DrPython–WEB: a tool to help teaching well-written Python programs

Tommaso Battistini[1], Nicolò Isaia[1],
Andrea Sterbini[1][0000−0002−5361−2786], and
Marco Temperini[2][0000−0002−8597−4634]

[1] Computer Science Dept. - Sapienza University of Rome - Italy
sterbini@di.uniroma1.it
[2] Dept. of Computer, Control and Management Engineering
Sapienza University of Rome - Italy
marte@diag.uniroma1.it

Abstract. A good percentage of students, while learning how to program for the first time in a higher education course, often write very bad code, i.e. code which is difficult to read, badly organized, not commented. Writing inelegant code reduces the student's professional opportunities. In this paper we present DrPython–WEB, a web application capable to automatically extract linguistic, structural and style-related features, from students' programs, and to grade them with respect to a teacher-defined assessment rubric. The aim of DrPython–WEB is to make the students accustomed to good coding practices, and stylistic features, and make their code better. There are other systems able to perform code analysis through quality measures: The novelty of DrPythonWEB, with respect to such systems, is in that it analyzes also linguistic and stylistic features.

Keywords: Teaching programming · Python · Feature extraction · Good coding practices.

1 Introduction

One of the main tasks of a computer programming course is to allow the students to reach an adequate level of skills, so to be able to produce good quality programs. The difficulty of accomplishing such a task is particularly felt in Higher Education on Computer Science, as the students in that area will become, in a relatively close future, professionals with important responsibilities in private and public sectors [1–3].

Students' skills, to produce programs showing good, or even high, "quality", are acquired through practice, and are applied to various aspects of programming, ranging from the capability to define suitable algorithms to solve a given problem, through the ability to design a program and the relevant data structures, to the practical coding abilities, that allow a student to produce a readable

program, i.e. a program whose instructions are 1) textually formatted in a readable fashion, 2) easy to interpret, as far as their purposes are concerned, and 3) as clearly commented as possible.

Both Learning and Teaching of Computer Programming are challenging tasks, when the traditional approach to education is used [4]. Hence, the availability of a web-based, automated support can be of great value, especially in Higher Education, where often direct interactions between a student who is solving a programming task, and a teacher who could help, are not easily possible [5].

In this paper, we present a web-based system, DrPython–WEB, whose use could help a student to improve her/his coding skills, by pointing out and recognizing the "elegance" of the student's code, in an automated and real-time fashion.

By "elegance" we mean a subset of the several qualities of a program, mentioned earlier, related to structure, readability and maintainability. On these aspect DrPython–WEB focuses its program analysis, and evaluation. In particular, given a program, the analysis is performed on a set of features, extracted from the program (see later), as well as on the good naming quality of the identifiers (i.e., the names given by the programmer to certain structures of the program, such as types, variables, and functions).

We developed DrPython–WEB based on a twofold aim: on the one hand we would like to help encouraging students to practice and improve their coding style; on the other hand we wanted to support both student's awareness and teacher's assessment procedures, by providing them with visual summaries of data, reporting the elements on which the overall evaluation of the code was based.

DrPython–WEB is still undergoing a thorough experimentation, and we cannot yet present a comprehensive analysis of the actual effects of its use for the student and for the teacher. So, in this paper we present the system, and its features, showing how we used it on a relatively large dataset of programs (produced by students during a recent edition of a course on *Basics in Programming* held in our University). Such dataset is comprised of programs produced to solve tasks related to the several mandatory homework requested during the course, and the solutions submitted for final exams.

The main goals in this paper are then the following:

Goal 1: *To show that DrPython–WEB can automatically extract the stylistic features of a program, and assess their usage to push students towards a better programming style.*
We will see that DrPython–WEB is able to 1) perform an automatic check of the hundreds of programs in our sample, 2) analyze, in such programs, the coding qualities we associated above to "elegance", and 3) express a quality grade for each program.

Goal 2: *To personalize the assessment depending on the teacher's preferences, the moment in the course, or just the specific assignment's characteristics.*
In this respect, we will see that the analysis performed by DrPython–WEB can be configured by the teacher, who is able to finely-tune the assessment

by specifying her/his preferences about the features to be taken into consideration, and their weight in the computation of the overall quality grade. In particular, the possibility to configure the assessment undertaken by DrPython–WEB allows the teacher to adapt the analysis of a given batch of programs, depending on the relevant characteristics of a given task, and/or the aspects to be taken care of at a given point-in-time of the course.

In the following sections we will:

1. present the software library *DrPython*, which we developed to provide core functionalities for the analysis of a program: On these functionalities DrPython–WEB was developed.
2. present the use of DrPython–WEB on a set of sample programs, in order to show the characteristics of the system and see its potential application on the field.
3. present some conclusions, submitting that DrPython–WEB, although subject to further improvements, can be an effective means to persuade the students to improve their coding style.

2 DrPython: feature extraction module

DrPython–WEB is based on the feature extraction library (named DrPython) that we developed to analyze the student's program and algorithm description to recognize/extract three type of features:

- **code syntax features:** the number of specific language constructs in the program: (functions, classes, super-classes of each class, methods, try-except, list-comprehensions, if-then-else, generators, lambda, recursive functions, variables, arguments),
- **code quality measures:**
  - McCabe's cyclomatic complexity [6], that captures how much a function control flow is intricate,
  - Halstead's measures [7], that captures a function's conceptual complexity from its vocabulary size and number of operators used
  - *code smells* [8], i.e., code structures that often imply bad coding practices
- **linguistic features:**
  - *good identifiers*, i.e., self-explanatory names that convey the meaning of their function. This relieves the programmer from having to recall what type of data is in a variable and its place in the algorithm, as well as the action performed by a function/method,
  - *good documentation practices* i.e. using comments and doc-strings to describe the reason for particular programming choices. This helps the reader to better understand the meaning of the algorithm implemented.
  - the usage of *pertinent keywords* related to the exercise description both in comments/doc-strings or in the algorithm description. This allows DrPython to automatically check (roughly) if the documentation is adequate to the task.

The code syntax features are extracted/counted by means of the **redbaron**[3] source code analysis library that allows to easily query the code structure for specific constructs. Redbaron queries use a syntax similar to CSS selectors (as it's done in jQuery w.r.t. the DOM of HTML pages). This in turn will allow us to easily expand in future the set of code syntax features extracted.

The code quality measures are computed by means of the **radon**[4] library.

Finally, to extract the linguistic features DrPython uses the automatic term extraction module **pyate** [9] to select the 25 highest ranked keywords returned by its **Combo Basic** algorithm [10], and the text analysis library **spacy**[5] to analyze the documentation/comments and the algorithm description. To decide if a particular identifier used by a student is of good/medium/bad quality, DrPython performs the following steps:

- it extracts the pertinent keywords with pyATE from the teacher's exercise task description
- it decomposes the identifier in its component words
- it compares the words (by means of Spacy semantic similarity and the WordNet semantic network) to grade their similarity to the pertinent keywords
- it classifies the identifier in the top/medium/bad group depending on having its max similarity to a keyword above 90%, between 40% and 90% or lower than 40%, respectively.

DrPython can be used both as a stand-alone program, to be run from the command line, or integrated in the DrPython–WEB web-based application described below.

For example, with DrPython one can analyze many student files and collect all extracted features as a CSV file, and one can study, for example:

- how the extracted features correlate with each other or with other data (exam grades or readability judgements manually collected)
- how different assessment templates will produce different grade distributions

To make the assessment templates more easy to use, and to automate the submission and assessment of the programs, we have developed the web-based application (DrPython–WEB).

3 Dr.Python-WEB: The System

The DrPython–WEB system allows the teacher to define one or more **assessment templates** to grade the submitted programs/algorithms depending on the features extracted, in order to encourage students to use more readable Python constructs, a better linguistic style, and to better modularize their code.

DrPython–WEB is a classic LAMP based web-application written in Python where:

[3] https://redbaron.readthedocs.io
[4] https://radon.readthedocs.io
[5] https://spacy.io

Homework 1 (Not mapped)	Edit template	Show me all student results	Show me all student evaluations
Feature	**Weight value**	**Minimum range**	**Maximum range**
Effort	1.5	5.0	100.0
Effort	1.0	100.0	500.0
Effort	0.5	500.0	5000.0
Cyclomatic complexity	0.8	2.0	15.0
Cyclomatic complexity	0.5	15.0	30.0
% Good identificators	1.2	70.0	100.0

Fig. 1. Assessment template that awards more points for lower cyclomatic complexity, lower Halstead's effort and high percentage of good identifiers

- the teacher defines assessment templates depending on the exercise and/or the course phase
- the students submit their code to get the style assessment grade and compare their results with each other's

Assessment templates are defined by the teacher by specifying what are the features assessed and what is their weight for a given range of values.

In figure 1 we show an assessment template that awards more points to a lower *Halstead's effort*, to a lower *cyclomatic complexity*, and to a high *percentage of good identifiers* depending on the range of values observed.

A template like this one, for example, is built to convince a student to modularize her program into smaller less complex functions (with lower cyclomatic complexity), to write more readable code (using mainly self-explanatory identifiers) and with a less complex algorithm (with lower Halstead's effort).

Notice that an assessment template can assign different weights to different range of feature values extracted, as shown in the figure, where we show three different ranges for the Halstead's effort measured. This way, the teacher could associate to each feature a weight function with complex shape.

After assessment the students' results are shown in the DrPython–WEB leaderboard, so that each student can compare her program style with others, as shown in fig. 2

Notice that the only features shown are those included in the assessment template.

Exercise 1								
		% Good Identificators		Cyclomatic complexity		Effort		
Student	Name	Feature value	Feature evaluation	Feature value	Feature evaluation	Feature value	Feature evaluation	Total
student0	program01.py	85.71	1.2	3.0	0.8	49.76	1.5	3.5
student2	program01.py	20.0	0.0	2.0	0.8	6.96	1.5	2.29
student3	program01.py	42.85	0.0	3.0	0.8	241.76	1.0	1.8
student4	program01.py	11.11	0.0	5.0	0.8	143.25	1.0	1.8
student6	program01.py	25.0	0.0	4.0	0.8	30.31	1.5	2.29
student7	program01.py	11.11	0.0	4.0	0.8	68.33	1.5	2.29
student8	program01.py	80.0	1.2	4.0	0.8	167.59	1.0	3.0
student12	program01.py	60.0	0.0	3.0	0.8	182.64	1.0	1.8
student13	program01.py	60.0	0.0	4.0	0.8	61.02	1.5	2.29
student14	program01.py	23.07	0.0	6.0	0.8	52.0	1.5	2.29
student15	program01.py	20.0	0.0	4.0	0.8	48.0	1.5	2.29
student16	program01.py	20.0	0.0	4.0	0.8	15.5	1.5	2.29
student17	program01.py	50.0	0.0	4.0	0.8	30.31	1.5	2.29
student18	program01.py	50.0	0.0	3.0	0.8	474.54	1.0	1.8
student19	program01.py	60.0	0.0	4.0	0.8	69.3	1.5	2.29

Fig. 2. Leaderboard example, showing the features checked for this exercise and the points assigned as defined in the previous assessment template.

4 Conclusions and future work

We have shown a novel library (DrPython) which extracts structural, quality and linguistic features from the programs and documentation submitted by students. DrPython is used within the novel DrPython–WEB application, that allows the teacher to build assessment templates specific both to the point in time during the course and/or to the specific exercise.

We plan to use the DrPython–WEB system on our next courses to collect data on the student's submissions and check that its usage improves the student's program quality.

Moreover, we plan to collect readability assessments from the students during the course to study both how the exercise readability improves with time and how the code readability perception of the students changes while they are learning.

From the collected data we intend to study if we can define a program readability measure that takes into consideration the linguistic features also.

Finally, we intend to study how the readability of a program is related to its grade, and/or to the grade received in the final lab-based exam.

References

1. Breuker D M, Derriks J, Brunekreef J. Measuring static quality of student code. In: Proceedings of the 16th Annual Joint Conference on Innovation and Technology in

Computer Science Education. 2011, 13–17
2. Yao LU, Xinjun MAO, Tao WANG1, Gang YIN, Zude LI. Improving students' programming quality with the continuous inspection process: a social coding perspective Front. Comput. Sci., 2020, 14(5): 145205 https://doi.org/10.1007/s11704-019-9023-2
3. Radermacher A, Walia G, Knudson D. Investigating the skill gap between graduating students and industry expectations. In: Proceedings of the 36th International Conference on Software Engineering Companion. 2014, 291–300
4. Feldman Y A. Teaching quality object-oriented programming. Technology on Educational Resources in Computing, 2005, 5(1): 1
5. Chen W K, Tu P Y. Grading code quality of programming assignments based on bad smells. In: Proceedings of the 24th IEEE-CS Conference on Software Engineering Education and Training. 2011, 559
6. McCabe (December 1976). "A Complexity Measure". IEEE Transactions on Software Engineering (4): 308–320. https://doi.org/10.1109/tse.1976.233837
7. Halstead, Maurice H. (1977). Elements of Software Science. Amsterdam: Elsevier North-Holland, Inc. ISBN 0-444-00205-7.
8. https://wiki.c2.com/?CodeSmell
9. Lu, Kevin. (2021, June 28). kevinlu1248/pyate: Python Automated Term Extraction (Version v0.5.3). Zenodo. http://doi.org/10.5281/zenodo.5039289
10. Astrakhantsev, N.: Methods and software for terminology extraction from domain-specific text collection. Ph.D. thesis, Institute for System Programming of Russian Academy of Sciences (2015)