Besides the main `PonyGE.py` file that can be found in the source directory, a number of extra scripts are provided with PonyGE2. These are located in the `scripts` folder. These extra scripts have been designed to work either as standalone files, or to work in tandem with PonyGE2. Various functions from within these scripts can provide extra functionality to PonyGE2.

Basic Experiment Manager

A basic experiment manager is provided in the `scripts` folder. This experiment manager allows users to execute multiple evolutionary runs across multiple cores using python's `multiprocessing` library. Experiments are saved in `results/[EXPERIMENT_NAME]` where `[EXPERIMENT_NAME]` is a parameter which specifies the name of the experiment. This can be set with the argument:

```
--experiment_name [EXPERIMENT_NAME]
```

or by setting the parameter `EXPERIMENT_NAME` to `[EXPERIMENT_NAME]` in either a parameters file or in the params dictionary, where `[EXPERIMENT_NAME]` is a string which specifies the desired name of the experiment.

***NOTE** that the `EXPERIMENT_NAME` parameter **must** be set when using the experiment manager.*

The number of evolutionary runs to be executed can be set with the arguemt:

```
--runs [INT]
```

or by setting the parameter `RUNS` to `[INT]` in either a parameters file or in the params dictionary, where `[INT]` is an integer which specifies the number of evolutionary runs to be completed. The experiment manager initialises each evolutionary run with a different unique random seed. The random seeds for a batch of evolutionary experiments are the indexes of the individual experiments (i.e. the first run will have seed 0, the second will have seed 1, and so on up to seed `[INT] - 1`.)

***NOTE** that the experiment manager uses pythons `multiprocessing` library to launch multiple runs simultaneously. As such, it is not possible to use the experiment manager with the `MULTICORE` parameter set to `True`. If the `MULTICORE` parameter is already set to `True`, it will be turned off automatically.*

Since python uses the central `algorithm.parameters.params` and `stats.stats.stats` dictionaries to manage various aspects of individual runs, it is not currently possible to launch multiple simultaneous runs of PonyGE2 from within a Python environment as the central dictionaries would be overwritten by the concurrent processes. As such, the experiment manager calls individual PonyGE2 runs from the command line using Python's `subprocess.call()` function.

***NOTE** that all functionality available to the main PonyGE file is available to the experiment manager, i.e. all command line arguments can be used including the specification of parameters files.*

To run the experiment manager, type:

```
$ python scripts/experiment_manager.py --experiment_name [EXPERIMENT_NAME] --runs
[INT]
```

where `[EXPERIMENT_NAME]` is a string which specifies the desired name of the experiment and where `[INT]` is an integer which specifies the number of evolutionary runs to be completed.

***NOTE** that since the `[MULTICORE]` parameter in PonyGE2 does not work with Windows operating systems, at present the experiment manager will not work with Windows operating systems.*

Post-run Analysis

A basic statistics parser is also included in the `scripts` folder. This parser is called automatically by the experiment manager upon successful completion of all specified runs, but can also be called on its own if so desired. The statistics parser generates summary `.csv` files and `.pdf` graphs for all stats generated by all runs saved in an `[EXPERIMENT_NAME]` folder.

The statistics parser extracts the `stats.tsv` files from all runs contained in the specified `[EXPERIMENT_NAME]` folder. For each stat, a unique `.csv` file is generated containing that statistic across all stats files. Average and standard deviations for each stat are calculated, and graphs displaying the average values (with standard deviations) across all generations are generated. Finally, the statistics parser saves a main `full_stats.csv` file containing all statistics across all runs in a single file. All `.csv` summary files can be used with any numerical statistics package, such as R.

While the experiment manager calls the statistics parser after all experiments have been completed, it is possible to call the statistics parser as a standalone program to generate these files for any given `[EXPERIMENT_NAME]` folder. This can be done from the command line by typing:

```
$ python scripts/parse_stats.py --experiment_name [EXPERIMENT_NAME]
```

where `[EXPERIMENT_NAME]` is a string which specifies the desired name of the experiment contained in the `results` folder.

GE LR Parser

A powerful script that has been included with PonyGE2 is the deterministic GE LR Parser. This script will parse a given target string using a specified `.bnf` grammar and will return a PonyGE2 individual that can be used in PonyGE2. Provided the target string can be fully and correctly represented by the specified grammar, the LR parser uncovers a derivation tree which matches the target string by building the overall tree from the terminals used in the solution. A repository of phenotypically correct sub-trees whose outputs match portions of the target string (termed 'snippets') is compiled. Deterministic concatenation operators are employed to build the desired solution. Provided the grammar remains unchanged, these reverse-engineered solutions can be saved and used in an evolutionary setting.

Since the GE LR Parser is fully deterministic, the same GE individual will be returned every time it is executed.

To run the GE LR Parser, only two parameters need to be specified:

```
--grammar_file [FILE_NAME.bnf]

--reverse_mapping_target [TARGET_STRING]
```

where `[TARGET_STRING]` is a string specifying the target string to be parsed by the GE LR Parser.

***NOTE** that the full file extension for the grammar file (e.g. ".bnf") **must** be specified, but the full file path for the grammar file (e.g. `grammars/example_grammar.bnf`) **does not** need to be specified.*

Alternatively, both the `GRAMMAR_FILE` and `REVERSE_MAPPING_TARGET` can be specified in either the `algorithm.parameters.params` dictionary or in a separate parameters file. An example parameters file can be seen in the parameters folder. To run this example, type:

```
$ python scripts/GE_LR_parser.py --parameters GE_parse.txt
```

Grammar Analyser

A simple grammar analyser has been provided with PonyGE2. Given a specified grammar file, this analyser will simply print out the number of unique combinations and permutations of phenotypes that are possible at a number of derivation tree depths.

Only one parameter needs to be specified with the grammar analyser:

```
--grammar_file [FILE_NAME.bnf]
```

where `[FILE_NAME.bnf]` is the name of the desired grammar file (including file extension).

The number of derivation tree depths across which permutations are displayed can be set with the argument:

```
--permutation_ramps [INT]
```

or by setting the parameter `PERMUTATION_RAMPS` to `[INT]` in the default params dictionary, where `[INT]` specifies the number of desired derivation tree depths.

***NOTE** that the `PERMUTATION_RAMPS` parameter specifies the number of derivation tree depths across which permutations will be shown, starting from the minimum path of the grammar. It does not necessarily specify the maximum derivation tree depth that permutations are to be calculated to.*

To run the Grammar Analyser, simply type:

```
$ python scripts/grammar_analyser.py --grammar_file [FILE_NAME.bnf]
```

NOTE *that parameters files cannot be specified with the grammar analyser.*