Lab Utils Documentation

Release 0.5.11

Carlos Vigo

May 12, 2021

CONTENTS

lab_utils is a collection of useful modules for Python development in the context of scientific laboratory work. It was created to standardize common tasks and features used by many different apps and users. The package aims to provide simple, efficient and robust protocols in the following areas:

- **PostgreSQL Database Interface**. The *database* module provides a simple interface to manage connections to a PostgreSQL database. It uses the package psycopg2 (a Python wrapper of the C library libpq) to provide simple database functionality.
- **Daemon-like TCP Server**. The *socket_comm* module provides a daemon-like TCP server base class. The *lab_utils.socket_comm.Server* is supposed to start and run indefinitely in the background, while listening for incoming communications over a TCP port. When a message is received, a string parser calls the appropriate method. Users should develop their own child class inheriting from *lab_utils.socket_comm.Server* and implementing the necessary methods for their particular needs.

  The module also provides a simple *lab_utils.socket_comm.Client* to send messages to a running *lab_utils.socket_comm.Server* and receive the reply.

If this is your first time using *lab_utils*, have a look at our *Readme* for a more detailed summary and installation instructions. If you're already familiar with this package, or you want to dive straight in, you can jump to the *API reference*. There are also some *examples* which demonstrate specific applications of the modules.

CHAPTER

ONE

README

1.1 Lab Utils, a collection of useful Python modules

This package contains several useful *modules* to help build Python applications. All modules provide support for a configuration file using configparser and standard logging.

Available modules:

- *database*: simple interface to manage connections to a PostgreSQL database
- *socket_comm*: TCP server/client communication for daemon-like apps.

1.2 Dependencies

The package **lab-utils** has the following pre-requisites:

- libpq, a C library that implements connections to the PostgreSQL backend server. The python package psycopg2 needed by the module *database* is built around this library, and it is strongly recommended to have it installed. If for some reason you can't install it or don't have access to it, a precompiled binary package is also available. Please notice that using precompiled binaries can lead to other problems.
- Python 3.6 and pip 10.0 are the minimum required versions to build **lab-utils** and its dependencies. It is recommended to install and run **lab-utils** (and any other package, for that matter) under a virtual environment.

1.3 Getting Started

These instructions will install the package **lab-utils** and let you import its modules in your own apps. It is assumed that you have successfully installed the *prerequisites* and are running inside a virtual environment.

1. Install the package and its dependencies

   ```
   python -m pip install lab-utils
   ```

   If you don't have PostgreSQL and libpq installed, you can use

   ```
   python -m pip install lab-utils --only-binary psycopg2
   ```

2. Run some examples to test that everything works

```
cd examples
python database/create_column.py
```

1.4 Import into your own app

To use a <module> from the **lab_utils** collection in your own Python app, simply add

```
from lab_utils import <module>
```

1.5 Modules

All the modules provided by the package provide support for:

- Usage of a configuration file via the <module>.config(*filename*) method
- Standard Python logging.

1.5.1 database

This module is a simple interface to manage connections to a PostgreSQL database based on the psycopg2 library. The main features are:

- Database connection and closing
- Create a new TimescaleDB table
- Check if column and/or table exist in a given database
- Create a new column in a table, with optional constraints

1.5.2 socket_comm

This module implements a simple TCP server/client structure to develop daemon-like application.

1.6 Authors

- **Carlos Vigo** - *Initial work* - GitLab

1.7 Contributing

Please read our contributing policy for details on our code of conduct, and the process for submitting pull requests to us.

1.8 Versioning

We use Git for versioning. For the versions available, see the tags on this repository.

1.9 License

This project is licensed under the GNU GPLv3 License

1.10 Built With

- PyCharm Professional 2020 - The IDE used
- Sphinx - Documentation

1.11 Acknowledgments

- Nobody so far

CHAPTER

TWO

API REFERENCE

Description

Collection of useful modules to build consistent Python apps. All modules share some basic principles to increase app compatibility and facilitate development:

- **Settings**. The modules have a `config()` method based in the standard library `configparser`. Documentation about the different configuration files can be found in the *examples section*.
- **Logging**. The modules use the standard `logging` library to manage logs at all levels. Each method will produce logs using a logger named like the method itself, so an app importing the module can easily modify the logging behaviour on a per-method basis. This is shown in the example TODO.

Modules

database	Basic interface to a PostgreSQL database.
socket_comm	Server/client communication via TCP sockets.
custom_logging	Implements a custom logging service.

2.1 database

Description

Basic interface to a PostgreSQL database.

The module consists of a main class *Database* which implements methods for connection and disconnection, table verification and data insertion.

The database settings are set with a *config file* and the standard library `configparser`.

Classes

Constraint	List of accepted constraints for new columns.
DataType	List of accepted data types for new columns.
Database	Manages connections and operations with a PostgreSQL database.

2.1.1 Constraint

Description

class lab_utils.database.**Constraint**(*value*)

List of accepted constraints for new columns. The constraints are hard-coded for safety reasons: SQL insertions are potentially dangerous.

Attributes

Constraint.positive	The variable must be greater or equal to 0
Constraint.positive_strict	The variable must be strictly positive

Constraint.positive

Constraint.**positive = ' CHECK({column_name} >= 0) '**

The variable must be greater or equal to 0

Constraint.positive_strict

Constraint.**positive_strict = ' CHECK({column_name} > 0) '**

The variable must be strictly positive

2.1.2 DataType

Description

class lab_utils.database.**DataType**(*value*)

List of accepted data types for new columns. The types are hard-coded for safety reasons: SQL insertions are potentially dangerous. See here for more information.

Attributes

DataType.bool	Boolean
DataType.double	Floating-point number, 8 bytes
DataType.float	Floating-point number, 4 bytes
DataType.int	Integer (4 bytes, range is -2,147,483,648 to +2,147,483,647)
DataType.long	Integer (8 bytes, range is -9,223,372,036,854,775,808 to +9,223,372,036,854,775,807)
DataType.short	Integer (2 bytes, range is -32,768 to +32,767)
DataType.string	String, unlimited length
DataType.time	Time stamp, with time zone information

DataType.bool

DataType.**bool = 'BOOLEAN'**
Boolean

DataType.double

DataType.**double = 'FLOAT(53)'**
Floating-point number, 8 bytes

DataType.float

DataType.**float = 'FLOAT(24)'**
Floating-point number, 4 bytes

DataType.int

DataType.**int = 'INTEGER'**
Integer (4 bytes, range is -2,147,483,648 to +2,147,483,647)

DataType.long

DataType.**long = 'BIGINT'**
Integer (8 bytes, range is -9,223,372,036,854,775,808 to +9,223,372,036,854,775,807)

DataType.short

DataType.**short = 'SMALLINT'**
Integer (2 bytes, range is -32,768 to +32,767)

DataType.string

DataType.**string = 'TEXT'**
String, unlimited length

DataType.time

DataType.**time = 'TIMESTAMPTZ'**
Time stamp, with time zone information

2.1.3 Database

Description

class lab_utils.database.**Database**(*config_file: Optional[str] = None*, *host: Optional[str] = None*, *port: Optional[int] = None*, *database: Optional[str] = None*, *user: Optional[str] = None*, *passfile: Optional[str] = None*)
Manages connections and operations with a PostgreSQL database. The class is based on the psycopg2 library and on this tutorial.

Attributes

Database.connection	Connection object returned by psycopg2.connect()
Database.cursor	Cursor provided by connection.cursor() to execute an SQL query
Database.database	The database name to connect to
Database.db_version	Database version
Database.host	The host name where the database is located
Database.passfile	Location of the pgpass file with the credentials
Database.port	Connection port
Database.user	User name

Database.connection

Database.**connection = None**
Connection object returned by psycopg2.connect()

Database.cursor

Database.**cursor = None**
Cursor provided by connection.cursor() to execute an SQL query

Database.database

Database.**database: str = 'beam'**
The database name to connect to

Database.db_version

Database.**db_version: str = ''**
Database version

Database.host

Database.**host: str = 'localhost'**
The host name where the database is located

Database.passfile

Database.**passfile: str = '~/.pgpass'**
Location of the pgpass file with the credentials

Database.port

Database.**port: int = 5432**
Connection port

Database.user

Database.**user: str = 'cw-beam'**
User name

Methods

Database.__init__	Initializes the *Database* object.
Database.check_column	Checks if a column exists in a given table.
Database.check_empty_table	Checks whether a table is empty or not
Database.check_table	Checks if a table exists.
Database.close	Closes the connection to the database.
Database.config	Loads the configuration.
Database.connect	Connects to the database.
Database.create_aggregate_view	Creates a set of aggregate views in a given table.
Database.create_database	Creates a database named *db_name*.
Database.create_timescale_db	Creates a TimescaleDB table.

continues on next page

Table 6 – continued from previous page

Database.fetch_latest_value	Retrieves the latest values of a time-ordered table.
Database.fetch_next_serial_id	Checks the latest ID of a serialized table and returns the next available one.
Database.get_list_columns	Fetches the list of columns in a given table.
Database.new_column	Creates a new column in a given table.
Database.new_entry	Inserts data into a given table.
Database.update_measurement	Updates the entry of a measurement.

Database.__init__

Database.**__init__**(*config_file: Optional[str] = None*, *host: Optional[str] = None*, *port: Optional[int] = None*, *database: Optional[str] = None*, *user: Optional[str] = None*, *passfile: Optional[str] = None*)

Initializes the *Database* object. If a *configuration file name* is given, the constructor calls the method *config()* and overrides the default attributes

Parameters

- **config_file** (*str, optional*) – Configuration file name, default is *None*. See *here* for a configuration file example.
- **host** (*str, optional*) – Database host.
- **port** (*int, optional*) – Database port.
- **database** (*str, optional*) – Database name.
- **user** (*str, optional*) – Database user.
- **passfile** (*str, optional*) – Local file with PostgreSQL password.

Raises **configparser.Error** – If a configuration file name was given, the method *config()* can fail raising this exception.

Database.check_column

Database.**check_column**(*table_name*, *column_name*) → bool

Checks if a column exists in a given table.

Parameters

- **table_name** (*str*) – The table where the column has to be checked
- **column_name** (*str*) – The column to be checked

Returns *True* if the column exists, *False* otherwise.

Return type bool

Raises **psycopg2.Error** – Base exception for all kind of database errors

Database.check_empty_table

Database.**check_empty_table**(*table_name: str*) → bool
Checks whether a table is empty or not

Parameters **table_name**¶ (*str*) – Name of the table where the column has to be retrieved from.

Returns True if the table is empty, False if it is not empty.

Return type bool

Raises **psycopg2.Error** – Base exception for all kind of database errors.

Database.check_table

Database.**check_table**(*table_name*) → bool
Checks if a table exists.

Parameters **table_name**¶ (*str*) – The name of the table to be checked

Returns *True* if the table exists, *False* otherwise.

Return type bool

Raises **psycopg2.Error** – Base exception for all kind of database errors

Database.close

Database.**close**()
Closes the connection to the database.

Raises **psycopg2.Error** – Base exception for all kind of database errors

Database.config

Database.**config**(*config_file: str*)
Loads the configuration.

Reads the `config_file` using the `configparser` library. The structure of the file is shown in the *examples section*.

Parameters **config_file**¶ (*str*) – Configuration file name.

Raises **configparser.Error** – Error while parsing the file, e.g. no file was found, a parameter is missing or it has an invalid value.

Database.connect

Database.**connect**(*print_version: bool = False*)
Connects to the database.

If the connection was successful and the flag *print* was set, it also prints the database version as a connection check.

Parameters **print_version**¶ (*bool, optional*) – Print the database version, default is 'False'.

Raises **psycopg2.Error** – Base exception for all kind of database errors

Database.create_aggregate_view

Database.**create_aggregate_view**(*table_name: str*, *index_name: str = 'time'*, *recreate: bool = True*)
Creates a set of aggregate views in a given table.

Parameters **table_name** (*str*) – Name of the table where the column has to be created.

index_name [str, optional] Column to be used as index for the aggregate view, default is 'time'.

recreate [bool, optional] Recreate the aggregate view if it exists, default is 'True'.

Raises **psycopg2.Error** – Base exception for all kind of database errors. In particular, it is raised if the table does not exist or the aggregate view could not be created.

Database.create_database

Database.**create_database**(*db_name: str*, *owner: str = 'postgres'*)
Creates a database named *db_name*. If *timescaledb_extension* is set (default is 'True'), the TimescaleDB extension is installed in the database to allow TimescaleDB hypertables.

Parameters

- **db_name** (*str*) – The name of the database to be created
- **owner** (*str, optional*) – Database owner, default is 'postgres'.

Raises **psycopg2.Error** – Base exception for all kind of database errors

Database.create_timescale_db

Database.**create_timescale_db**(*table_name: str*, *default_now: bool = True*)
Creates a TimescaleDB table.

The table has a single column named 'time' with type 'TIMESTAMPTZ'. If the flag *default_now* is set (default is 'True'), the column 'time' will default to 'NOW()'

Parameters

- **table_name** (*str*) – The name of the table to be checked
- **default_now** (*bool, optional*) – Set the 'time' column default to 'NOW()', default is *True*.

Raises **psycopg2.Error** – Base exception for all kind of database errors

Database.fetch_latest_value

Database.**fetch_latest_value**(*table_name: str*, *column_name: Optional[List[str]] = None*) → tuple
Retrieves the latest values of a time-ordered table. If paramref:'column_name' is not given, all values of the table are retrieved.

Parameters

- **table_name** (*str*) – Name of the table where the column has to be retrieved from.
- **column_name** (*str*) – Column[s] to be retrieved, default is all columns.

Returns Latest data. First element is the time of the latest entry. Consecutive elements are the value of each column.

Return type tuple

Raises psycopg2.Error – Base exception for all kind of database errors.

Database.fetch_next_serial_id

Database.**fetch_next_serial_id**(*table_name: str*) → int

Checks the latest ID of a serialized table and returns the next available one.

Parameters **table_name**¶ (*str*) – Name of the table to check.

Returns Next available serial ID.

Return type int

Raises psycopg2.Error – Base exception for all kind of database errors.

Database.get_list_columns

Database.**get_list_columns**(*table_name: str*) → List[List[str]]

Fetches the list of columns in a given table.

Returns Column data: list of pairs <name, type>.

Return type List[List[str]]

Raises psycopg2.Error – Base exception for all kind of database errors. In particular, it is raised if the table does not exist or the aggregate view could not be created.

Database.new_column

Database.**new_column**(*table_name: str*, *column_name: str*, *data_type:* lab_utils.database.DataType, *constraints: Optional[list] = None*)

Creates a new column in a given table.

If the column already exists, it just returns. If the table does not exist or the column could not be created, it raises a psycopg2.Error.

Parameters

- **table_name**¶ (*str*) – Name of the table where the column has to be created
- **column_name**¶ (*str*) – Name of the column to be created
- **data_type**¶ (*DataType*) – Data type of the new column
- **constraints**¶ (*list, optional*) – List of *Constraints*, default is 'None'

Raises

- **TypeError** – Invalid constraint or data type
- **ValueError** – Invalid constraint or data type
- **psycopg2.Error** – Base exception for all kind of database errors

Database.new_entry

Database.**new_entry**(*table_name: str*, *columns: list*, *data: list*, *check_columns: bool = False*)

Inserts data into a given table.

See *this example* for usage examples

Parameters

- **table_name** (*str*) – Name of the table where the data has to be inserted
- **columns** (*list[str]*) – List of columns names corresponding to the data
- **data** (*list*) – Values of the new data entry
- **check_columns** (*bool, optional*) – Check that columns exist before insertion, default is False

Raises

- **TypeError** – Invalid data
- **ValueError** – Invalid data
- **psycopg2.Error** – Base exception for all kind of database errors

Database.update_measurement

Database.**update_measurement**(*table_name: str*, *measurement_id: int*, *columns: Optional[List[str]] = None*, *values: Optional[List] = None*, *pairs: Optional[List[Union[Tuple, List]]] = None*)

Updates the entry of a measurement.

Parameters

- **table_name** (*str*) – Measurement type.
- **measurement_id** (*int*) – Measurement ID.
- **columns** (*List[str]*) – List of columns names corresponding to the data
- **values** (*List*) – Values of the new data entry
- **pairs** (*List[tuple]*) – List of pairs (variable, value)

Returns True if the table is empty, False if it is not empty.

Return type bool

Raises **psycopg2.Error** – Base exception for all kind of database errors.

2.2 socket_comm

Description

Server/client communication via TCP sockets. The module implements TCP communication between a daemon-like *Server* and a simple *Client*.

The *Server* class is meant to be run as a daemon-like app. The user should override the *create_parser()* method to define the daemon behaviour upon reception of a message from a *Client*. The base class provides support for the message 'quit', which will terminate the daemon. Any other message will be met with a help-like reply.

The *Client* class communicates with the *Server* sending a text string.

The *ArgumentParser* class and *MessageError* exception are necessary to override some unwanted default behaviour of the argparse library.

The module is based upon this tutorial.

lab_utils.socket_comm.**buffer_size**
Maximum length of a transmitted messages

Type int, 4096

Classes

ArgumentParser	Modifies some annoying behaviours of the argparse library.
Client	Simple TCP client to communicate with a running *Server*.
Server	Daemon-like TCP server.

2.2.1 ArgumentParser

Description

class lab_utils.socket_comm.**ArgumentParser**(*prog=None*, *usage=None*, *description=None*, *epilog=None*, *parents=None*, *formatter_class=<class 'argparse.HelpFormatter'>*, *prefix_chars='-'*, *fromfile_prefix_chars=None*, *argument_default=None*, *conflict_handler='error'*, *add_help=False*, *allow_abbrev=True*)
Modifies some annoying behaviours of the argparse library.

Methods

ArgumentParser.__init__	Overrides the default initialization of add_help to False.
ArgumentParser.error	Avoids the call to sys.exit() when an error is encountered.
ArgumentParser.full_help	Creates a complete help message for the daemon usage.

ArgumentParser.__init__

ArgumentParser.**__init__**(*prog=None*, *usage=None*, *description=None*, *epilog=None*, *parents=None*, *formatter_class=<class 'argparse.HelpFormatter'>*, *prefix_chars='-'*, *fromfile_prefix_chars=None*, *argument_default=None*, *conflict_handler='error'*, *add_help=False*, *allow_abbrev=True*)

Overrides the default initialization of `add_help` to False. It also fixes the 'default value is mutable' warning.

ArgumentParser.error

ArgumentParser.**error**(*message: str*)

Avoids the call to `sys.exit()` when an error is encountered.

Raises *MessageError* – Custom exception just for this purpose.

ArgumentParser.full_help

ArgumentParser.**full_help**() → str

Creates a complete help message for the daemon usage. The –help option of `argparse` does not provide the possibility to print a monolithic help message including the subparsers.

Returns Full help message.

Return type str

2.2.2 Client

Description

class lab_utils.socket_comm.**Client**(*config_file: Optional[str] = None*, *host: Optional[str] = None*, *port: Optional[int] = None*)

Simple TCP client to communicate with a running *Server*. It sends a message and receives the reply from the server.

Attributes

Client.host	Host address.
Client.port	Connection port.

Client.host

Client.**host**: `str` = `'localhost'`

Host address.

Client.port

Client.**port**: int = 1507
Connection port.

Methods

Client.__init__	Initializes the *Client* object.
Client.config	Loads the TCP configuration from the file *config_file*.
Client.send_message	Complete communication process.

Client.__init__

Client.**__init__**(*config_file: Optional[str] = None*, *host: Optional[str] = None*, *port: Optional[int] = None*)
Initializes the *Client* object. If a *config_file* is given, the constructor calls the *config()* method and overrides the default attributes. If the parameters `host` and `port` are given, they will override the configuration file.

Parameters

- **config_file** (*str, optional*) – Configuration file name, default is *None*. Same as See the example TODO.
- **host** (*str, optional*) – Host address, default is *None*.
- **port** (*int, optional*) – Connection port, default is *None*.

Raises **configparser.Error** – If a configuration file name was given, the method *config()* can fail raising this exception.

Client.config

Client.**config**(*config_file: str*)
Loads the TCP configuration from the file *config_file*.

The method reads the file using the library `configparser`..

Parameters **config_file** (*str*) – TCP configuration file, full path.

Raises **configparser.Error** – Error while parsing the file, e.g. no file was found, a parameter is missing or it has an invalid value.

Client.send_message

Client.**send_message**(*message: str*) → str
Complete communication process. Connects to the *Server*, sends a *message*, gets the reply and closes the connection.

Parameters **message** (*str*) – Message for the *Server*.

Raises **OSError** – Various socket errors, e.g. address or timeout

Returns Reply from the server

Return type str

2.2.3 Server

Description

class lab_utils.socket_comm.**Server**(*config_file: Optional[str] = None*, *pid_file_name: Optional[str] = None*, *host: Optional[str] = None*, *port: Optional[int] = None*)

Daemon-like TCP server. The server connects to the specified *host* and *port* and locks a *PID file* to ensure it is the only instance running.

If successful, the server will then listen indefinitely, waiting for a client to connect. Upon connection, a *message* is received and passed to the *parser*. If the message is valid, the parser will call the respective method. The base class provides only the *quit()* method; users should create new methods suitable for their needs. These methods should always set an appropriate *reply*, which will be then sent back to the client.

If a message is not valid (i.e. the parser does not support it), an error message and a complete help string is sent back to the client. The help string by the `argparse` library is not complete and hence is overridden by the *ArgumentParser.full_help()* method.

Attributes

Server.address	TCP binding address.
Server.delimiter_left	Left delimiter
Server.delimiter_right	Left delimiter
Server.host	Host address.
Server.lock	`LockFile` object.
Server.logger	Single *CustomLogger* for the whole class.
Server.max_backlog	TCP connection queue.
Server.message	Message from the client.
Server.namespace	Container to hold message options.
Server.parser	Argument parser.
Server.pid_file_name	The PID file name
Server.port	Connection port.
Server.quit_flag	Internal flag to stop the daemon.
Server.regex	Parsing pattern.
Server.reply	Reply to the client.
Server.sock	Connection socket.
Server.socket_timeout	Socket time-out, used for Ctrl+C handling
Server.sp	Argument subparser

Server.address

Server.**address**: str = None
TCP binding address.

Server.delimiter_left

Server.**delimiter_left**: chr = None
Left delimiter

Server.delimiter_right

Server.**delimiter_right**: chr = None
Left delimiter

Server.host

Server.**host**: str = 'localhost'
Host address.

Server.lock

Server.**lock**: zc.lockfile.LockFile = None
LockFile object.

Server.logger

Server.**logger**: *lab_utils.custom_logging.CustomLogger* = None
Single *CustomLogger* for the whole class.

Server.max_backlog

Server.**max_backlog**: int = 1
TCP connection queue.

Server.message

Server.**message**: str = ''
Message from the client.

Server.namespace

Server.**namespace**: *argparse.Namespace* = **None**
Container to hold message options.

Server.parser

Server.**parser**: *lab_utils.socket_comm.ArgumentParser* = **None**
Argument parser.

Server.pid_file_name

Server.**pid_file_name**: *str* = **'/tmp/socket_comm.pid'**
The PID file name

Server.port

Server.**port**: *int* = **1507**
Connection port.

Server.quit_flag

Server.**quit_flag**: *bool* = **False**
Internal flag to stop the daemon.

Server.regex

Server.**regex**: *Pattern* = **None**
Parsing pattern.

Server.reply

Server.**reply**: *str* = **''**
Reply to the client.

Server.sock

Server.**sock**: *_socket.socket* = **None**
Connection socket.

Server.socket_timeout

Server.**socket_timeout**: *float* = **1.0**
Socket time-out, used for Ctrl+C handling

Server.sp

Server.**sp**: **argparse._SubParsersAction = None**
Argument subparser

Methods

Server.__init__	Initializes and runs the *Server* object.
Server.close	Releases the PID lock file and the TCP socket.
Server.config	Loads the server configuration from a file.
Server.create_parser	Configures the message *parser*, which will call the appropriate method upon reception of a message.
Server.daemonize	Locks a PID file to ensure that a single instance of the server is running.
Server.quit	User-defined task example.
Server.signal_handler	
Server.start_daemon	Starts the server.

Server.__init__

Server.**__init__**(*config_file: Optional[str] = None*, *pid_file_name: Optional[str] = None*, *host: Optional[str] = None*, *port: Optional[int] = None*)
Initializes and runs the *Server* object. The constructor calls the *config()* method to read out the server attributes, and initializes the *logger* and the message *parser*. Finally, the method *daemonize()* tries to lock the PID file *pid_file_name*.

Parameters

- **config_file** (*str, optional*) – Configuration file, default is *None*.
- **pid_file_name** (*str, optional*) – If given, overrides the default *PID file name*.
- **host** (*int, optional*) – If given, overrides the server *host*.
- **port** (*int, optional*) – If given, overrides the server *port*.

Raises

- **configparser.Error** – Configuration file error
- **LockError** – The PID file could not be locked (see here).
- **OSError** – Various socket errors, e.g. address or timeout

Server.close

Server.**close**()
Releases the PID lock file and the TCP socket.

Server.config

Server.**config**(*filename: str*)
Loads the server configuration from a file.

Parameters **filename**¶ (*str*) – The file name to be read.

Raises **configparser.Error** – If an error happened while parsing the file, e.g. no file was found

Server.create_parser

Server.**create_parser**()
Configures the message *parser*, which will call the appropriate method upon reception of a message. Other arguments given to the parser will be available in the *namespace*.

As an example, the subparser for the message 'quit' is implemented. The user should override the *quit()* method, as well as implement other methods for the particular daemon tasks.

Server.daemonize

Server.**daemonize**()
Locks a PID file to ensure that a single instance of the server is running. It is based on the (poorly documented) zc.lockfile package.

Raises **LockError** – The PID file could not be locked.

Server.quit

Server.**quit**()
User-defined task example. The method is called by the *parser* when the message 'quit' is received. For the base class, it just says goodbye to the client. Users should override it to do proper clean-up of their daemon.

Server.signal_handler

Server.**signal_handler**(_, __)

Server.start_daemon

Server.**start_daemon**()
Starts the server. The server will run in an endless loop until the message 'quit' is received. Clients can connect to the TCP port and send a text string. The message will be parsed by the *parser*, which will call the respective function. If the message is invalid, a help string is sent to the client.

The binding to the TCP port might fail for several reasons (e.g. the port is already in use by another process or requires admin rights), in which an OSError exception is raised. If the binding is successful, the server should be able to manage all exceptions, log them, and continue normal operations.

Raises OSError – Various socket errors, e.g. address or timeout

Exceptions

MessageError	Invalid message.

2.2.4 MessageError

exception lab_utils.socket_comm.**MessageError**
Invalid message.

2.3 custom_logging

Description

Implements a custom logging service. The logging setup is based upon the standard Python logging library and offers some advantages:

- Standard logging across multiple apps using this module.
- Ease of use and configuration, with only necessary options.
- Extra functionality:
  - E-mail notification over TLS.
  - Coloured output to a terminal
  - Improved file rotation naming.
  - Logging over TCP socket compatible with the *socket_comm* module.

lab_utils.custom_logging.**SUCCESS**
New log level (25) intended to report successful events to Slack, between INFO (20) and WARNING (30).

Type int

Classes

ColoredFormatter	Console formatter that prepends the coloured severity level of the message.
CustomLogger	Custom logging class based on the default Python Logger.
CustomTimedRotatingFileHandler	Variation of TimedRotatingFileHandler.
NonPickledSocketHandler	Socket handler that sends non-pickled strings.
TlsSMTPHandler	SMTPHandler with TLS support.

2.3.1 ColoredFormatter

Description

class lab_utils.custom_logging.**ColoredFormatter**(*fmt=None*, *datefmt=None*, *style='%'*)

Console formatter that prepends the coloured severity level of the message. Based upon this gist.

Attributes

ColoredFormatter.colour_map	Colour mapping.
ColoredFormatter.colours	Terminal colour codes.
ColoredFormatter.prefix	Prefix to modify terminal output colour.
ColoredFormatter.suffix	Suffix to modify terminal output colour.

ColoredFormatter.colour_map

ColoredFormatter.**colour_map** = **{'CRITICAL': 'bgred', 'DEBUG': 'bggrey', 'ERROR': 'red', 'EXCEPTION': 'bgred', 'INFO': 'cyan', 'SUCCESS': 'green', 'WARNING': 'yellow'}**

Colour mapping.

ColoredFormatter.colours

ColoredFormatter.**colours** = **{'bggrey': 100, 'bgred': 41, 'black': 30, 'blue': 34, 'cyan': 36, 'green': 32, 'magenta': 35, 'red': 31, 'white': 37, 'yellow': 33}**

Terminal colour codes.

ColoredFormatter.prefix

ColoredFormatter.**prefix** = **'\x1b['**

Prefix to modify terminal output colour.

ColoredFormatter.suffix

ColoredFormatter.**suffix = '\x1b[0m'**
Suffix to modify terminal output colour.

Methods

ColoredFormatter.format	Prepends a fixed-length, coloured trailer with the log level.

ColoredFormatter.format

ColoredFormatter.**format**(*record: logging.LogRecord*) → str
Prepends a fixed-length, coloured trailer with the log level.

Parameters **record** (LogRecord) – The record to be logged.

Returns The formatted message

Return type str

2.3.2 CustomLogger

Description

class lab_utils.custom_logging.**CustomLogger**(*name*, *level=0*)
Custom logging class based on the default Python Logger. It introduces the new logging level *SUCCESS* = 25, meant to be used to notify Slack about important, non-error events.

Attributes

—

Methods

CustomLogger.__init__	Calls the parent constructor and adds the *SUCCESS* = 25 logging level.
CustomLogger.success	Creates a log entry with level *SUCCESS*, similar to the standard error() and info().

CustomLogger.__init__

CustomLogger.**__init__**(*name*, *level=0*)
Calls the parent constructor and adds the *SUCCESS* = 25 logging level.

CustomLogger.success

CustomLogger.**success**(*message*, **args*, ***kws*)
Creates a log entry with level *SUCCESS*, similar to the standard `error()` and `info()`.

2.3.3 CustomTimedRotatingFileHandler

Description

class lab_utils.custom_logging.**CustomTimedRotatingFileHandler**(*path: str*, *basename: str*, *extension: str = '.log'*)
Variation of `TimedRotatingFileHandler`. The handler will produce daily log files to a given directory, appending the date to a given base name.

Attributes

CustomTimedRotatingFileHandler.basename	Complete file base name where date will be appended, without extension.
CustomTimedRotatingFileHandler.extension	Log file extension.
CustomTimedRotatingFileHandler.path	Parent directory to save all logs.
CustomTimedRotatingFileHandler.rolloverAt	Next rollover time
CustomTimedRotatingFileHandler.stream	File stream

CustomTimedRotatingFileHandler.basename

CustomTimedRotatingFileHandler.**basename: str = None**
Complete file base name where date will be appended, without extension.

CustomTimedRotatingFileHandler.extension

CustomTimedRotatingFileHandler.**extension**: str = None
Log file extension.

CustomTimedRotatingFileHandler.path

CustomTimedRotatingFileHandler.**path**: str = None
Parent directory to save all logs.

CustomTimedRotatingFileHandler.rolloverAt

CustomTimedRotatingFileHandler.**rolloverAt** = None
Next rollover time

CustomTimedRotatingFileHandler.stream

CustomTimedRotatingFileHandler.**stream**: TextIO = None
File stream

Methods

CustomTimedRotatingFileHandler.__init__	Calls the parent constructor and creates the logging directory, if it does not exist.
CustomTimedRotatingFileHandler.doRollover	Rotates log files on daily basis.

CustomTimedRotatingFileHandler.__init__

CustomTimedRotatingFileHandler.**__init__**(*path: str*, *basename: str*, *extension: str = '.log'*)
Calls the parent constructor and creates the logging directory, if it does not exist.

Parameters

- **path** (*str*) – Parent directory to save all logs.
- **basename** (*str*) – File base name where date will be appended, without extension.
- **extension** (*str, optional*) – Log file extension, default is 'log'.

Raises OSError – The logging directory could not be created. The handler should not be used if this exception is raised.

CustomTimedRotatingFileHandler.doRollover

CustomTimedRotatingFileHandler.**doRollover**()
Rotates log files on daily basis. The file name of the current logfile is *basename* + strftime() + *extension*, with time format '%Y-%m-%d'.

2.3.4 NonPickledSocketHandler

Description

class lab_utils.custom_logging.**NonPickledSocketHandler**(*host*, *port*)
Socket handler that sends non-pickled strings. Such strings are compatible with a *Server* listening on the appropriate TCP port.

Attributes

—

Methods

NonPickledSocketHandler.emit	Encodes a *record* and writes it to the socket.

NonPickledSocketHandler.emit

NonPickledSocketHandler.**emit**(*record: logging.LogRecord*)
Encodes a *record* and writes it to the socket. If there is an error with the socket, silently drops the packet. If there was a problem with the socket, re-establishes the socket.

Parameters **record**¶ (LogRecord) – The record to be emitted.

2.3.5 TlsSMTPHandler

Description

class lab_utils.custom_logging.**TlsSMTPHandler**(*mailhost*, *fromaddr*, *toaddrs*, *subject*, *credentials=None*, *secure=None*, *timeout=5.0*)
SMTPHandler with TLS support. Based upon this gist.

Attributes

TlsSMTPHandler.fromaddr	Sender address (inherited).
TlsSMTPHandler.mailhost	Mail provider (inherited).
TlsSMTPHandler.mailport	Mail port (inherited).
TlsSMTPHandler.password	Login password (inherited).
TlsSMTPHandler.toaddrs	Recipients addresses (inherited).
TlsSMTPHandler.username	Login username (inherited).

TlsSMTPHandler.fromaddr

TlsSMTPHandler.**fromaddr**: str = None
Sender address (inherited).

TlsSMTPHandler.mailhost

TlsSMTPHandler.**mailhost**: str = None
Mail provider (inherited).

TlsSMTPHandler.mailport

TlsSMTPHandler.**mailport**: int = None
Mail port (inherited).

TlsSMTPHandler.password

TlsSMTPHandler.**password**: str = None
Login password (inherited).

TlsSMTPHandler.toaddrs

TlsSMTPHandler.**toaddrs**: List[str] = None
Recipients addresses (inherited).

TlsSMTPHandler.username

TlsSMTPHandler.**username**: str = None
Login username (inherited).

Methods

TlsSMTPHandler.emit	Emits a record.

TlsSMTPHandler.emit

TlsSMTPHandler.**emit**(*record*)
Emits a record. Opens a TLS SMTP connection using the `smtplib` library and sends an `EmailMessage`.

Functions

configure_logging	Sets up the custom logger.
getLogger	Overrides the Python standard `logging.getLogger()` to fix type completion hints, referring them to *CustomLogger* instead of `Logger`.
string_to_bool	Parses a variety of strings (e.g.

2.3.6 configure_logging

lab_utils.custom_logging.**configure_logging**(*config_file: Optional[str] = None*, *fallback: bool = False*, *logger_name: str = 'root'*, *log_level: Optional[int] = None*)

Sets up the custom logger. Loads the configuration from *config_file* using the `configparser` library.

Parameters

- **config_file** (*str*) – Configuration file name.
- **fallback** (*bool, optional*) – If 'True' and the logger setup fails, fall back to the default `Logger`.
- **logger_name** (*str, optional*) – Logger name.
- **log_level** (*int, optional*) – Initial logging level, overrides the configuration file.

Raises `configparser.Error` – Error while parsing the file, e.g. no file was found, a parameter is missing or it has an invalid value.

2.3.7 getLogger

lab_utils.custom_logging.**getLogger**(*name: Optional[str] = None*) → *lab_utils.custom_logging.CustomLogger*

Overrides the Python standard `logging.getLogger()` to fix type completion hints, referring them to *CustomLogger* instead of `Logger`. Taken from a StackOverflow question.

Parameters **name** (*str, optional*) – The logger name

Returns A named instance of the logger.

Return type *CustomLogger*

2.3.8 string_to_bool

lab_utils.custom_logging.**string_to_bool**(*s: str*) → bool

Parses a variety of strings (e.g. 'true' or '1') to a boolean.

Parameters **s** (*str*) – The string to parse

Returns

True if `s` is one of:

- 'true'
- '1'
- 't'

- 'y'
- 'yes'
- 'on'

Return type bool

CHAPTER

THREE

EXAMPLES

This documentation is intended to show practical usage examples of the different modules included in the *lab_utils* package.

3.1 Configuration files

The `config` method of each module expects a configuration file with a specific pattern. In addition, a sample file accepted by the standard `logging.config` method is also provided.

3.1.1 Logging Configuration File

The logging configuration file

3.1.2 Database Configuration File

The database configuration file

3.2 Database

3.2.1 Installing

lab_utils can be obtained from pip via

```
pip install lab_utils
```

You can also get *lab_utils* from its current source on GitHub, to get all the latest and greatest features. *lab_utils* is under active development, and many new features are being added. However, note that the API is currently unstable at this time.

```
git clone https://github.com/mrocklin/sparse.git
cd ./sparse/
pip install .
```

CHAPTER

FOUR

CHANGELOG

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog, and this project adheres to Semantic Versioning.

4.1 [0.5.11] - 2021-05-12

- **custom_logging**: fix bug in midnight file rotation

4.2 [0.5.10] - 2021-04-28

- **custom_logging**: minor bug fixes

4.3 [0.5.9] - 2021-04-22

- **database**: add method to update an existing measurement entry

4.4 [0.5.8] - 2021-04-22

- **database**: add methods to check whether a table is empty and to retrieve the next serial ID

4.5 [0.5.7] - 2021-04-16

- **custom_logging**: add parameter to *configure_logging* for initial logging level setup

4.6 [0.5.6] - 2021-04-16

- **database**: add method *get_list_columns* to retrieve all columns in a table

4.7 [0.5.5] - 2021-04-16

- **database**: add method *fetch_latest_value* to retrieve latest data from a time-ordered table

4.8 [0.5.4] - 2021-04-14

- **database**: remove aggregate views for 10 s and 10 min, only 1 min is left

4.9 [0.5.2] - 2020-10-26

- Changes to **socket_comm** module:
  - Fix bug when the argparse option **choices** is used for an argument
  - Increase TCP buffer size to 4096

4.10 0.5.1 - 2020-06-22

- Changes to **database** module:
  - Fix method create_aggregate_view

4.11 0.5.0 - 2020-06-09

- Changes to **database** module:
  - Add methods create_database and create_aggregate_view
  - Minor improvements and typos fixed

4.12 0.4.0 - 2020-05-25

- Improve log formatting
- Implement new logging system in the database and socket_comm modules
- Minor changes to documentation

4.13 0.3.0 - 2020-05-20

- Fix Server class destructor.
- Add module **custom_logging** for homogeneous logging setup across apps with the following handlers:
  - Console (with coloured code).
  - File (with daily rotation).
  - TCP socket, to notify a central alarm management app.
  - Email (SMTP over TLS).
  - Slack notification.
- Implement new logging schema in the examples.
- Improve documentation and other minor fixes.

4.14 0.2.0 - 2020-05-08

- Implement CI with .__gitlab-ci.yml.
- Improve documentation
- Module socket_comm:
- Implement method to send a complete help message to the client.
- Implement signal ahndler to deal with Ctrl+C nicely
- Expand examples

4.15 0.1.0 - 2020-05-05

- First release of the **lab-utils** package
- Installation instructions and setup
- Modules available: **database** and **socket_comm**

CHAPTER

FIVE

CONTRIBUTING

When contributing to this repository, please first discuss the change you wish to make via issue, email, or any other method with the owners of this repository before making a change.

Please note we have a code of conduct, please follow it in all your interactions with the project.

5.1 Pull Request Process

1. Ensure any install or build dependencies are removed before the end of the layer when doing a build.
2. Update the README.md with details of changes to the interface, this includes new environment variables, exposed ports, useful file locations and container parameters.
3. Increase the version numbers in any examples files and the README.md to the new version that this Pull Request would represent. The versioning scheme we use is SemVer.
4. You may merge the Pull Request in once you have the sign-off of two other developers, or if you do not have permission to do that, you may request the second reviewer to merge it for you.

5.2 Code of Conduct

5.2.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

5.2.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

5.2.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

5.2.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

5.2.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at [INSERT EMAIL ADDRESS]. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

5.2.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at http://contributor-covenant.org/version/1/4

CHAPTER

SIX

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

6.1 Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: **(1)** assert copyright on the software, and **(2)** offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

6.2 TERMS AND CONDITIONS

6.2.1 0. Definitions

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that **(1)** displays an appropriate copyright notice, and **(2)** tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

6.2.2 1. Source Code

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that **(a)** is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and **(b)** serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs

which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

6.2.3 2. Basic Permissions

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

6.2.4 3. Protecting Users' Legal Rights From Anti-Circumvention Law

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

6.2.5 4. Conveying Verbatim Copies

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

6.2.6 5. Conveying Modified Source Versions

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- **a)** The work must carry prominent notices stating that you modified it, and giving a relevant date.
- **b)** The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- **c)** You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- **d)** If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6.2.7 6. Conveying Non-Source Forms

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- **a)** Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- **b)** Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either **(1)** a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or **(2)** access to copy the Corresponding Source from a network server at no charge.
- **c)** Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- **d)** Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- **e)** Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either **(1)** a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or **(2)** anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

6.2.8 7. Additional Terms

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- **a)** Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- **b)** Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- **c)** Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- **d)** Limiting the use for publicity purposes of names of licensors or authors of the material; or
- **e)** Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

- **f)** Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

6.2.9 8. Termination

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated **(a)** provisionally, unless and until the copyright holder explicitly and finally terminates your license, and **(b)** permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

6.2.10 9. Acceptance Not Required for Having Copies

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

6.2.11 10. Automatic Licensing of Downstream Recipients

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

6.2.12 11. Patents

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either **(1)** cause the Corresponding Source to be so available, or **(2)** arrange to deprive yourself of the benefit of the patent license for this particular work, or **(3)** arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license **(a)** in connection with copies of the covered work conveyed by you (or copies made from those copies), or **(b)** primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

6.2.13 12. No Surrender of Others' Freedom

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

6.2.14 13. Use with the GNU Affero General Public License

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

6.2.15 14. Revised Versions of this License

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

6.2.16 15. Disclaimer of Warranty

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

6.2.17 16. Limitation of Liability

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

6.2.18 17. Interpretation of Sections 15 and 16

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

6.3 How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

PYTHON MODULE INDEX

INDEX

Symbols

A

B

C

D

E

F

G

H

I

L

M

N

P

Q

R

S

T

U