1. To write a wrapper and/or tuner for a solver, make a new python file (version >= 3.6, tested on 3.9) and import the `Solver` and `Parameter` classes from the generator. Make sure that you use the correct path, e.g. if you're in the `scripts` folder then use `from generator.generator`.
2. Specify tuning parameters by creating a list with instances of the `Parameter` class using the following arguments:

```
name : str
  Required argument that specifies the name of the parameter.

value : float | int | str
  Required argument with a predefined value of the parameter.

tuning_range : list(float | int | str)
  Optional argument that specifies the interval of continuous or integer
  tuning variables or all the options of categorical ones.
  Defaults to empty list.

input_type : type
  Optional argument that specifies what kind of values the
  parameter attains.
  Must be `float`, `int` or `str`.
  Defaults to type of given `value`.

optimization_type : str
  Optional argument that specifies how the variable is treated
  when tuning.
  Must be `"continuous"`, `"integer"` or `"categorical"`.
  Default depends on type of given `value` as follows:
  `float` -> `"continuous"`,
  `int` -> `"integer"`,
  `str` -> `"categorical"`

description : str
  Optional argument that specifies a short description of the parameter
  and should fit into half a line.
  Defaults to empty string.

info : str
  Optional argument that provides more information than the description.
  Will be broken into multiple lines if necessary.
  Defaults to empty string.
```

3. Create an instance of `Solver` using the following arguments:

```
name : str
  Required argument that specifies the name of the solver.

params : list(Parameter)
  Optional argument that defines all the tuning parameters.
  Defaults to empty list.

paths : list(str)
  Optional argument that specifies all paths of the solver
  which are needed for calling it and will be added to the workspace.
  Defaults to empty list.

preparation : str
  Optional argument that is valid MATLAB code, which is needed
  to prepare all input options of the solver.
  Defaults to empty string.

call : str
  Optional argument that is valid MATLAB code, which calls the solver
  and processes the output as needed.
  Defaults to empty string.

tune_cases : list(list(Parameter))
  Optional argument that specifies more predefined parameter options
  that can be selected when calling the wrapper.
  Defaults to empty list.
```

4. Call the desired wrapper or tuner writing method on the defined solver instance:

```
Solver.write_wrapper(save_path)
  save_path : str
    Optional argument that specifies path to which the wrapper
    will be saved.
    Defaults to path that the script is executed from.

Solver.write_tuner(save_path)
  save_path : str
    Optional argument that specifies path to which the tuner
    will be saved.
    Defaults to path that the script is executed from.
```

To make structural changes to all wrappers and tuners edit the `WRAPPER_TEMPLATE.txt` or `TUNER_TEMPLATE.txt`. Make sure to not alter the keywords, unless you intend to edit them in `data.py` too. All solver scripts that require this change then need to be run again.

See `example.py` for a showcase of an ideal structure for such a script. The files `wSOLVER.m` and `tSOLVER.m` were generated with this script.

Author: Maximilian Stollmayer, maximilian.stollmayer@univie.ac.at