db_lib Documentation

William Stearns

Tue Jan 3 11:27:34 EST 2023

Contents

Module `db_lib`

Test library for sqlite storage.

Functions

Function `add_to_db_dict`

```
def add_to_db_dict(
    dbfiles: Union[str, list],
    key_value_dict: dict
) -> bool
```

Inside the given database, process multiple key/value lists/tuples. For each value, add it to the existing list if not already there.

Function `add_to_db_list`

```
def add_to_db_list(
    dbfiles: Union[str, list],
    key_str: str,
    new_value: str
) -> bool
```

Inside the given database, add the new_value to the list for key_str and write it back if changed.

Function `add_to_db_list_large_value`

```
def add_to_db_list_large_value(
    dbfiles: Union[str, list],
    large_dbfiles: Union[str, list],
    key_str: str,
    new_value: str,
    max_adds: int
) -> bool
```

Inside the given database, add the new_value to the list for key_str and write it back if changed.

Function `add_to_db_multiple_lists`

```
def add_to_db_multiple_lists(
    dbfiles: Union[str, list],
    key_value_list: list
) -> bool
```

Inside the given database, process multiple key/value lists/tuples. For each value, add it to the existing list if not already there.

Function `buffer_delete_vals`

```
def buffer_delete_vals(
    dbfiles: Union[str, list],
    key_str: str,
    delete_values: list,
    max_dels: int
) -> bool
```

Buffer up values that will eventually get removed from their respective databases. You *must* call this with buffer_delete_vals('','', [], 0) to flush any remaining writes before shutting down.

Function `buffer_merges`

```
def buffer_merges(
    dbfiles: Union[str, list],
    key_str: str,
    new_values: list,
    max_adds: int
) -> bool
```

Buffer up writes that will eventually get merged into their respective databases. You *must* call this with buffer_merges('','', [], 0) to flush any remaining writes before shutting down.

Function `delete_key`

```
def delete_key(
        dbfiles: Union[str, list],
        key_str: str
    ) -> bool
```

Delete row with key_str and associated object from database.

Function `insert_key`

```
    def insert_key(
        dbfiles: Union[str, list],
        key_str: str,
        value_obj: Any
    ) -> bool
```

Inserts key_str and its associated python object into database serializing the object on the way in.

Function `insert_key_large_value`

```
    def insert_key_large_value(
        dbfiles: Union[str, list],
        large_dbfiles: Union[str, list],
        key_str: str,
        value_obj: Any
    ) -> bool
```

Inserts key_str and its associates python object into database serializing the object on the way in.

Function `is_sha256_sum`

```
    def is_sha256_sum(
        possible_hash_string: str
    ) -> bool
```

Check if the string is valid hex. Not that it won't correctly handle strings starting with 0x.

Function `remove_from_db_multiple_lists`

```
    def remove_from_db_multiple_lists(
        dbfiles: Union[str, list],
        key_value_list: list
    ) -> bool
```

Inside the given database, process multiple key/value lists/tuples. For each value, remove it from the existing list if there.

Function `select_all`

```
    def select_all(
        dbfiles: Union[str, list],
        return_values: bool = True
    ) -> list
```

Returns all entries from database. Optional parameter return_values decides whether key, value or just key comes back in the list.

Function `select_key`

```
    def select_key(
        dbfiles: Union[str, list],
        key_str: str
    )
```

Searches for key_str from database. If the key_str is found, the obj is unserialized and returned as the original type of that value.

Function `select_key_large_value`

```
def select_key_large_value(
    dbfiles: Union[str, list],
    large_dbfiles: Union[str, list],
    key_str: str
)
```

Searches for key_str from database. If the key_str is found, the obj is unserialized and returned as the original type of that value.

Function `select_random`

```
def select_random(
    dbfiles: Union[str, list]
) -> tuple
```

Selects a random key,value tuple from from all databases (both the sole read-write database at position 0 and the remaining read-only databases.). The return value is a single key,value tuple (unless all databases have no k,v pairs, in which case we return (’’, []) .

Function `setup_db`

```
def setup_db(
    dbfiles: Union[str, list]
) -> bool
```

Create Sqlite3 DB with all required tables.

Function `sha256_sum`

```
def sha256_sum(
    raw_object
) -> str
```

Creates a hex format sha256 hash/checksum of the given string/bytes object.

Function `should_add`

```
def should_add(
    dbfiles: Union[str, list],
    key_str: str,
    existing_list: list,
    new_value: str
) -> bool
```

Make a decision about whether we should add a new value to an existing list.

Module `merge_into_db`

Import pipe-separated key-value pairs and merge into the database specified on the command line.

Module `remove_from_db`

Import pipe-separated key-value pairs and remove the values (and key, if no more values) from the database specified on the command line.

Module `unittest_db_lib`

Perform unit tests for the db_lib library.

Classes

Class `DbFunctionsTest`

```
class DbFunctionsTest(
    methodName='runTest'
)
```

Tests for the db_lib library.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

Ancestors (in MRO)

- unittest.case.TestCase

Methods

Method `test001MakeDB`

```
def test001MakeDB(
    self
)
```

Set up the base databases.

Method `test002DbExists`

```
def test002DbExists(
    self
)
```

Check that it's on disk.

Method `test003AddKeys`

```
def test003AddKeys(
    self
)
```

Add a few keys.

Method `test004CheckThere`

```
def test004CheckThere(
    self
)
```

See that they are in there.

Method `test005AppendValue`

```
def test005AppendValue(
    self
)
```

Add new items to a row value.

Method `test006AddLarge`

```
def test006AddLarge(
    self
)
```

Make sure we can add large values across databases.

Method `test007BufferedMerges`

```
def test007BufferedMerges(
    self
)
```

Test that buffering works correctly.

Method `test999Shutdown`

```
def test999Shutdown(
    self
)
```

Remove test files.
