
T-reX Documentation

Release 0.2.2

Stewart Charles McDowall

Apr 23, 2024

CONTENTS

1	Introduction	3
2	Installation	5
2.1	Dependencies	5
2.2	Installation Instructions	5
3	Usage	7
3.1	Command Line	7
3.2	Python Module	7
4	Configuration modules	9
4.1	CustomConfig Module	9
4.2	General Settings: <code>user_settings.py</code>	9
4.3	Waste Search Settings: <code>queries_waste.py</code>	10
4.4	Material Search Settings: <code>queries_materials.py</code>	11
5	T-reX Modules	13
5.1	FutureScenarios	13
5.2	ExplodeDatabase	14
5.3	SearchWaste	15
5.4	SearchMaterial	17
5.5	MakeCustomDatabase	18
5.6	MethodEditor	19
5.7	ExchangeEditor	20
5.8	VerifyDatabase	22
6	Examples	25
7	Example Output	27
8	Source code	29
9	API Reference	31
9.1	T-reX	31
	Python Module Index	59
	Index	61

The T-reX tool is a python package that allows one to easily and flexibly calculate the waste and material footprint of any product or service inside of life cycle assessment (LCA) databases. Currently it has been tested with ecoinvent 3.8, 3.9, 3.9.1 and 3.10.

[View the example output here](#)



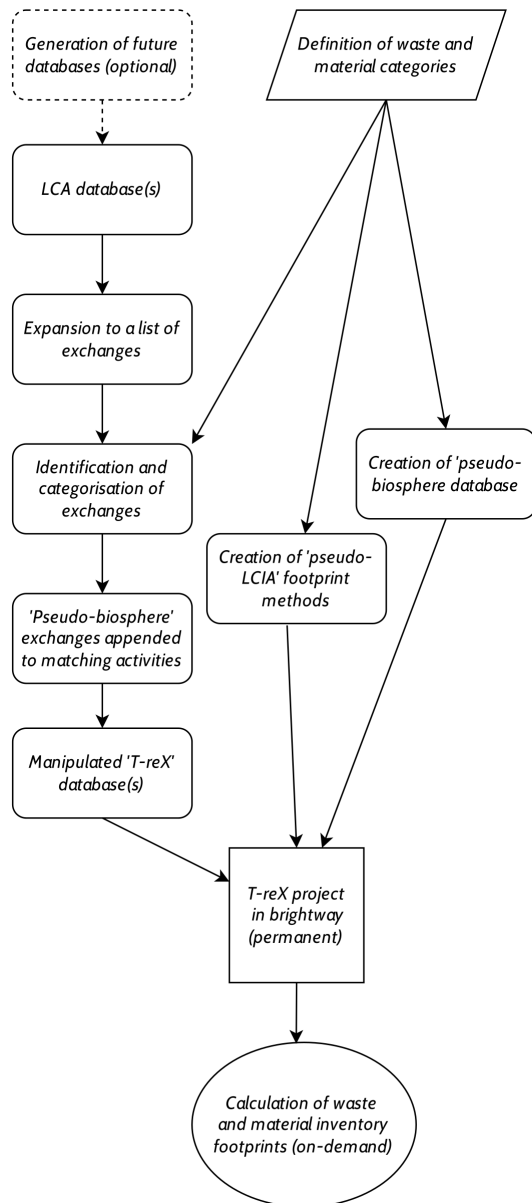
INTRODUCTION

T-reX [T(ool) reX (reduce, recycle, etc.)] is a python package that makes it easy to calculate the waste generation and material demand footprints of any product or service inside of future and prospective life cycle assessment (LCA) databases. Currently tested with the database ecoinvent 3.5–3.10.

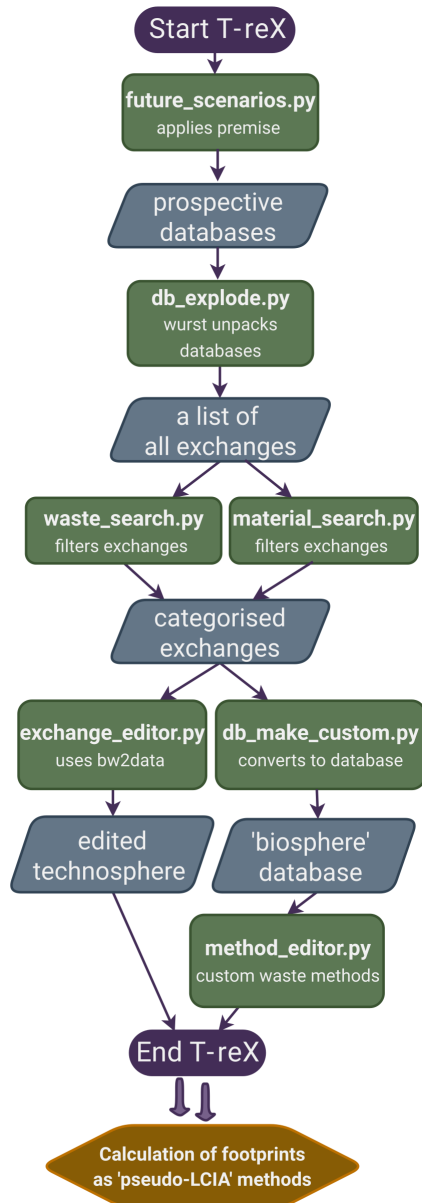
For more information on the background and motivation of T-reX, see the manuscript on its [Github repo](#).

The T-reX tool for LCA

a. Generalised method



b. Computational method



INSTALLATION

2.1 Dependencies

The program is written in Python and the required packages are listed in the *requirements.txt* file. These should be installed automatically when installing the program.

The main dependencies are:

- brightway2 components: bw2io, bw2data, bw2calc
- premise
- wurst

2.2 Installation Instructions

It is recommended to use a fresh virtual environment to install the program.

You can simply clone or download the repo and run it without installation:

```
python venv -m <name>  
source <name>/bin/activate
```

This will not install any of the dependencies, so you will need to install them manually if you don't already have them.

```
pip install -r requirements.txt
```

Alternatively: the program can be installed using pip:

Either from PyPI:

```
pip install T_reX_LCA
```

Or, probably best to install the latest version from GitHub:

```
pip install git+https://github.com/Stew-McD/T-reX.git
```

Or for an editable install (good for development and testing):

```
git clone https://github.com/Stew-McD/T-reX.git  
cd T-reX  
pip install -e .
```


USAGE

The program can be used directly from the command line, or imported as a Python module. This will run the program using the default settings. See the configuration section for more information on how to change the settings.

3.1 Command Line

You should clone the repo, navigate to the T-reX folder, and then run the program using:

```
python src/T-reX/main.py
```

Configuration files can be found in *src/T-reX/config/*. These can be edited before running the main script.

3.2 Python Module

The program can be imported as a Python module if it has been installed with pip. The main function can then be run using:

```
import T-reX as T-reX
T-reX.run()
```

As with the command line, the configuration files can be found in *src/T-reX/config/*. These can be edited before running the main script. It is also possible to edit the configuration settings directly in the Python script, and accessed in interactive terminal sessions like iPython and Jupyter. For example:

```
>>> import T-reX as T-reX

>>> T-reX.user_settings.use_premise
True
>>> T-reX.user_settings.use_T-reX
False
>>> T-reX.user_settings.use_T-reX = True
>>> T-reX.user_settings.use_T-reX
True
>>> T-reX.user_settings.project_base
'SSP-cutoff'
>>> T-reX.user_settings.project_base = "other project"
>>> T-reX.user_settings.project_base
'other project'
```

The individual modules can also be imported and used separately. For example:

```
.. import T-reX as T-reX

.. # only use premise
.. T-reX.FutureScenarios.MakeFutureScenarios()

.. # only do waste or material searches
.. database = 'my database'
.. project = 'my project'
.. T-reX.ExplodeDatabase(project, database)
.. T-reX.SearchWaste(project, database)
.. T-reX.SearchMaterial(project, database)

.. # check databases

.. T-reX.VerifyDatabase(project, database)

.. # or with the internal settings:

.. database = T-reX.user_settings.database_name

.. # check original database
.. project_base = T-reX.user_settings.project_base
.. T-reX.VerifyDatabase(project_base, database)
.. # (this will return '0', because it was not edited)

.. # check final database
.. project = T-reX.user_settings.project_T-reX
.. T-reX.VerifyDatabase(project, database)
.. # (this will return '1', if it was edited correctly)
```

CONFIGURATION MODULES

4.1 CustomConfig Module

This module provides functions for managing the configuration of the T-reX package. This is only needed if you have installed the package from PyPi, if you have cloned the repository you can just edit the configuration files directly.

By running the following command, the program will create a folder `config` in the current working directory containing the default configuration files:

```
T-reX.config.config_setup()
```

You can then edit these files to match your project structure and your needs. And then run the following command to reload the configuration:

```
T-reX.config.config_reload()
```

You will then need to restart your python session for the changes to take effect. If you want to reset the configuration to the default values, you can run:

```
T-reX.config.config_reset()
```

4.2 General Settings: `user_settings.py`

This is the main configuration file, the one that you might want to edit to match your project structure and your needs. By default, the program will take a brightway2 project named `default` and copy that to a new project named `SSP-cutoff`, which is then copied to a new project named `T_reXootprint-SSP-cutoff`.

Doing it this way isolates the components and allows you to keep your original brightway2 project as it was. If space is an issue, you can set all of the project names to be the same.

If you are happy with the default settings, you can just run the program and it will create the databases for you. If you want to change the settings, you can edit the `user_settings.py` file that you can find in the `config` directory of your working directory.

These are some extracts from `user_settings.py` with the most important settings (the ones you might want to change) and their default values:

```
# Choose whether to use premise to create future scenario databases
use_premise = True
# Choose whether to use T-reX to edit the databases (you could also turn this off and
→ just use the package as an easy way to make a set of future scenario databases)
```

(continues on next page)

(continued from previous page)

```
use_T_reX = True

# Choose the names of the projects to use
project_premise_base = "default"
project_premise = "SSP-cutoff"
project_base = project_premise
project_T_reX = f"T_reXootprint-{project_base}"

# Choose the name of the database to use (needed for premise only, the T-reX tool will
↳run all databases except the biospheres)
database_name = "ecoinvent-3.9.1-cutoff"

# if you want to use a fresh project
delete_existing_premise_project = False
delete_existing_T_reX_project = False

# Choose the premise scenarios to generate (see FutureScenarios.py for more details)
# Not all combinations are available, the code in FutureScenarios.py will filter out the
↳scenarios that are not possible
# the default is to have an optimistic and a pessimistic scenario with SSP2 for 2030,
↳2065 and 2100

models = ["remind"]
ssps = ["SSP2"]
rcps = ["Base", "PkBudg500"]
years = [2030, 2065, 2100, ]
```

4.3 Waste Search Settings: queries_waste.py

This file sets up search parameters for different T-reX flow categories, crucial for the SearchWaste.py script. It leverages a .pickle file created by ExplodeDatabase.py.

4.3.1 Categories

Handles various categories like digestion, composting, incineration, recycling, landfill, etc.

4.3.2 Query Types

Two sets of queries are created:

1. queries_kg for waste flows in kilograms.
2. queries_m3 for waste flows in cubic meters.

4.3.3 Adjusting Search Terms

- **Search Keywords:** Tweak the AND, OR, NOT lists to refine your search.

4.3.4 Category-Specific Changes

- **Adding Categories:** You can add new categories to the `names` list.
- **Modifying Queries:** Update the query parameters for each category based on your requirements.

4.3.5 Optimising Search Efficiency

You can choose to include or exclude whatever you want. For instance, “non-hazardous” is not included as it’s derivable from other categories and slows down the process.

4.3.6 Validating Search Terms

Isolate the function of `SearchWaste.py` to validate your search terms. That means, turning off the other functions in `user_settings.py`, or running the module directly. You can achieve this by setting the following in `user_settings.py`:

```
use_premise = False
do_search = True
do_methods = False
do_edit = False
```

4.4 Material Search Settings: `queries_materials.py`

The `queries_materials` module creates demand methods in the T-reX tool. It aligns with the EU CRM list 2023 and the ecoinvent database, incorporating additional strategic materials for comprehensive analysis. More can be easily added, as wished by the user.

This function uses the string tests `startswith` in `SearchMaterial.py` to identify activities beginning with the specified material name. This allows one to be more specific with the search terms (the `,` can be critical sometimes).

4.4.1 Structure and Customisation

Tuple Structure

- **First Part (Activity Name):** Specifies the exact activity in the database (e.g., `market for chromium`).
- **Second Part (Material Category):** Aggregates related activities under a common category (e.g., `chromium`), enhancing data processing efficiency.

Customisation Options

- **Add or Remove Materials:** Adapt the tuple list by including new materials or removing irrelevant ones.
- **Refine Search Terms:** Update material categories for a better fit with your database, ensuring precision in naming, especially with the use of commas.

4.4.2 Usage Considerations

- **Material Quantity:** The current list comprises over 40 materials. Modify this count to suit your project's scope.
- **Database Alignment:** Check that the material names correspond with your specific database version, likeecoinvent v3.9.1.

Example Tuples

- ("market for chromium", "chromium")
- ("market for coal", "coal")
- ("market for cobalt", "cobalt")
- ("market for coke", "coke")
- ("market for copper", "copper")
- ("market for tap water", "water")
- ("market for water,", "water")

T-REX MODULES

This section provides an overview of the various modules within the T-reX program, each with a brief description and a link to its detailed documentation.

- **`FutureScenarios`**: Contains scenarios for future environmental impact assessments.
- **`ExplodeDatabase`**: Responsible for expanding a Brightway2 database into detailed exchange lists.
- **`SearchWaste`**: Provides functions for searching and categorizing waste-related data.
- **`SearchMaterial`**: Enables detailed search and categorization of material-related data within a database.
- **`MakeCustomDatabase`**: Facilitates the creation of custom databases for use in environmental impact assessments.
- **`MethodEditor`**: Manages methods within the T-reX program for waste and material footprint calculations.
- **`ExchangeEditor`**: Handles the editing and appending of exchanges in Brightway2 databases.
- **`VerifyDatabase`**: Performs verification of databases by calculating LCA scores for random activities.

5.1 FutureScenarios

The *FutureScenarios* module is an integral part of the T-reX project, enabling the creation of future scenario databases. It leverages the [premise](#) Python package to generate future scenario databases based on integrated assessment models (IAMs) for brightway2 projects.

5.1.1 Configuration

Before using the *FutureScenarios* module, it's crucial to set up your desired scenarios in the *user_settings.py* file. This configuration determines the specific scenarios the module will process.

Configuring Projects and Premise Settings

- **Project Settings**: Define the base and target projects for the scenarios. For instance, *project_premise_base* as the source, and *project_premise* as the destination project.
- **Database Selection**: Specify the database to use, like *ecoinvent-3.9.1-cutoff*.
- **Premise Key**: A key is required for accessing premise functionality. It can be hardcoded or read from a file.
- **Multiprocessing and Batch Size**: Decide on using multiprocessing and the size of batches to process scenarios.
- **Deletion Settings**: Choose whether to delete existing projects before creating new ones.

Selecting Scenarios

Define the models, SSPs (Shared Socioeconomic Pathways), RCPs (Representative Concentration Pathways), and years you want to explore. The script will then attempt to create databases for each combination of these parameters, provided they are available.

5.1.2 Implementation

Module Overview

The *FutureScenarios* module consists of several key functions and a main script that orchestrates the entire process of creating future scenario databases.

Key Functions

- **make_possible_scenario_list**: Generates a list of feasible scenarios based on user preferences and available data.
- **check_existing**: Checks if a scenario already exists in the project to avoid redundancy.
- **FutureScenarios**: The core function that invokes the *premise* module to create new databases for each specified scenario.
- **MakeFutureScenarios**: The main function that calls *FutureScenarios* if the *use_premise* flag is set to True.

Process Flow

1. **Initialisation**: The script starts by importing necessary libraries and user settings. It sets up logging and changes the working directory if necessary.
2. **Scenario Filtering**: It filters out unavailable or existing scenarios.
3. **Project Preparation**: Depending on user settings, it either uses an existing project or creates a new one.
4. **Scenario Processing**: For each scenario, it calls *premise* to update or create a database reflecting that scenario.
5. **Database Writing**: After processing, the script writes the new databases to Brightway2.
6. **Cleanup and Conclusion**: The script concludes by adding GWP factors and returning to the original directory.

5.2 ExplodeDatabase

The ExplodeDatabase module is the first fundamental component of the T-reX tool, designed for processing Brightway2 life cycle assessment (LCA) databases. It provides a mechanism to expand a given LCA database into a detailed list of all exchanges, facilitating subsequent analysis with the search modules.

5.2.1 Primary Function

The module's core functionality is to 'explode' a Brightway2 database. This term refers to the process of decomposing the database into a single-level, comprehensive list of all exchanges. Each exchange in the database is extracted and documented, including detailed attributes such as name, amount, unit, product, production volume, type, and location.

5.2.2 Process Overview

1. Utilising the wurst library, the module opens the specified Brightway2 database.
2. It extracts information from each process in the database.
3. The extracted data is converted into a pandas DataFrame for ease of analysis and manipulation.
4. The module then expands ('explodes') this DataFrame to detail each exchange individually.
5. The resulting data, now a comprehensive list of exchanges, is saved in a .pickle binary file for efficient storage and retrieval.

5.2.3 Usage

The ExplodeDatabase function is invoked with a single argument, the name of the Brightway2 database to be processed. It performs the exploding process, logs the operation, and saves the resulting DataFrame as a pickle file. The function is designed for internal use within the T-reX tool and does not return a value but rather saves the output for subsequent use.

5.3 SearchWaste

The SearchWaste module is a part of the T-reX tool, dedicated to processing waste-related data. It loads data from a specified '< db name >_exploded.pickle' file, executes predefined search queries on this data, and generates CSV files containing the results along with corresponding log entries. The search queries are structured as dictionaries, specified in the *config/queries_waste.py* file, and include fields such as NAME, CODE, and search terms like keywords_AND, keywords_OR, and keywords_NOT.

5.3.1 Functionality

The module provides the `SearchWaste()` function, which is responsible for three main actions:

1. Loading data from the '< db name >_exploded.pickle' file.

```
pickle_path = os.path.join(dir_tmp, db_name + "_exploded.pickle")
df = pd.read_pickle(pickle_path)
```

2. Running the specified search queries on this data. These queries are designed to filter and identify relevant waste exchanges based on specific criteria.

This functionality is implemented in the `search()` function, which is a subfunction of the `SearchWaste()` function. The `search()` function takes one argument, a search query from the list that is produced by the configuration module *queries_waste.py*.

The search function is applied as follows, where *df* is the dataframe of the exploded database and *query* is a dictionary:

```
NAME_BASE = query["name"]
UNIT = query["unit"]
NAME = NAME_BASE + "-" + UNIT
CODE = NAME.replace(" ", "")
query.update({"code": NAME, "db_name": db_name})
AND = query["AND"]
OR = query["OR"]
NOT = query["NOT"]
DBNAME = query["db_name"]

# Apply the search terms to the dataframe
df_results = df[
    (df["ex_name"].apply(lambda x: all(i in x for i in AND)))
    & (df["ex_unit"] == UNIT)
    & (df["ex_amount"] != 0)
    # & (df["ex_amount"] != -1)
].copy()

# Apply OR and NOT search filters
if OR:
    df_results = df_results[
        df_results["ex_name"].apply(lambda x: any(i in x for i in OR))
    ]
if NOT:
    df_results = df_results[
        df_results["ex_name"].apply(lambda x: not any(i in x for i in NOT))
    ]
```

3. Producing CSV files to store the results of these queries and creating log entries for each search operation. When customising the search configuration, it is important to check these files to see that the correct exchanges are being captured. The files are used by the subsequent modules to edit the exchanges and to produce LCIA methods.

5.3.2 Usage

```
T-reX.SearchWaste(db_name, output_dir)
```

The `SearchWaste()` function is invoked with two arguments: the name of the Brightway2 database to be processed and the name of the directory to store the results. The search queries are specified in the `config/queries_waste.py` file. The function is designed for internal use within the T-reX tool and does not return a value but rather saves the output for subsequent use. It could be used separately, if you would have a .pickle file with exploded database as well as the config files in the right locations.

5.4 SearchMaterial

The SearchMaterial module's primary function is to load data from '< db name >_exploded.pickle', execute search queries based on the list of materials, and store the results in a CSV file along with a corresponding log entry. The search queries are formatted as tuples, where the first entry is the name of the activity and the second is the material grouping, e.g. ("market for sodium borates", "borates"). These queries are defined by the list in *queries_materials.py*, which can be easily modified by the user to change the scope or the groupings as desired.

5.4.1 Functionality

The module's core function, `SearchMaterial()`, takes a database name and a Brightway2 project as inputs. It searches for materials within the specified database, extracts relevant information such as ISIC and CPC classifications, and saves this information in CSV format. It also extracts and saves related material exchanges and saves them as separate files based on the material grouping. The files are used by the subsequent modules to edit the exchanges and to produce LCIA methods.

5.4.2 Search queries

The module *queries_material.py* contains the list of materials used to create demand methods in the T-reX tool. The materials are chosen to match activities in the ecoinvent v3.9.1 database and future databases made with premise, and to align with the EU CRM list 2023. Additional strategic materials are included, along with some other materials of interest.

NOTE: The search function in *SearchMaterial.py* uses *str.startswith*, finding all activities that start with the material name.

```
# Filter activities based on the materials list
acts_all = pd.DataFrame([x.as_dict() for x in db])
materials_df = pd.DataFrame(queries_materials, columns=["name", "group"])

acts = acts_all[
    acts_all["name"].apply(
        lambda x: any(x.startswith(material) for material in materials_df.name)
    )
].reset_index(drop=True)
```

The comma in material names is crucial for filtering specific activities in the database. For example, with the comma, the query *market for gold*, will catch the activity *market for nickel* and *market for nickel, 90%* but not *market for nickel plating*.

```
queries_materials = [
    ("market for aluminium", "aluminium"),
    ("market for antimony", "antimony"),
    # ... [rest of the list] ...
    ("market for zirconium", "zirconium"),
]
```

5.5 MakeCustomDatabase

The MakeCustomDatabase module's function is to facilitate the creation and integration of custom databases into the Brightway2 project. This module is responsible for generating an `xlsx` representation of a Brightway2 database based on the output of the modules *SearchWaste* and *SearchMaterial* and importing it into as a database with *bwio* from the Brightway2 framework.

5.5.1 Main Functions

- **dbWriteExcel**: This function creates an `xlsx` file that represents a custom Brightway2 database, using data from predefined directories and database settings.
- **dbExcel2BW**: This function imports the custom database, created by `dbWriteExcel`, into Brightway2, making it available for the *ExchangeEditor* and *MethodEditor* modules.

5.5.2 Important Code Snippets

- **Function to Collect Filenames from Directories:**

```
def get_files_from_tree(dir_searchmaterial_results, dir_searchwaste_results):  
    # Implementation of the function
```

- **Function to Create Excel File for Custom Database:**

```
def dbWriteExcel():  
    #.... snippet to show structure of the activities in the database ...  
    for NAME in names:  
        count += 1  
        CODE = NAME  
        UNIT = determine_unit_from_name(NAME)  
  
        if "Waste" in NAME:  
            TYPE = "waste"  
            CODE = (  
                NAME.replace("WasteFootprint_", "")  
                .capitalize()  
                .replace("kilogram", "(kg)")  
                .replace("cubicmeter", "(m3)")  
                .replace("-", " ")  
            )  
        elif "Material" in NAME:  
            TYPE = "natural resource"  
            CODE = NAME.replace("MaterialFootprint_", "").capitalize()  
        else:  
            TYPE = "?"  
  
        db_entry = {  
            "Activity": NAME,  
            "categories": "water, air, land",  
            "code": CODE,  
            "unit": UNIT,  
            "type": TYPE,
```

(continues on next page)

(continued from previous page)

```
}
# ... ..
```

- **Function to Determine Unit from Name:**

```
def determine_unit_from_name(name):
    # Implementation of the Function
```

- **Function to Import Database into Brightway2:**

```
def dbExcel2BW():
    # Implementation of the Function
```

5.6 MethodEditor

The MethodEditor module is a key component of the T-reX tool, offering functionalities to manage methods related to waste and material footprints in a project. This module is instrumental in adding, deleting, and checking methods within the context of environmental impact assessments.

5.6.1 Function Summary

- **`AddMethods`**: This function adds new methods to a project based on entries in a custom biosphere database. It is useful for incorporating specific waste and material footprint methodologies into the project's analysis framework.
- **`DeleteMethods`**: This function removes specific methods from a project, particularly those associated with waste and material footprints, aiding in maintaining the relevance and accuracy of the project's methodological toolbox.
- **`CheckMethods`**: This function lists and checks the methods in a project, focusing on those linked to waste and material footprints, ensuring that the methods are correctly implemented and aligned with the project's objectives.

5.6.2 Important Code Snippets

- **Function to Add Methods to a Project:**

```
def AddMethods():
    # ... snippet to show structure of method keys added ...

    # Assign characterization factor based on type
    ch_factor = -1.0 if m_type == "waste" else 1.0

    # Assign method key and description based on type
    if m_type == "waste":
        name_combined = m_code.split(" ")[0] + " combined"
        method_key = (
            "T-reX",
            "Waste: " + name_combined,
            m_code,
```

(continues on next page)

(continued from previous page)

```

    )
    description = "For estimating the waste footprint of an activity"
else:
    method_key = ("T-reX", "Demand: " + m_code, m_code)
    description = "For estimating the material demand footprint of an activity"

# ... end snippet ...

```

- **Function to Delete Methods from a Project:**

```

def DeleteMethods():
    # Implementation of the function

```

- **Function to Check Methods in a Project:**

```

def CheckMethods():
    # ... snippet to show how methods are accessed ...

    methods_T_reX = [
        x for x in list(bd.methods) if "WasteAndMaterial Footprint" == x[0]
    ]

    for m in methods_T_reX:
        method = bd.Method(m)
        print(method.load())
        print(method.metadata)

    print(len(methods_T_reX))

# ... end snippet ...

```

5.7 ExchangeEditor

The ExchangeEditor module is the final major part of the T-reX tool, responsible for editing exchanges with wurst and Brightway2. It appends relevant exchanges from the *db_T_reX* (a database containing T-reX exchange details) to activities identified by *WasteAndMaterialSearch()* in a specified project's database (*db_name*).

This module takes the longest time to run, as it iterates over each exchange that was found by the search modules, which can be >200,000 exchanges. For each database, this can take around 15-20 minutes to run.

5.7.1 Function Summary

- **`ExchangeEditor`:** This function modifies the specified project's database by appending exchanges from the *db_T_reX* to activities identified by *WasteAndMaterialSearch()*. Each appended exchange replicates the same amount and unit as the original technosphere waste and material exchange.

5.7.2 Important Code Snippets

- Function to Append Exchanges to Activities in a Database:

```
def ExchangeEditor(project_T_reX, db_name, db_T_reX_name):
    # ... snippet to show addition of the custom exchange ...

    # Iterate over each category (NAME)
    for NAME, df in sorted(file_dict.items(), reverse=False):
        countNAME += 1
        progress_db = f"{countNAME:2}/{len(file_dict.items())}"
        count = 0

        # For each exchange in the current category's DataFrame
        for exc in tqdm(
            df.to_dict("records"),
            desc=f" - {progress_db} : {NAME} ",
            bar_format=bar_format,
            colour="magenta",
            smoothing=0.01,
        ):
            # Extract details of the exchange
            code, name, location, ex_name, amount, unit, ex_location, database = (
                exc["code"],
                exc["name"],
                exc["location"],
                exc["ex_name"],
                exc["ex_amount"],
                exc["ex_unit"],
                exc["ex_location"],
                db_name,
            )

            KEY = (database, code)
            T_reX_KEY = (
                db_T_reX_name,
                NAME.split("_")[1]
                .capitalize()
                .replace("_", " ")
                .replace("-", " ")
                .replace("kilogram", "(kg)")
                .replace("cubicmeter", "(m3)"),
            )

            # Retrieve the process and T_reX exchange from the databases
            try:
                process = bd.get_activity(KEY)
                T_reX_ex = bd.get_activity(T_reX_KEY)
                before = len(process.exchanges())

                process.new_exchange(
                    input=T_reX_ex,
                    amount=amount,
                    unit=unit,
```

(continues on next page)

(continued from previous page)

```

        type="biosphere",
    ).save()
    after = len(process.exchanges())

# ... end of snippet ...

```

5.8 VerifyDatabase

The VerifyDatabase module contains functionality for verifying an edited T-reX database within a given project in Brightway2. It performs this verification by calculating LCA scores for random activities within the specified database using selected methods.

5.8.1 Function Summary

- **VerifyDatabase**: This function verifies a database within a Brightway2 project by calculating LCA scores for random activities using Waste Footprint and Material Demand Footprint methods. Since it is not expected that every activity and method combination would result in a non-zero score, the function allows users to specify the number of activities and methods to be used in the verification process. The function also allows users to specify whether they want to check the Waste Footprint, Material Demand Footprint, or both.

```

def VerifyDatabase(project_name, database_name, check_material=True, check_
    waste=True, log=True):
    # ... snippet to show verification process ... #

    print(f"\n** Verifying database {database_name} in project {project_name} **\n")

    # Loop until a non-zero score is obtained
    while lca_score == 0 and count < 5:
        try:
            count += 1
            # Get a random activity from the database
            act = bd.Database(database_name).random()

            # Initialize the list of methods
            methods = []

            # Find methods related to Waste Footprint
            if check_waste:
                methods_waste = [x for x in bd.methods if "Waste" in x[1]]
                methods += methods_waste

            # Find methods related to Material Demand Footprint
            if check_material:
                methods_material = [x for x in bd.methods if "Demand" in x[1]]
                methods += methods_material

            if not check_waste and not check_material:
                method = bd.methods.random()
                methods.append(method)

```

(continues on next page)

(continued from previous page)

```

# Choose a random method
method = choice(methods)

# Perform LCA calculation
lca = bc.LCA({act: 1}, method)
lca.lci()
lca.lcia()

# Get the lca score
lca_score = lca.score

# Print the result
log_statement = f"\tScore: {lca_score:2.2e} \n\tMethod: {method[2]} \n\
↪\tActivity: {act['name']} \n\tDatabase: {database_name}\n"

except Exception as e:
    # Print any errors that occur
    log_statement = (
        f"@@@@@@@@ Error occurred with '{database_name}': {e}! @@@@@@"
    )

# ... end snippet ... #

```

5.8.2 Usage

This function is called automatically after each database is processed, and again after all databases have been processed. The function can also be called manually by the user by invoking the following command:

```

T-reX.VerifyDatabase(project_name, database_name, check_material=True, check_waste=True, ↪
↪log=True)

```


EXAMPLES

See the *examples* directory for an examples of how to use the T-reX package. The folder *batteries* contains a small case study using the package to calculate the f and material footprints of several battery technologies in ecoinvent 3.10.

EXAMPLE OUTPUT

[View the example output here](#)

There is also an .ipynb and .pdf version of the example output in the *examples/package_test* directory of the repository.

SOURCE CODE

The source code is available [here](#).

API REFERENCE

This page contains auto-generated API reference documentation¹.

9.1 T-reX

9.1.1 Subpackages

T-reX.config

Submodules

T-reX.config.queries_materials

This module contains the list of materials that will be used to create demand methods in the T-reX tool. The materials are specifically chosen to match with the activities in the ecoinvent v3.9.1 database (and the future databases made with premise) and to align with the EU CRM list 2023. Additional strategic materials are also included, as well as other materials of interest.

Note:

- The search function in *SearchMaterial.py* uses ‘startswith’, so it will catch all activities that start with the material name.
- The comma in material names is important for filtering specific activities in the database.

queries_materials

List of tuples where each tuple contains the activity name (as appears in the database) and its corresponding material category. This list is used for creating demand methods in the T-reX tool.

¹ Created with sphinx-autoapi

Module Contents

Functions

<code>get_queries_materials(list_materials)</code>	Just a wrapper to return the list of materials
----------------------------------------------------	------------------------------------------------

Attributes

<code>list_materials</code>
<code>queries_materials</code>

```
T-reX.config.queries_materials.list_materials = [('market for aluminium', 'aluminium'),  
('market for antimony', 'antimony'), ('market for...
```

```
T-reX.config.queries_materials.get_queries_materials(list_materials)
```

Just a wrapper to return the list of materials

args: list_materials returns: list queries_materials

```
T-reX.config.queries_materials.queries_materials
```

T-reX.config.queries_waste

This module defines the search parameters for each waste and material flow category. It is used in conjunction with *SearchWaste.py* and requires a .pickle file generated by *ExplodeDatabase.py*.

The queries are set up for different waste flow categories like digestion, composting, incineration, recycling, and landfill, among others. Each query is a dictionary containing search terms for the respective category.

The module creates two sets of queries: 1. *queries_kg* for waste flows measured in kilograms. 2. *queries_m3* for waste flows measured in cubic meters.

These queries are combined into the *queries_waste* list, which is then used by *SearchWaste.py* for filtering and extracting relevant data from the database.

Module Contents

Functions

<code>make_queries_waste()</code>	This function creates the queries_waste list, which is used by SearchWaste.py for filtering and extracting relevant data from the database.
-----------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------

Attributes

`queries_waste`

`T-reX.config.queries_waste.make_queries_waste()`

This function creates the `queries_waste` list, which is used by `SearchWaste.py` for filtering and extracting relevant data from the database.

The queries are set up for different waste flow categories like digestion, composting, incineration, recycling, and landfill, among others. Each query is a dictionary containing search terms for the respective category.

args: None returns: list `queries_waste`

`T-reX.config.queries_waste.queries_waste`

`T-reX.config.user_settings`

Configure Project and Database Settings for T-reX.

This script is used to configure the project and database settings, as well as set up the essential paths for the data, config, and result directories.

The script allows for two modes of operation:

1. Single Mode (*single* is True): Set the project and database names to a single specified value.
2. Multiple projects/databases mode (*single* is False): Facilitates batch processing of multiple projects and databases.

Additionally, the script allows for the use of multiprocessing (*use_multiprocessing* is True).

Premise can also be used to make future databases (*use_premise* is True).

Module Contents

Functions

`generate_args_list([single_database])`

Generate a list of argument dictionaries for processing multiple projects and databases.

Attributes

`custom_bw2_dir`

`project_premise_base`

`project_premise`

continues on next page

Table 1 – continued from previous page

<i>project_base</i>
<i>project_T_reX</i>
<i>use_T_reX</i>
<i>db_T_reX_name</i>
<i>single</i>
<i>single_database</i>
<i>delete_T_reX_project</i>
<i>use_multiprocessing</i>
<i>verbose</i>
<i>do_search</i>
<i>do_methods</i>
<i>do_edit</i>
<i>use_premise</i>
<i>database_name</i>
<i>delete_existing_premise_project</i>
<i>use_mp</i>
<i>batch_size</i>
<i>premise_quiet</i>
<i>project_T_reX</i>
<i>premise_key</i>
<i>key_path</i>
<i>models</i>
<i>ssps</i>
<i>rcps</i>
<i>years</i>
<i>desired_scenarios</i>

continues on next page

Table 1 – continued from previous page

<i>cwd</i>
<i>dir_config</i>
<i>dir_data</i>
<i>dir_tmp</i>
<i>dir_logs</i>
<i>dir_searchwaste_results</i>
<i>dir_searchmaterial_results</i>
<i>dir_databases_T_reX</i>
<i>dir_T_reX</i>

T-reX.config.user_settings.custom_bw2_dir

T-reX.config.user_settings.project_premise_base = 'default'

T-reX.config.user_settings.project_premise = 'premise-SSP2-cutoff'

T-reX.config.user_settings.project_base

T-reX.config.user_settings.project_T_reX

T-reX.config.user_settings.use_T_reX = True

T-reX.config.user_settings.db_T_reX_name = 'biosphere_T-reX'

T-reX.config.user_settings.single = False

T-reX.config.user_settings.single_database = 'ecoinvent_cutoff_3.9_remind_SSP2-Base_2065'

T-reX.config.user_settings.delete_T_reX_project = True

T-reX.config.user_settings.use_multiprocessing = False

T-reX.config.user_settings.verbose = False

T-reX.config.user_settings.do_search = True

T-reX.config.user_settings.do_methods = True

T-reX.config.user_settings.do_edit = True

T-reX.config.user_settings.use_premise = True

T-reX.config.user_settings.database_name = 'ecoinvent-3.9.1-cutoff'

T-reX.config.user_settings.delete_existing_premise_project = False

T-reX.config.user_settings.use_mp = True

```
T-reX.config.user_settings.batch_size = 3
T-reX.config.user_settings.premise_quiet = True
T-reX.config.user_settings.project_T_reX
T-reX.config.user_settings.premise_key = 'tUePmX_S5B8ieZkkM7WUU2Cn08SmShwmAeWK9x2rTFo='
T-reX.config.user_settings.key_path
T-reX.config.user_settings.models = ['remind']
T-reX.config.user_settings.ssps = ['SSP2']
T-reX.config.user_settings.rcps = ['Base', 'PkBudg500']
T-reX.config.user_settings.years = [2020, 2025, 2030, 2035, 2040, 2045, 2050, 2055, 2060,
2065, 2070, 2075, 2080, 2085, 2090, 2095, 2100]
T-reX.config.user_settings.desired_scenarios = []
```

T-reX.config.user_settings.generate_args_list(*single_database=None*)

Generate a list of argument dictionaries for processing multiple projects and databases.

This function is used when the tool is set to operate in multiple projects/databases mode. It generates a list of argument dictionaries, each containing project and database settings.

Returns

A list of dictionaries with project and database settings for batch processing.

```
T-reX.config.user_settings.cwd
T-reX.config.user_settings.dir_config
T-reX.config.user_settings.dir_data
T-reX.config.user_settings.dir_tmp
T-reX.config.user_settings.dir_logs
T-reX.config.user_settings.dir_searchwaste_results
T-reX.config.user_settings.dir_searchmaterial_results
T-reX.config.user_settings.dir_databases_T_reX
T-reX.config.user_settings.dir_T_reX
```

Package Contents

Functions

<code>generate_args_list([single_database])</code>	Generate a list of argument dictionaries for processing multiple projects and databases.
----------------------------------------------------	------------------------------------------------------------------------------------------

Attributes

custom_bw2_dir

db_T_reX_name

delete_T_reX_project

dir_logs

dir_tmp

project_base

project_premise

project_T_reX

use_multiprocessing

use_premise

use_T_reX

do_search

do_methods

do_edit

single_database

queries_waste

queries_materials

`T-reX.config.custom_bw2_dir`

`T-reX.config.db_T_reX_name = 'biosphere-T-reX'`

`T-reX.config.delete_T_reX_project = True`

`T-reX.config.dir_logs`

`T-reX.config.dir_tmp`

`T-reX.config.generate_args_list(single_database=None)`

Generate a list of argument dictionaries for processing multiple projects and databases.

This function is used when the tool is set to operate in multiple projects/databases mode. It generates a list of argument dictionaries, each containing project and database settings.

Returns

A list of dictionaries with project and database settings for batch processing.

```
T-reX.config.project_base
T-reX.config.project_premise = 'premise-SSP2-cutoff'
T-reX.config.project_T_reX
T-reX.config.use_multiprocessing = False
T-reX.config.use_premise = True
T-reX.config.use_T_reX = True
T-reX.config.do_search = True
T-reX.config.do_methods = True
T-reX.config.do_edit = True
T-reX.config.single_database = 'ecoinvent_cutoff_3.9_remind_SSP2-Base_2065'
T-reX.config.queries_waste
T-reX.config.queries_materials
```

T-reX.config_backup

Submodules

T-reX.config_backup.queries_materials

This module contains the list of materials that will be used to create demand methods in the T-reX tool. The materials are specifically chosen to match with the activities in the ecoinvent v3.9.1 database (and the future databases made with premise) and to align with the EU CRM list 2023. Additional strategic materials are also included, as well as other materials of interest.

Note:

- The search function in *SearchMaterial.py* uses 'startswith', so it will catch all activities that start with the material name.
- The comma in material names is important for filtering specific activities in the database.

queries_materials

List of tuples where each tuple contains the activity name (as appears in the database) and its corresponding material category. This list is used for creating demand methods in the T-reX tool.

Module Contents

Functions

<code>get_queries_materials(list_materials)</code>	Just a wrapper to return the list of materials
----------------------------------------------------	------------------------------------------------

Attributes

<code>list_materials</code>
<code>queries_materials</code>

`T-reX.config_backup.queries_materials.list_materials = [('market for aluminium', 'aluminium'), ('market for antimony', 'antimony'), ('market for...`

`T-reX.config_backup.queries_materials.get_queries_materials(list_materials)`

Just a wrapper to return the list of materials

args: list_materials returns: list queries_materials

`T-reX.config_backup.queries_materials.queries_materials`

T-reX.config_backup.queries_waste

This module defines the search parameters for each waste and material flow category. It is used in conjunction with *SearchWaste.py* and requires a .pickle file generated by *ExplodeDatabase.py*.

The queries are set up for different waste flow categories like digestion, composting, incineration, recycling, and landfill, among others. Each query is a dictionary containing search terms for the respective category.

The module creates two sets of queries: 1. *queries_kg* for waste flows measured in kilograms. 2. *queries_m3* for waste flows measured in cubic meters.

These queries are combined into the *queries_waste* list, which is then used by *SearchWaste.py* for filtering and extracting relevant data from the database.

Module Contents

Functions

<code>make_queries_waste()</code>	This function creates the queries_waste list, which is used by SearchWaste.py for filtering and extracting relevant data from the database.
-----------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------

Attributes

`queries_waste`

`T-reX.config_backup.queries_waste.make_queries_waste()`

This function creates the `queries_waste` list, which is used by `SearchWaste.py` for filtering and extracting relevant data from the database.

The queries are set up for different waste flow categories like digestion, composting, incineration, recycling, and landfill, among others. Each query is a dictionary containing search terms for the respective category.

args: None returns: list `queries_waste`

`T-reX.config_backup.queries_waste.queries_waste`

`T-reX.config_backup.user_settings`

Configure Project and Database Settings for T-reX.

This script is used to configure the project and database settings, as well as set up the essential paths for the data, config, and result directories.

The script allows for two modes of operation:

1. Single Mode (*single* is True): Set the project and database names to a single specified value.
2. Multiple projects/databases mode (*single* is False): Facilitates batch processing of multiple projects and databases.

Additionally, the script allows for the use of multiprocessing (*use_multiprocessing* is True).

Premise can also be used to make future databases (*use_premise* is True).

Module Contents

Functions

`generate_args_list([single_database])`

Generate a list of argument dictionaries for processing multiple projects and databases.

Attributes

`custom_bw2_dir`

`project_premise_base`

`project_premise`

continues on next page

Table 2 – continued from previous page

<i>project_base</i>
<i>project_T_reX</i>
<i>use_T_reX</i>
<i>db_T_reX_name</i>
<i>single</i>
<i>single_database</i>
<i>delete_T_reX_project</i>
<i>use_multiprocessing</i>
<i>verbose</i>
<i>do_search</i>
<i>do_methods</i>
<i>do_edit</i>
<i>use_premise</i>
<i>database_name</i>
<i>delete_existing_premise_project</i>
<i>use_mp</i>
<i>batch_size</i>
<i>premise_quiet</i>
<i>project_T_reX</i>
<i>premise_key</i>
<i>key_path</i>
<i>models</i>
<i>ssps</i>
<i>rcps</i>
<i>years</i>
<i>desired_scenarios</i>

continues on next page

Table 2 – continued from previous page

<i>cwd</i>
<i>dir_config</i>
<i>dir_data</i>
<i>dir_tmp</i>
<i>dir_logs</i>
<i>dir_searchwaste_results</i>
<i>dir_searchmaterial_results</i>
<i>dir_databases_T_reX</i>
<i>dir_T_reX</i>

T-reX.config_backup.user_settings.custom_bw2_dir

T-reX.config_backup.user_settings.project_premise_base = 'default'

T-reX.config_backup.user_settings.project_premise = 'premise-SSP2-cutoff'

T-reX.config_backup.user_settings.project_base

T-reX.config_backup.user_settings.project_T_reX

T-reX.config_backup.user_settings.use_T_reX = True

T-reX.config_backup.user_settings.db_T_reX_name = 'biosphere_T-reX'

T-reX.config_backup.user_settings.single = False

T-reX.config_backup.user_settings.single_database =
'ecoinvent_cutoff_3.9_remind_SSP2-Base_2065'

T-reX.config_backup.user_settings.delete_T_reX_project = True

T-reX.config_backup.user_settings.use_multiprocessing = False

T-reX.config_backup.user_settings.verbose = False

T-reX.config_backup.user_settings.do_search = True

T-reX.config_backup.user_settings.do_methods = True

T-reX.config_backup.user_settings.do_edit = True

T-reX.config_backup.user_settings.use_premise = True

T-reX.config_backup.user_settings.database_name = 'ecoinvent-3.9.1-cutoff'

T-reX.config_backup.user_settings.delete_existing_premise_project = True

```

T-reX.config_backup.user_settings.use_mp = True
T-reX.config_backup.user_settings.batch_size = 9
T-reX.config_backup.user_settings.premise_quiet = True
T-reX.config_backup.user_settings.project_T_reX
T-reX.config_backup.user_settings.premise_key =
'tUePmX_S5B8ieZkkM7WUU2Cn08SmShwmAeWK9x2rTFo='
T-reX.config_backup.user_settings.key_path
T-reX.config_backup.user_settings.models = ['remind']
T-reX.config_backup.user_settings.ssps = ['SSP2']
T-reX.config_backup.user_settings.rcps = ['Base', 'PkBudg500']
T-reX.config_backup.user_settings.years = [2020, 2025, 2030, 2035, 2040, 2045, 2050,
2055, 2060, 2065, 2070, 2075, 2080, 2085, 2090, 2095, 2100]
T-reX.config_backup.user_settings.desired_scenarios = []

```

T-reX.config_backup.user_settings.generate_args_list(*single_database=None*)

Generate a list of argument dictionaries for processing multiple projects and databases.

This function is used when the tool is set to operate in multiple projects/databases mode. It generates a list of argument dictionaries, each containing project and database settings.

Returns

A list of dictionaries with project and database settings for batch processing.

```

T-reX.config_backup.user_settings.cwd
T-reX.config_backup.user_settings.dir_config
T-reX.config_backup.user_settings.dir_data
T-reX.config_backup.user_settings.dir_tmp
T-reX.config_backup.user_settings.dir_logs
T-reX.config_backup.user_settings.dir_searchwaste_results
T-reX.config_backup.user_settings.dir_searchmaterial_results
T-reX.config_backup.user_settings.dir_databases_T_reX
T-reX.config_backup.user_settings.dir_T_reX

```

Package Contents

Functions

<code>generate_args_list([single_database])</code>	Generate a list of argument dictionaries for processing multiple projects and databases.
----------------------------------------------------	------------------------------------------------------------------------------------------

Attributes

<code>custom_bw2_dir</code>
<code>db_T_reX_name</code>
<code>delete_T_reX_project</code>
<code>dir_logs</code>
<code>dir_tmp</code>
<code>project_base</code>
<code>project_premise</code>
<code>project_T_reX</code>
<code>use_multiprocessing</code>
<code>use_premise</code>
<code>use_T_reX</code>
<code>do_search</code>
<code>do_methods</code>
<code>do_edit</code>
<code>single_database</code>
<code>queries_waste</code>
<code>queries_materials</code>

`T-reX.config_backup.custom_bw2_dir`

`T-reX.config_backup.db_T_reX_name = 'biosphere_T-reX'`

`T-reX.config_backup.delete_T_reX_project = True`

T-reX.config_backup.dir_logs

T-reX.config_backup.dir_tmp

T-reX.config_backup.generate_args_list(*single_database=None*)

Generate a list of argument dictionaries for processing multiple projects and databases.

This function is used when the tool is set to operate in multiple projects/databases mode. It generates a list of argument dictionaries, each containing project and database settings.

Returns

A list of dictionaries with project and database settings for batch processing.

T-reX.config_backup.project_base

T-reX.config_backup.project_premise = 'premise-SSP2-cutoff'

T-reX.config_backup.project_T_reX

T-reX.config_backup.use_multiprocessing = False

T-reX.config_backup.use_premise = True

T-reX.config_backup.use_T_reX = True

T-reX.config_backup.do_search = True

T-reX.config_backup.do_methods = True

T-reX.config_backup.do_edit = True

T-reX.config_backup.single_database = 'ecoinvent_cutoff_3.9_remind_SSP2-Base_2065'

T-reX.config_backup.queries_waste

T-reX.config_backup.queries_materials

9.1.2 Submodules

T-reX.CustomConfig

Module Contents

Functions

<code>config_setup()</code>	Copies the config files to the current working directory and adds the config directory to sys.path
<code>config_reload()</code>	Copies the config files back to the package directory. The user will need to restart the python session to reload the T-reX module with the updated configuration files
<code>config_reset()</code>	Reset config to defaults that are stored in the root directory of the package

T-reX.CustomConfig.config_setup()

Copies the config files to the current working directory and adds the config directory to sys.path args: None
returns: None

T-reX.CustomConfig.config_reload()

Copies the config files back to the package directory. The user will need to restart the python session to reload the T-reX module with the updated configuration files

args: None returns: None

T-reX.CustomConfig.config_reset()

Reset config to defaults that are stored in the root directory of the package

args: None returns: None

T-reX.ExchangeEditor

This module is responsible for editing exchanges with wurst and Brightway2. It appends relevant exchanges from the *db_T_reX* (database containing waste and material exchange details) to activities identified by *WasteAndMaterialSearch()* in the specified project's database (*db_name*). Each appended exchange replicates the same amount and unit as the original technosphere waste and material exchange.

Module Contents**Functions**

<i>ExchangeEditor</i> (project_T_reX, db_T_reX_name)	db_name,	Append relevant exchanges from <i>db_T_reX</i> to each activity in <i>db_name</i> identified by <i>WasteAndMaterialSearch()</i> .
---------------------------------------------------------	----------	-----------------------------------------------------------------------------------------------------------------------------------

T-reX.ExchangeEditor.ExchangeEditor(project_T_reX, db_name, db_T_reX_name)

Append relevant exchanges from *db_T_reX* to each activity in *db_name* identified by *WasteAndMaterialSearch()*.

This function modifies the specified project's database by appending exchanges from the *db_T_reX* to activities identified by *WasteAndMaterialSearch()*. The appended exchanges mirror the quantity and unit of the original technosphere waste and material exchange.

Parameters

- **project_T_reX** (*str*) – Name of the Brightway2 project to be modified.
- **db_name** (*str*) – Name of the database within the project where activities and exchanges are stored.
- **db_T_reX_name** (*str*) – Name of the database containing waste and material exchange details.

Returns

None. Modifies the given Brightway2 project by appending exchanges and logs statistics about the added exchanges.

Return type

None

Raises

Exception – If any specified process or exchange is not found in the database.

T-reX.ExplodeDatabase

This module is responsible for exploding a Brightway2 database into a single-level list of all exchanges. It utilizes the wurst package to unpack the database, explode it to a list of all exchanges, and save this data in a DataFrame as a .pickle binary file.

Module Contents

Functions

<i>ExplodeDatabase</i> (db_name)	Explode a Brightway2 database into a single-level list of all exchanges using wurst.
----------------------------------	--------------------------------------------------------------------------------------

T-reX.ExplodeDatabase.**ExplodeDatabase**(db_name)

Explode a Brightway2 database into a single-level list of all exchanges using wurst.

Parameters

db_name (*str*) – Name of the Brightway2 database to be exploded.

Returns

None The function saves the output to a file and logs the operation, but does not return any value.

Return type

None

T-reX.FutureScenarios

This module is responsible for creating future databases with premise.

Module Contents

Functions

<i>make_possible_scenario_list</i> (filenames, ...)	Make a list of dictionaries with scenario details based on the available scenarios and the desired scenarios.
<i>check_existing</i> (desired_scenarios)	Check the project to see if the desired scenarios already exist, and if so, remove them from the list of scenarios to be created.
<i>FutureScenarios</i> (scenario_list)	Create future databases with premise.
<i>MakeFutureScenarios</i> ()	Main function to run the FutureScenarios module.

Attributes

dir_premise

T-reX.FutureScenarios.**dir_premise**

T-reX.FutureScenarios.**make_possible_scenario_list**(*filenames, desired_scenarios, years*)

Make a list of dictionaries with scenario details based on the available scenarios and the desired scenarios.

args: filenames (list): list of filenames of available scenarios
desired_scenarios (list): list of dictionaries with scenario details
years (list): list of years to be used

returns: scenarios (list): list of dictionaries with scenario details that are available and desired

T-reX.FutureScenarios.**check_existing**(*desired_scenarios*)

Check the project to see if the desired scenarios already exist, and if so, remove them from the list of scenarios to be created. Quite useful when running many scenarios, as it can take a long time to create them all, sometimes crashes, etc.

args: desired_scenarios (list): list of dictionaries with scenario details

returns: new_scenarios (list): list of dictionaries with scenario details that do not already exist in the project

T-reX.FutureScenarios.**FutureScenarios**(*scenario_list*)

Create future databases with premise.

This function processes scenarios and creates new databases based on the premise module. It configures and uses user-defined settings and parameters for database creation and scenario processing.

Returns

None The function does not return any value but performs operations to create and configure databases in Brightway2 based on specified scenarios.

Raises

Exception – If an error occurs during the processing of scenarios or database creation.

T-reX.FutureScenarios.**MakeFutureScenarios**()

Main function to run the FutureScenarios module. Only activated if *use_premise* is set to True in *user_settings.py*.

Calls the *FutureScenarios* function to create new databases based on the list of scenarios and settings specified in *user_settings.py*.

T-reX.MakeCustomDatabase

This module contains functions for creating an xlsx representation of a Brightway2 database and importing it into Brightway2.

Main functions: - dbWriteExcel: Creates an xlsx file representing a custom Brightway2 database. - dbExcel2BW: Imports the custom database (created by dbWriteExcel) into Brightway2.

Module Contents

Functions

<code>get_files_from_tree(dir_searchmaterial_results, ...)</code>	Collects filenames from the SearchMaterial and SearchWasteResults directories.
<code>dbWriteExcel()</code>	Create an xlsx file representing a custom Brightway2 database.
<code>determine_unit_from_name(name)</code>	Determine the unit based on the name.
<code>dbExcel2BW()</code>	Import the custom database (created by dbWriteExcel) into Brightway2.

T-reX.MakeCustomDatabase.**get_files_from_tree**(*dir_searchmaterial_results*, *dir_searchwaste_results*)
Collects filenames from the SearchMaterial and SearchWasteResults directories.

Parameters

- **dir_searchmaterial_results** – Directory path for SearchMaterial results.
- **dir_searchwaste_results** – Directory path for SearchWasteResults.

Returns

Sorted list of filenames.

T-reX.MakeCustomDatabase.**dbWriteExcel**()

Create an xlsx file representing a custom Brightway2 database.

This function generates an Excel file which represents a custom database for Brightway2, using predefined directory and database settings.

Returns

Path to the generated xlsx file.

T-reX.MakeCustomDatabase.**determine_unit_from_name**(*name*)

Determine the unit based on the name.

Parameters

name – The name from which to infer the unit.

Returns

The inferred unit as a string.

T-reX.MakeCustomDatabase.**dbExcel2BW**()

Import the custom database (created by dbWriteExcel) into Brightway2.

This function imports a custom Brightway2 database from an Excel file into the Brightway2 software, making it available for further environmental impact analysis.

Returns

None

T-reX.MethodEditor

This module provides functions for adding, deleting, and checking methods related to waste and material footprints in a project.

Function Summary:

- *AddMethods*: Adds new methods to a project based on a custom biosphere database.
- *DeleteMethods*: Removes specific methods from a project, particularly those related to waste and material footprints.
- *CheckMethods*: Lists and checks the methods in a project, focusing on those associated with waste and material footprints.

Module Contents

Functions

<i>AddMethods()</i>	Add methods to the specified project based on entries in the custom biosphere database.
<i>DeleteMethods()</i>	Delete methods associated with the "WasteAndMaterial Footprint" in the specified project.
<i>CheckMethods()</i>	Check methods associated with the "WasteAndMaterial Footprint" in the specified project.

T-reX.MethodEditor.AddMethods()

Add methods to the specified project based on entries in the custom biosphere database.

Parameters

- **project_T_reX** – Name of the project.
- **db_T_reX_name** – Name of the database.

T-reX.MethodEditor.DeleteMethods()

Delete methods associated with the “WasteAndMaterial Footprint” in the specified project.

Parameters

project_T_reX – Name of the project.

T-reX.MethodEditor.CheckMethods()

Check methods associated with the “WasteAndMaterial Footprint” in the specified project.

Parameters

project_T_reX – Name of the project.

T-reX.SearchMaterial

This script loads data from ‘<db name>_exploded.pickle’, runs search queries, and produces a CSV to store the results and a log entry. The search queries are formatted as dictionaries with fields NAME, CODE, and search terms keywords_AND, keywords_OR, and keywords_NOT. These queries are defined in *config/queries_waste.py*.

Module Contents

Functions

<code>SearchMaterial(db_name[, project_T_reX])</code>	Search for materials in a specified database and extract related information.
-------------------------------------------------------	-------------------------------------------------------------------------------

T-reX.SearchMaterial.**SearchMaterial**(*db_name*, *project_T_reX=project_T_reX*)

Search for materials in a specified database and extract related information.

This function takes a database name as input, sets the project to the respective database, and looks for activities involving a predefined list of materials. It extracts relevant details of these activities, such as ISIC and CPC classifications, and saves the details to a CSV file. It also extracts related material exchanges and saves them to another CSV file.

Parameters

- **db_name** – The name of the database to search in.
- **project_T_reX** – The Brightway2 project to set as current for the search.

Returns

None

Raises

Exception – If there is any error in reading the materials list from the file.

T-reX.SearchWaste

This script loads data from ‘<db name>_exploded.pickle’, runs search queries, and produces CSV files to store the results and a log entry. The search queries are formatted as dictionaries with fields NAME, CODE, and search terms keywords_AND, keywords_OR, and keywords_NOT. These queries are defined in *config/queries_waste.py*.

Functionality

Provides a function, `SearchWaste()`, that loads data from ‘<db name>_exploded.pickle’, runs search queries, and produces result CSVs and log entries.

Module Contents

Functions

<code>SearchWaste(db_name[, dir_searchwaste_results])</code>	Load data from '<db name>_exploded.pickle', run search queries, and produce
--------------------------------------------------------------	-----------------------------------------------------------------------------

T-reX.SearchWaste.**SearchWaste**(*db_name*, *dir_searchwaste_results*=*dir_searchwaste_results*)

Load data from '<db name>_exploded.pickle', run search queries, and produce result CSVs and log entries.

This function processes waste-related data from a given database and runs predefined queries to identify relevant waste exchanges. The results are saved in CSV files and log entries are created for each search operation.

Parameters

db_name (*str*) – The database name to be used in the search operation.

Note: The queries are defined in *config/queries_waste.py*.

T-reX.VerifyDatabase

This module contains a function to verify a (T-reX) database within a given project in Brightway2. It performs a verification by calculating LCA scores for random activities within the specified database using selected methods.

Module Contents

Functions

<code>VerifyDatabase(project_name, database_name[, ...])</code>	Verifies a database within a given project in Brightway2 by calculating LCA scores
-----------------------------------------------------------------	------------------------------------------------------------------------------------

Attributes

<code>project_name</code>

T-reX.VerifyDatabase.**VerifyDatabase**(*project_name*, *database_name*, *check_material*=*True*, *check_waste*=*True*, *log*=*True*)

Verifies a database within a given project in Brightway2 by calculating LCA scores for random activities using selected methods.

This function assesses the integrity and validity of a specified database within a Brightway2 project. It performs LCA calculations on random activities using Waste Footprint and Material Demand Footprint methods, and logs the results.

Parameters

- **project_name** (*str*) – The name of the Brightway2 project.
- **database_name** (*str*) – The name of the database to be verified.

- **check_material** (*bool*) – If True, checks for Material Demand Footprint methods.
- **check_waste** (*bool*) – If True, checks for Waste Footprint methods.
- **log** (*bool*) – If True, logs the results.

Returns

Exit code (0 for success, 1 for failure).

`T-reX.VerifyDatabase.project_name = 'T-reX-default'`

T-reX.main

Main module of the *T-reX* tool.

This script serves as the entry point for the *T-reX* tool. It orchestrates the overall process, including the setup and execution of various subprocesses like database explosion, material and waste searches, and the editing of exchanges.

The script supports both single and multiple project/database modes, as well as the option to use multiprocessing. It also facilitates the use of the premise module to generate future scenario databases.

Customisation:

- Project and database names, and other settings can be edited in *config/user_settings.py*.
- Waste search query terms can be customised in *config/queries_waste.py*.
- The list of materials can be modified in *config/queries_materials.py*.

Usage:

To use the default settings, run the script with *python main.py*. Arguments can be provided to change project/database names or to delete the project before running.

Module Contents

Functions

<code>run()</code>	Main function serving as the wrapper for the T-reX tool.
<code>ExplodeAndSearch(args)</code>	Exploding the database into separate exchanges, searching for waste and
<code>EditExchanges(args)</code>	Edit exchanges in the database.

Attributes

`num_cpus`

`script_dir`

T-reX.main.**num_cpus**

T-reX.main.**script_dir**

T-reX.main.**run()**

Main function serving as the wrapper for the T-reX tool.

This function coordinates the various components of the tool, including:

creating future scenario databases, setting up and processing each database for waste and material foot-printing, and combining results into a custom database. adding LCIA methods to the project for each of the waste/material flows.

The function supports various modes of operation based on the settings in *config/user_settings.py*. Specifications for material and waste searches can be customised in *queries_materials*.

T-reX.main.**ExplodeAndSearch**(*args*)

Exploding the database into separate exchanges, searching for waste and material flows, and processing these results.

This includes:

- `ExplodeDatabase.py`
- `SearchWaste.py`
- `SearchMaterial.py`

Parameters

args – Dictionary containing database and project settings.

Returns

None

T-reX.main.**EditExchanges**(*args*)

Edit exchanges in the database.

This function adds waste and material flows to the activities and verifies the database.

Parameters

args – Dictionary containing database and project settings.

Returns

None

9.1.3 Package Contents

Functions

<code>run()</code>	Main function serving as the wrapper for the T-reX tool.
<code>config_setup()</code>	Copies the config files to the current working directory and adds the config directory to sys.path
<code>config_reload()</code>	Copies the config files back to the package directory. The user will need to restart the python session to reload the T-reX module with the updated configuration files
<code>config_reset()</code>	Reset config to defaults that are stored in the root directory of the package
<code>ExchangeEditor(project_T_reX, db_name, db_T_reX_name)</code>	Append relevant exchanges from <i>db_T_reX</i> to each activity in <i>db_name</i> identified by <i>WasteAndMaterialSearch()</i> .
<code>ExplodeDatabase</code>	ExplodeDatabase Module
<code>MakeFutureScenarios()</code>	Main function to run the FutureScenarios module.
<code>dbWriteExcel()</code>	Create an xlsx file representing a custom Brightway2 database.
<code>dbExcel2BW()</code>	Import the custom database (created by dbWriteExcel) into Brightway2.
<code>AddMethods()</code>	Add methods to the specified project based on entries in the custom biosphere database.
<code>DeleteMethods()</code>	Delete methods associated with the "WasteAndMaterial Footprint" in the specified project.
<code>CheckMethods()</code>	Check methods associated with the "WasteAndMaterial Footprint" in the specified project.
<code>SearchMaterial(db_name[, project_T_reX])</code>	Search for materials in a specified database and extract related information.
<code>SearchWaste(db_name[, dir_searchwaste_results])</code>	Load data from '<db name>_exploded.pickle', run search queries, and produce
<code>VerifyDatabase(project_name, database_name[, ...])</code>	Verifies a database within a given project in Brightway2 by calculating LCA scores

T-reX.run()

Main function serving as the wrapper for the T-reX tool.

This function coordinates the various components of the tool, including:

creating future scenario databases, setting up and processing each database for waste and material footprinting, and combining results into a custom database. adding LCIA methods to the project for each of the waste/material flows.

The function supports various modes of operation based on the settings in *config/user_settings.py*. Specifications for material and waste searches can be customised in *queries_materials*.

T-reX.config_setup()

Copies the config files to the current working directory and adds the config directory to sys.path args: None returns: None

T-reX.config_reload()

Copies the config files back to the package directory. The user will need to restart the python session to reload the T-reX module with the updated configuration files

args: None returns: None

T-reX.config_reset()

Reset config to defaults that are stored in the root directory of the package

args: None returns: None

T-reX.ExchangeEditor(*project_T_reX*, *db_name*, *db_T_reX_name*)

Append relevant exchanges from *db_T_reX* to each activity in *db_name* identified by *WasteAndMaterialSearch()*.

This function modifies the specified project's database by appending exchanges from the *db_T_reX* to activities identified by *WasteAndMaterialSearch()*. The appended exchanges mirror the quantity and unit of the original technosphere waste and material exchange.

Parameters

- **project_T_reX** (*str*) – Name of the Brightway2 project to be modified.
- **db_name** (*str*) – Name of the database within the project where activities and exchanges are stored.
- **db_T_reX_name** (*str*) – Name of the database containing waste and material exchange details.

Returns

None. Modifies the given Brightway2 project by appending exchanges and logs statistics about the added exchanges.

Return type

None

Raises

Exception – If any specified process or exchange is not found in the database.

T-reX.ExplodeDatabase(*db_name*)

Explode a Brightway2 database into a single-level list of all exchanges using wurst.

Parameters

db_name (*str*) – Name of the Brightway2 database to be exploded.

Returns

None The function saves the output to a file and logs the operation, but does not return any value.

Return type

None

T-reX.MakeFutureScenarios()

Main function to run the FutureScenarios module. Only activated if *use_premise* is set to True in *user_settings.py*.

Calls the *FutureScenarios* function to create new databases based on the list of scenarios and settings specified in *user_settings.py*.

T-reX.dbWriteExcel()

Create an xlsx file representing a custom Brightway2 database.

This function generates an Excel file which represents a custom database for Brightway2, using predefined directory and database settings.

Returns

Path to the generated xlsx file.

T-reX.dbExcel2BW()

Import the custom database (created by dbWriteExcel) into Brightway2.

This function imports a custom Brightway2 database from an Excel file into the Brightway2 software, making it available for further environmental impact analysis.

Returns

None

T-reX.AddMethods()

Add methods to the specified project based on entries in the custom biosphere database.

Parameters

- **project_T_reX** – Name of the project.
- **db_T_reX_name** – Name of the database.

T-reX.DeleteMethods()

Delete methods associated with the “WasteAndMaterial Footprint” in the specified project.

Parameters

project_T_reX – Name of the project.

T-reX.CheckMethods()

Check methods associated with the “WasteAndMaterial Footprint” in the specified project.

Parameters

project_T_reX – Name of the project.

T-reX.SearchMaterial(*db_name*, *project_T_reX*=*project_T_reX*)

Search for materials in a specified database and extract related information.

This function takes a database name as input, sets the project to the respective database, and looks for activities involving a predefined list of materials. It extracts relevant details of these activities, such as ISIC and CPC classifications, and saves the details to a CSV file. It also extracts related material exchanges and saves them to another CSV file.

Parameters

- **db_name** – The name of the database to search in.
- **project_T_reX** – The Brightway2 project to set as current for the search.

Returns

None

Raises

Exception – If there is any error in reading the materials list from the file.

T-reX.SearchWaste(*db_name*, *dir_searchwaste_results*=*dir_searchwaste_results*)

Load data from ‘<db name>_exploded.pickle’, run search queries, and produce result CSVs and log entries.

This function processes waste-related data from a given database and runs predefined queries to identify relevant waste exchanges. The results are saved in CSV files and log entries are created for each search operation.

Parameters

db_name (*str*) – The database name to be used in the search operation.

Note: The queries are defined in *config/queries_waste.py*.

T-reX.VerifyDatabase(*project_name*, *database_name*, *check_material*=*True*, *check_waste*=*True*, *log*=*True*)

Verifies a database within a given project in Brightway2 by calculating LCA scores for random activities using selected methods.

This function assesses the integrity and validity of a specified database within a Brightway2 project. It performs LCA calculations on random activities using Waste Footprint and Material Demand Footprint methods, and logs the results.

Parameters

- **project_name** (*str*) – The name of the Brightway2 project.
- **database_name** (*str*) – The name of the database to be verified.
- **check_material** (*bool*) – If True, checks for Material Demand Footprint methods.
- **check_waste** (*bool*) – If True, checks for Waste Footprint methods.
- **log** (*bool*) – If True, logs the results.

Returns

Exit code (0 for success, 1 for failure).

PYTHON MODULE INDEX

t

- T-reX, [31](#)
- T-reX.config, [31](#)
- T-reX.config.queries_materials, [31](#)
- T-reX.config.queries_waste, [32](#)
- T-reX.config.user_settings, [33](#)
- T-reX.config_backup, [38](#)
- T-reX.config_backup.queries_materials, [38](#)
- T-reX.config_backup.queries_waste, [39](#)
- T-reX.config_backup.user_settings, [40](#)
- T-reX.CustomConfig, [45](#)
- T-reX.ExchangeEditor, [46](#)
- T-reX.ExplodeDatabase, [47](#)
- T-reX.FutureScenarios, [47](#)
- T-reX.main, [53](#)
- T-reX.MakeCustomDatabase, [48](#)
- T-reX.MethodEditor, [50](#)
- T-reX.SearchMaterial, [51](#)
- T-reX.SearchWaste, [51](#)
- T-reX.VerifyDatabase, [52](#)

A

AddMethods() (in module *T-reX*), 57
AddMethods() (in module *T-reX.MethodEditor*), 50

B

batch_size (in module *T-reX.config.user_settings*), 35
batch_size (in module *T-reX.config_backup.user_settings*), 43

C

check_existing() (in module *T-reX.FutureScenarios*), 48
CheckMethods() (in module *T-reX*), 57
CheckMethods() (in module *T-reX.MethodEditor*), 50
config_reload() (in module *T-reX*), 55
config_reload() (in module *T-reX.CustomConfig*), 46
config_reset() (in module *T-reX*), 55
config_reset() (in module *T-reX.CustomConfig*), 46
config_setup() (in module *T-reX*), 55
config_setup() (in module *T-reX.CustomConfig*), 45
custom_bw2_dir (in module *T-reX.config*), 37
custom_bw2_dir (in module *T-reX.config.user_settings*), 35
custom_bw2_dir (in module *T-reX.config_backup*), 44
custom_bw2_dir (in module *T-reX.config_backup.user_settings*), 42
cwd (in module *T-reX.config.user_settings*), 36
cwd (in module *T-reX.config_backup.user_settings*), 43

D

database_name (in module *T-reX.config.user_settings*), 35
database_name (in module *T-reX.config_backup.user_settings*), 42
db_T_reX_name (in module *T-reX.config*), 37
db_T_reX_name (in module *T-reX.config.user_settings*), 35
db_T_reX_name (in module *T-reX.config_backup*), 44
db_T_reX_name (in module *T-reX.config_backup.user_settings*), 42
dbExcel2BW() (in module *T-reX*), 56

dbExcel2BW() (in module *T-reX.MakeCustomDatabase*), 49
dbWriteExcel() (in module *T-reX*), 56
dbWriteExcel() (in module *T-reX.MakeCustomDatabase*), 49
delete_existing_premise_project (in module *T-reX.config.user_settings*), 35
delete_existing_premise_project (in module *T-reX.config_backup.user_settings*), 42
delete_T_reX_project (in module *T-reX.config*), 37
delete_T_reX_project (in module *T-reX.config.user_settings*), 35
delete_T_reX_project (in module *T-reX.config_backup*), 44
delete_T_reX_project (in module *T-reX.config_backup.user_settings*), 42
DeleteMethods() (in module *T-reX*), 57
DeleteMethods() (in module *T-reX.MethodEditor*), 50
desired_scenarios (in module *T-reX.config.user_settings*), 36
desired_scenarios (in module *T-reX.config_backup.user_settings*), 43
determine_unit_from_name() (in module *T-reX.MakeCustomDatabase*), 49
dir_config (in module *T-reX.config.user_settings*), 36
dir_config (in module *T-reX.config_backup.user_settings*), 43
dir_data (in module *T-reX.config.user_settings*), 36
dir_data (in module *T-reX.config_backup.user_settings*), 43
dir_databases_T_reX (in module *T-reX.config.user_settings*), 36
dir_databases_T_reX (in module *T-reX.config_backup.user_settings*), 43
dir_logs (in module *T-reX.config*), 37
dir_logs (in module *T-reX.config.user_settings*), 36
dir_logs (in module *T-reX.config_backup*), 44
dir_logs (in module *T-reX.config_backup.user_settings*), 43
dir_premise (in module *T-reX.FutureScenarios*), 48
dir_searchmaterial_results (in module *T-reX.config.user_settings*), 36

- dir_searchmaterial_results (in module *T-reX.config.backup.user_settings*), 43
- dir_searchwaste_results (in module *T-reX.config.user_settings*), 36
- dir_searchwaste_results (in module *T-reX.config.backup.user_settings*), 43
- dir_T_reX (in module *T-reX.config.user_settings*), 36
- dir_T_reX (in module *T-reX.config.backup.user_settings*), 43
- dir_tmp (in module *T-reX.config*), 37
- dir_tmp (in module *T-reX.config.user_settings*), 36
- dir_tmp (in module *T-reX.config.backup*), 45
- dir_tmp (in module *T-reX.config.backup.user_settings*), 43
- do_edit (in module *T-reX.config*), 38
- do_edit (in module *T-reX.config.user_settings*), 35
- do_edit (in module *T-reX.config.backup*), 45
- do_edit (in module *T-reX.config.backup.user_settings*), 42
- do_methods (in module *T-reX.config*), 38
- do_methods (in module *T-reX.config.user_settings*), 35
- do_methods (in module *T-reX.config.backup*), 45
- do_methods (in module *T-reX.config.backup.user_settings*), 42
- do_search (in module *T-reX.config*), 38
- do_search (in module *T-reX.config.user_settings*), 35
- do_search (in module *T-reX.config.backup*), 45
- do_search (in module *T-reX.config.backup.user_settings*), 42
- ## E
- EditExchanges() (in module *T-reX.main*), 54
- ExchangeEditor() (in module *T-reX*), 56
- ExchangeEditor() (in module *T-reX.ExchangeEditor*), 46
- ExplodeAndSearch() (in module *T-reX.main*), 54
- ExplodeDatabase() (in module *T-reX*), 56
- ExplodeDatabase() (in module *T-reX.ExplodeDatabase*), 47
- ## F
- FutureScenarios() (in module *T-reX.FutureScenarios*), 48
- ## G
- generate_args_list() (in module *T-reX.config*), 37
- generate_args_list() (in module *T-reX.config.user_settings*), 36
- generate_args_list() (in module *T-reX.config.backup*), 45
- generate_args_list() (in module *T-reX.config.backup.user_settings*), 43
- get_files_from_tree() (in module *T-reX.MakeCustomDatabase*), 49
- get_queries_materials() (in module *T-reX.config.queries_materials*), 32
- get_queries_materials() (in module *T-reX.config.backup.queries_materials*), 39
- ## K
- key_path (in module *T-reX.config.user_settings*), 36
- key_path (in module *T-reX.config.backup.user_settings*), 43
- ## L
- list_materials (in module *T-reX.config.queries_materials*), 32
- list_materials (in module *T-reX.config.backup.queries_materials*), 39
- ## M
- make_possible_scenario_list() (in module *T-reX.FutureScenarios*), 48
- make_queries_waste() (in module *T-reX.config.queries_waste*), 33
- make_queries_waste() (in module *T-reX.config.backup.queries_waste*), 40
- MakeFutureScenarios() (in module *T-reX*), 56
- MakeFutureScenarios() (in module *T-reX.FutureScenarios*), 48
- models (in module *T-reX.config.user_settings*), 36
- models (in module *T-reX.config.backup.user_settings*), 43
- module
- T-reX*, 31
 - T-reX.config*, 31
 - T-reX.config.queries_materials*, 31
 - T-reX.config.queries_waste*, 32
 - T-reX.config.user_settings*, 33
 - T-reX.config.backup*, 38
 - T-reX.config.backup.queries_materials*, 38
 - T-reX.config.backup.queries_waste*, 39
 - T-reX.config.backup.user_settings*, 40
 - T-reX.CustomConfig*, 45
 - T-reX.ExchangeEditor*, 46
 - T-reX.ExplodeDatabase*, 47
 - T-reX.FutureScenarios*, 47
 - T-reX.main*, 53
 - T-reX.MakeCustomDatabase*, 48
 - T-reX.MethodEditor*, 50
 - T-reX.SearchMaterial*, 51
 - T-reX.SearchWaste*, 51
 - T-reX.VerifyDatabase*, 52
- ## N
- num_cpus (in module *T-reX.main*), 54

P

premise_key (in module *T-reX.config.user_settings*), 36
 premise_key (in module *T-reX.config_backup.user_settings*), 43
 premise_quiet (in module *T-reX.config.user_settings*), 36
 premise_quiet (in module *T-reX.config_backup.user_settings*), 43
 project_base (in module *T-reX.config*), 38
 project_base (in module *T-reX.config.user_settings*), 35
 project_base (in module *T-reX.config_backup*), 45
 project_base (in module *T-reX.config_backup.user_settings*), 42
 project_name (in module *T-reX.VerifyDatabase*), 53
 project_premise (in module *T-reX.config*), 38
 project_premise (in module *T-reX.config.user_settings*), 35
 project_premise (in module *T-reX.config_backup*), 45
 project_premise (in module *T-reX.config_backup.user_settings*), 42
 project_premise_base (in module *T-reX.config.user_settings*), 35
 project_premise_base (in module *T-reX.config_backup.user_settings*), 42
 project_T_reX (in module *T-reX.config*), 38
 project_T_reX (in module *T-reX.config.user_settings*), 35, 36
 project_T_reX (in module *T-reX.config_backup*), 45
 project_T_reX (in module *T-reX.config_backup.user_settings*), 42, 43

Q

queries_materials (in module *T-reX.config*), 38
 queries_materials (in module *T-reX.config_queries_materials*), 32
 queries_materials (in module *T-reX.config_backup*), 45
 queries_materials (in module *T-reX.config_backup_queries_materials*), 39
 queries_waste (in module *T-reX.config*), 38
 queries_waste (in module *T-reX.config_queries_waste*), 33
 queries_waste (in module *T-reX.config_backup*), 45
 queries_waste (in module *T-reX.config_backup_queries_waste*), 40

R

rcps (in module *T-reX.config.user_settings*), 36
 rcps (in module *T-reX.config_backup.user_settings*), 43
 run() (in module *T-reX*), 55
 run() (in module *T-reX.main*), 54

S

script_dir (in module *T-reX.main*), 54
 SearchMaterial() (in module *T-reX*), 57
 SearchMaterial() (in module *T-reX.SearchMaterial*), 51
 SearchWaste() (in module *T-reX*), 57
 SearchWaste() (in module *T-reX.SearchWaste*), 52
 single (in module *T-reX.config.user_settings*), 35
 single (in module *T-reX.config_backup.user_settings*), 42
 single_database (in module *T-reX.config*), 38
 single_database (in module *T-reX.config.user_settings*), 35
 single_database (in module *T-reX.config_backup*), 45
 single_database (in module *T-reX.config_backup.user_settings*), 42
 ssps (in module *T-reX.config.user_settings*), 36
 ssps (in module *T-reX.config_backup.user_settings*), 43

T

T-reX
 module, 31
 T-reX.config
 module, 31
 T-reX.config.queries_materials
 module, 31
 T-reX.config.queries_waste
 module, 32
 T-reX.config.user_settings
 module, 33
 T-reX.config_backup
 module, 38
 T-reX.config_backup.queries_materials
 module, 38
 T-reX.config_backup.queries_waste
 module, 39
 T-reX.config_backup.user_settings
 module, 40
 T-reX.CustomConfig
 module, 45
 T-reX.ExchangeEditor
 module, 46
 T-reX.ExplodeDatabase
 module, 47
 T-reX.FutureScenarios
 module, 47
 T-reX.main
 module, 53
 T-reX.MakeCustomDatabase
 module, 48
 T-reX.MethodEditor
 module, 50
 T-reX.SearchMaterial
 module, 51

T-reX.SearchWaste

module, [51](#)

T-reX.VerifyDatabase

module, [52](#)

U

use_mp (in module *T-reX.config.user_settings*), [35](#)

use_mp (in module *T-reX.config_backup.user_settings*),
[42](#)

use_multiprocessing (in module *T-reX.config*), [38](#)

use_multiprocessing (in module *T-reX.config.user_settings*), [35](#)

use_multiprocessing (in module *T-reX.config_backup*), [45](#)

use_multiprocessing (in module *T-reX.config_backup.user_settings*), [42](#)

use_premise (in module *T-reX.config*), [38](#)

use_premise (in module *T-reX.config.user_settings*), [35](#)

use_premise (in module *T-reX.config_backup*), [45](#)

use_premise (in module *T-reX.config_backup.user_settings*), [42](#)

use_T_reX (in module *T-reX.config*), [38](#)

use_T_reX (in module *T-reX.config.user_settings*), [35](#)

use_T_reX (in module *T-reX.config_backup*), [45](#)

use_T_reX (in module *T-reX.config_backup.user_settings*), [42](#)

V

verbose (in module *T-reX.config.user_settings*), [35](#)

verbose (in module *T-reX.config_backup.user_settings*),
[42](#)

VerifyDatabase() (in module *T-reX*), [57](#)

VerifyDatabase() (in module *T-reX.VerifyDatabase*),
[52](#)

Y

years (in module *T-reX.config.user_settings*), [36](#)

years (in module *T-reX.config_backup.user_settings*), [43](#)