One of the central components of PonyGE is the `algorithm.parameters.params` dictionary. This dictionary is referenced throughout the entire program and is used to streamline the whole process by keeping all optional parameters in the one place. This also means that there is little to no need for arguments to the various functions in PonyGE, as these arguments can often be read directly from the parameters dictionary. Furthermore, the parameters dictionary is used to specify and store optional functions such as `initialisation`, `crossover`, `mutation`, and `replacement`.

# Setting Parameters

There are three different ways to specify operational parameters with PonyGE: by editing the dictionary itself, by setting a parameters file, and by specifying parameters from the command line.

## The Parameters Dictionary

The most basic way to set specific parameters values in PonyGE2 is to directly modify the `algorithm.parameters.params` dictionary in the code. This is not encouraged, as the `algorithm.parameters.params` dictionary contains the default values for many parameters. As such, PonyGE2 may perform in an unpredictable manner in some cases.

## Parameters Files

The most versatile and robust method for specifying desired parameters is to list your desired parameters in a specialised parameters text file. Specific parameters files can then be read in by PonyGE2, and will set any parameters specified therein.

Parameters files are simple text files (with no commenting or other documentation) that simply list all of the desired parameters to be set in the central `algorithm.parameters.params` dictionary. Parameters can be specified in a parameters file using the following formatting:

```
PARAMETER_NAME: PARAMETER_VALUE
```

where `PARAMETER_NAME` is the name of the desired parameter from the central `algorithm.parameters.params` dictionary to be set, and `PARAMETER_VALUE` is the desired value to be stored for that parameter.

Entire lines can be commented out of parameters files simply by prefacing the line with a `#` symbol, as with normal Python code. The use of comments allows parameters files to be neatly and clearly organised into grouped sections if so desired. Note that blank lines in parameters files are ignored.

***NOTE** that parameter names and values must be colon-separated, i.e. each line of the parameters file will be read in as a string, and the string will be parsed based on the location of the first instance of a colon. **NOTE** that this **does** preclude the use of colons in parameter **names** (i.e. the keys of the params dictionary), but that this does **not** preclude the use of colons in the parameter **values**.*

Example parameters files are located in the `parameters` folder. When using parameters files, it is necessary to specify the desired parameter file from the command line. This is done with the argument:

```
--parameters [FULL FILE NAME INCLUDING EXTENSION]
```

## Command-Line Arguments

The third and final method for setting parameters is to list desired parameters directly from the command line. PonyGE2 has an extensive command line parser, located in `utilities.algorithm.command_line_parser`. All stock parameters listed in the original `algorithm.parameters.params` dictionary can be set or changed via command line arguments. Command line arguments for all parameters are the same as the dictionary keys, except all in lower case. For example, the dictionary key for the size of the evolutionary population is `POPULATION_SIZE`. This value can be set from the command line with the argument:

```
--population_size [SIZE]
```

where `[SIZE]` is an integer specifying the desired population size.

To see a list of all currently available command-line arguments implemented in the parser, type

```
$ python ponyge.py --help
```

## Order of Setting Parameters

Parameters are set in the `algorithm.parameters.params` dictionary in strict order based on the above three methods. Initial values are set in directly by the [[dictionary|Evolutionary-Parameters#the-parameters-dictionary]] itself. Any specified [[parameters file|Evolutionary-Parameters#parameters-files]] is then read in, and any values specified therein will over-write the values contained so far in the dictionary. Finally, any specified [[command-line arguments|Evolutionary-Parameters#command-line-arguments]] will over-write all previously set parameters.

## Path Specification

The modular structure of PonyGE2 has specialised modules (folders) for each aspect of the overall system:

1. [[Datasets|Evaluation#datasets]] *must* be stored in the `datasets` folder,
2. [[Grammar files|Grammars]] *must* be stored in the `grammars` folder,
3. [[Parameters files|Evolutionary-Parameters#parameters-files]] *must* be stored in the `parameters` folder,
4. [[Population seeds|Seeding#2-seeding-runs-with-one-or-more-target-solutions]] *must* be stored in the `seeds` folder,
5. [[Fitness functions|Evaluation#fitness-functions]] *must* be stored in `src.fitness`,
6. [[Error metrics|Evaluation#error-metrics]] *must* be stored in `utilities.fitness.error_metric`,
7. Operators are typically stored in the operators folder:
    1. [[Initialisation]] functions are stored in `src.operators.initialisation`,
    2. [[Selection]] functions are stored in `src.operators.selection`,

    3. [[Crossover|Variation#crossover]] functions are stored in `src.operators.crossover`,
    4. [[Mutation|Variation#mutation]] functions are stored in `src.operators.mutation`,
    5. [[Replacement]] functions are stored in `src.operators.replacement`.
8. [[Search framework functions|Search-Options]] are typically stored in the algorithm folder:
    1. [[Search Loop|Search-Options#search-loop]] functions are typically stored in `src.algorithm.search_loop`,
    2. [[Step|Search-Options#step]] functions are typically stored in `src.algorithm.step`.

Numbers 1-6 above are hard-coded into the PonyGE2 framework, i.e. these various components of PonyGE2 *must* be stored in their respective folders. However, optional operators have more freedom in the PonyGE2 framework (this will be explained presently).

For all specified arguments listed above, PonyGE2 automatically parses the correct path. Since numbers 1-6 above are hard-coded, only the direct names of the desired modules or functions within the respective folders/modules need to be specified. For example, when specifying a grammar file, one simply needs to specify the path to the file itself *from within the `grammars` folder*, i.e. `letter.bnf` or `supervised_learning/Dow.bnf`. When specifying a fitness function, one simply needs to specify the name of the function class itself *from within the `fitness` module*, i.e. `string_match` or `supervised_learning.regression`. There is no need to specify the containing folder or module (e.g. `grammars` or `fitness` in the above two cases, respectively). Doing so will produce an error.

However, operators and search framework functions (as listed in points 7 and 8 above) have more freedom in where they can be saved. These functions can either be located in their respective default `operators` or `algorithm` modules, or in any desired location within the overall PonyGE2 `src` directory.

If you are importing operators that are saved in their respective default locations, there is no need to specify the full module path; you can simply just specify the name of the desired operator function. For example, if you are seeking to use PonyGE2's built-in [[subtree crossover|Variation#subtree]] function, then you simply need to specify the function name itself, `subtree`, for the relevant parameter input (e.g. `CROSSOVER: subtree` in a parameters file, or `--crossover subtree` from the command line). To summarize, if a single function name is specified, PonyGE2 will attempt to load that function from its default location.

On the other hand, you can specify full module paths (from within the `src` directory) to operator or search functions should you so desire. This enables if you to import these methods from any location within the `src` directory. For example, if you are seeking to use PonyGE2's built-in [[Late-Acceptance Hill Climbing|Search-Options#hill-climbing]] function, you can specify `algorithm.hill_clilmbing.LAHC_search_loop` for the relevant parameter input (as above). PonyGE2 will attempt to import the module directly from the full specified path, and if this fails, it will then try to import the module from the `algorithm` module location (taking the specified path into account).

If PonyGE2 cannot find the specified functions or modules, error reports will be printed out specifying exactly where PonyGE2 is trying to search. Should they appear, these reports should help you to diagnose what has gone awry.

# Adding New Parameters

---

It is possible to add new parameters to the `algorithm.parameters.params` dictionary either by editing the dictionary directly (this is not recommended), or simply by specifying any desired parameters in a parameters file. Since parameters files are used to set new or over-write existing key:value pairs in the parameters dictionary, it is possible to specify any number of new/unseen parameters in a parameters file (even if they do not already appear in the default `algorithm.parameters.params` dictionary). Such new or custom parameters can then be accessed throughout the entire PonyGE2 code base. However, it is not currently possible to specify new parameters directly from the command line. This is because any existing parameters that were merely mis-spelled from the command line would then create new parameters, and the evolutionary system would not perform as expected. Thus, any parameters entered from the command line that do not exist in the command line parser in `utilities.algorithm.command_line_parser` will produce an error. Of course, it is entirely possible to modify the command line parser to accept new arguments.

***NOTE*** *that parameter names should follow the naming convention demonstrated with the existing parameters. All names should contain only uppercase letters and underscores. Avoid the use of spaces where possible and do not use a colon (`:`) in the name of a parameter.*