PILeT: an Interactive Learning Tool To Teach Python

Bedour Alshaigy
Department of Computing and Communication Technologies
Oxford Brookes University
Oxford, United Kingdom
12012361@brookes.ac.uk

Samia Kamal
Department of Computing and Communication Technologies
Oxford Brookes University
Oxford, United Kingdom
skamal@brookes.ac.uk

Faye Mitchell
Department of Computing and Communication Technologies
Oxford Brookes University
Oxford, United Kingdom
frmitchell@brookes.ac.uk

Clare Martin
Department of Computing and Communication Technologies
Oxford Brookes University
Oxford, United Kingdom
cemartin@brookes.ac.uk

Arantza Aldea
Department of Computing and Communication Technologies
Oxford Brookes University
Oxford, United Kingdom
aaldea@brookes.ac.uk

ABSTRACT

This paper describes a planned investigation into how learning styles and pedagogical methodologies can be embedded into an e-learning tool to assist students' learning programming. The objective of the research is to test the hypothesis that combining multiple teaching methods to accommodate different learners' preferences will significantly improve comprehension of concepts, which in turn increases students' confidence and as a consequence performance in programming. An interactive learning tool to teach Python programming language to students, called PILeT, has been developed to test the hypothesis. The tool aims to be adaptable to the students' learning style and as such it will teach programming using several techniques (e.g. visual, textual, puzzles) to appeal to each preference. PILeT is suitable for secondary school students or teachers wishing to undertake CPD (Continuing Professional Development). PILeT will be tested on first year undergraduate students at Oxford Brookes University.

CCS Concepts

•**Social and professional topics** → **Computer science education;** *Student assessment;*

Keywords

Computer science education, e-learning tools, introductory programming

1. INTRODUCTION

For many students, programming is considered the most difficult task in computer science classes. Although several factors have been attributed to students' failure to acquire programming skills such as problem solving abilities, self-efficacy and mental models to name a few [11], we are far from fully understanding the underlying reasons behind different progression rates amongst them. Evidence from the literature review strongly implicated teaching techniques adopted whilst teaching programming [2], students' learning preference [6], in addition to the choice of the first programming language taught [7].

With the prevalence of mobile devices and e-learning, many instructors are in favour of using innovative courseware in teaching [5]. Some of these tools have been proven effective in improving students' retention and engagement by using games to teach a concept [3] or providing a user friendly environment with less cryptic error messages and feedback to support students' programming [8]. However many fail to address critical cognitive and programming skills, and do not take into account the different learning styles found in a diverse cohort of students.

This paper introduces PILeT, a web-based interactive tool designed to teach Python to anyone with an interest in learning programming. The tool's novel contribution is that it will offer a combination of pedagogical methods to support the student's learning style. Therefore, each programming concept will be explained using videos, reading material, examples, exercises and puzzles adequately on its own or in combination with the other methods. Additionally multiple choice questions are available at the end of each lesson to assess the students understanding of the taught concept. This way, each student can learn from the teaching technique they are most comfortable with, or use a mixture of several ways to support their learning [10].

2. PYTHON INTERACTIVE LEARNING TOOL (PILET)

In light of the literature review findings, a prototype of a web-based, Python Interactive Learning Tool (PILeT) has been developed. The tool incorporates several teaching methods (visualisation, example and exercises, and puzzles) suitable for different learners.

PILeT consists of three layers (Figure 1):

WiPSCE '15, November 09-11, 2015, London, United Kingdom

DOI: http://dx.doi.org/10.1145/2818314.2818319

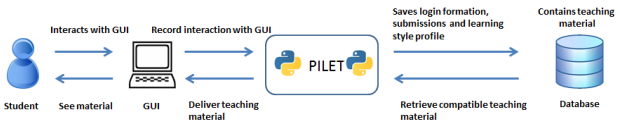

Figure 1: PILeT Architecture

1. A front end graphical user interface (GUI) that displays the teaching material to the user.

2. PILeT system that provides different learning resources to support their learning.

3. A back end database that stores the user's login information, course submissions, their learning style profile, and the different learning resources used to teach the module

Additionally, Google Analytics tracks and measures the student's interactions with the interface such as number of video plays and mouse clicks.

The student accesses the course material using the GUI. PILeT records the student's behaviour with different GUI elements and creates a learning style profile for the student based on those interactions. PILeT then uses the student's profile to select the most suitable teaching material and deliver it to the adaptable interface for the user. Hence, the more the student uses the tool, the higher the match between the material presented to the student and their learning preference. The material is:

- Textual description of the concept. The content was teaching material in form of lecture notes or slides.

- A visual explanation of the concept, either:

  - A video with a live demonstration of the concept.
  - Static tables, flowcharts and illustrations that describe the different states of the program or function activity.
  - Online Python Tutor [4]: an embedded step-by-step animated visualisation of executed code and data structures.

- Executable Python examples (Figure 2) that can be compiled and edited. This encourages students to experiment with the code (e.g. "What happens if I do this instead?" ...).

- Programming exercises that can be implemented in an embedded interpreter, with a sample solution and discussion tab. The sample solution will be revealed to the student after 10 tries. Students are encouraged to talk about different problem solutions and code efficiency in their discussions. The instructor will screen the submissions to ensure that no obvious hints or solutions are posted.

- Parson's programming puzzles [9]: an exercise that involves rearranging code blocks in the correct order (Figure 3). The nature of the exercises promotes reasoning and logical skills and motivates students into learning.

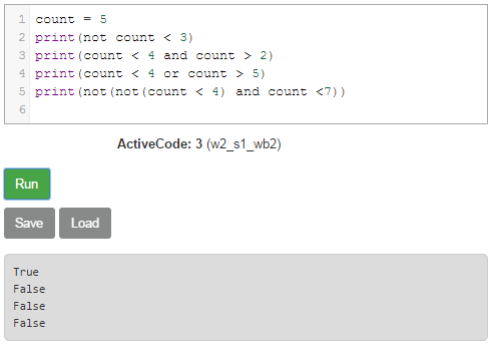

Figure 2: Executable Python Examples

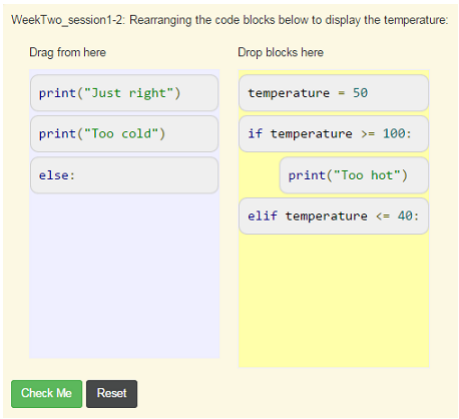

Figure 3: Parson's Programming Puzzles

- Assessment at the end of each session in the form of multiple choice questions and checkboxes. This helps the students revise and retain the learned constructs and to identify any gaps in knowledge where the instructor can provide extra support.

This framework combines several of the most popular programming pedagogies that effectively increases retention and motivation in students with different abilities, and appeals to their learning style. It also allows the instructor to monitor the learners' development and support the struggling group. As such, this framework could be adopted to teach other programming languages such as Java.

3. MODULE DELIVERY MODE

The Module "Modern Computing Technology" is a general introduction to the stages of designing and implementing programs using Python programming language. It is a compulsory module for first year undergraduate students at Oxford Brookes University. The 12 week module consists of two 3 hour practical sessions per week. The module is delivered using the apprenticeship model [1] where students are exposed to a programming concepts using PowerPoint slides then spend time practising these concepts using short Python programming exercises under the supervision of their lecturer. The module also provides a list of recommended books in addition to online resources (e.g. e-books, YouTube). At the end of the practical session, students

are expected to solve a multiple choice quiz and submit the answers to the weekly exercises on the university's virtual learning environment (VLE). While students are advised to take advantage of all of the learning materials available for them, they mostly stick with what is used in the sessions, the slides and exercises, for two reasons: one because that is what the instructor used to teach them in the lab and the other because of all the effort that involves finding just the right material to suit their learning.

4. PRELIMINARY STUDY DESIGN

The study is quantitative in nature. The collected data will be in the form of questionnaire results, exercise submissions and a record of the users' interaction with the website using Google Analytics. In order to prove the hypothesis, three stemmed research questions must be addressed:

- RQ1: are students capable of identifying their own learning style?
- RQ2: does a relationship exist between a specific teaching method and a student's preference in learning programming?
- RQ3: do students adhere to the original learning style they identify with or rely on a combination of learning techniques?

The experiment will take place from week 1 till week 8 in semester 1 in which the students will use PILeT on three separate occasions (during week 1, week 4 and week 7). In each week, they will learn a specific programming concept that will eventually be taught the following week in the lecture (week 2, week 5 and week 8). Their learning progress will be constantly monitored by tracking their online activity and examining their exercise submissions.

4.1 Research Instruments

1. *Index of Learning Style Questionnaire:* developed by Felder-Silverman [12], the questionnaire will be used to identify the learning style students associate themselves with the most. It will be distributed to participants twice; first at the beginning of the study in week 1 and again in week 8. Responses from week 1 will be compared to responses in week 8 to detect any changes in preferences during the semester and therefore answering RQ1.
2. *Google Analytics*: the website interactions will be recoded and analysed to verify which elements (i.e. visual, textual) participants heavily relied on to learn the concepts. The tracking records will be used in combination with their responses to the Felder-Silverman questionnaire to answer RQ2 and RQ3. This will also determine whether some concepts are taught better using a specific teaching technique.
3. *Exercise submissions:* all the students enrolled in the module will be asked to solve compulsory exercises at the end of each lesson. The results of the experimental group (PILeT users) will be compared to the control group (non PILeT users) to measure the differences in performance.

4.2 Participants

Participants in this study will be first year undergraduate students at Oxford Brookes University. The study will be introduced in week 1. Participation is voluntary, however, in order to meet the participation requirements students must commit to using PILeT on the specified occasions. Google Analytics will be used to monitor their use and corroborate with that fact. If they agree to take part, a 10 minutes demonstration will take place at a computer lab to demonstrate the features and functions of PILeT.

4.3 PILeT Implementation

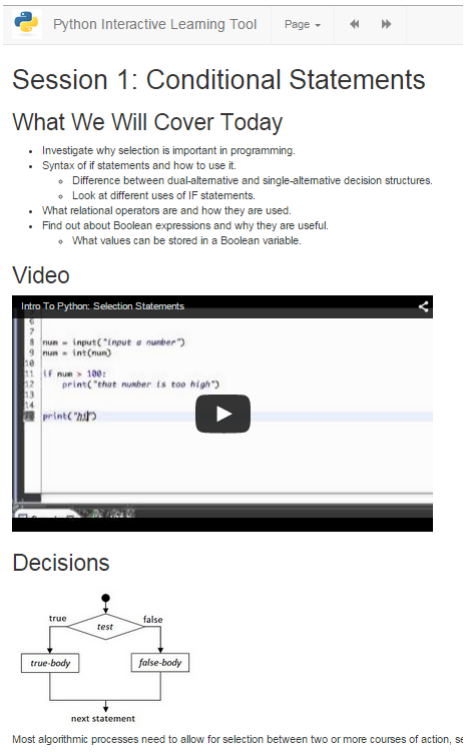

Figure 4: Screenshot of PILeT

New students will be asked to register to PILeT. After submitting profile information (i.e. username, first name, surname, email, password and course name) a personal account is created and students can log in. The course name field is used as a unique field to prevent other users who are not affiliated with the course from accessing it. After logging in, the student has the freedom to navigate between different Python concepts.

When a student is logged in, they can select between 10 Python concepts to learn: Python primers, conditional statements, loops, logic circuits, functions, global and local variables, software development process, searching and sorting, lists and classes and objects. Each concept is broken down into two sessions to prevent the student of being overwhelmed with several new ideas. Sessions are taught using different teaching methodologies.

A normal session starts with an embedded video at the beginning explaining the selected topic ideal for visual learners (Figure 4). This is followed by an alternative which is a written explanation for verbal users. Small executable examples

with nested code are readily available for execution for active learners, and alternatively exercises for reflective students. Sequential learners can use the Python Tutor to view a step by step execution of the code or can use in combination with the written material to reinforce their learning. Each student is required to solve all the exercises in each session in addition to the assessment questions to assess their learning. The submitted answers will be saved in the database. The instructor will be able to mark the questions, provide prompt feedback, and examine the progression rate of student overall.

The first version of PILeT is not personalised, instead it will display all of the different representations of the learning material to examine the effect of combining multiple teaching resources on learning a single programming concept.

5. CONCLUSION AND FUTURE WORK

This paper describes an ongoing preliminary study designed to test the hypothesis that offering a combination of pedagogical methods to accommodate to different learning styles will significantly increase programming comprehension and skills. The literature review that motivated the study implicated teaching methodologies, learning styles and the choice of the first programming language taught as the major difficulties that learners encounter whilst programming. Other than general recommendations and strategies for teaching programming, the literature does not contain any direct initiatives that take into consideration the students' learning preference. Additionally, although other educational tools in use by other institutions show promising results, they still ignore specific students' needs by adopting a single pedagogy. PILeT was developed to counter these problems by consolidating different teaching methods that individually cater to a group of learners. The tool will be tested on campus. The study results will establish whether a relationship does exist between the teaching process and the preferred learning method in programming.

The tool's framework enables it to be independent of the programming language taught and could be used to teach other programming languages (e.g. Java) or other computing concepts. PILeT might be particularly useful in large classroom settings in secondary schools as a self learning tool where students progress at different rates or for teachers training to teach Python in school.

The next step is to collect, analyse and publish the research results. We are hoping to replicate this study in the following academic term to improve the validity and reliability of the findings. The future plans include making the tool adaptable for each student by defining the most appropriate pedagogical practice to deliver each concept based on their learning profile.

6. ACKNOWLEDGMENTS

The authors would like to thank Oxford Brookes University for supporting the study. The main author would also like to thank the Saudi Arabian Cultural Bureau in London for sponsoring her PhD research.

References

[1] A. Aldea, N. Crook, D. Duce, P. Marshall, C. Martin, and D. Sutton. Reflections on the evolution of the teaching of programming to undergraduates at oxford brookes university. In *Brookes eJournal of Learning and Teaching - Volume 7*. Oxford Brookes University, 2015.

[2] J. Allert. Learning style and factors contributing to success in an introductory computer science course. In *Advanced Learning Technologies, 2004. Proceedings. IEEE International Conference on*, pages 385–389. IEEE, 2004.

[3] M. Eagle and T. Barnes. Wu's castle: teaching arrays and loops in a game. In *ACM SIGCSE Bulletin*, volume 40, pages 245–249. ACM, 2008.

[4] P. J. Guo. Online python tutor: embeddable web-based program visualization for cs education. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 579–584. ACM, 2013.

[5] A. Klašnja-Milićević, B. Vesin, M. Ivanović, and Z. Budimac. E-learning personalization based on hybrid recommendation strategy and learning style identification. *Computers & Education*, 56(3):885–899, 2011.

[6] E. Lahtinen, K. Ala-Mutka, and H.-M. Järvinen. A study of the difficulties of novice programmers. In *ACM SIGCSE Bulletin*, volume 37, pages 14–18. ACM, 2005.

[7] A. Mcgettrick, R. Boyle, R. Ibbett, J. Lloyd, G. Lovegrove, and K. Mander. Grand challenges in computing education: a summary. *The Computer Journal*, 48(1):42–48, 2005.

[8] C. Murphy, E. Kim, G. Kaiser, and A. Cannon. Backstop: a tool for debugging runtime errors. In *ACM SIGCSE Bulletin*, volume 40, pages 173–177. ACM, 2008.

[9] D. Parsons and P. Haden. Parson's programming puzzles: a fun and effective learning tool for first programming courses. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*, pages 157–163. Australian Computer Society, Inc., 2006.

[10] L. Pollock and T. Harvey. Combining multiple pedagogies to boost learning and enthusiasm. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, pages 258–262. ACM, 2011.

[11] V. Ramalingam, D. LaBelle, and S. Wiedenbeck. Self-efficacy and mental models in learning to program. In *ACM SIGCSE Bulletin*, volume 36, pages 171–175. ACM, 2004.

[12] B. A. Soloman and R. M. Felder. Index of learning styles questionnaire. *NC State University. Available online at: http://www.engr.ncsu.edu/learningstyles/ilsweb.html (last visited on 14.02.2015)*, 2005.