MATH20622 Python Course

Lecturer: Dr Stefan Guettel Delivery: 2016 Nr of students: 50

Marking Scheme

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

The deadline for submitting the coursework was 12/05/2016 at 12:00 noon and all students have submitted on time. The submission system stayed open until 12/05/2016 at 13:00.

Each coursework submission is a supermarket.py text file and it has been marked semi-automatically based on four criteria:

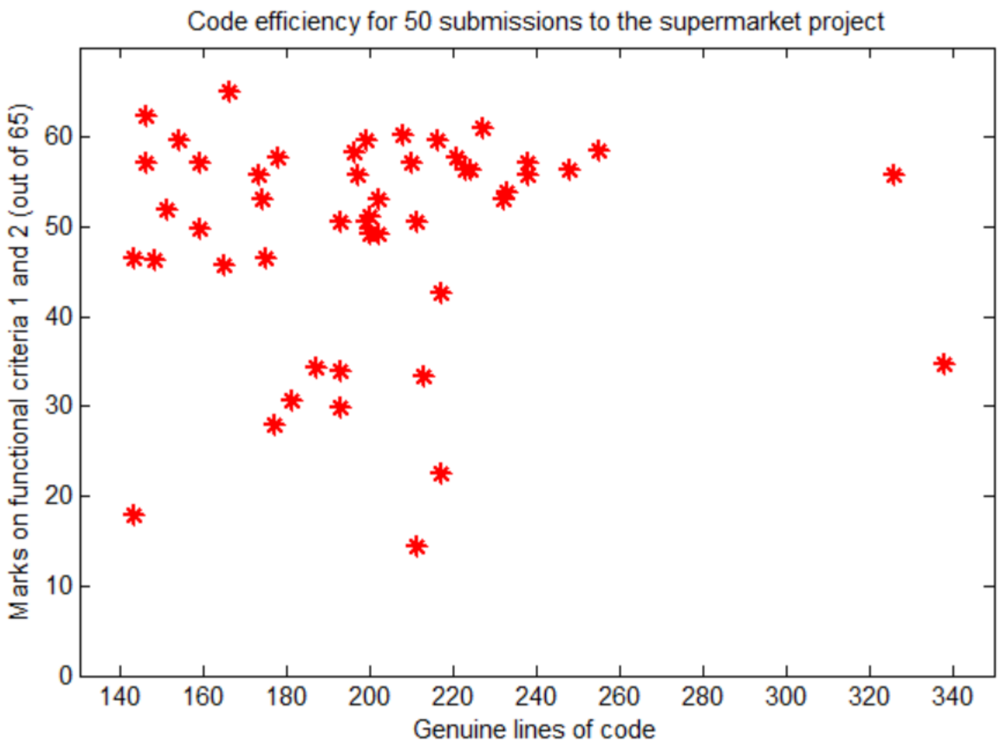
1.) Unit testing (50%): another Python script is prepended to the original code and runs 38 unit tests, each giving either 0 or 1 mark, depending on whether a test is completed successfully or it failed. The number of lines that needed to be changed in order to make at least some of the tests run is subtracted from the 38 marks.
2.) Manual testing (15%): by running the originally submitted code, eleven predetermined transactions will be tested, each giving between 0 and 2 marks.
3.) Documentation (15%): the seven functions as well as the main program are assessed on their documentation, giving 0-2 marks in each case. In particular, the inputs and outputs of each function need to be explained explicitly.
4.) 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. The number of lines in the resulting code is a measure for the length of the code. This is put in relation to the marks gained in part 1 and 2 above, and scaled to a number between 0 and 1, giving a score for the code efficiency (essentially, functional marks per line of code).

Feedback

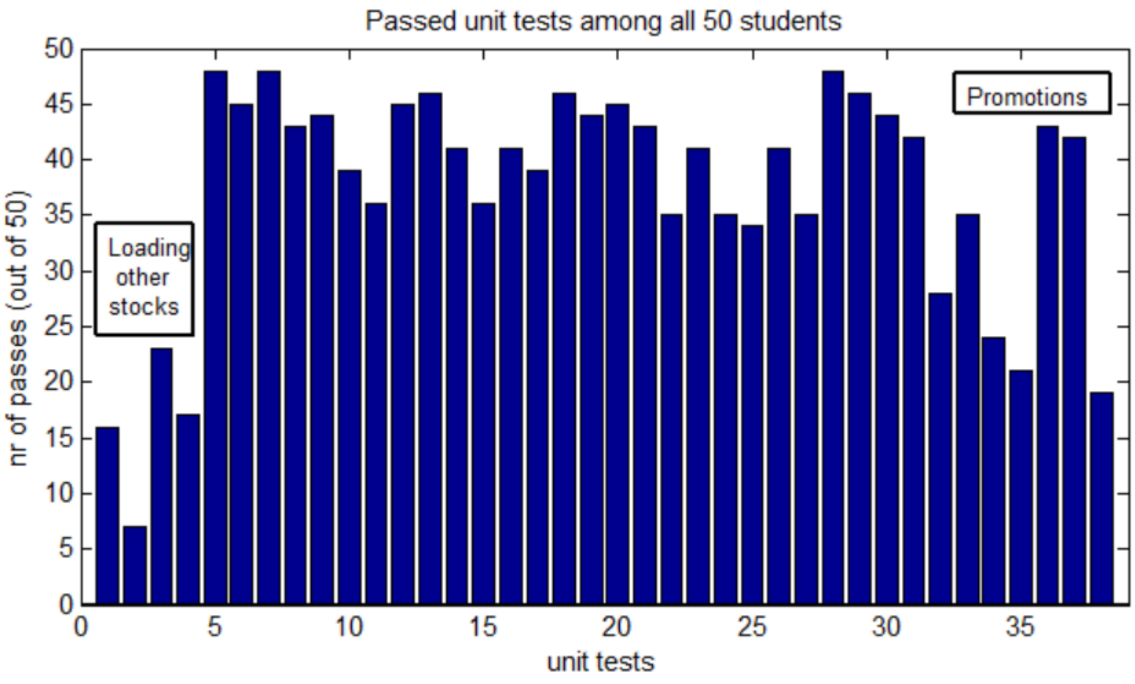
Most students did a great job and some came up with very sophisticated and efficient codes! Common problems where some marks may have been lost were:

- The names of the functions or the number of input arguments do not follow exactly the coursework description. For example, `loadStockFromFile()` is not supposed to take any input arguments per coursework specification.
- The loading function fails when given another stock.csv file. Corrupted lines which do not match the format outlined in the coursework description should just be skipped, and not break the code.
- Items in unit “pieces” formatted with decimals in the table. Also these items should only be stored and operated on in integer amounts.
- Program crashes on invalid inputs.
- Promotion get4pay3 works only for 4 groups.
- Promotion get4pay3 does not find the cheapest item per group.
- Documentation lacking, e.g., input parameters not specified correctly.
- The code is too long and hence inefficient. The whole coursework could be solved in about 160 lines of genuine code (i.e., excluding empty lines, comments, docstrings, and commands broken over multiple lines). See the plot below.

The following plot visualizes the code efficiency of all 50 submissions. Points in the top-left are very efficient (few lines of code, good functionality), whereas codes in the bottom-left are inefficient (many lines of code but little functionality).



The next plot shows how many students have passed each of the 38 unit tests. It indicates that all unit tests were solvable. The most difficult unit test, only passed by 7 students, is number 2. This test loads a modified stock3.csv file with some corrupted entries and then checks whether the returned stock dictionary has the correct length.


Passed unit tests among all 50 students
nr of passes (out of 50)
unit tests
Loading other stocks
Promotions