Simplifying Multimedia Programming for Novice Programmers: MediaLib and Its Learning Materials

Adam Wynn
Durham University
Durham, United Kingdom
adam.t.wynn@durham.ac.uk

Jingyun Wang
Durham University
Durham, United Kingdom
jingyun.wang@durham.ac.uk

Andrea Valente
University of Southern Denmark
Odense, Denmark
anva@mmmi.sdu.dk

ABSTRACT

Beginner programmers can develop an intuitive understanding of programming by leveraging the motivating field of multimedia to visually inspect outputs and experiment with different ways to solve problems. This paper presents MediaLib, a Python library designed to facilitate multimedia programming and lessen the cognitive load associated with programming for novice programmers. In addition, we designed an official MediaLib website which contains the library itself, two tutorials, and clear documentation. The tutorial clearly presents the learning objectives of each lesson and contains exercises related to MediaLib. We designed these exercises to help students gain knowledge incrementally, without requiring in-depth maths knowledge.

CCS CONCEPTS

• Social and professional topics → Computational thinking; **Computer science education**; **• General and reference → Empirical studies**.

KEYWORDS

multimedia programming, beginners, Python, programming skills, computational thinking, library, teaching materials

ACM Reference Format:
Adam Wynn, Jingyun Wang, and Andrea Valente. 2024. Simplifying Multimedia Programming for Novice Programmers: MediaLib and Its Learning Materials. In *Proceedings of the 2024 Innovation and Technology in Computer Science Education V. 2 (ITiCSE 2024), July 8–10, 2024, Milan, Italy.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3649405.3659521

1 INTRODUCTION

Computational thinking (CT) is a way of thinking that involves breaking down complex problems into smaller parts, focusing on the relevant information [12]. One of the core concepts of CT is abstraction, which is the process of ignoring some irrelevant details of problems that are not needed in order to focus on the essential ones [1]. Despite abstraction being a key skill helping learners to focus on the important features, some of the existing teaching materials are designed without considering this pedagogical theory, making novice learners reluctant to accept CT [9].

Therefore, a meaningful pedagogical tool for CT education should be accessible to all types of learners, and adaptable to different levels of difficulty and complexity [3]. Block-based tools such as Scratch [7] help learners from different domains to build knowledge about programming concepts relieving learners from having to write code on their own. Block-based programming allows learners to focus on problem-solving logic rather than syntax, facilitating computational problem-solving [2]. However, beyond introductory contexts, it is difficult for learners to develop transferable skills and a deeper understanding of programming languages required when learning specific programming languages such as Python to solve real-world problems [3].

Therefore, text-based languages may be more suitable for preparing learners to use existing programming libraries [10], as they enforce syntax, structure, and logic rules. Dual-modality environments, such as Blockly, Phaser, P5, and Processing, integrate block-based and text-based features into a single interface, providing scaled-down views of professional programming languages. Guzdial et al. [4] developed a course aiming at teaching introductory programming to non-technical students focusing on media computation to provide a more engaging learning experience to these students. Students primarily completed tasks such as manipulating images, music, and video. Whilst effective in providing an interactive learning experience, their approach primarily involved students modifying existing programs rather than developing code from scratch.

These text-based environments, along with teaspoon languages, present challenges for tool creators in deciding which functions to simplify for beginners without causing confusion during the transition to advanced languages [5]. To address these issues, we introduce MediaLib, which offers a simplified version of Python to familiarize beginners with the language and enable them to create programs from scratch using a small number of predefined functions and basic Python concepts, expanding beyond media computation across different domains beyond manipulating media. The design of Medialib is intended to bridge the gap between existing block-based and text-based programming languages by means of enabling learners to work in a traditional programming environment.

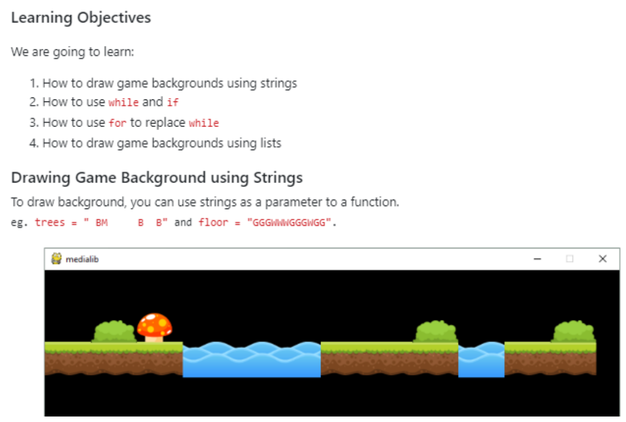
In this paper, we present a Python library called MediaLib (https://medialib.club/) and its exercises designed to aid beginners in practicing Python syntax incrementally with the support of multimedia. MediaLib is a novel way to structure beginners' Python courses which moves away from the classic approaches based on math-related problems and leverages multimedia as a motivational and powerful general-purpose domain to ground programming and scaffold its understanding.

2 DESIGN OF MEDIALIB AND ITS WEBSITE

MediaLib's design is centered around simplifying Python to a core fragment, retaining imperative features like sequences, primitive datatypes, choice, and iteration. The reduced Python fragment is then extended to multimedia by introducing minimal instructions for drawing images, controlling timing, and user interaction. The design principles emphasise sequential execution, basic variable types, absence of event handling for simplicity, and supporting a use-modify-create approach. To avoid mental overload for beginners, we minimised the number of functions in MediaLib. The current version (3.0), consists of 11 void functions and 8 fruitful functions, allowing learners to engage with graphics, text, input, other multimedia elements, and more advanced utility functions.

MediaLib exercises, which are available on the MediaLib website (https://medialib.club/tutorials) introduce fundamental concepts such as variables and functions, gradually incorporating iterative and conditional statements to create animations. Rather than relying on deep mathematical knowledge, these exercises focus on multimedia effects, providing a hands-on approach for learners to develop programming logic. The exercises not only promote understanding of MediaLib functions like draw() and clear() but also allow for the use of MediaLib with external libraries such as reading data from and saving data to csv files using Pandas [8] [11] for game development scenarios, or using Matplotib [6] for visualisation purposes. We aim to create interactive and engaging experiences by incorporating elements from other fields including game development to help learners develop more versatile skills.

As shown in Figure 1, the learning objectives of each MediaLib exercise were clearly stated at the beginning of the exercise in the tutorial, with images, animations, code snippets, and links to the documentation. For each MediaLib exercise, beginners need to use no more than three MediaLib functions to solve the problem.


Learning Objectives

We are going to learn:

1. How to draw game backgrounds using strings
2. How to use `while` and `if`
3. How to use `for` to replace `while`
4. How to draw game backgrounds using lists

Drawing Game Background using Strings

To draw background, you can use strings as a parameter to a function.

eg. `trees = " BM     B  B"` and `floor = "GGGWWWGGGWGG"`.

Figure 1: An example of an exercise on the MediaLib website.

Moreover, we have designed two tutorials: 1) a tutorial for non-technical (specifically addressing business analytics master) students featuring exercises related to MediaLib with simpler mathematical content, and 2) a tutorial designed for computer science students who are Python beginners and contains exercises related to both MediaLib and programming logic. The tutorial for non-technical students includes topics such as Variables, Void and Fruitful Functions, Conditional Statements, For Loops, While Loops, and Lists, whereas the tutorial for computer science students includes topics such as Lists, Libraries, Conditional Statements, Multiple Branching, For Loops, Nested Loops, While Loops, Dictionaries and Tuples, and File Operations.

3 CONCLUSION

Overall, MediaLib has the potential to support novice programmers in an introductory university course by providing motivating, visual exercises that aid comprehension and application of programming basics. The future improvement of MediaLib does not aim to expand its scope to cover all aspects of Python programming or to support specific Python libraries. Instead, MediaLib is designed as a pedagogical tool specifically for beginners to enhance their problem-solving skills in Python and gradually acquaint themselves with the language syntax. The intention is for learners to use MediaLib for a limited time span, allowing them to become familiar with the fundamentals of Python.

The tutorials can be used to alleviate the burden on teachers in various ways. Providing clear learning objectives and related exercises for each tutorial lesson can enable teachers to easily identify learning goals in a logical sequence, and linking the tutorial directly to the documentation reduces the need for the teacher to spend time explaining the functions. The inclusion of images, animations and code snippets in the tutorial easily demonstrated programming concepts to students without requiring further explanations from the teacher.

REFERENCES

[1] Valerie Barr and Chris Stephenson. 2011. Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? *ACM Inroads* 2 (03 2011). https://doi.org/10.1145/1929887.1929905

[2] Eugene Geist. 2016. Robots, programming and coding, oh my! *Child. Educ.* 92, 4 (July 2016), 298–304.

[3] Shuchi Grover and Roy Pea. 2013. Computational Thinking in K–12 A Review of the State of the Field. *Educational Researcher* 42 (02 2013), 38–43.

[4] Mark Guzdial. 2003. A media computation course for non-majors. *SIGCSE Bull.* 35, 3 (jun 2003), 104–108. https://doi.org/10.1145/961290.961542

[5] Mark Guzdial. 2022. Teaspoon Languages for Integrating Programming into Social Studies, Language Arts, and Mathematics Secondary Courses. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2* (Providence, RI, USA) *(SIGCSE 2022)*. Association for Computing Machinery, New York, NY, USA, 1027. https://doi.org/10.1145/3478432.3499240

[6] J. D. Hunter. 2007. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* 9, 3 (2007), 90–95. https://doi.org/10.1109/MCSE.2007.55

[7] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. The Scratch Programming Language and Environment. *ACM Trans. Comput. Educ.* 10, 4, Article 16 (nov 2010), 15 pages.

[8] The pandas development team. 2020. *pandas-dev/pandas: Pandas.* https://doi.org/10.5281/zenodo.3509134

[9] Yingxiao Qian and Ikseon Choi. 2022. Tracing the essence: ways to develop abstraction in computational thinking. *Educational technology research and development* 71, 3 (2022), 1055–1078. https://doi.org/10.1007/s11423-022-10182-0

[10] David Weintrop and Uri Wilensky. 2019. Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms. *Computers Education* 142 (2019), 103646. https://doi.org/10.1016/j.compedu.2019.103646

[11] Wes McKinney. 2010. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman (Eds.). 56 – 61. https://doi.org/10.25080/Majora-92bf1922-00a

[12] Jeannette M. Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (mar 2006), 33–35. https://doi.org/10.1145/1118178.1118215