local$_{s}imple_{d}atabaseDocumentation$

stanislav

Feb 12, 2022

Contents

CHAPTER 1

LOCAL_SIMPLE_DATABASE

Table of Contents

1.1 Short Overview.

local_simple_database is a simple Python package(**py>=2.7 or py>=3.4**) with the main purpose to help storing and retrieving data from human-readable .txt files with one line of code. All the interactions with files are being made in a processes-threads safe manner.

1.2 Long Overview.

This package consists of 2 main classes with which user should interact:

1. LocalSimpleDatabase
2. LocalDictDatabase

1.2.1 One small example

Let's say you want to store file with int variable with name int_times_I_ve_eaten.

Then, using this package, you can do it like this:

```
from local_simple_database import LocalSimpleDatabase
LSD = LocalSimpleDatabase(path_to_dir_where_to_save_file)
```

and then just use everywhere in your code **LSD["int_times_I_ve_eaten"]** like if it was usual dictionary.

```
LSD["int_times_I_ve_eaten"] += 1  # To increase value in the file
LSD["int_times_I_ve_eaten"]  # To get current value from the file
```

After running this code with:
path_to_dir_where_to_save_file = "./folder_with_all_my_databases"
Inside directory *./folder_with_all_my_databases*
will be created file *"int_times_I_ve_eaten.txt"* with current value.

Value is stored in a human-readable .txt file, so you can always access it.
To get it some time later or from another process just use:

```
int_value_I_was_afraid_to_lose = LSD["int_times_I_ve_eaten"]
```

How to name databases

Name of database should satisfy template "type_name"

Examples: int_balls, float_seconds_left, str_my_name, dict_useless_heap

So just by the name you can define the type of database, isn't it awesome.

1.3 Installation

- Install via pip:

```
pip install local_simple_database
```

1.4 Basic usage.

1.4.1 1) LocalSimpleDatabase

This class is built to handle (saving-retrieving) one value data like integer or float.

For now supported types of databases are:

- ["int", "float", "str", "datetime"] (Probably will be enhanced soon)
- This means that one file with database can handle only type data

Initialization of databases handler

```
from local_simple_database import LocalSimpleDatabase
LSD = LocalSimpleDatabase(
    str_path_database_dir=".",
)
```

Arguments:

1. **str_path_database_dir:**

   If the explicit path is not given or variable is not set at all,
   then will be used path "./local_database"

Folder for database will be created automatically

A few examples of Usage

After you've initialized LSD object you can use:

1) Integer database

If you want to store/access/modify simple int in file:

```
# Process 1
LSD["int_red_cars_drove"] += 1
LSD["int_red_cars_drove"] += 2
# Oh now, last one was burgundy
LSD["int_red_cars_drove"] -= 1

# Process 2
print("red cars already found", LSD["int_red_cars_drove"])
# If there was no such DataBase yet, than in will be created and 0 value will be␣
↪returned.
LSD["int_red_cars_drove"] = 5
print("Red cars already found: ", LSD["int_red_cars_drove"])
```

2) Float database

```
LSD["float_last_price_of_watermelons"] = 7.49
# Too many watermelons this year, need to apply 30% discount
LSD["float_last_price_of_watermelons"] *= 0.7
print(
    "Hello my best customer, current price on watermelons is: ",
    LSD["float_last_price_of_watermelons"]
)
```

3) Datetime database

```
import datetime
# Saving datetime in file in ISO format (E.G. 2020-05-16T18:00:41.780534)
LSD["datetime_now"] = datetime.datetime.now()

# Load datetime obj from DataBase
# if DB not found will be retunrs datetime for 1970-01-01
print("Hour was a moment ago: ", LSD["datetime_now"].hour)

# Use DataBase value to find timedelta
int_seconds_gone = (datetime.datetime.now() - LSD["datetime_now"]).seconds
print("Seconds gone: ", int_seconds_gone)
```

4) Date database

Very similar to datetime database, but only date will by saved

```
import datetime
# Saving datetime in file in ISO format (E.G. 2020-05-16)
LSD["date_now"] = datetime.datetime.now()

# Load datetime obj from DataBase
# if DB not found will be retunrs datetime for 1970-01-01
print("Date today: ", LSD["date_now"])

# Use DataBase value to find timedelta
if datetime.datetime.now().date() == LSD["date_now"]:
    int_seconds_gone_today = (datetime.datetime.now() - LSD["date_now"]).seconds
    print("Seconds already gone: ", int_seconds_gone_today)
```

1.4.2 2) LocalDictDatabase

This class was built to handle (saving-retrieving) dictionary with data from a file.

Work with such database is a little different from **LocalSimpleDatabase** so it was necessary to put it in a separate class

Initialization of databases handler

```
from local_simple_database import LocalDictDatabase
LDD = LocalDictDatabase(
    str_path_database_dir=".",
    default_value=None,
)
```

Arguments:

1. **str_path_database_dir:**

   If the explicit path is not given or variable is not set at all,
   then will be used path "./local_database"
   Folder for databases will be created automatically

2. **default_value: value to use for any database if key in it is not found.**

   LDD[database_name][key] = default_value

A few examples of Usage

1) Basic store/access/modify data from a dict database.

```
# Set methods
## Set value for whole LDD:
LDD["dict_very_useful_heap"] = {"Mike": 50, "Stan": 1000000}

## Set keys for one dictionary LDD
## If there is no file with asked dict database then it will be created automatically
LDD["dict_useless_heap"]["random_key"] = 1
LDD["dict_useless_heap"]["random_key"] += 3
LDD["dict_useless_heap"][2] = ["Oh my God, what a list is doing here", "Aaa"]
```

(continues on next page)

(continued from previous page)

```
LDD["dict_useless_heap"][99] = {"Are you serious?": {"You'd better be!": "Bbb"}}

# Get methods
## To get whole dict for LDD, please use:
LDD["dict_useless_heap"].get_value()  # Sorry for that, I don't know how to do it␣
↪without additional method

## To get string representation of whole dict:
print(LDD["dict_useless_heap"])

## To get one key from dict:
int_random_key = LDD["dict_useless_heap"]["random_key"]
```

2) Set default value:

```
# You can set the default value for all databases OR for only one:

## 1) Set default value for any database when can't find key:
LDD.change_default_value(0)

## 2) Set default value for one database:
LDD["cars"].change_default_value(0)

# They you can use LDD similarly to collections.defaultdict
LDD["cars"]["red"] += 1
# Oh no, that was burgundy once again
LDD["cars"]["red"] -= 1
LDD["cars"]["burgundy"] += 1
```

1.5 Advanced usage (can be skipped, you already know enough to use it)

1.5.1 1) class database additional arguments

Both 2 main classes (**LocalSimpleDatabase**, **LocalDictDatabase**) have additional arguments:

1) **str_datetime_template_for_rolling=""**

   This variable allows setting rolling save of database results using the DateTime template.

   If the value is not empty, then saving/retrieving results will be done from deeper folders with names satisfy the evaluation of the DateTime string template.

   E.G. To save daily results use "%Y%m%d" (Then deeper folder names will be like "20191230", "20191231", . . .)

   E.G. To save hourly results use "%Y%m%d_%H" (Then deeper folder names will be like "20191230_0", "20191230_23", . . .)

2) **float_max_seconds_per_file_operation=0.01**

   This variable is necessary for multiprocessing safe work.

   It setting time in which LSD file accessed by process can't be accessed by any other process.

   By default, it is set to 10 ms for simple database and 20 ms for dict database.

If you use operations which from accessing value till setting new value needs more time, you are more than welcome to increase it.

You can set it to 0.0 if you are not using threads-processes and want the maximum speed.

```
# Full definition of LocalSimpleDatabase
LSD = LocalSimpleDatabase(
    str_path_database_dir=".",
    float_max_seconds_per_file_operation=0.05,
    str_datetime_template_for_rolling=""
)
```

```
# Full definition of LocalDictDatabase
LDD = LocalDictDatabase(
    str_path_database_dir=".",
    default_value=None,
    float_max_seconds_per_file_operation=0.05,
    str_datetime_template_for_rolling=""
)
```

1.5.2 2) Rolling example

```
LSD_daily_rolling = LocalSimpleDatabase(
    str_path_database_dir=".",
    str_datetime_template_for_rolling="%Y%m%d"
)
```

1.5.3 3) Get values for ALL databases in the directory.

To get a dictionary with data in all databases by database name, use:

```
LSD.get_dict_data_by_db_name()
```

If you were using rolling, then you can get dictionary with results like {"datetime_1": dict_all_DBs_date_1, }

```
LSD.get_dict_every_DB_by_datetime()
```

If you were using rolling, and interested only in one database. {"datetime_1": database_value_1, . . . }

```
# Please replace *str_database_name* on name of LSD which values you want to get
LSD.get_one_db_data_daily(
    str_database_name,
    value_to_use_if_db_not_found=None
)
```

1.6 Links

- PYPI
- readthedocs
- GitHub

1.7 Project local Links

- CHANGELOG.
- CONTRIBUTING.

1.8 Contacts

- Email: stas.prokopiev@gmail.com
- vk.com
- Facebook

1.9 License

CHAPTER 2

Contents

2.1 License

2.2 Authors

- Stanislav Prokopyev <stas.prokopiev@gmail.com>

2.2.1 Contacts

- email: stas.prokopiev@gmail.com
- vk.com
- Facebook

2.3 Changelog

2.3.1 Version 0.0.1

- First release, waiting to see tons of errors.

2.4 local_simple_database

2.4.1 local_simple_database package

Submodules

local_simple_database.class_dict_database_handler module

Module with class to handle dict databases

class DictDatabaseHandler(*local_dict_database_obj*, *str_db_name*)

Bases: `object`

This class was built to handle one DICT DataBase

...

`self.`**local_dict_database_obj**

Handler of all DICT database-s in the folder

Type object of class: LocalDictDatabase

`self.`**str_db_name**

Name of DataBase which to use

Type str

`self.`**default_value**

Value to use if key in database is not found

Type object

change_default_value(*new_default_value*)

Change default value for exactly one DataBase

Parameters **new_default_value** (*object*) – Value to use if key in database is not found

get_value()

Get current dict value of whole DataBase

set_value(*dict_values_to_set*)

Setting whole value of DataBase to given dictionary

Parameters **dict_values_to_set** (*dict*) – Any dictionary

local_simple_database.class_local_dict_database module

module with main class to process Local Databases with dictionaries

class LocalDictDatabase(*str_path_database_dir='.', float_max_seconds_per_file_operation=0.01, default_value=None, str_datetime_template_for_rolling=''*)

Bases: `local_simple_database.virtual_class_all_local_databases.VirtualAnyLocalDatabase`

This class was built to handle dictionary DataBase-s

...

self.str_path_main_database_dir [str] Path to main folder with DataBase-s

self.str_datetime_template_for_rolling [str] Datetime template for folder name if to use rolling

self.list_supported_types [list] DataBase Types with which this local database can work

self.dict_file_lock_by_fila_path [dict] {file_path_1: FileLock object, ...}

self.float_max_seconds_per_file_operation [float] Seconds per file operation, need it for multiprocessing safety

self.**default_value**

Value to use if key in database is not found

Type object

self.**dict_db_handler_by_str_db_name**

{database_name: handler (special object to handle access to one DB)}

Type dict

change_default_value(*new_default_value*)

Changing default value to use for all DICT DataBase-s

Parameters **new_default_value** (*obj*) – Value to use if key in database is not found

init_new_class_obj(***kwargs*)

Create a new instance of the same class object

local_simple_database.class_local_simple_database module

Module with class to handle all simple local databases

class LocalSimpleDatabase(*str_path_database_dir='.', float_max_seconds_per_file_operation=0.01, str_datetime_template_for_rolling=''*)

Bases: `local_simple_database.virtual_class_all_local_databases.VirtualAnyLocalDatabase`

This class was built to handle all one value DataBase-s

...

self.str_path_main_database_dir [str] Path to main folder with DataBase-s

self.str_datetime_template_for_rolling [str] Datetime template for folder name if to use rolling

self.list_supported_types [list] DataBase Types with which this local database can work

self.dict_file_lock_by_fila_path [dict] {file_path_1: FileLock object, ...}

self.float_max_seconds_per_file_operation [float] Seconds per file operation, need it for multiprocessing safety

self.**dict_str_db_type_by_str_db_name**

{database_1_name: str_value_type, ...}

Type dict

self.**dict_list_db_allowed_types_by_str_db_name**
{database_1_name: list_allowed_types_for_set_value, ...}

Type dict

self.**dict_func_db_getter_by_str_db_name**
{database_1_name: func_to_convert_str_to_value, ...}

Type dict

self.**dict_func_db_setter_by_str_db_name**
{database_1_name: func_to_convert_value_to_str, ...}

Type dict

init_new_class_obj(***kwargs*)
Create a new instance of the same class object

init_new_simple_database(*str_db_name*)
Method for first preparings for new database

Parameters **str_db_name** (*str*) – Name of DataBase which to use

local_simple_database.virtual_class_all_local_databases module

Module with virtual class to contain all common methods for any database

class VirtualAnyLocalDatabase(*str_path_database_dir='.', float_max_seconds_per_file_operation=0.01, str_datetime_template_for_rolling=''*)

Bases: object

This is virtual class to handle all needs for all child DataBases

...

self.**str_path_main_database_dir**
Path to main folder with DataBase-s

Type str

self.**str_datetime_template_for_rolling**
Datetime template for folder name if to use rolling

Type str

self.**list_supported_types**
DataBase Types with which this local database can work

Type list

self.**dict_file_lock_by_file_path**
{file_path_1: FileLock object, ...}

Type dict

self.**float_max_seconds_per_file_operation**
Seconds per file operation, need it for multiprocessing safety

Type float

define_type_of_db_by_name(*str_db_name*)
Define type on database if it name follows given template type_name

Parameters **str_db_name** (*str*) – Name of asked database

get_dict_data_by_db_name()
Getting dict with data of every database in the dir of DataBase

get_dict_every_DB_by_datetime()
Getting {date_1: dict_results_of_all_DBs_for_date_1, date_2: ... }

get_dir_names_of_all_dirs_with_rolling_DBs()
Getting names of dir-s with rolling results for DataBase

get_file_path_with_db_file(*str_db_name*)
Get path to file with DataBase

Parameters **str_db_name** (*str*) – Name of asked database

get_folder_for_databases()
Getting folder where should be file with database

get_list_names_of_all_files_with_dbs_in_dir()
Getting all names of databases in DB-handler folder

get_names_of_files_in_dbs_dir()
Get names of files in current directory of DataBase

get_one_db_data_daily(*str_db_name*, *value_to_use_if_db_not_found=None*)
Getting {date_1: value_1, date_2: value_2, ... } for one database

Parameters

- **str_db_name** (*str*) – Name of DataBase which to use
- **value_to_use_if_db_not_found** (*object*) – value to set if results for some days not found

read_file_content(*str_db_name*)
Read whole content of file with DataBase in multiprocess safe way

Parameters **str_db_name** (*str*) – Name of asked database

save_file_content(*str_content*, *str_db_name*)
Save content to file with DataBase in multiprocess safe way

Parameters

- **str_content** (*str*) – Content to save in database file
- **str_db_name** (*str*) – Name of asked database

Module contents

class LocalSimpleDatabase(*str_path_database_dir='.'*, *float_max_seconds_per_file_operation=0.01*, *str_datetime_template_for_rolling=''*)
Bases: `local_simple_database.virtual_class_all_local_databases.VirtualAnyLocalDatabase`

This class was built to handle all one value DataBase-s

...

self.str_path_main_database_dir [str] Path to main folder with DataBase-s

self.str_datetime_template_for_rolling [str] Datetime template for folder name if to use rolling

self.list_supported_types [list] DataBase Types with which this local database can work

self.dict_file_lock_by_fila_path [dict] {file_path_1: FileLock object, ... }

self.float_max_seconds_per_file_operation [float] Seconds per file operation, need it for multiprocessing safety

`self.`**`dict_str_db_type_by_str_db_name`**
{database_1_name: str_value_type, . . . }

Type dict

`self.`**`dict_list_db_allowed_types_by_str_db_name`**
{database_1_name: list_allowed_types_for_set_value, . . . }

Type dict

`self.`**`dict_func_db_getter_by_str_db_name`**
{database_1_name: func_to_convert_str_to_value, . . . }

Type dict

`self.`**`dict_func_db_setter_by_str_db_name`**
{database_1_name: func_to_convert_value_to_str, . . . }

Type dict

`init_new_class_obj`(***kwargs*)
Create a new instance of the same class object

`init_new_simple_database`(*str_db_name*)
Method for first preparings for new database

Parameters **`str_db_name`** (`str`) – Name of DataBase which to use

`class LocalDictDatabase`(*str_path_database_dir='.'*, *float_max_seconds_per_file_operation=0.01*, *default_value=None*, *str_datetime_template_for_rolling=''*)
Bases: `local_simple_database.virtual_class_all_local_databases.VirtualAnyLocalDatabase`

This class was built to handle dictionary DataBase-s

. . .

self.str_path_main_database_dir [str] Path to main folder with DataBase-s

self.str_datetime_template_for_rolling [str] Datetime template for folder name if to use rolling

self.list_supported_types [list] DataBase Types with which this local database can work

self.dict_file_lock_by_fila_path [dict] {file_path_1: FileLock object, . . . }

self.float_max_seconds_per_file_operation [float] Seconds per file operation, need it for multiprocessing safety

`self.`**`default_value`**
Value to use if key in database is not found

Type object

`self.`**`dict_db_handler_by_str_db_name`**
{database_name: handler (special object to handle access to one DB)}

Type dict

`change_default_value`(*new_default_value*)
Changing default value to use for all DICT DataBase-s

Parameters **`new_default_value`** (`obj`) – Value to use if key in database is not found

init_new_class_obj(***kwargs*)

Create a new instance of the same class object

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

l

Index

R

S

V