# RandomFileTree Documentation

*Release 1.1.0*

**Kilian Lieret**

**Oct 05, 2021**

# Contents

Contents:

CHAPTER 1

# Readme

# CHAPTER 2

## Module content

randomfiletree.core.**choose_random_elements**(*basedir*, *n_dirs*, *n_files*, *onfail='raise'*)

Select random files and directories. If all directories and files must be unique, use sample_random_elements instead.

**Args:** basedir: Directory to scan n_dirs: Number of directories to pick n_files: Number of files to pick onfail: What to do if there are no files or folders to pick from?

Either 'raise' (raise ValueError) or 'ignore' (return empty list)

**Returns:** (List of dirs, List of files), all as pathlib.Path objects.

randomfiletree.core.**iterative_gaussian_tree**(*basedir: Union[str, pathlib.PurePath], nfiles=2, nfolders=1, repeat=1, maxdepth=None, sigma_folders=1, sigma_files=1, min_folders=0, min_files=0, filename=<function random_string>, payload: Optional[Callable[pathlib.Path, Generator[[pathlib.Path, None], None]]] = None*)

Create a random set of files and folders by repeatedly walking through the current tree and creating random files or subfolders (the number of files and folders created is chosen from a Gaussian distribution).

**Args:** basedir: Directory to create files and folders in nfiles: Average number of files to create nfolders: Average number of folders to create repeat: Walk this often through the directory tree to create new

subdirectories and files

**maxdepth: Maximum depth to descend into current file tree. If None,** infinity.

sigma_folders: Spread of number of folders sigma_files: Spread of number of files min_folders: Minimal number of folders to create. Default 0. min_files: Minimal number of files to create. Default 0. filename: Callable to generate filename. Default returns short

random string

**payload: Use this argument to generate files with content: Specify a** function that takes a directory `dir` (`Path` object) as argument, picks a name `name`, creates the corresponding file `dir/name` and yields `name`. Overrides `filename` argument if both are passed. Takes Path object as catalog where to create file and returns Path of created file. If this option is not specified, all created files will be empty.

**Returns:** (List of dirs, List of files), all as `pathlib.Path` objects.

randomfiletree.core.**iterative_tree**(*basedir: Union[str, pathlib.PurePath], nfolders_func: Callable, nfiles_func: Callable, repeat=1, maxdepth=None, filename=<function random_string>, payload: Optional[Callable[pathlib.Path, Generator[[pathlib.Path, None], None]]] = None*) → Tuple[List[pathlib.Path], List[pathlib.Path]]

Create a random set of files and folders by repeatedly walking through the current tree and creating random files or subfolders (the number of files and folders created is chosen by evaluating a depth dependent function).

**Args:** basedir: Directory to create files and folders in nfolders_func: (depth) that returns the number of folders to be

created in a folder of that depth.

**nfiles_func: Function(depth) that returns the number of files to be** created in a folder of that depth.

**repeat: Walk this often through the directory tree to create new** subdirectories and files

**maxdepth: Maximum depth to descend into current file tree. If None,** infinity.

**filename: Callable to generate filename. Default returns short** random string

**payload: Use this argument to generate files with content: Specify a** function that takes a directory `dir` (`Path` object) as argument, picks a name `name`, creates the corresponding file `dir/name` and yields `name`. Overrides `filename` argument if both are passed. Takes Path object as catalog where to create file and yields Path of created file. If this option is not specified, all created files will be empty.

**Returns:** (List of dirs, List of files), all as pathlib.Path objects.

randomfiletree.core.**random_string**(*min_length=5*, *max_length=10*) → str

Get a random string.

**Args:** min_length: Minimal length of string max_length: Maximal length of string

**Returns:** Random string of ascii characters

randomfiletree.core.**sample_random_elements**(*basedir*, *n_dirs*, *n_files*, *onfail='raise'*)

Select random distinct files and directories. If the directories and files do not have to be distinct, use choose_random_elements instead.

**Args:** basedir: Directory to scan n_dirs: Number of directories to pick n_files: Number of files to pick onfail: What to do if there are no files or folders to pick from?

Either 'raise' (raise ValueError) or 'ignore' (return list with fewer elements)

**Returns:** (List of dirs, List of files), all as pathlib.Path objects.

CHAPTER 3

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## r

# Index