lam8 Documentation

Release 0.1.2.dev0

Chris Cummins

Feb 26, 2018

Contents

1 labm8 3

2 labm8.cache 5

3 labm8.crypto 7

4 labm8.db 9

5 labm8.fmt 15

6 labm8.fs 17

7 labm8.io 23

8 labm8.jsonutil 25

9 labm8.latex 27

10 labm8.lockfile 29

11 labm8.make 31

12 labm8.math 33

13 labm8.modules 37

14 labm8.prof 39

15 labm8.system 41

16 labm8.tar 45

17 labm8.text 47

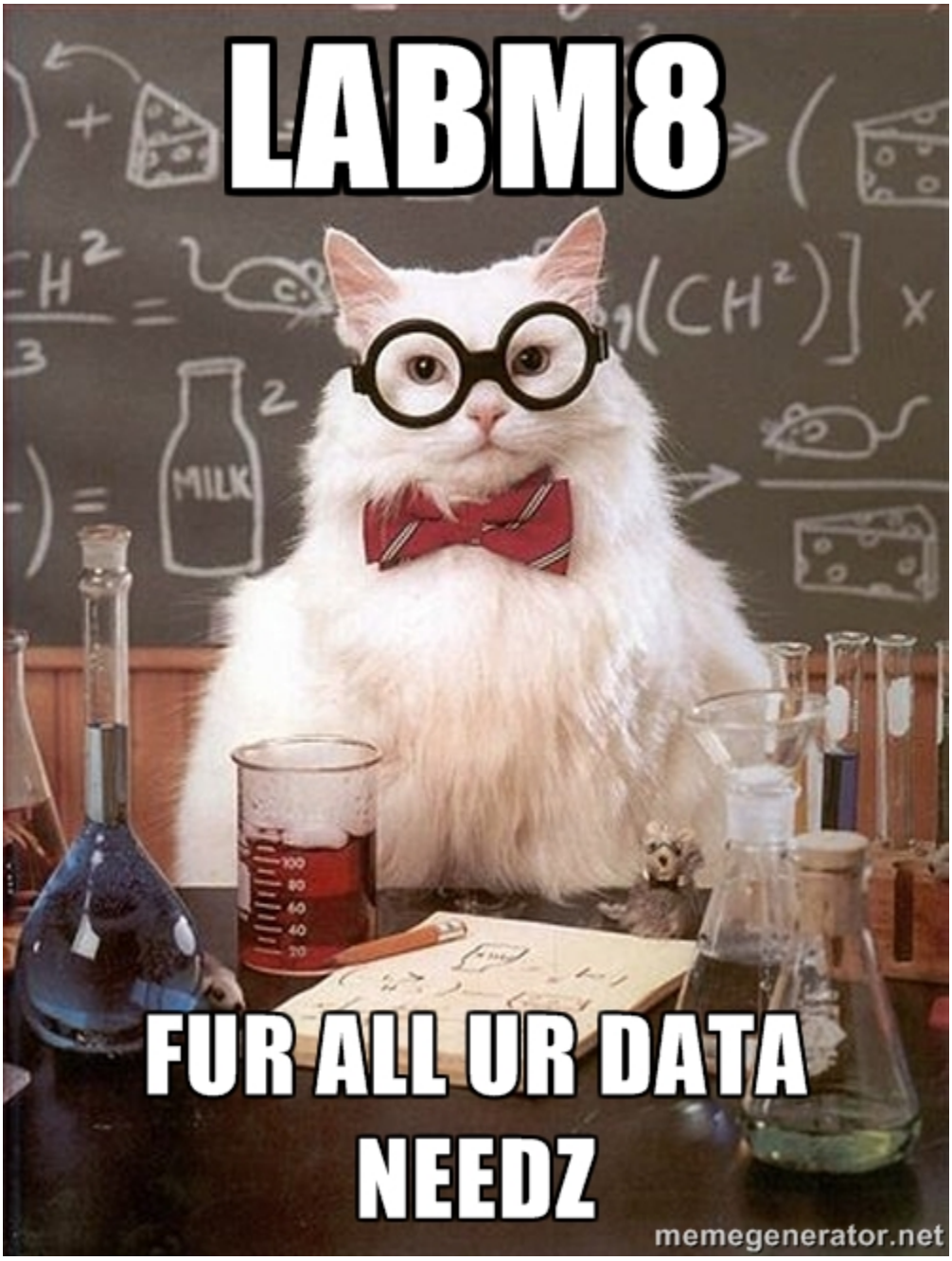
18 labm8.time 49

19 labm8.types 51

20 labm8.viz 53

21 Indices and tables 55

Python Module Index 57




```
$ pip install labm8
```

Supports Python versions 2.7 and >= 3.4.

Contents:

CHAPTER 1

labm8

Utils for manipulating quantitative experimental data.

labm8.**exit**(*status=0*)
Terminate the program with the given status code.

labm8.**is_python3**()
Returns whether the Python version is >= 3.0.

This is for compatability purposes, where you need to implement different code for Python 2 and 3.

Example

To import the StringIO class:

if is_python3(): from io import StringIO

else: from StringIO import StringIO

Returns True if Python >= 3, else False.

Return type bool

CHAPTER 2

labm8.cache

Transient and persistent caching mechanisms.

class labm8.cache.**Cache**
Cache for storing (key,value) relational data.

A cache is a dictionary with a limited subset of a the functionality.

clear()
Remove all items from cache.

get(*key*, *default=None*)
Retrieve an item from cache.

Parameters

- **key** – Item key.
- **default** (*optional*) – Default value if item not found.

items()
Returns a generator for iterating over (key, value) pairs.

class labm8.cache.**FSCache**(*root*, *escape_key=<function hash_key>*)
Persistent filesystem cache.

Each key uniquely identifies a file. Each value is a file path.

Adding a file to the cache moves it into the cahce directory.

Members: path (str): Root cache. escape_key (fn): Function to convert keys to file names.

clear()
Empty the filesystem cache.

This deletes the entire cache directory.

get(*key*, *default=None*)
Fetch from cache.

Parameters

- **key** – Key.
- **default** (*optional*) – Value returned if key not found.

Returns Path to cached file.

Return type str

keypath(*key*)
Get the filesystem path for a key.

Parameters **key** – Key.

Returns Absolute path.

Return type str

ls(***kwargs*)
List files in cache.

Parameters ****kwargs** – Keyword options to pass to fs.ls().

Returns List of files.

Return type iterable

class labm8.cache.**JsonCache**(*path*, *basecache=None*)
A persistent, JSON-backed cache.

Requires that (key, value) pairs are JSON serialisable.

write()
Write contents of cache to disk.

class labm8.cache.**TransientCache**(*basecache=None*)
An in-memory only cache.

clear()

get(*key*, *default=None*)

items()

labm8.cache.**escape_path**(*key*)
Convert a key to a filename by escaping invalid characters.

labm8.cache.**hash_key**(*key*)
Convert a key to a filename by hashing its value.

CHAPTER 3

labm8.crypto

Hashing and cryptography utils.

labm8.crypto.**md5**(*data*)
Return the md5 of "data".

Parameters **data** (*bytes*) – Data.

Returns Hex encoded.

Return type str

labm8.crypto.**md5_file**(*path*)
Return the md5 of file at "path".

Parameters **path** (*str*) – Path to file

Returns Hex encoded.

Return type str

labm8.crypto.**md5_list**(**elems*)
Return the md5 of all elements of a list.

Parameters ***elems** – List of stringifiable data.

Returns Hex encoded.

Return type str

labm8.crypto.**md5_str**(*string*, *encoding='utf-8'*)
Return the md5 of string "data".

Parameters **string** – String.

Returns Hex encoded.

Return type str

labm8.crypto.**sha1**(*data*)
Return the sha1 of "data".

Parameters **data** (*bytes*) – Data.

Returns Hex encoded.

Return type str

labm8.crypto.**sha1_file**(*path*)
Return the sha1 of file at "path".

Parameters **path** (*str*) – Path to file

Returns Hex encoded.

Return type str

labm8.crypto.**sha1_list**(**elems*)
Return the sha1 of all elements of a list.

Parameters ***elems** – List of stringifiable data.

Returns Hex encoded.

Return type str

labm8.crypto.**sha1_str**(*string*, *encoding='utf-8'*)
Return the sha1 of string "data".

Parameters **string** – String.

Returns Hex encoded.

Return type str

labm8.crypto.**sha256**(*data*)
Return the sha256 of "data".

Parameters **data** (*bytes*) – Data.

Returns Hex encoded.

Return type str

labm8.crypto.**sha256_file**(*path*)
Return the sha256 of file at "path".

Parameters **path** (*str*) – Path to file

Returns Hex encoded.

Return type str

labm8.crypto.**sha256_list**(**elems*)
Return the sha256 of all elements of a list.

Parameters ***elems** – List of stringifiable data.

Returns Hex encoded.

Return type str

labm8.crypto.**sha256_str**(*string*, *encoding='utf-8'*)
Return the sha256 of string "data".

Parameters **string** – String.

Returns Hex encoded.

Return type str

CHAPTER 4

labm8.db

Extended SQL database interface.

class labm8.db.**Database**(*path*, *tables={}*, *enable_traces=True*)

attach(*path*, *name*)
Attach a database.

Parameters

- **path** (*str*) – Path to the database to merge.
- **name** (*str*) – Name to attach database as.

close()
Close a database connection.

commit()
Commit the current transaction.

Make sure to call this method after you've modified the database's state!

copy_table(*src*, *dst*)
Create a carbon copy of the source table.

Parameters

- **src** (*str*) – The name of the table to copy.
- **dst** (*str*) – The name of the target duplicate table.

Raises sql.OperationalError – If source table does not exist.

create_table(*name*, *schema*)
Create a new table.

If the table already exists, nothing happens.

Example

```
>>> db.create_table("foo", (("id", "integer primary key"),
                            ("value", "text")))
```

Parameters

- **name** (*str*) – The name of the table to create.
- **schema** (*sequence of tuples*) – A list of (name, type) tuples representing each of the columns.

create_table_from(*name*, *src*)
Create a new table with same schema as the source.

If the named table already exists, nothing happens.

Parameters

- **name** (*str*) – The name of the table to create.
- **src** (*str*) – The name of the source table to duplicate.

Raises sql.OperationalError – If source table does not exist.

detach(*name*)
Detach a database.

Parameters **name** (*str*) – Name of database to detach.

drop_table(*name*)
Drop an existing table.

If the table does not exist, nothing happens.

Parameters **name** (*str*) – The name of the table to drop.

empty_table(*name*)
Delete all rows in a table.

If the table does not exist, nothing happens.

Parameters **name** (*str*) – The name of the table to empty.

execute(**args*)
Execute the given arguments.

executemany(**args*)
Execute the given arguments.

executescript(**args*)
Execute the given arguments.

export_csv(*table*, *output=None*, *columns='*'*, ***kwargs*)
Export a table to a CSV file.

If an output path is provided, write to file. Else, return a string.

Wrapper around pandas.sql.to_csv(). See: http://pandas.pydata.org/pandas-docs/stable/io.html#io-store-in-csv

Parameters

- **table** (*str*) – Name of the table to export.

- **output** (*str, optional*) – Path of the file to write.
- **columns** (*str, optional*) – A comma separated list of columns to export.
- ****kwargs** – Additional args passed to pandas.sql.to_csv()

Returns CSV string, or None if writing to file.

Return type str

Raises

- `IOError` – In case of error writing to file.
- *SchemaError* – If the named table is not found.

isempty(*tables=None*)

Return whether a table or the entire database is empty.

A database is empty is if it has no tables. A table is empty if it has no rows.

Parameters **tables** (*sequence of str, optional*) – If provided, check that the named tables are empty. If not provided, check that all tables are empty.

Returns True if tables are empty, else false.

Return type bool

Raises `sql.OperationalError` – If one or more of the tables do not exist.

num_rows(*table*)

Return the number of rows in the named table.

Example

```
>>> db.num_rows("foo")
3
```

Parameters **table** (*str*) – The name of the table to count the rows in.

Returns The number of rows in the named table.

Return type int

Raises `sql.OperationalError` – If the named table does not exist.

schema

Returns the schema of all tables.

For each table, return the name, and a list of tuples representing the columns. Each column tuple consists of a (name, type) pair. Note that additional metadata, such as whether a column may be null, or whether a column is a primary key, is not returned.

Example

```
>>> db.schema
[("bar", (("id", "integer"), ("name", "table"))]
```

Returns

Each tuple has the format (name, columns), where ”columns” is a list of tuples of the form (name, type).

Return type list of tuples

table_info(*table*)
Returns information about the named table.

See: https://www.sqlite.org/pragma.html#pragma_table_info

Example

```
>>> db.table_info("foo")
[{"name": "id", "type": "integer", "primary key": True,
  "notnull": False, "default_value": None}]
```

Parameters **name** (*str*) – The name of the table to lookup.

Returns

One dict per column. Each dict contains the keys: “name”, “type”, “primary key”, “notnull”, and “default_value”.

Return type list of dicts

Raises sql.OperationalError – If table does not exist.

tables
Returns a list of table names.

Example

```
>>> db.tables
["bar", "foo"]
```

Returns One string for each table name.

Return type list of str

exception labm8.db.**Error**
Module-level base error class.

exception labm8.db.**SchemaError**
Error thrown in case of conflicting schemas.

labm8.db.**placeholders**(**columns*)
Generate placeholders string for given arguments.

Examples

```
>>> placeholders(a, b, c)
"(?,?,?)"
```

```
>>> placeholders(*sequence)
"(?,?,?,?,?,?,?,?,?,?)"
```

```
>>> db.execute("INSERT INTO foo VALUES " +
               placeholders(a, b, c), (a, b, c))
```

```
>>> db.execute("INSERT INTO bar VALUES " +
               placeholders(*sequence), sequence)
```

Parameters ***columns** – Arguments to generate placeholders for.

Returns Bracketed, comma separated placeholders for given arguments.

Return type str

labm8.db.**where**(**columns*)

String together multiple where clauses.

Examples

```
>>> where("a", "b", "c")
"a=? AND b=? AND c=?"
```

```
>>> db.execute("SELECT * FROM foo WHERE {}"
               .format(where("a", "b", "c")),
               (a, b, c))
```

Parameters ***columns** – Arguments to generate '<name>=?' placeholders for.

Returns Equivalence checks for all columns.

Return type str

CHAPTER 5

labm8.fmt

String formatting utils.

exception labm8.fmt.**Error**
Module-level error.

labm8.fmt.**table**(*rows*, *columns=None*, *output=None*, *data_args={}*, ***kwargs*)
Return a formatted string of "list of list" table data.

See: http://pandas.pydata.org/pandas-docs/dev/generated/pandas.DataFrame.html

Examples

```
>>> fmt.print([("foo", 1), ("bar", 2)])
     0  1
0  foo  1
1  bar  2
```

```
>>> fmt.print([("foo", 1), ("bar", 2)], columns=("type", "value"))
  type  value
0  foo      1
1  bar      2
```

Parameters

- **rows** (*list of list*) – Data to format, one row per element, multiple columns per row.
- **columns** (*list of str, optional*) – Column names.
- **output** (*str, optional*) – Path to output file.
- **data_args** (*dict, optional*) – Any additional kwargs to pass to pandas.DataFrame constructor.
- ****kwargs** – Any additional arguments to pass to pandas.DataFrame.to_string().

Returns Formatted data as table.

Return type str

Raises *Error* – If number of columns (if provided) does not equal number of columns in rows; or if number of columns is not consistent across all rows.

CHAPTER 6

labm8.fs

High level filesystem interface.

exception labm8.fs.**Error**

exception labm8.fs.**File404**

labm8.fs.**abspath**(**components*)
Get an absolute file path.

Concatenate all components into an absolute path.

labm8.fs.**basename**(**components*)
Return the basename of a given file path.

labm8.fs.**cd**(*path*)
Change working directory.

Returns absolute path to new working directory.

labm8.fs.**cdpop**()
Return the last directory.

Returns absolute path to new working directory.

labm8.fs.**cp**(*src*, *dst*)
Copy a file or directory.

If source is a directory, this recursively copies the directory and its contents. If the destination is a directory, then this creates a copy of the source in the destination directory with the same basename.

If the destination already exists, this will attempt to overwrite it.

Parameters

- **src** (*string*) – path to the source file or directory.
- **dst** (*string*) – path to the destination file or directory.

Raises IOError – if source does not exist.

labm8.fs.**dirname**(**components*)
Return the directory name of a given file path.

labm8.fs.**du**(**components*, ***kwargs*)
Get the size of a file in bytes or as a human-readable string.

Parameters

- ***components** (*str[]*) – Path to file.
- ****kwargs** – If "human_readable" is True, return a formatted string, e.g. "976.6 KiB" (default True)

Returns If "human_readble" kwarg is True, return str, else int.

Return type int or str

labm8.fs.**exists**(**components*)
Return whether a file exists.

labm8.fs.**files_from_list**(**paths*)
Return a list of all file paths from a list of files or directories.

For each path in the input: if it is a file, return it; if it is a directory, return a list of files in the directory.

Parameters **paths** (*list of str*) – List of file and directory paths.

Returns Absolute file paths.

Return type list of str

Raises *File404* – If any of the paths do not exist.

labm8.fs.**is_subdir**(*child*, *parent*)
Determine if "child" is a subdirectory of "parent".

If child == parent, returns True.

labm8.fs.**isdir**(**components*)
Return whether a path exists, and is a directory.

labm8.fs.**isexe**(**components*)
Return whether a path is an executable file.

Parameters **path** (*str*) – Path of the file to check.

Examples

```
>>> fs.isexe("/bin/ls")
True
```

```
>>> fs.isexe("/home")
False
```

```
>>> fs.isexe("/not/a/real/path")
False
```

Returns True if file is executable, else false.

Return type bool

labm8.fs.**isfile**(**components*)
Return whether a path exists, and is a file.

labm8.fs.**ls**(*root='.'*, *abspaths=False*, *recursive=False*)
Return a list of files in directory.

Directory listings are sorted alphabetically. If the named directory is a file, return it's path.

Examples

```
>>> fs.ls("foo")
["a", "b", "c"]
```

```
>>> fs.ls("foo/a")
["foo/a"]
```

```
>>> fs.ls("foo", abspaths=True)
["/home/test/foo/a", "/home/test/foo/b", "/home/test/foo/c"]
```

```
>>> fs.ls("foo", recursive=True)
["a", "b", "b/d", "b/d/e", "c"]
```

Parameters

- **root** (*str*) – Path to directory. Can be relative or absolute.
- **abspaths** (*bool, optional*) – Return absolute paths if true.
- **recursive** (*bool, optional*) – Recursively list subdirectories if true.

Returns A list of paths.

Return type list of str

Raises OSError – If root directory does not exist.

labm8.fs.**lsdirs**(*root='.'*, ***kwargs*)
Return only subdirectories from a directory listing.

Parameters

- **root** (*str*) – Path to directory. Can be relative or absolute.
- ****kwargs** – Any additional arguments to be passed to ls().

Returns A list of directory paths.

Return type list of str

Raises OSError – If root directory does not exist.

labm8.fs.**lsfiles**(*root='.'*, ***kwargs*)
Return only files from a directory listing.

Parameters

- **root** (*str*) – Path to directory. Can be relative or absolute.
- ****kwargs** – Any additional arguments to be passed to ls().

Returns A list of file paths.

Return type list of str

Raises OSError – If root directory does not exist.

labm8.fs.**mkdir**(**components*, ***kwargs*)
Make directory "path", including any required parents. If directory already exists, do nothing.

labm8.fs.**mkopen**(*p*, **args*, ***kwargs*)
A wrapper for the open() builtin which makes parent directories if needed.

labm8.fs.**must_exist**(**components*)
Ensure path exists.

Parameters ***components** (*str[]*) – Path components.

Returns File path.

Return type str

Raises *File404* – If path does not exist.

labm8.fs.**mv**(*src*, *dst*)
Move a file or directory.

If the destination already exists, this will attempt to overwrite it.

Parameters

- **src** (*string*) – path to the source file or directory.
- **dst** (*string*) – path to the destination file or directory.

Raises

- *File404* – if source does not exist.
- IOError – in case of error.

labm8.fs.**path**(**components*)
Get a file path.

Concatenate all components into a path.

labm8.fs.**pwd**()
Return the path to the current working directory.

labm8.fs.**read**(**components*, ***kwargs*)
Read file and return a list of lines. If comment_char is set, ignore the contents of lines following the comment_char.

Raises IOError – if reading path fails

labm8.fs.**read_file**(*path*)
Read file to string.

Parameters **path** (*str*) – Source.

labm8.fs.**rm**(**components*, ***kwargs*)
Remove a file or directory.

If path is a directory, this recursively removes the directory and any contents. Non-existent paths are silently ignored.

Supports Unix style globbing by default (disable using glob=False). For details on globbing pattern expansion, see:

https://docs.python.org/2/library/glob.html

Parameters

- ***components** (*string[]*) – path to the file or directory to remove. May be absolute or relative. May contain unix glob
- ****kwargs** – if "glob" is True, perform Unix style pattern expansion of paths (default: True).

labm8.fs.**rmtrash**(**components*)

Move a file or directory to trash.

If file does not exist, nothing happens.

Examples

```
>>> fs.rmtrash("foo", "bar")
```

```
>>> fs.rmtrash("/home/labm8/file.txt")
```

Parameters ***components** (*string[]*) – path to the file or directory.

labm8.fs.**write_file**(*path*, *contents*)

Write string to file.

Parameters

- **path** (*str*) – Destination.
- **contents** (*str*) – Contents.

CHAPTER 7

labm8.io

Logging interface.

class labm8.io.**Colours**

Shell escape colour codes.

BLUE = '\x1b[94m'

GREEN = '\x1b[92m'

RED = '\x1b[91m'

RESET = '\x1b[0m'

YELLOW = '\x1b[93m'

labm8.io.**colourise**(*colour*, **args*)

labm8.io.**debug**(**args*, ***kwargs*)

labm8.io.**error**(**args*, ***kwargs*)

labm8.io.**fatal**(**args*, ***kwargs*)

labm8.io.**info**(**args*, ***kwargs*)

labm8.io.**pprint**(*data*, ***kwargs*)

labm8.io.**printf**(*colour*, **args*, ***kwargs*)

labm8.io.**prof**(**args*, ***kwargs*)

Print a profiling message.

Profiling messages are intended for printing runtime performance data. They are prefixed by the "PROF" title.

Parameters ****kwargs** (**args,*) – Message payload.

labm8.io.**warn**(**args*, ***kwargs*)

CHAPTER 8

labm8.jsonutil

JSON parser which supports comments.

labm8.jsonutil.**format_json**(*data*)
Pretty print JSON.

Parameters **data** (*dict*) – JSON blob.

Returns Formatted JSON

Return type str

labm8.jsonutil.**loads**(*text*, ***kwargs*)
Deserialize *text* (a *str* or *unicode* instance containing a JSON document with Python or JavaScript like comments) to a Python object.

Supported comment types: *// comment* and *# comment*.

Taken from commentjson, written by Vaidik Kapoor.

Copyright (c) 2014 Vaidik Kapoor, MIT license.

Parameters

- **text** (*str*) – serialized JSON string with or without comments.
- ****kwargs** (*optional*) – all the arguments that json.loads accepts.

Returns Decoded JSON.

Return type *dict* or *list*

labm8.jsonutil.**read_file**(**components*, ***kwargs*)
Load a JSON data blob.

Parameters

- **path** (*str*) – Path to file.
- **must_exist** (*bool, otional*) – If False, return empty dict if file does not exist.

Returns JSON data.

Return type array or dict

Raises

- `File404` – If path does not exist, and must_exist is True.
- `InvalidFile` – If JSON is malformed.

`labm8.jsonutil.`**`write_file`**(*path*, *data*, *format=True*)

Write JSON data to file.

Parameters

- **`path`** (*`str`*) – Destination.
- **`data`** (*`dict or list`*) – JSON serializable data.
- **`format`** (*`bool, optional`*) – Pretty-print JSON data.

CHAPTER 9

labm8.latex

Utilities for generating LaTeX.

exception labm8.latex.**Error**
Module-level error.

labm8.latex.**escape**(*text*)

labm8.latex.**table**(*rows*, *columns=None*, *output=None*, *data_args={}*, ***kwargs*)
Return a LaTeX formatted string of "list of list" table data.

See: http://pandas.pydata.org/pandas-docs/dev/generated/pandas.DataFrame.html

Requires the "booktabs" package to be included in LaTeX preamble:

usepackage{booktabs}

Examples

```
>>> fmt.print([("foo", 1), ("bar", 2)])
     0  1
0  foo  1
1  bar  2
```

```
>>> fmt.print([("foo", 1), ("bar", 2)], columns=("type", "value"))
  type  value
0  foo      1
1  bar      2
```

Parameters

- **rows** (*list of list*) – Data to format, one row per element, multiple columns per row.
- **columns** (*list of str, optional*) – Column names.
- **output** (*str, optional*) – Path to output file.

- **data_args** (*dict, optional*) – Any additional kwargs to pass to pandas.DataFrame constructor.
- ****kwargs** – Any additional arguments to pass to pandas.DataFrame.to_latex().

Returns Formatted data as LaTeX table.

Return type str

Raises *Error* – If number of columns (if provided) does not equal number of columns in rows; or if number of columns is not consistent across all rows.

labm8.latex.**wrap_bold**(*text*)

labm8.latex.**write_table_body**(*data*, *output=None*, *headers=None*, *header_fmt=<function wrap_bold>*, *hline_after_header=True*, *hline_before=False*, *hline_after=False*)

CHAPTER 10

labm8.lockfile

Lock file mechanism.

exception labm8.lockfile.**Error**

class labm8.lockfile.**LockFile**(*path*)
A lock file.

path
str – Path of lock file.

acquire(*replace_stale=False*, *force=False*)
Acquire the lock.

A lock can be claimed if any of these conditions are true:

1. The lock is unheld by anyone.
2. The lock is held but the 'force' argument is set.
3. The lock is held by the current process.

Parameters

- **replace_stale** (*bool, optional*) – stale processes. A stale process is one which currently owns the parent lock, but no process with that PID is alive.
- **force** (*bool, optional*) – If true, ignore any existing lock. If false, fail if lock already claimed.

Returns self.

Return type *LockFile*

Raises *UnableToAcquireLockError* – If the lock is already claimed (not raised if force option is used).

date
The date that the lock was acquired. Value is None if lock is unclaimed.

islocked
Whether the directory is locked.

owned_by_self
Whether the directory is locked by the current process.

pid
The process ID of the lock. Value is None if lock is not claimed.

static read(*path*)
Read the contents of a LockFile.

Parameters **path** (*str*) – Path to lockfile.

Returns

The integer PID of the lock owner, and the date the lock was required. If the lock is not claimed, both values are None.

Return type Tuple(int, datetime)

release(*force=False*)
Release lock.

To release a lock, we must already own the lock.

Parameters **force** (*bool, optional*) – If true, ignore any existing lock owner.

Raises *UnableToReleaseLockError* – If the lock is claimed by another process (not raised if force option is used).

static write(*path*, *pid*, *timestamp*)
Write the contents of a LockFile.

Parameters

- **path** (*str*) – Path to lockfile.
- **pid** (*int*) – The integer process ID.
- **timestamp** (*datetime*) – The time the lock was aquired.

exception labm8.lockfile.**UnableToAcquireLockError**(*lock*)
thrown if cannot acquire lock

exception labm8.lockfile.**UnableToReleaseLockError**(*lock*)
thrown if cannot release lock

CHAPTER 11

labm8.make

Wrapper for invoking Makefiles.

exception labm8.make.**Error**
Module-level error class.

exception labm8.make.**MakeError**
Thrown if make command fails.

exception labm8.make.**NoMakefileError**
Thrown if a directory does not contain a Makefile.

exception labm8.make.**NoTargetError**
Thrown if there is no rule for the requested target.

labm8.make.**clean**(***kwargs*)
Run make clean.

Equivalent to invoking make() with target="clean".

Parameters ****kwargs** (*optional*) – Any additional arguments to be passed to make().

Returns

The first element is the return code of the make command. The second and third elements are the stdout and stderr of the process.

Return type (int, str, str)

Raises

- *NoMakefileError* – In case a Makefile is not found in the target directory.
- *NoTargetError* – In case the Makefile does not support the requested target.
- *MakeError* – In case the target rule fails.

labm8.make.**make**(*target='all'*, *dir='.'*, ***kwargs*)
Run make.

Parameters

- **target** (*str, optional*) – Name of the target to build. Defaults to "all".
- **dir** (*str, optional*) – Path to directory containing Makefile.
- ****kwargs** (*optional*) – Any additional arguments to be passed to system.run().

Returns

The first element is the return code of the make command. The second and third elements are the stdout and stderr of the process.

Return type (int, str, str)

Raises

- *NoMakefileError* – In case a Makefile is not found in the target directory.
- *NoTargetError* – In case the Makefile does not support the requested target.
- *MakeError* – In case the target rule fails.

CHAPTER 12

labm8.math

Math utils. Import as "labmath" to prevent conflicts with system math package.

labm8.math.**ceil**(*number*)

Return the ceiling of a number as an int.

This is the smallest integral value >= number.

Example

```
>>> labmath.ceil(1.5)
2
```

Parameters **number** (*float*) – A numeric value.

Returns Smallest integer >= number.

Return type int

Raises TypeError – If argument is not a numeric value.

labm8.math.**confinterval**(*array*, *conf=0.95*, *normal_threshold=30*, *error_only=False*, *array_mean=None*)

Return the confidence interval of a list for a given confidence.

Parameters

- **array** (*list*) – Sequence of numbers.
- **conf** (*float*) – Confidence interval, in range 0 <= ci <= 1
- **normal_threshold** (*int*) – The number of elements in the array is < normal_threshold, use a T-distribution. If the number of elements in the array is >= normal_threshold, use a normal distribution.
- **error_only** (*bool, optional*) – If true, return only the size of symmetrical confidence interval, equal to ci_upper - mean.

- **array_mean** (*float, optional*) – Optimisation trick for if you already know the arithmetic mean of the array to prevent this function from re-calculating.

Returns

Lower and upper bounds on confidence interval, respectively.

Return type (float, float)

labm8.math.**filter_iqr**(*array*, *lower*, *upper*)

Return elements which falls within specified interquartile range.

Parameters

- **array** (*list*) – Sequence of numbers.
- **lower** (*float*) – Lower bound for IQR, in range 0 <= lower <= 1.
- **upper** (*float*) – Upper bound for IQR, in range 0 <= upper <= 1.

Returns

Copy of original list, with elements outside of IQR removed.

Return type list

labm8.math.**floor**(*number*)

Return the floor of a number as an int.

This is the largest integral value <= number.

Example

```
>>> labmath.floor(1.5)
1
```

Parameters **number** (*float*) – A numeric value.

Returns Largest integer <= number.

Return type int

Raises TypeError – If argument is not a numeric value.

labm8.math.**geomean**(*array*)

Return the mean value of a list of divisible numbers.

labm8.math.**iqr**(*array*, *lower*, *upper*)

Return interquartile range for given array.

Parameters

- **lower** (*float*) – Lower bound for IQR, in range 0 <= lower <= 1.
- **upper** (*float*) – Upper bound for IQR, in range 0 <= upper <= 1.

Returns Lower and upper IQR values.

Return type (float, float)

labm8.math.**mean**(*array*)

Return the mean value of a list of divisible numbers.

labm8.math.**median**(*array*)
Return the median value of a list of numbers.

labm8.math.**range**(*array*)
Return the range between min and max values.

labm8.math.**sqrt**(*number*)
Return the square root of a number.

labm8.math.**stdev**(*array*)
Return the standard deviation of a list of divisible numbers.

labm8.math.**variance**(*array*)
Return the variance of a list of divisible numbers.

CHAPTER 13

labm8.modules

Utils for handling python modules.

labm8.modules.**import_foreign**(*name*, *custom_name=None*)

Import a module with a custom name.

NOTE this is only needed for Python2. For Python3, import the module using the "as" keyword to declare the custom name.

For implementation details, see: http://stackoverflow.com/a/6032023

Example

To import the standard module "math" as "std_math":

if labm8.is_python3(): import math as std_math

else: std_math = modules.import_foreign("math", "std_math")

Parameters

- **name** (*str*) – The name of the module to import.
- **custom_name** (*str, optional*) – The custom name to assign the module to.

Raises ImportError – If the module is not found.

CHAPTER 14

labm8.prof

Profiling API for timing critical paths in code.

labm8.prof.**disable**()

labm8.prof.**enable**()

labm8.prof.**is_enabled**()

labm8.prof.**isrunning**(*name*)
Check if a timer is running.

Parameters **name** (*str, optional*) – The name of the timer to check.

Returns True if timer is running, else False.

Return type bool

labm8.prof.**profile**(*fun*, **args*, ***kwargs*)
Profile a function.

labm8.prof.**start**(*name*)
Start a new profiling timer.

Parameters

- **name** (*str, optional*) – The name of the timer to create. If no name is given, the resulting timer is anonymous. There can only be one anonymous timer.
- **unique** (*bool, optional*) – If true, then ensure that timer name is unique. This is to prevent accidentally resetting an existing timer.

Returns Whether or not profiling is enabled.

Return type bool

labm8.prof.**stop**(*name*, *file=<open file '<stderr>', mode 'w'>*)
Stop a profiling timer.

Parameters **name** (*str*) – The name of the timer to stop. If no name is given, stop the global anonymous timer.

Returns Whether or not profiling is enabled.

Return type bool

Raises `KeyError` – If the named timer does not exist.

`labm8.prof.`**`timers`**`()`
Iterate over all timers.

Returns An iterator over all time names.

Return type Iterable[str]

CHAPTER 15

labm8.system

Utilies for grokking the underlying system.

Variables:

- *HOSTNAME* (str) System hostname.
- *USERNAME* (str) Username.
- *UID* (int) User ID.
- *PID* (int) Process ID.

exception labm8.system.**CommandNotFoundError**
Error thrown a system command is not found.

exception labm8.system.**Error**

exception labm8.system.**ScpError**(*stdout*, *stderr*)
Error thrown if scp file transfer fails.

class labm8.system.**Subprocess**(*cmd*, *shell=False*, *stdout=-1*, *stderr=-1*, *decode_out=True*)
Subprocess abstraction.

Wrapper around subprocess.Popen() which provides the ability to force a timeout after a number of seconds have elapsed.

run(*timeout=-1*)
Run the subprocess.

Parameters **timeout** (*optional*) – the given number of seconds.

Raises *SubprocessError If subprocess has not completed after "timeout"* – seconds.

exception labm8.system.**SubprocessError**
Error thrown if a subprocess fails.

labm8.system.**echo**(**args*, ***kwargs*)
Write a message to a file.

Parameters **A list of arguments which make up the message. The last argument** (*args*) – is the path to the file to write to.

labm8.system.**is_linux**()

labm8.system.**is_mac**()

labm8.system.**is_windows**()

labm8.system.**isprocess**(*pid*, *error=False*)
Check that a process is running.

Parameters **pid** (*int*) – Process ID to check.

Returns True if the process is running, else false.

labm8.system.**run**(*command*, *num_retries=1*, *timeout=-1*, ***kwargs*)
Run a command with optional timeout and retries.

Provides a convenience method for executing a subprocess with additional error handling.

Parameters

- **command** (*list of str*) – The command to execute.
- **num_retries** (*int, optional*) – If the subprocess fails, the number of attempts to execute it before failing.
- **timeout** (*float, optional*) – If positive, the number of seconds to wait for subprocess completion before failing.
- ****kwargs** – Additional args to pass to Subprocess.__init__()

Returns Where the variables represent (exit status, stdout, stderr).

Return type Tuple of (int, str, str)

Raises *SubprocessError* – If the command fails after the given number of retries.

labm8.system.**scp**(*host*, *src*, *dst*, *user=None*, *path=None*)
Copy a file or directory from a remote location.

A thin wrapper around the scp (1) system command.

If the destination already exists, this will attempt to overwrite it.

Parameters

- **host** (*str*) – name of the host
- **src** (*str*) – path to the source file or directory.
- **dst** (*str*) – path to the destination file or directory.
- **user** (*str, optional*) – Alternative username for remote access. If not provided, the default scp behaviour is used.
- **path** (*str, optional*) – Directory containing scp command. If not provided, attempt to locate scp using which().

Raises

- *CommandNotFoundError* – If scp binary not found.
- IOError – if transfer fails.

labm8.system.**sed**(*match*, *replacement*, *path*, *modifiers=''*)
Perform sed text substitution.

labm8.system.**which**(*program*, *path=None*)

Returns the full path of shell commands.

Replicates the functionality of system which (1) command. Looks for the named program in the directories indicated in the $PATH environment variable, and returns the full path if found.

Examples

```
>>> system.which("ls")
"/bin/ls"
```

```
>>> system.which("/bin/ls")
"/bin/ls"
```

```
>>> system.which("not-a-real-command")
None
```

```
>>> system.which("ls", path=("/usr/bin", "/bin"))
"/bin/ls"
```

Parameters

- **program** (*str*) – The name of the program to look for. Can be an absolute path.
- **path** (*sequence of str, optional*) – A list of directories to look for the pgoram in. Default value is system $PATH.

Returns Full path to program if found, else None.

Return type str

CHAPTER 16

labm8.tar

CHAPTER 17

labm8.text

Text utilities.

exception labm8.text.**Error**
Module-level error.

exception labm8.text.**TruncateError**
Thrown in case of truncation error.

labm8.text.**diff**(*s1*, *s2*)
Return a normalised Levenshtein distance between two strings.

Distance is normalised by dividing the Levenshtein distance of the two strings by the max(len(s1), len(s2)).

Examples

```
>>> text.diff("foo", "foo")
0
```

```
>>> text.diff("foo", "fooo")
1
```

```
>>> text.diff("foo", "")
1
```

```
>>> text.diff("1234", "1 34")
1
```

Parameters

- **s1** (*str*) – Argument A.
- **s2** (*str*) – Argument B.

Returns Normalised distance between the two strings.

Return type float

labm8.text.**get_substring_idxs**(*substr*, *string*)
Return a list of indexes of substr. If substr not found, list is empty.

Parameters

- **substr** (*str*) – Substring to match.
- **string** (*str*) – String to match in.

Returns Start indices of substr.

Return type list of int

labm8.text.**levenshtein**(*s1*, *s2*)
Return the Levenshtein distance between two strings.

Implementation of Levenshtein distance, one of a family of edit distance metrics.

Based on: https://en.wikibooks.org/wiki/Algorithm_Implementation/Strings/Levenshtein_distance#Python

Examples

```
>>> text.levensthein("foo", "foo")
0
```

```
>>> text.levensthein("foo", "fooo")
1
```

```
>>> text.levensthein("foo", "")
3
```

```
>>> text.levensthein("1234", "1 34")
1
```

Parameters

- **s1** (*str*) – Argument A.
- **s2** (*str*) – Argument B.

Returns Levenshtein distance between the two strings.

Return type int

labm8.text.**truncate**(*string*, *maxchar*)
Truncate a string to a maximum number of characters.

If the string is longer than maxchar, then remove excess characters and append an ellipses.

Parameters

- **string** (*str*) – String to truncate.
- **maxchar** (*int*) – Maximum length of string in characters. Must be >= 4.

Returns Of length <= maxchar.

Return type str

Raises *TruncateError* – In case of an error.

CHAPTER 18

labm8.time

Time utilities.

labm8.time.**now**()
: Get the current datetime.

labm8.time.**nowstr**(*format='%Y-%m-%d %H:%M:%S'*)
: Convenience wrapper to get the current time as a string.

  Equivalent to invoking strfmt(now()).

labm8.time.**strfmt**(*datetime*, *format='%Y-%m-%d %H:%M:%S'*)
: Format date to string.

CHAPTER 19

labm8.types

Python type utilities.

labm8.types.**dict_values**(*src*)
Recursively get values in dict.

Unlike the builtin dict.values() function, this method will descend into nested dicts, returning all nested values.

Parameters **src** (*dict*) – Source dict.

Returns List of values.

Return type list

labm8.types.**flatten**(*lists*)
Flatten a list of lists.

labm8.types.**get_class_that_defined_method**(*meth*)
Return the class that defines a method.

Parameters **meth** (*str*) – Class method.

Returns Class object, or None if not a class method.

Return type class

labm8.types.**is_dict**(*obj*)
Check if an object is a dict.

labm8.types.**is_seq**(*obj*)
Check if an object is a sequence.

labm8.types.**is_str**(*s*)
Return whether variable is string type.

On python 3, unicode encoding is *not* string type. On python 2, it is.

Parameters **s** – Value.

Returns True if is string, else false.

Return type bool

labm8.types.**update**(*dst*, *src*)

Recursively update values in dst from src.

Unlike the builtin dict.update() function, this method will decend into nested dicts, updating all nested values.

Parameters

- **dst** (*dict*) – Destination dict.
- **src** (*dict*) – Source dict.

Returns dst updated with entries from src.

Return type dict

CHAPTER 20

labm8.viz

Graphing helper.

exception labm8.viz.**Error**
Visualisation module base error class.

labm8.viz.**finalise**(*output=None*, *figsize=None*, *tight=True*, ***kwargs*)
Finalise a plot.

Display or show the plot, then close it.

Parameters

- **output** (*str, optional*) – Path to save figure to. If not given, show plot.
- **figsize** (*(float, float), optional*) – Figure size in inches.
- ****kwargs** – Any additional arguments to pass to plt.savefig(). Only required if output is not None.

CHAPTER 21

Indices and tables

- genindex
- modindex
- search

Python Module Index

l

labm8, 3
labm8.cache, 5
labm8.crypto, 7
labm8.db, 9
labm8.fmt, 15
labm8.fs, 17
labm8.io, 23
labm8.jsonutil, 25
labm8.latex, 27
labm8.lockfile, 29
labm8.make, 31
labm8.math, 33
labm8.modules, 37
labm8.prof, 39
labm8.system, 41
labm8.text, 47
labm8.time, 49
labm8.types, 51
labm8.viz, 53

Index

A

abspath() (in module labm8.fs), 17
acquire() (labm8.lockfile.LockFile method), 29
attach() (labm8.db.Database method), 9

B

basename() (in module labm8.fs), 17
BLUE (labm8.io.Colours attribute), 23

C

Cache (class in labm8.cache), 5
cd() (in module labm8.fs), 17
cdpop() (in module labm8.fs), 17
ceil() (in module labm8.math), 33
clean() (in module labm8.make), 31
clear() (labm8.cache.Cache method), 5
clear() (labm8.cache.FSCache method), 5
clear() (labm8.cache.TransientCache method), 6
close() (labm8.db.Database method), 9
colourise() (in module labm8.io), 23
Colours (class in labm8.io), 23
CommandNotFoundError, 41
commit() (labm8.db.Database method), 9
confinterval() (in module labm8.math), 33
copy_table() (labm8.db.Database method), 9
cp() (in module labm8.fs), 17
create_table() (labm8.db.Database method), 9
create_table_from() (labm8.db.Database method), 10

D

Database (class in labm8.db), 9
date (labm8.lockfile.LockFile attribute), 29
debug() (in module labm8.io), 23
detach() (labm8.db.Database method), 10
dict_values() (in module labm8.types), 51
diff() (in module labm8.text), 47
dirname() (in module labm8.fs), 17
disable() (in module labm8.prof), 39
drop_table() (labm8.db.Database method), 10
du() (in module labm8.fs), 18

E

echo() (in module labm8.system), 41
empty_table() (labm8.db.Database method), 10
enable() (in module labm8.prof), 39
Error, 12, 15, 17, 27, 29, 31, 41, 47, 53
error() (in module labm8.io), 23
escape() (in module labm8.latex), 27
escape_path() (in module labm8.cache), 6
execute() (labm8.db.Database method), 10
executemany() (labm8.db.Database method), 10
executescript() (labm8.db.Database method), 10
exists() (in module labm8.fs), 18
exit() (in module labm8), 3
export_csv() (labm8.db.Database method), 10

F

fatal() (in module labm8.io), 23
File404, 17
files_from_list() (in module labm8.fs), 18
filter_iqr() (in module labm8.math), 34
finalise() (in module labm8.viz), 53
flatten() (in module labm8.types), 51
floor() (in module labm8.math), 34
format_json() (in module labm8.jsonutil), 25
FSCache (class in labm8.cache), 5

G

geomean() (in module labm8.math), 34
get() (labm8.cache.Cache method), 5
get() (labm8.cache.FSCache method), 5
get() (labm8.cache.TransientCache method), 6
get_class_that_defined_method() (in module labm8.types), 51
get_substring_idxs() (in module labm8.text), 48
GREEN (labm8.io.Colours attribute), 23

H

hash_key() (in module labm8.cache), 6

I

import_foreign() (in module labm8.modules), 37
info() (in module labm8.io), 23
iqr() (in module labm8.math), 34
is_dict() (in module labm8.types), 51
is_enabled() (in module labm8.prof), 39
is_linux() (in module labm8.system), 42
is_mac() (in module labm8.system), 42
is_python3() (in module labm8), 3
is_seq() (in module labm8.types), 51
is_str() (in module labm8.types), 51
is_subdir() (in module labm8.fs), 18
is_windows() (in module labm8.system), 42
isdir() (in module labm8.fs), 18
isempty() (labm8.db.Database method), 11
isexe() (in module labm8.fs), 18
isfile() (in module labm8.fs), 18
islocked (labm8.lockfile.LockFile attribute), 29
isprocess() (in module labm8.system), 42
isrunning() (in module labm8.prof), 39
items() (labm8.cache.Cache method), 5
items() (labm8.cache.TransientCache method), 6

J

JsonCache (class in labm8.cache), 6

K

keypath() (labm8.cache.FSCache method), 6

L

labm8 (module), 3
labm8.cache (module), 5
labm8.crypto (module), 7
labm8.db (module), 9
labm8.fmt (module), 15
labm8.fs (module), 17
labm8.io (module), 23
labm8.jsonutil (module), 25
labm8.latex (module), 27
labm8.lockfile (module), 29
labm8.make (module), 31
labm8.math (module), 33
labm8.modules (module), 37
labm8.prof (module), 39
labm8.system (module), 41
labm8.text (module), 47
labm8.time (module), 49
labm8.types (module), 51
labm8.viz (module), 53
levenshtein() (in module labm8.text), 48
loads() (in module labm8.jsonutil), 25
LockFile (class in labm8.lockfile), 29
ls() (in module labm8.fs), 19
ls() (labm8.cache.FSCache method), 6
lsdirs() (in module labm8.fs), 19
lsfiles() (in module labm8.fs), 19

M

make() (in module labm8.make), 31
MakeError, 31
md5() (in module labm8.crypto), 7
md5_file() (in module labm8.crypto), 7
md5_list() (in module labm8.crypto), 7
md5_str() (in module labm8.crypto), 7
mean() (in module labm8.math), 34
median() (in module labm8.math), 34
mkdir() (in module labm8.fs), 20
mkopen() (in module labm8.fs), 20
must_exist() (in module labm8.fs), 20
mv() (in module labm8.fs), 20

N

NoMakefileError, 31
NoTargetError, 31
now() (in module labm8.time), 49
nowstr() (in module labm8.time), 49
num_rows() (labm8.db.Database method), 11

O

owned_by_self (labm8.lockfile.LockFile attribute), 30

P

path (labm8.lockfile.LockFile attribute), 29
path() (in module labm8.fs), 20
pid (labm8.lockfile.LockFile attribute), 30
placeholders() (in module labm8.db), 12
pprint() (in module labm8.io), 23
printf() (in module labm8.io), 23
prof() (in module labm8.io), 23
profile() (in module labm8.prof), 39
pwd() (in module labm8.fs), 20

R

range() (in module labm8.math), 35
read() (in module labm8.fs), 20
read() (labm8.lockfile.LockFile static method), 30
read_file() (in module labm8.fs), 20
read_file() (in module labm8.jsonutil), 25
RED (labm8.io.Colours attribute), 23
release() (labm8.lockfile.LockFile method), 30
RESET (labm8.io.Colours attribute), 23
rm() (in module labm8.fs), 20
rmtrash() (in module labm8.fs), 21
run() (in module labm8.system), 42
run() (labm8.system.Subprocess method), 41

S

schema (labm8.db.Database attribute), 11
SchemaError, 12
scp() (in module labm8.system), 42
ScpError, 41
sed() (in module labm8.system), 42
sha1() (in module labm8.crypto), 7
sha1_file() (in module labm8.crypto), 8
sha1_list() (in module labm8.crypto), 8
sha1_str() (in module labm8.crypto), 8
sha256() (in module labm8.crypto), 8
sha256_file() (in module labm8.crypto), 8
sha256_list() (in module labm8.crypto), 8
sha256_str() (in module labm8.crypto), 8
sqrt() (in module labm8.math), 35
start() (in module labm8.prof), 39
stdev() (in module labm8.math), 35
stop() (in module labm8.prof), 39
strfmt() (in module labm8.time), 49
Subprocess (class in labm8.system), 41
SubprocessError, 41

T

table() (in module labm8.fmt), 15
table() (in module labm8.latex), 27
table_info() (labm8.db.Database method), 12
tables (labm8.db.Database attribute), 12
timers() (in module labm8.prof), 40
TransientCache (class in labm8.cache), 6
truncate() (in module labm8.text), 48
TruncateError, 47

U

UnableToAcquireLockError, 30
UnableToReleaseLockError, 30
update() (in module labm8.types), 52

V

variance() (in module labm8.math), 35

W

warn() (in module labm8.io), 23
where() (in module labm8.db), 13
which() (in module labm8.system), 42
wrap_bold() (in module labm8.latex), 28
write() (labm8.cache.JsonCache method), 6
write() (labm8.lockfile.LockFile static method), 30
write_file() (in module labm8.fs), 21
write_file() (in module labm8.jsonutil), 26
write_table_body() (in module labm8.latex), 28

Y

YELLOW (labm8.io.Colours attribute), 23