MATH20622 Python Course

Lecturer: Dr Stefan Guettel   Delivery: 2018   Nr of students: 99

Marking Scheme

The overall mark consists of lab test results (30%) and the final coursework (70%).

The deadline for submitting the coursework was 10/05/2018 at 13:00. The submission system stayed open until 14:00 that day and 94 submissions have been received.

Each coursework submission is a `stocktrader.py` text file and it has been marked based on four criteria:

1.) Unit testing and manual inspection (65%): a Python script runs 36 unit tests for testing each of the nine tasks, each giving either 0 or 1 mark, depending on whether a test is completed successfully or failed. The maximal number of marks on unit tests is 36. The codes were also inspected manually, and in some cases, minor modifications were made to improve the marks for unit testing.

2.) Docstrings (15%): using `print(functionname.__doc__)` the docstrings associated with the functions in Tasks 1-5 were extracted and for each of them 0-2 marks given, based on the length of the docstring as well as the content. To get full marks, a docstring must be absolutely clear about the inputs and outputs of its function and the possible exceptions being raised. Also, if the function opens/writes some files, the docstring should mention this. A well-written docstring should reveal all info required to use the function without looking at the code itself.

3.) Code efficiency (20%): The code is minified using the `pyminifier` module, which removes all comments and combines print commands and dictionary definitions spanning several lines. Also, any code related to `tradeStrategy2()` was excluded. The number of lines in the resulting code is put in relation to the marks gained in part 1., and scaled to a number between 0 and 1, giving a score for the code efficiency (essentially, functional marks per line of code).

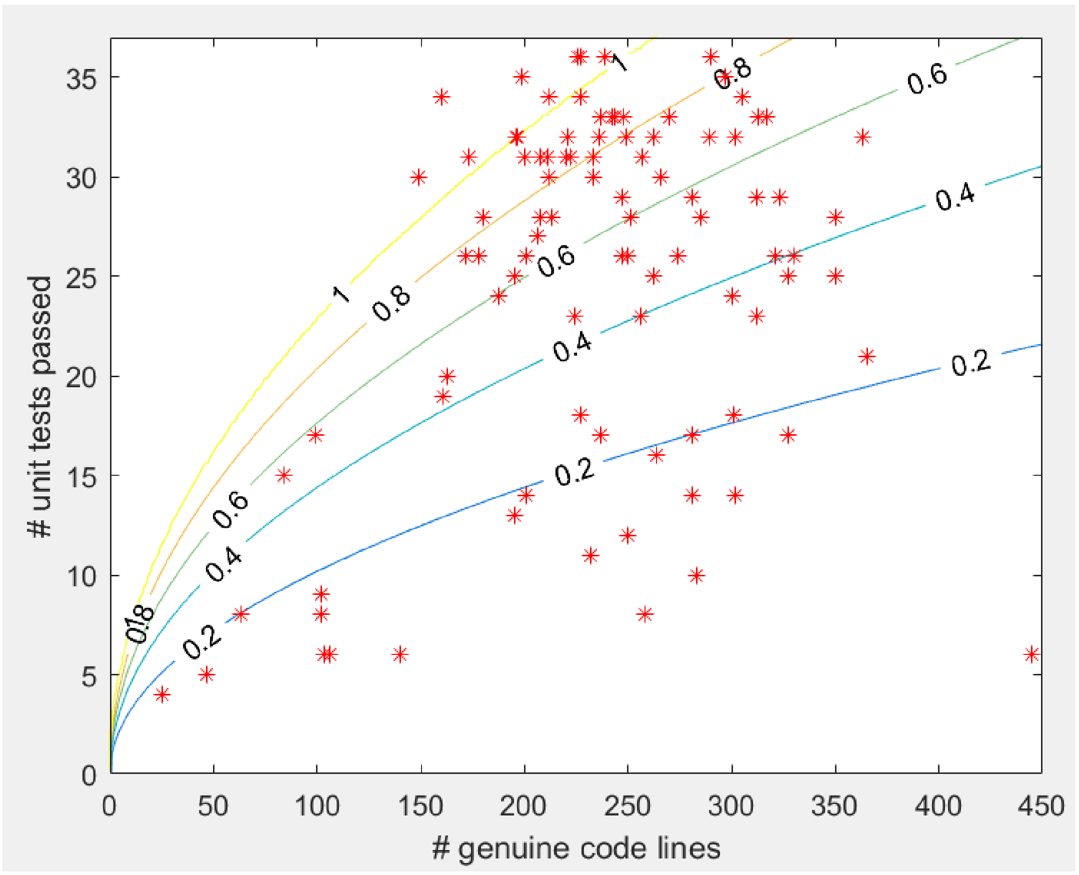
Feedback

Many students did a good job and some came up with sophisticated and rather efficient codes. Common problems where marks have been lost were:

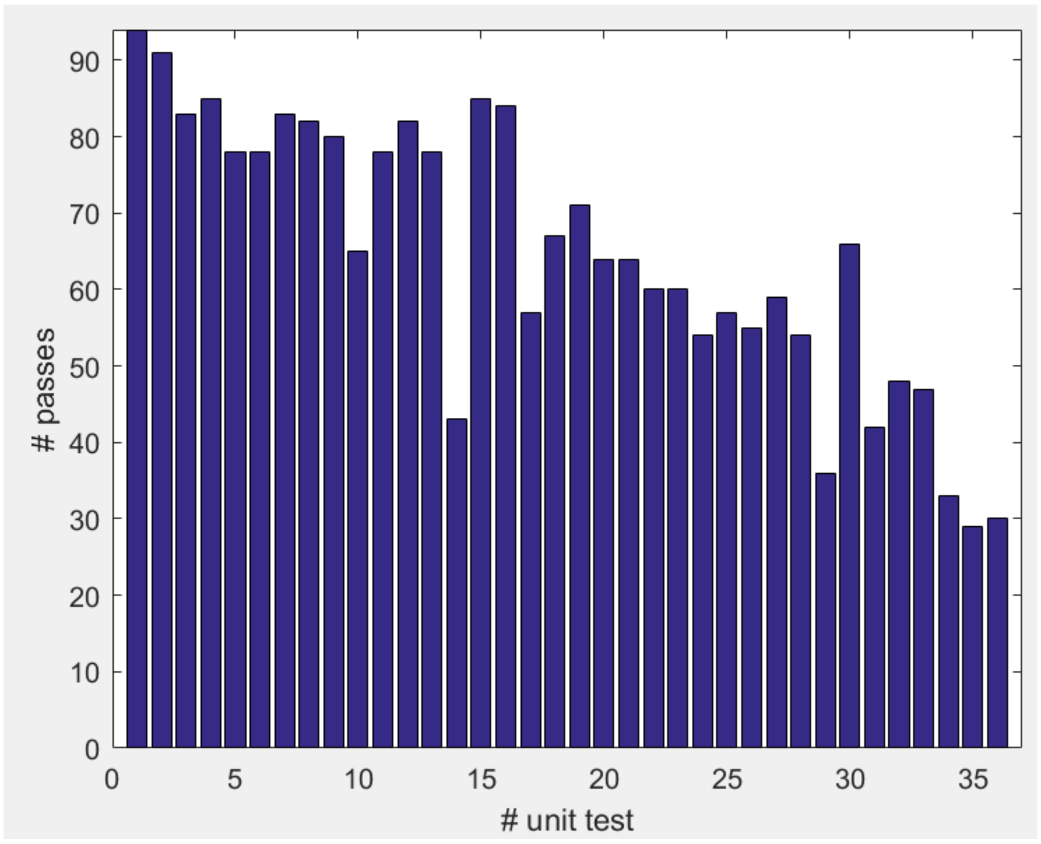
- The names of the functions or the number of input arguments, or the used data structures, do not follow exactly the coursework description. For example, `loadPortfolio()` should be callable without any input parameters, in which case 'portfolio.csv' is assumed. Also, `stocks` has to be dictionary of dictionaries exactly as specified in the coursework description.
- There was some confusion about `fname` and `symbol` being provided with or without the CSV extension, respectively. The coursework specifies that `fname` is a filename of a portfolio CSV which also contains the file extension, whereas symbol is just a string without the ".csv" part.

- Some modules appeared to write to the files in the stockdata subfolder, but there is no task that would require this. Also, some modules attempted to change the current working directory (using `os.chdir`). This might cause serious unexpected behaviour when the module is used in combination with others, and it is strongly recommended to avoid this.
- Some programs crashed on invalid input arguments or did not raise exceptions as required. Exceptions are raised with the `raise` keyword; this is different from just printing to the console.
- Marks have often been lost due to lacking or incomplete docstrings. It was strictly required to document all inputs and outputs of each function, including the data types of the arguments, and to mention which exceptions might be raised. Docstrings must appear inside their functions, immediately after the line with the `def` keyword.
- The code is too long and hence inefficient. The whole coursework could be solved in about 250 lines of genuine code (i.e., excluding empty lines, comments, docstrings, `tradeStrategy2()`, and commands broken over multiple lines).

The following plot visualizes the code efficiency of all 94 submissions. Points in the top-left are very efficient (few lines of code, good functionality), whereas codes in the bottom-right are inefficient (many lines of code but little functionality). The quadratic level lines show the corresponding scores for code efficiency.



The final graph shows how many of the 94 codes have passed each of the 36 unit tests. It proves that all unit tests were solvable. The most difficult unit test, passed by only 29 students, is number 35. This test runs `tradeStrategy1()` using 'SKY' and 'EZJ' shares and then verifies the final portfolio value.


# passes
# unit test