```
sudo pip install pyangext
# drop sudo if you are using a virtualenv or pyenv
```

There are some python packages that register themselves as pyang plugins using setuptools entry-points. While pyang does not natively support it, pyangext will consider it and generate a complete plugin path. You can activate it by doing:

```
eval $(pyangext --export-path)
```

If you like it, you can also include it in your `.(ba|z)shrc` file.

DONE

If you ARE a plugin writer

You have probably noticed that `pyang` does not support the standard `setuptools` entry-points way of writing plugins. Instead it requires that the user either copies the plugin to the `pyang` plugins directory, or changes manually the `PYANG_PLUGINPATH` env var. Sometimes this makes difficult to describe how to use your plugin, e.g. pyangbind.

Using, `pyangext` you can:

1. Create an empty plugin package inside your project (folder with just and empty `__init__.py` file inside).
2. Put just your plugin modules inside it (`.py` files containing `pyang_plugin_init` function).
3. Register a `setuptools` entry-point under the `yang.plugins` section, with the name of your plugin, pointint to that function.
4. Ask your users in the documentation to use `eval $(pyangext --export-path)` before runing `pyang`, **or** exchange the `pyang` shell commad by `pyangext run` with the same arguments.
5. Distribute your package using PyPI/pip tools.

Additionally `pyangext` provides two other submodules with functions that can be used in your code. The *`pyangext.utils`* module provides functions like `create_context`, `parse`, `dump`, `walk`. This functions are very useful, and a little example is provided bellow:

```
from pyangext.utils import create_context, dump, find, parse, walk
ctx = create_context(keep_comments=True, features=['if:if-mib'])
ast = parse('leaf id { type int32; }', ctx)  # tree-ish structure
print(dump(ast, ctx))  # produce YANG code
used_types = walk(ast,
                  select=lambda node: node.keyword == 'type',
                  apply=lambda node: node.arg)
# => ['int32']
int32_nodes = find(ast, 'type', 'int32')  # list with 1 object
```

`pyang.Context` object plays a central role in the `pyang` architecture. The `create_context` can be used to create this object in a similar way it is created by the `pyang` CLI.

The *`pyangext.definitions`* on the other hand provides some constants like the `BUILT_IN_TYPES` list.

Note: There are few well known issues with `create_context` and `parse` functions preventing them to be used by standalone python scripts, like the lack of YANG `deviation` support. Despite they can be used in most situations, the prefered way of manipulating the YANG Abstract Syntax Tree (AST) is yet writing a plugin.

See also:

`pyangext.cli` *`pyangext.utils`*

1.1.3 Stuff Doesn't Work

This work was tested and I think it's stable, but any feedback you can give me on this would be gratefully received (see section **Reporting a Bug** at Contribution Guidelines.).

1.1.4 Can I help?

Yes, please! Contributions of any kind are welcome, and also feel free to ask your questions!

Please take a look at the Contribution Guidelines.

Well-known list of TODOs

- Make sure `augment`, `deviation` and `include` work with both `ctx.add_module` and `parse`. (by writing tests and making it pass).
- Use `ctx.add_module` under the hood when a file name is passed to `parse`. If it is a module, why not add it to context as well?
- Make `parse` and `dump` work with `yin` format.

Doubts

- Perform `ctx.validate` and `validate_module` under the hood?
- Abstract Context and `i_` magic method?

Ultimate Goals

- Allow `pyang` plugins to be written as standalone python scripts. (I think it is better to have small focused scripts, instead of a huge amount of options in the `pyang` CLI)
- Merged into `pyang` own code base.

1.2 pyangext

1.2.1 pyangext package

Submodules

pyangext.cli module

Extension for the pyang command line interface.

This module includes tools for augmenting `PYANG_PLUGINPATH` with the location of auto-discovered pyang plugins.

Pyang do not use the `setuptools` to register plugins. Instead it requires that the paths of all directories containing plugins to be present in the `PYANG_PLUGINPATH` environment variable.

`pyangext` reads all entry points under `yang.plugins`, detect the path to the file that contains the function registered, and builds a list with the containing directories.

In this sense, `pyangext run` command can be used as a bridge to the `pyang` command, but using the auto-discovery feature.

Note: Including non pyang-plugin python files alongside pyang-plugins python files (in the same directory) will result in a pyang CLI crash.

It is recommended that the function registered as entry-point follows the proprietary pyang plugin convention, or in other words:

- it should be named `pyang_plugin_init`
- it should call `pyang.plugin.register_plugin` with an instance of `pyang.plugin.PyangPlugin` as argument.

See also:

https://pythonhosted.org/setuptools/setuptools.html#dynamic-discovery-of-services-and-plugins

Command Line Interface

Usage: `pyangext [OPTIONS] COMMAND [ARGS]`

Options:

-h, --help	Show this message and exit.
-v, --version	Show the version and exit.
--path	Prints the auto discovered plugin path. Python packages that register an entry-point inside `yang.plugins` will be auto-detected.
--init, --export-path	Prints an export shell statement with the auto discovered plugin path. This may be used by shell script to configure `PYANG_PLUGINPATH` environment variable. Example:
--help	Show this message and exit.

Commands:

`call` invoke pyang script with plugin path adjusted using auto-discovery.

`pyangext.cli.`**`export_path`**(*ctx*, *_*, *value*)

Prints an export shell statement with the auto discovered plugin path.

This may be used by shell script to configure `PYANG_PLUGINPATH` environment variable.

Example

```
eval $(pyangext --export-path)
```

`pyangext.cli.`**`print_path`**(*ctx*, *_*, *value*)

Prints the auto discovered plugin path.

Packages that register an `yang.plugins` entry-point will be auto-detected.

pyangext.definitions module

Meta information about YANG modeling language.

See also:

https://tools.ietf.org/html/rfc6020

pyangext.definitions.**BUILT_IN_TYPES = ['binary', 'bits', 'boolean', 'decimal64', 'empty', 'enumeration', 'identityref**
Types supported by default in the YANG language.

pyangext.definitions.**DATA_STATEMENTS = ['container', 'leaf', 'leaf-list', 'list', 'anyxml']**
Statements that denote a data node in the abstract tree.

pyangext.definitions.**HEADER_STATEMENTS = ['organization', 'contact', 'revision', 'yang-version']**
Descriptive statements used in the header of a module or submodule.

pyangext.definitions.**ID_STATEMENTS = ['namespace', 'prefix']**
Statements used to identify the module.

pyangext.definitions.**PREFIX_SEPARATOR = ':'**
Character used to denote prefix in YANG language.

pyangext.definitions.**YANG_KEYWORDS = ['action', 'anydata', 'anyxml', 'argument', 'augment', 'base', 'belongs-to', 'b**
YANG language Keywords.

pyangext.paths module

Automatically discover pyang plugins by reading setuptools entry-points.

pyangext.paths.**discover**()
Discovers pyang plugins registered using setuptools entry points.

Collects the path for all python modules that have functions registered as an entry point inside `yang.plugins` group.

Ideally the function registered should be named `pyang_plugin_init`. It is also important to not include non-pyang-plugin python modules in the same directory of this module.

Returns Array of paths that contains python modules with pyang plugins.

Reference: https://pythonhosted.org/setuptools/setuptools.html#dynamic-discovery-of-services-and-plugins

pyangext.paths.**expanded**()
Combines the auto-discovered plugin paths with env `PYANG_PLUGINPATH`.

This function appends paths discovered using `discover` function to the list provided by `PYANG_PLUGINPATH` environment variable. It also removes duplicated entries from the resulting list.

Returns Array of paths that contains python modules with pyang plugins.

pyangext.utils module

Utility belt for working with `pyang` and `pyangext`.

pyangext.utils.**create_context**(*path='.'*, **options*, ***kwargs*)
Generates a pyang context.

The dict options and keyword arguments are similar to the command line options for `pyang`. For `plugindir` use env var `PYANG_PLUGINPATH`. For `path` option use the argument with the same name, or `PYANG_MODPATH` env var.

Parameters

- **path** (*str*) – location of YANG modules. (Join string with `os.pathsep` for multiple locations). Default is the current working dir.

- ***options** – list of dicts, with options to be passed to context. See bellow.
- ****kwargs** – similar to `options` but have a higher precedence. See bellow.

Keyword Arguments

- **print_error_code** (*bool*) – On errors, print the error code instead of the error message. Default `False`.
- **warnings** (*list*) – If contains `error`, treat all warnings as errors, except any other error code in the list. If contains `none`, do not report any warning.
- **errors** (*list*) – Treat each error code container as an error.
- **ignore_error_tags** (*list*) – Ignore error code. (For a list of error codes see `pyang --list-errors`).
- **ignore_errors** (*bool*) – Ignore all errors. Default `False`.
- **canonical** (*bool*) – Validate the module(s) according to the canonical YANG order. Default `False`.
- **yang_canonical** (*bool*) – Print YANG statements according to the canonical order. Default `False`.
- **yang_remove_unused_imports** (*bool*) – Remove unused import statements when printing YANG. Default `False`.
- **trim_yin** (*bool*) – In YIN input modules, trim whitespace in textual arguments. Default `False`.
- **lax_xpath_checks** (*bool*) – Lax check of XPath expressions. Default `False`.
- **strict** (*bool*) – Force strict YANG compliance. Default `False`.
- **max_line_len** (*int*) – Maximum line length allowed. Disabled by default.
- **max_identifier_len** (*int*) – Maximum identifier length allowed. Disabled by default.
- **features** (*list*) – Features to support, default all. Format `<modname>:[<feature>,]*`.
- **keep_comments** (*bool*) – Do not discard comments. Default `True`.
- **no_path_recurse** (*bool*) – Do not recurse into directories in the yang path. Default `False`.

Returns Context object for `pyang` usage

Return type pyang.Context

`pyangext.utils.`**`compare_prefixed`**(*arg1*, *arg2*, *prefix_sep=':'*, *ignore_prefix=False*)

Compare 2 arguments : prefixed strings or tuple `(prefix, string)`

Parameters

- **arg1** (*`str or tuple`*) – first argument
- **arg2** (*`str or tuple`*) – first argument
- **prefix_sep** (*`str`*) – prefix string separator (default: `':'`)

Returns bool

`pyangext.utils.`**`qualify_str`**(*arg*, *prefix_sep=':'*)

Transform prefixed strings in tuple `(prefix, string)`

`pyangext.utils.`**`select`**(*statements*, *keyword=None*, *arg=None*, *ignore_prefix=False*)

Given a list of statements filter by keyword, or argument or both.

Parameters

- **statements** (*list of pyang.statements.Statement*) – list of statements to be filtered.
- **keyword** (*str*) – if specified the statements should have this keyword
- **arg** (*str*) – if specified the statements should have this argument

`keyword` and `arg` can be also used as keyword arguments.

Returns nodes that matches the conditions

Return type list

`pyangext.utils.`**find**(*parent*, *keyword=None*, *arg=None*, *ignore_prefix=False*)

Select all sub-statements by keyword, or argument or both.

See also:

function *select()*

`pyangext.utils.`**dump**(*node*, *file_obj=None*, *prev_indent=''*, *indent_string=' '*, *ctx=None*)

Generate a string representation of an abstract syntax tree.

Parameters

- **node** (*pyang.statements.Statement*) – object to be represented
- **file_obj** (*file*) – *file-like* object where the representation will be dumped. If nothing is passed, the method returns a string

Keyword Arguments

- **prev_indent** (*str*) – string to be added to the produced indentation
- **indent_string** (*str*) – string to be used as indentation
- **ctx** (*pyang.Context*) – context object used to generate string representation. If no context is passed, a dummy object is used with default configuration

Returns text content if `file_obj` is not specified

Return type str

`pyangext.utils.`**check**(*ctx*, *rescue=False*)

Check existence of errors or warnings in context.

Code mostly borrowed from `pyang` script.

Parameters **ctx** (*pyang.Context*) – pyang context to be checked.

Keyword Arguments **rescue** (*bool*) – if `True`, no exception/warning will be raised.

Raises `SyntaxError` – if errors detected

Warns **SyntaxWarning** – if warnings detected

Returns (list of errors, list of warnings), if `rescue` is `True`

Return type tuple

`pyangext.utils.`**parse**(*text*, *ctx=None*)

Parse a YANG statement into an Abstract Syntax subtree.

Parameters

- **text** (*str*) – file name for a YANG module or text

- **ctx** (*optional pyang.Context*) – context used to validate text

Returns Abstract syntax subtree

Return type pyang.statements.Statement

Note: The `parse` function can be used to parse small amounts of text. If yout plan to parse an entire YANG (sub)module, please use instead:

```
ast = ctx.add_module(module_name, text_contet)
```

It is also well known that `parse` function cannot solve YANG deviations yet.

---

pyangext.utils.**walk**(*parent*, *select=<function <lambda>>*, *apply=<function <lambda>>*, *key='substmts'*)

Recursivelly find nodes and/or apply a function to them.

Parameters

- **parent** (*pyang.statements.Statement*) – root of the subtree were the search will take place.
- **select** – optional callable that receives a node and returns a bool (True if the node matches the criteria)
- **apply** – optional callable that are going to be applied to the node if it matches the criteria
- **key** (*str*) – property where the children nodes are stored, default is `substmts`

Returns results collected from the apply function

Return type list

Module contents

Critical missing features for `pyang` plugin users and authors.

1.3 License

```
Mozilla Public License, version 2.0

1. Definitions

1.1. "Contributor"

     means each individual or legal entity that creates, contributes to the
     creation of, or owns Covered Software.

1.2. "Contributor Version"

     means the combination of the Contributions of others (if any) used by a
     Contributor and that particular Contributor's Contribution.

1.3. "Contribution"

     means Covered Software of a particular Contributor.
```

```
1.4. "Covered Software"

     means Source Code Form to which the initial Contributor has attached the
     notice in Exhibit A, the Executable Form of such Source Code Form, and
     Modifications of such Source Code Form, in each case including portions
     thereof.

1.5. "Incompatible With Secondary Licenses"
     means

     a. that the initial Contributor has attached the notice described in
        Exhibit B to the Covered Software; or

     b. that the Covered Software was made available under the terms of
        version 1.1 or earlier of the License, but not also under the terms of
        a Secondary License.

1.6. "Executable Form"

     means any form of the work other than Source Code Form.

1.7. "Larger Work"

     means a work that combines Covered Software with other material, in a
     separate file or files, that is not Covered Software.

1.8. "License"

     means this document.

1.9. "Licensable"

     means having the right to grant, to the maximum extent possible, whether
     at the time of the initial grant or subsequently, any and all of the
     rights conveyed by this License.

1.10. "Modifications"

     means any of the following:

     a. any file in Source Code Form that results from an addition to,
        deletion from, or modification of the contents of Covered Software; or

     b. any new file in Source Code Form that contains any Covered Software.

1.11. "Patent Claims" of a Contributor

      means any patent claim(s), including without limitation, method,
      process, and apparatus claims, in any patent Licensable by such
      Contributor that would be infringed, but for the grant of the License,
      by the making, using, selling, offering for sale, having made, import,
      or transfer of either its Contributions or its Contributor Version.

1.12. "Secondary License"

      means either the GNU General Public License, Version 2.0, the GNU Lesser
      General Public License, Version 2.1, the GNU Affero General Public
      License, Version 3.0, or any later versions of those licenses.
```

```
1.13. "Source Code Form"

      means the form of the work preferred for making modifications.

1.14. "You" (or "Your")

      means an individual or a legal entity exercising rights under this
      License. For legal entities, "You" includes any entity that controls, is
      controlled by, or is under common control with You. For purposes of this
      definition, "control" means (a) the power, direct or indirect, to cause
      the direction or management of such entity, whether by contract or
      otherwise, or (b) ownership of more than fifty percent (50%) of the
      outstanding shares or beneficial ownership of such entity.


2. License Grants and Conditions

2.1. Grants

     Each Contributor hereby grants You a world-wide, royalty-free,
     non-exclusive license:

     a. under intellectual property rights (other than patent or trademark)
        Licensable by such Contributor to use, reproduce, make available,
        modify, display, perform, distribute, and otherwise exploit its
        Contributions, either on an unmodified basis, with Modifications, or
        as part of a Larger Work; and

     b. under Patent Claims of such Contributor to make, use, sell, offer for
        sale, have made, import, and otherwise transfer either its
        Contributions or its Contributor Version.

2.2. Effective Date

     The licenses granted in Section 2.1 with respect to any Contribution
     become effective for each Contribution on the date the Contributor first
     distributes such Contribution.

2.3. Limitations on Grant Scope

     The licenses granted in this Section 2 are the only rights granted under
     this License. No additional rights or licenses will be implied from the
     distribution or licensing of Covered Software under this License.
     Notwithstanding Section 2.1(b) above, no patent license is granted by a
     Contributor:

     a. for any code that a Contributor has removed from Covered Software; or

     b. for infringements caused by: (i) Your and any other third party's
        modifications of Covered Software, or (ii) the combination of its
        Contributions with other software (except as part of its Contributor
        Version); or

     c. under Patent Claims infringed by Covered Software in the absence of
        its Contributions.

     This License does not grant any rights in the trademarks, service marks,
     or logos of any Contributor (except as may be necessary to comply with
```

```
     the notice requirements in Section 3.4).

2.4. Subsequent Licenses

     No Contributor makes additional grants as a result of Your choice to
     distribute the Covered Software under a subsequent version of this
     License (see Section 10.2) or under the terms of a Secondary License (if
     permitted under the terms of Section 3.3).

2.5. Representation

     Each Contributor represents that the Contributor believes its
     Contributions are its original creation(s) or it has sufficient rights to
     grant the rights to its Contributions conveyed by this License.

2.6. Fair Use

     This License is not intended to limit any rights You have under
     applicable copyright doctrines of fair use, fair dealing, or other
     equivalents.

2.7. Conditions

     Sections 3.1, 3.2, 3.3, and 3.4 are conditions of the licenses granted in
     Section 2.1.


3. Responsibilities

3.1. Distribution of Source Form

     All distribution of Covered Software in Source Code Form, including any
     Modifications that You create or to which You contribute, must be under
     the terms of this License. You must inform recipients that the Source
     Code Form of the Covered Software is governed by the terms of this
     License, and how they can obtain a copy of this License. You may not
     attempt to alter or restrict the recipients' rights in the Source Code
     Form.

3.2. Distribution of Executable Form

     If You distribute Covered Software in Executable Form then:

     a. such Covered Software must also be made available in Source Code Form,
        as described in Section 3.1, and You must inform recipients of the
        Executable Form how they can obtain a copy of such Source Code Form by
        reasonable means in a timely manner, at a charge no more than the cost
        of distribution to the recipient; and

     b. You may distribute such Executable Form under the terms of this
        License, or sublicense it under different terms, provided that the
        license for the Executable Form does not attempt to limit or alter the
        recipients' rights in the Source Code Form under this License.

3.3. Distribution of a Larger Work

     You may create and distribute a Larger Work under terms of Your choice,
     provided that You also comply with the requirements of this License for
```

```
     the Covered Software. If the Larger Work is a combination of Covered
     Software with a work governed by one or more Secondary Licenses, and the
     Covered Software is not Incompatible With Secondary Licenses, this
     License permits You to additionally distribute such Covered Software
     under the terms of such Secondary License(s), so that the recipient of
     the Larger Work may, at their option, further distribute the Covered
     Software under the terms of either this License or such Secondary
     License(s).

3.4. Notices

     You may not remove or alter the substance of any license notices
     (including copyright notices, patent notices, disclaimers of warranty, or
     limitations of liability) contained within the Source Code Form of the
     Covered Software, except that You may alter any license notices to the
     extent required to remedy known factual inaccuracies.

3.5. Application of Additional Terms

     You may choose to offer, and to charge a fee for, warranty, support,
     indemnity or liability obligations to one or more recipients of Covered
     Software. However, You may do so only on Your own behalf, and not on
     behalf of any Contributor. You must make it absolutely clear that any
     such warranty, support, indemnity, or liability obligation is offered by
     You alone, and You hereby agree to indemnify every Contributor for any
     liability incurred by such Contributor as a result of warranty, support,
     indemnity or liability terms You offer. You may include additional
     disclaimers of warranty and limitations of liability specific to any
     jurisdiction.

4. Inability to Comply Due to Statute or Regulation

   If it is impossible for You to comply with any of the terms of this License
   with respect to some or all of the Covered Software due to statute,
   judicial order, or regulation then You must: (a) comply with the terms of
   this License to the maximum extent possible; and (b) describe the
   limitations and the code they affect. Such description must be placed in a
   text file included with all distributions of the Covered Software under
   this License. Except to the extent prohibited by statute or regulation,
   such description must be sufficiently detailed for a recipient of ordinary
   skill to be able to understand it.

5. Termination

5.1. The rights granted under this License will terminate automatically if You
     fail to comply with any of its terms. However, if You become compliant,
     then the rights granted under this License from a particular Contributor
     are reinstated (a) provisionally, unless and until such Contributor
     explicitly and finally terminates Your grants, and (b) on an ongoing
     basis, if such Contributor fails to notify You of the non-compliance by
     some reasonable means prior to 60 days after You have come back into
     compliance. Moreover, Your grants from a particular Contributor are
     reinstated on an ongoing basis if such Contributor notifies You of the
     non-compliance by some reasonable means, this is the first time You have
     received notice of non-compliance with this License from such
     Contributor, and You become compliant prior to 30 days after Your receipt
     of the notice.
```

```
5.2. If You initiate litigation against any entity by asserting a patent
     infringement claim (excluding declaratory judgment actions,
     counter-claims, and cross-claims) alleging that a Contributor Version
     directly or indirectly infringes any patent, then the rights granted to
     You by any and all Contributors for the Covered Software under Section
     2.1 of this License shall terminate.

5.3. In the event of termination under Sections 5.1 or 5.2 above, all end user
     license agreements (excluding distributors and resellers) which have been
     validly granted by You or Your distributors under this License prior to
     termination shall survive termination.

6. Disclaimer of Warranty

   Covered Software is provided under this License on an "as is" basis,
   without warranty of any kind, either expressed, implied, or statutory,
   including, without limitation, warranties that the Covered Software is free
   of defects, merchantable, fit for a particular purpose or non-infringing.
   The entire risk as to the quality and performance of the Covered Software
   is with You. Should any Covered Software prove defective in any respect,
   You (not any Contributor) assume the cost of any necessary servicing,
   repair, or correction. This disclaimer of warranty constitutes an essential
   part of this License. No use of  any Covered Software is authorized under
   this License except under this disclaimer.

7. Limitation of Liability

   Under no circumstances and under no legal theory, whether tort (including
   negligence), contract, or otherwise, shall any Contributor, or anyone who
   distributes Covered Software as permitted above, be liable to You for any
   direct, indirect, special, incidental, or consequential damages of any
   character including, without limitation, damages for lost profits, loss of
   goodwill, work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses, even if such party shall have been
   informed of the possibility of such damages. This limitation of liability
   shall not apply to liability for death or personal injury resulting from
   such party's negligence to the extent applicable law prohibits such
   limitation. Some jurisdictions do not allow the exclusion or limitation of
   incidental or consequential damages, so this exclusion and limitation may
   not apply to You.

8. Litigation

   Any litigation relating to this License may be brought only in the courts
   of a jurisdiction where the defendant maintains its principal place of
   business and such litigation shall be governed by laws of that
   jurisdiction, without reference to its conflict-of-law provisions. Nothing
   in this Section shall prevent a party's ability to bring cross-claims or
   counter-claims.

9. Miscellaneous

   This License represents the complete agreement concerning the subject
   matter hereof. If any provision of this License is held to be
   unenforceable, such provision shall be reformed only to the extent
   necessary to make it enforceable. Any law or regulation which provides that
   the language of a contract shall be construed against the drafter shall not
   be used to construe this License against a Contributor.
```

```
10. Versions of the License

10.1. New Versions

      Mozilla Foundation is the license steward. Except as provided in Section
      10.3, no one other than the license steward has the right to modify or
      publish new versions of this License. Each version will be given a
      distinguishing version number.

10.2. Effect of New Versions

      You may distribute the Covered Software under the terms of the version
      of the License under which You originally received the Covered Software,
      or under the terms of any subsequent version published by the license
      steward.

10.3. Modified Versions

      If you create software not governed by this License, and you want to
      create a new license for such software, you may create and use a
      modified version of this License if you rename the license and remove
      any references to the name of the license steward (except to note that
      such modified license differs from this License).

10.4. Distributing Source Code Form that is Incompatible With Secondary
      Licenses If You choose to distribute Source Code Form that is
      Incompatible With Secondary Licenses under the terms of this version of
      the License, the notice described in Exhibit B of this License must be
      attached.

Exhibit A - Source Code Form License Notice

      This Source Code Form is subject to the
      terms of the Mozilla Public License, v.
      2.0. If a copy of the MPL was not
      distributed with this file, You can
      obtain one at
      http://mozilla.org/MPL/2.0/.

If it is not possible or desirable to put the notice in a particular file,
then You may include the notice in a location (such as a LICENSE file in a
relevant directory) where a recipient would be likely to look for such a
notice.

You may add additional accurate notices of copyright ownership.

Exhibit B - "Incompatible With Secondary Licenses" Notice

      This Source Code Form is "Incompatible
      With Secondary Licenses", as defined by
      the Mozilla Public License, v. 2.0.
```

1.4 Developers

- Anderson Bravalheri <andersonbravalheri@gmail.com>

1.5 Changelog

1.5.1 Version 0.0.1 (2016-06-01)

Features

- **ast:**
- add select, find, from_tuple, append (67724329)
- add walk function to traverse tree (1f334d3d)
- **auto-discover:** add plugins, setuptools way (10794a94)
- **ctx:** add utility function to context creation (72f43d90)
- **parse:** add parse function (str => AST) (ab3b465b)

Documentation

- improve doc generation and contents (e471c595)
- **create_context:** document options for context (c2ad6d0f)
- **project:** improve overall project docs. (e1b60ecd)
- **requirements:** create a separated req file (94882cd4)

Test

- **cli:** Add cli tests (11501c44)

1.6 How to contribute

Pull-requests and discussions are essential for any open-source project. Any contribution to this project will be considered lovely. Here's just a quick guide to help you in this journey.

Please have in mind that nothing can be considered 100% truth and immutable (including this statement). This project will not adhere to any `strict` way of development.

1.6.1 Pull-Requests

Github has two great GREAT articles about contributing: Contributing to Open Source on GitHub and Using pull requests. Please make sure to read it in your lifetime (everyone that reads became a better person).

Note: Oh man, guides.github.com and help.github.com are astonishing!

Please, try to keep your commit messages as communicative as possible. There is a good reference for it as well.

Note: I usually think in the commit itself as an implicit subject of commit message. For example: `[This commit] Add .gitignore` Also take a look at this commit message format proposal, that borrows some convention from AngularJS.

Communication is *always* handy! If you have any doubt or would like to discuss your thoughts, you are more than welcome to send me a message! Please comment directly on the code, open an issue, submit a pull request, mention me anywhere... I think GitHub has good tools to help developers communicate and share experiences.

Code Guidelines

This repository try to adhere to PEP8 as much as possible.

Please make use of tools like flake8, pylint, isort, and pre-commit before submitting your code. There are configuration files for all these tools in the root of the repository and the easiest way of starting is by doing:

```
sudo pip install pre-commit
# drop sudo if you are using a virtualenv or pyenv
# inside project directory:
pre-commit install
pre-commit run --all-files
```

Please also consider running the test suite before submitting a pull request:

```
python setup.py test
```

1.6.2 Reporting a Bug

- Update to the most recent master release if possible. Someone may have already fixed your bug (such a wonderful scenario!)
- Search for similar issues. It's possible somebody has encountered this bug already. In this case comment your experience too!
- Clearly describe the issue including steps to reproduce when it is a bug and preferably send a script that does so. Try to keep all the things fully operational with the exception of the bug you want to demonstrate. (Ok, I admit this is boring, but is probably the fastest way to get thing working).
- Keep up to date with feedback from the project team, maybe you can help us to test ;)
- If possible, submit a Pull Request with a failing test. It would be wonderful to increase the test coverage!
- Consider the challenge of fixing the bug, I'm sure it can be funny or at least very aggrandizing.

1.6.3 Requesting a Feature

- Search Issues for similar feature requests. It's possible somebody has already asked for this feature or provided a pull request that we're still discussing.
- Provide a clear and detailed explanation of the feature you want and why it's important to add. Keep in mind that features should be useful to the majority of users and not just a small subset. If you're just targeting a minority of users, consider writing an add-on library.
- If the feature is complex, consider writing some initial documentation for it. If we do end up accepting the feature it will need to be documented and this will also help us to understand it better ourselves.

- Attempt a Pull Request. If you're at all able, start writing some code. We always have more work to do than time to do it. If you can write some code then that will speed the process along.

Note: This guide was partially copied from

- ember.js
- factory_girl
- puppet
- rails

Please consider reading them. They are just great!

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

p

B

C

D

E

F

H

I

P

Q

S

W

Y