Lars 1.0 Documentation

Release 1.0

Dave Jones

Jan 04, 2018

Contents

1 Install **1**
- 1.1 Pre-requisites . . . 1
- 1.2 Ubuntu Linux . . . 1
- 1.3 Other Platforms . . . 1

2 Introduction **3**
- 2.1 Filtering rows . . . 3
- 2.2 Manipulating row content . . . 5

3 API Reference **7**
- 3.1 lars.apache - Reading Apache Logs . . . 7
- 3.2 lars.iis - Reading IIS Logs . . . 7
- 3.3 lars.csv - Writing CSV Files . . . 7
- 3.4 lars.sql - Direct Database Output . . . 9
- 3.5 lars.geoip - GeoIP Database Access . . . 9
- 3.6 lars.datatypes - Web Log Datatypes . . . 9
- 3.7 lars.progress - Rendering Progress . . . 9
- 3.8 lars.dns - DNS Resolution . . . 11
- 3.9 lars.cache - Cache Decorators . . . 12
- 3.10 lars.exc - Base Exceptions . . . 12

4 Change log **13**
- 4.1 Release 1.0 (2017-01-04) . . . 13
- 4.2 Release 0.3 (2014-09-07) . . . 13
- 4.3 Release 0.2 (2013-07-28) . . . 13
- 4.4 Release 0.1 (2013-06-09) . . . 13

5 License **15**
- 5.1 DateTime, Date, and Time documentation license . . . 15
- 5.2 _strptime license . . . 16
- 5.3 IPNetwork & IPAddress documentation license . . . 17

Python Module Index **19**

CHAPTER 1

Install

lars is distributed in several formats. The following sections detail installation on a variety of platforms.

1.1 Pre-requisites

Where possible, installation methods will automatically handle all mandatory pre-requisites. However, if your particular installation method does not handle dependency installation, then you will need to install the following Python packages manually:

- pygeoip[1] - The pure Python API for MaxMind GeoIP databases
- ipaddress[2] - Google's IPv4 and IPv6 address handling library. This is included as standard in Python 3.3 and above.

1.2 Ubuntu Linux

For Ubuntu Linux, it is simplest to install from the Waveform PPA[3] as follows (this also ensures you are kept up to date as new releases are made):

```
$ sudo add-apt-repository ppa://waveform/ppa
$ sudo apt-get update
$ sudo apt-get install python-lars
```

1.3 Other Platforms

If your platform is *not* covered by one of the sections above, lars is available from PyPI and can therefore be installed with the Python setuptools `easy_install` tool:

```
$ easy_install lars
```

[1] https://pypi.python.org/pypi/pygeoip/
[2] https://pypi.python.org/pypi/ipaddress/
[3] https://launchpad.net/~waveform/+archive/ppa

Or the (now deprecated) distribute `pip` tool:

```
$ pip install lars
```

If you do not have either of these tools available, please install the Python setuptools[4] package first.

[4] https://pypi.python.org/pypi/setuptools/

CHAPTER 2

Introduction

A typical lars script opens some log source, typically a file, and uses the source and target wrappers provided by lars to convert the log entries into some other format (potentially filtering and/or modifying the entries along the way). A trivial script to convert IIS W3C style log entries into a CSV file is shown below:

```
import io
from lars import iis, csv

with io.open('webserver.log', 'r') as infile, \
         io.open('output.csv', 'wb') as outfile:
    with iis.IISSource(infile) as source, csv.CSVTarget(outfile) as target:
        for row in source:
            target.write(row)
```

Going through this section by section we can see the following:

1. The first couple of lines import the necessary modules that we'll need; the standard Python `io`[5] module for opening files, and the `iis` and `csv`[6] modules from lars for converting the data.
2. Using `io.open()`[7] we open the input file (with mode `'r'` for reading) and the output file (with mode `'wb'` for creating a new file and writing (binary mode) to it)
3. We wrap `infile` (the input file) with `IISSource` to parse the input file, and `outfile` (the output file) with *`CSVTarget`* (page 7) to format the output file.
4. Finally, we use a simple loop to iterate over the rows in the source file, and the *`write()`* (page 8) method to write them to the target.

This is the basic structure of most lars scripts. Most extra lines for filtering and manipulating rows appear within the loop at the end of the file, although sometimes extra module configuration lines are required at the top.

2.1 Filtering rows

The row object declared in the loop has attributes named after the columns of the source (with characters that cannot appear in Python identifiers replaced with underscores). To see the structure of a row you can simply print one and then terminate the loop:

[5] https://docs.python.org/3.5/library/io.html#module-io
[6] https://docs.python.org/3.5/library/csv.html#module-csv
[7] https://docs.python.org/3.5/library/io.html#io.open

```
import io
from lars import iis, csv

with io.open('webserver.log', 'r') as infile, \
        io.open('output.csv', 'wb') as outfile:
    with iis.IISSource(infile) as source, csv.CSVTarget(outfile) as target:
        for row in source:
            print(row)
            break
```

Given the following input file (long lines indented for readability):

```
#Software: Microsoft Internet Information Services 6.0
#Version: 1.0
#Date: 2002-05-24 20:18:01
#Fields: date time c-ip cs-username s-ip s-port cs-method cs-uri-stem
    cs-uri-query sc-status sc-bytes cs-bytes time-taken cs(User-Agent)
    cs(Referrer)
2002-05-24 20:18:01 172.224.24.114 - 206.73.118.24 80 GET /Default.htm -
    200 7930 248 31
    Mozilla/4.0+(compatible;+MSIE+5.01;+Windows+2000+Server)
    http://64.224.24.114/
```

This will produce this output on the command line:

```
Row(date=Date(2002, 5, 24), time=Time(20, 18, 1),
    c_ip=IPv4Address(u'172.224.24.114'), cs_username=None,
    s_ip=IPv4Address(u'206.73.118.24'), s_port=80, cs_method=u'GET',
    cs_uri_stem=Url(scheme='', netloc='', path=u'/Default.htm', params='',
    query_str='', fragment=''), cs_uri_query=None, sc_status=200,
    sc_bytes=7930, cs_bytes=248, time_taken=31.0,
    cs_User_Agent=u'Mozilla/4.0 (compatible; MSIE 5.01; Windows 2000
    Server)', cs_Referrer=Url(scheme=u'http', netloc=u'64.224.24.114',
     path=u'/', params='', query_str='', fragment=''))
```

From this one can see that field names like `c-ip` have been converted into `c_ip` (`-` is an illegal character in Python identifiers). Furthermore it is apparent that instead of simple strings being extracted, the data has been converted into a variety of appropriate datatypes (`Date` for the `date` field, `Url` for the `cs-uri-stem` field, and so on). This significantly aids in filtering rows based upon sub-attributes of the extracted data.

For example, to filter on the year of the date:

```
if row.date.year == 2002:
    target.write(row)
```

Alternatively, you could filter on whether or not the client IP belongs in a particular network:

```
if row.c_ip in datatypes.network('172.0.0.0/8'):
    target.write(row)
```

Or use Python's string methods[8] to filter on any string:

```
if row.cs_User_Agent.startswith('Mozilla/'):
    target.write(row)
```

Or any combination of the above:

```
if row.date.year == 2002 and 'MSIE' in row.cs_User_Agent:
    target.write(row)
```

[8] http://docs.python.org/2/library/stdtypes.html#string-methods

2.2 Manipulating row content

If you wish to modify the output structure,the simplest method is to declare the row structure you want at the top of the file (using the `row()` function) and then construct rows with the new structure in the loop (using the result of the function):

```
import io
from lars import datatypes, iis, csv

NewRow = datatypes.row('date', 'time', 'client', 'url')

with io.open('webserver.log', 'r') as infile, \
        io.open('output.csv', 'wb') as outfile:
    with iis.IISSource(infile) as source, csv.CSVTarget(outfile) as target:
        for row in source:
            new_row = NewRow(row.date, row.time, row.c_ip, row.cs_uri_stem)
            target.write(new_row)
```

There is no need to convert column data back to strings for output; all datatypes produced by lars source adapters have built-in string conversions which all target adapters know to use.

CHAPTER 3

API Reference

The framework is designed in a modular fashion with a separate module for each log input format, each data output format, a few auxilliary modules for the datatypes exposed by the framework and their functionality. Where possible, standards dictating formats are linked in the API reference.

Each module comes with documentation including examples of usage. The best way to learn the framework is to peruse the API reference and try out the examples, modifying them to suit your purposes.

3.1 lars.apache - Reading Apache Logs

3.2 lars.iis - Reading IIS Logs

3.3 lars.csv - Writing CSV Files

This module provides a target wrapper for CSV (Comma Separated Values) formatted text files, which are typically used as a generic source format for bulk loading databases.

The *CSVTarget* (page 7) class is the major element that this module provides; it is a standard target class (a context manager with a *write()* (page 8) method that accepts row tuples).

3.3.1 Classes

class lars.csv.**CSVTarget**(*fileobj*, *header=False*, *dialect=CSV_DIALECT*, *encoding='utf-8'*, ***kwargs*)

Wraps a stream to format rows as CSV (Comma Separated Values).

This wrapper provides a simple *write()* (page 8) method which can be used to format row tuples as comma separated values in a variety of common dialects. The dialect defaults to *CSV_DIALECT* (page 8) which produces a typical CSV file compatible with the vast majority of products.

If you desire a different output format you can either specify a different value for the *dialect* parameter, or if you only wish to use a minimal modification of the dialect you can override its attributes with keyword arguments. For example:

```
CSVTarget(outfile, dialect=CSV_DIALECT, lineterminator='\n')
```

The *encoding* parameter controls the character set used in the output. This defaults to UTF-8 which is a sensible default for most modern systems, but is a multi-byte encoding which some legacy systems (notably mainframes) may have troubles with. In this case you can either select a single byte encoding like ISO-8859-1 or even EBCDIC. See Python standard encodings[9] for a full list of supported encodings.

Warning: The file that you wrap with *CSVTarget* (page 7) *must* be opened in binary mode (`'wb'`) partly because the dialect dictates the line terminator that is used, and partly because the class handles its own character encoding.

close()
Closes the CSV output. Further calls to *write()* (page 8) are not permitted after calling this method.

write(*row*)
Write the specified *row* (a tuple of values) to the wrapped output. All provided rows must have the same number of elements. There is no need to convert elements of the tuple to `str`[10]; this will be handled implicitly.

class lars.csv.**CSV_DIALECT**
This is the default dialect used by the *CSVTarget* (page 7) class which has the following attributes:

Attribute	Value
delimiter	`','` (comma)
quotechar	`'"'` (double-quote)
quoting	*QUOTE_MINIMAL* (page 8)
lineterminator	`'\r\n'` (DOS line breaks)
doublequote	True
escapechar	None

This dialect is compatible with Microsoft Excel and the vast majority of of other products which accept CSV as an input format. However, please note that some UNIX based database products require UNIX style line endings (`'\n'`) in which case you may wish to override the *lineterminator* attribute (see *CSVTarget* (page 7) for more information).

class lars.csv.**TSV_DIALECT**
This is a dialect which produces tab-delimited files, another common data exchange format also supported by Microsoft Excel and numerous database products. This dialect has the following properties:

Attribute	Value
delimiter	`'\t'` (tab)
quotechar	`'"'` (double-quote)
quoting	*QUOTE_MINIMAL* (page 8)
lineterminator	`'\r\n'` (DOS line breaks)
doublequote	True
escapechar	None

3.3.2 Data

lars.csv.**QUOTE_NONE**
This value indicates that no values should ever be quoted, even if they contain the delimiter character. In this case, any delimiter characters appearing the data will be preceded by the dialect's *escapechar* which should be set to an appropriate value. If *escapechar* is not set (None) an exception will be raised if any character that require quoting are encountered.

[9] http://docs.python.org/2/library/codecs.html#standard-encodings
[10] https://docs.python.org/3.5/library/stdtypes.html#str

lars.csv.**QUOTE_MINIMAL**
: This is the default quoting mode. In this mode the writer will only quote those values that contain the *delimiter* or *quotechar* characters, or any of the characters in *lineterminator*.

lars.csv.**QUOTE_NONNUMERIC**
: This value tells the writer to quote all numeric (int and float) values.

lars.csv.**QUOTE_ALL**
: This value simply tells the writer to quote all values written.

3.3.3 Examples

A typical example of working with the class is shown below:

```
import io
from lars import apache, csv

with io.open('/var/log/apache2/access.log', 'rb') as infile:
    with io.open('apache.csv', 'wb') as outfile:
        with apache.ApacheSource(infile) as source:
            with csv.CSVTarget(outfile, lineterminator='\n') as target:
                for row in source:
                    target.write(row)
```

3.4 lars.sql - Direct Database Output

3.5 lars.geoip - GeoIP Database Access

3.6 lars.datatypes - Web Log Datatypes

3.7 lars.progress - Rendering Progress

This module provides a wrapper that outputs simple progress meters to the command line based on source file positions, or an arbitrary counter. The *ProgressMeter* (page 9) class is the major element that this module provides.

3.7.1 Classes

class lars.progress.**ProgressMeter**(*fileobj=None*, *value=0*, *total=None*, *max_wait=0.1*, *stream=sys.stderr*, *mode='w'*, *style=BarStyle*, *hide_on_finish=True*)
: This class provides a simple means of rendering a progress meter at the command line. It can be driven either with a file object (in which case the current position of the file is used) or with an arbitrary value (which your code must provide). In the case of a file-object, the file must be seekable (so that the class can determine the overall length of the file). If *fileobj* is not specified, then *total* must be specified.

  The class is intended to be used as a context manager. Upon entry it will render an initial progress meter, and will update it at reasonable intervals (dictated by the max_wait parameter) in response to calls to the *update()* (page 10) method. When you leave the context, the progress meter will be automatically erased if *hide_on_finish* is True (which it is by default).

  Within the context, the *hide()* (page 10) and *show()* (page 10) methods can be used to temporarily hide and show the progress meter (in order to display some status text, for example).

  Parameters

- **fileobj** (*file*) – A file-like object from which to determine progress
- **value** (*int*[11]) – An arbitrary value from which to determine progress
- **total** (*int*[12]) – In the case that *value* is set, this must be set to the maximum value that *value* will take
- **max_wait** (*float*[13]) – The minimum length of time that must elapse before a screen update is permitted
- **stream** (*file*) – The stream object that output should be written to, defaults to stderr
- **style** – A reference to a class which will be used to render the progress meter, defaults to *BarStyle* (page 10)
- **hide_on_finish** (*bool*[14]) – If True (the default), the progress meter will be erased when the context exits

hide()
Hide the progress bar from the console (or whatever the output stream is connected to).

show()
Show the progress bar on the console (or whatever the output stream is connected to).

update(*value=None*)
Update the progress bar to position *value* (which must be less than the *total* value passed to the constructor).

class lars.progress.**SpinnerStyle**(*meter*)
A *ProgressMeter* (page 9) style that renders a simple spinning line.

class lars.progress.**PercentageStyle**(*meter*)
A *ProgressMeter* (page 9) style that renders a simple percentage counter.

class lars.progress.**EllipsisStyle**(*meter*)
A *ProgressMeter* (page 9) style that renders an looping series of dots.

class lars.progress.**BarStyle**(*meter*)
A *ProgressMeter* (page 9) style that renders a full progress bar and percentage.

class lars.progress.**HashStyle**(*meter*)
A *ProgressMeter* (page 9) style for those that remember FTP's `hash` command!

3.7.2 Examples

The most basic usage of this class is as follows:

```
import io
from lars import iis, csv, progress

with io.open('logs\iis.txt', 'rb') as infile, \
        io.open('iis.csv', 'wb') as outfile, \
        progress.ProgressMeter(infile) as meter, \
        iis.IISSource(infile) as source, \
        csv.CSVTarget(outfile) as target:
    for row in source:
        target.write(row)
        meter.update()
```

[11] https://docs.python.org/3.5/library/functions.html#int
[12] https://docs.python.org/3.5/library/functions.html#int
[13] https://docs.python.org/3.5/library/functions.html#float
[14] https://docs.python.org/3.5/library/functions.html#bool

Note that you do not need to worry about the detrimental performance effects of calling *update()* (page 10) too often; the class ensures that repeated calls are ignored until `max_wait` seconds have elapsed since the last update.

Alternatively, if you wish to update according to, say, the number of files to process you could use something like the following example (which also demonstrates temporarily hiding the progress meter in order to show the current filename):

```
import os
import io
from lars import iis, csv, progress
from pathlib import Path

files = list(Path('.').iterdir())
with progress.ProgressMeter(total=len(files),
                            style=progress.BarStyle) as meter:
    for file_num, file_name in enumerate(files):
        meter.hide()
        print "Processing %s" % file_name
        meter.show()
        with file_name.open('rb') as infile, \
                file_name.with_suffix('.csv').open('wb') as outfile, \
                iis.IISSource(infile) as source, \
                csv.CSVTarget(outfile) as target:
            for row in source:
                target.write(row)
        meter.update(file_num)
```

3.8 lars.dns - DNS Resolution

This module provides a couple of trivial DNS resolution functions, enhanced with LRU caches. Most users should never need to access these functions directly. Instead, use the `address` and `hostname` properties of relevant objects.

3.8.1 Functions

`lars.dns.`**`from_address`**(*address*)

Reverse resolve an address to a hostname.

Given a string containing an IPv4 or IPv6 address, this functions returns a hostname associated with the address, using an LRU cache to speed up repeat queries. If the address does not reverse, the function returns the original address.

Parameters **`address`** (*str*[15]) – The address to resolve to a hostname

Returns The resolved hostname

`lars.dns.`**`to_address`**(*hostname*, *family=<AddressFamily.AF_UNSPEC: 0>*, *socktype=<SocketKind.SOCK_STREAM: 1>*)

Resolve a hostname to an address, preferring IPv4 addresses.

Given a string containing a DNS hostname, this function resolves the hostname to an address, using an LRU cache to speed up repeat queries. The function prefers IPv4 addresses, but will return IPv6 addresses if no IPv4 addresses are present in the result from getaddrinfo. If the hostname does not resolve, the function returns None rather than raise an exception (this is preferable as it provides a negative lookup cache).

Parameters **`hostname`** (*str*[16]) – The hostname to resolve to an address

Returns The resolved address

[15] https://docs.python.org/3.5/library/stdtypes.html#str
[16] https://docs.python.org/3.5/library/stdtypes.html#str

3.9 lars.cache - Cache Decorators

This module provides a backport of the Python 3.3 LRU caching decorator. Users should never need to access this module directly; its contents are solely present to ensure DNS lookups can be cached under a Python 2.7 environment.

Source adapted from Raymond Hettinger's recipe[17] licensed under the MIT license[18].

3.9.1 Functions

lars.cache.**lru_cache**(*maxsize=100*, *typed=False*)
Least-recently-used cache decorator.

If *maxsize* is set to None, the LRU features are disabled and the cache can grow without bound.

If *typed* is True, arguments of different types will be cached separately. For example, f(3.0) and f(3) will be treated as distinct calls with distinct results.

Arguments to the cached function must be hashable.

View the cache statistics named tuple (hits, misses, maxsize, currsize) with f.cache_info(). Clear the cache and statistics with f.cache_clear(). Access the underlying function with f.__wrapped__.

3.10 lars.exc - Base Exceptions

Defines base exception and warnings types for the package.

3.10.1 Exceptions

exception lars.exc.**LarsError**
Base class for all errors generated by the lars package. This exists purely for ease of filtering / catching all such errors.

exception lars.exc.**LarsWarning**
Base class for all warnings generated by the lars package. This exists purely for ease of filtering / catching all such warnings.

[17] http://code.activestate.com/recipes/578078-py26-and-py30-backport-of-python-33s-lru-cache/
[18] http://opensource.org/licenses/MIT

CHAPTER 4

Change log

4.1 Release 1.0 (2017-01-04)

- Permit NULL values in first row when creating SQL tables (but warn as this is not encouraged)
- Permit sources and targets to be used outside of context handlers (makes experimentation in the REPL a bit nicer)
- Don't warn when request is NULL in Apache log sources (in certain configurations this is common when stringent timeouts are set)
- Fixed incorrect generation of Oracle multi-row INSERT statements
- Fixed operation of SQL target when row doesn't cover complete set of target table rows

4.2 Release 0.3 (2014-09-07)

- Implemented Python 3 compatibility (specifically 3.2 or above) and added debian packaging for Python 3 and docs

4.3 Release 0.2 (2013-07-28)

- Added ISP and organisation lookups to geoip module
- Added multi-row insertion support to the sql module
- Added Oracle specific target in the sql module
- Fixed the setup.py script (missing MANIFEST.in meant utils.py was excluded which setup.py relies upon)
- Fixed test coverage for the progress module

4.4 Release 0.1 (2013-06-09)

- Initial release

CHAPTER 5

License

5.1 DateTime, Date, and Time documentation license

The documentation for the `DateTime`, `Date`, and `Time` classes in this module are derived from the documentation sources for the datetime, date, and time classes in Python 2.7.4 and thus are subject to the following copyright and license:

5.1.1 PSF LICENSE AGREEMENT FOR PYTHON 2.7.4

1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"), and the Individual or Organization ("Licensee") accessing and otherwise using Python 2.7.4 software in source or binary form and its associated documentation.

[19] dave@waveform.org.uk
[20] info@mimeconsulting.co.uk

2. Subject to the terms and conditions of this License Agreement, PSF hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python 2.7.4 alone or in any derivative version, provided, however, that PSF's License Agreement and PSF's notice of copyright, i.e., "Copyright © 2001-2013 Python Software Foundation; All Rights Reserved" are retained in Python 2.7.4 alone or in any derivative version prepared by Licensee.
3. In the event Licensee prepares a derivative work that is based on or incorporates Python 2.7.4 or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python 2.7.4.
4. PSF is making Python 2.7.4 available to Licensee on an "AS IS" basis. PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 2.7.4 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.
5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 2.7.4 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 2.7.4, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.
6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.
7. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between PSF and Licensee. This License Agreement does not grant permission to use PSF trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.
8. By copying, installing or otherwise using Python 2.7.4, Licensee agrees to be bound by the terms and conditions of this License Agreement.

5.2 _strptime license

The `strptime` and `timezone` modules are derived from the _strptime and datetime modules in Python 3.2 respectively, and therefore are subject to the following license:

Copyright (c) 1990-2013, Python Software Foundation

5.2.1 PSF LICENSE AGREEMENT FOR PYTHON 3.2.3

1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"), and the Individual or Organization ("Licensee") accessing and otherwise using Python 3.2.3 software in source or binary form and its associated documentation.
2. Subject to the terms and conditions of this License Agreement, PSF hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python 3.2.3 alone or in any derivative version, provided, however, that PSF's License Agreement and PSF's notice of copyright, i.e., "Copyright © 2001-2012 Python Software Foundation; All Rights Reserved" are retained in Python 3.2.3 alone or in any derivative version prepared by Licensee.
3. In the event Licensee prepares a derivative work that is based on or incorporates Python 3.2.3 or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python 3.2.3.
4. PSF is making Python 3.2.3 available to Licensee on an "AS IS" basis. PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 3.2.3 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 3.2.3 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 3.2.3, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.
6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.
7. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between PSF and Licensee. This License Agreement does not grant permission to use PSF trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.
8. By copying, installing or otherwise using Python 3.2.3, Licensee agrees to be bound by the terms and conditions of this License Agreement.

5.3 IPNetwork & IPAddress documentation license

The documentation for the `IPv4Address`, `IPv4Network`, `IPv6Address`, and `IPv6Network` classes in lars are derived from the ipaddress[21] documentation sources which are subject to the following copyright and are licensed to the PSF under the contributor agreement which makes them subject to the PSF 3.2.3 license from the section above:

Copyright (c) 2007 Google Inc.

[21] http://code.google.com/p/ipaddr-py/

Python Module Index

l

lars, 3
lars.cache, 12
lars.csv, 7
lars.dns, 11
lars.exc, 12
lars.progress, 9

Index

B

BarStyle (class in lars.progress), 10

C

close() (lars.csv.CSVTarget method), 8
CSV_DIALECT (class in lars.csv), 8
CSVTarget (class in lars.csv), 7

E

EllipsisStyle (class in lars.progress), 10

F

from_address() (in module lars.dns), 11

H

HashStyle (class in lars.progress), 10
hide() (lars.progress.ProgressMeter method), 10

L

lars (module), 3
lars.cache (module), 12
lars.csv (module), 7
lars.dns (module), 11
lars.exc (module), 12
lars.progress (module), 9
LarsError, 12
LarsWarning, 12
lru_cache() (in module lars.cache), 12

P

PercentageStyle (class in lars.progress), 10
ProgressMeter (class in lars.progress), 9

Q

QUOTE_ALL (in module lars.csv), 9
QUOTE_MINIMAL (in module lars.csv), 8
QUOTE_NONE (in module lars.csv), 8
QUOTE_NONNUMERIC (in module lars.csv), 9

S

show() (lars.progress.ProgressMeter method), 10
SpinnerStyle (class in lars.progress), 10

T

to_address() (in module lars.dns), 11
TSV_DIALECT (class in lars.csv), 8

U

update() (lars.progress.ProgressMeter method), 10

W

write() (lars.csv.CSVTarget method), 8