Journal of Information Systems and Informatics
Vol. 6, No. 2, June 2024 e-ISSN: 2656-4882 p-ISSN: 2656-5935
DOI: 10.51519/journalisi.v6i2.751 Published By DRPM-UBD

PyLe: An Interactive Tool for Improving Python Syntax Mastery in Non-Computing Students

Alain Kabo Mbiada[1], Bassey Isong[2], Francis Lugayizi[3]

[12,3]Computer Science Department North-West University, Mafikeng, South Africa
Email: [1]mbiadaalain@gmail.com, [2]bassey.isong@nwu.ac.za, [3]francis.lugayizi@nwu.ac.za

Abstract

The learning and mastering of programming language syntax pose a significant challenge for non-computing students. Most teaching approaches and existing educational tools often fail to address this issue. Therefore, this paper introduces an interactive learning environment called PyLe, specifically designed for introductory programming in Python programming courses. We evaluated the effectiveness of PyLe on first-year students at North-West University in South Africa and the University of Yaoundé 1, Cameroon. Firstly, the study conducts an experiment to assess the effect of PyLe on the time taken to solve a problem and the response quality. Secondly, PyLe's usability and its instructional value were evaluated by the students and the instructors, respectively. The results from post-test method and a quantitative survey indicate that PyLe improves students' ability to learn and master program syntax and has a high usability rate. Moreover, feedback from students and teachers affirms PyLe's potential to address programming syntax challenges for non-computing students. However, the analyses revealed no real relationship between the time taken to complete a task in PyLe and the quality of the solution. This study contributes to improving the teaching and learning of computer programming, which has been considered difficult for both computing and non-computing students.

Keywords: Computing programming, Program Syntax, PyLe, Teaching/Learning environments.

1. INTRODUCTION

Computer programming is widely recognized as an increasingly valuable skill in most professional environments. It serves as an effective medium for fostering computational thinking, which empowers an individual to analyse a problem, comprehend its nature, and devise potential solutions [1]. Programming skills necessitate students to grasp two key concepts: syntax, which pertains to the structure of well-formed language programs, and semantics, which relates to the significance or meaning of these programs. Moreover, the process of teaching and learning computer programming is more complex than it appears. This is because the objective of programming instruction is to equip students with the core competencies needed to develop a program that addresses a problem by

articulating a given solution in a programming language that is considered inherently rigid and constrained. These programming languages comprise a vocabulary, grammar rules, and meanings, along with a translation environment designed to render their syntax machine-readable. Thus, the rigidity of programming languages, in comparison to natural languages, contributes to the difficulties faced by beginners in introductory programming (IP) modules. Given this ugly development, current research works in the field of computer science education (CSE) have proposed a plethora of teaching methods and tools to mitigate most of the challenges encountered by programming novices. These tools include the utilization of visual programming environments, visualization environments, and educational environments for programming [2]–[14]. However, despite acknowledging the significance of these suggested tools, a substantial body of research continues to reveal that students exhibit limited interest in programming courses [9], [10], [15], [16]. Most of them remain at a reproductive level of education when the course concludes [15]. This could potentially explain why a significant proportion of students fail to acquire the basic skills in IP [16]. Moreover, many students who complete such courses possess minimal confidence in their programming skills [9]. They often struggle with developing simple applications or even interpreting small fragments of code [10]. This collective evidence underscores the need for innovative approaches in teaching and learning programming.

Furthermore, in the context of programming, the understanding of programming language syntax is undeniably crucial to the creation and execution of appropriate, semantically correct programs. It is challenging for a student who has not grasped the syntax and usage of a basic programming concept to employ this concept to solve a given problem, especially for non-computing students. Syntax errors in programming often become a significant source of frustration, discouragement, or even cause for dropping out for novices in IP modules [17]. Moreover, most beginners encounter difficulties in understanding compiler error messages, constructing a robust mental model for logical problem-solving, and comprehending basic programming concepts. However, most of the existing tools do not explicitly consider non-computing students and fail to address the syntax-related issues they face. Consequently, in a previous study, we advocated for the development of learning materials specifically designed for non-computing students to enhance their IP skills [18].

This paper aims to design, implement, and evaluate an interactive web-based environment to improve non-computing students' understanding of programming language syntax based on the method proposed in [18]. The study approach is grounded in the cognitivism learning theory and incorporates interactive, visual, and organized resources and activities. These elements are designed to progressively prepare students to construct and comprehend code effectively. In response to the aforementioned frustration and demotivation, our proposed

approach also integrates misconceptions about basic programming concepts into the content. The motivation behind the development of this environment is to consolidate a diverse set of contents into a single course that can be utilized in IP courses. This consolidation aims to help non-computing students enhance their understanding of syntax and the application of basic programming concepts. Furthermore, the proposed tool can also assist teachers in an online or blended learning scenario to swiftly process the syntax of basic programming concepts, allowing them to devote the remainder of their time to problem-solving. This is because, in most traditional IP courses, teachers spend more time grappling with syntax issues than instructing beginners on problem-solving, often within a very limited timeframe. In addition, it can be employed to subsequently identify specific misconceptions related to non-computing students. Therefore, this paper's objective is to answer the following research questions (RQs):

RQ1: Which novel learning environment can be designed to enhance students' programming syntax skills?
RQ2: How does such a learning environment influence students' performance in terms of learning and mastering program syntax?
RQ3: How good is the learning environment's usability and what potential effects does it have on the instructional process?

These RQs guide the design, development and evaluation of the proposed learning environment's potential for enhancing the learning experience of non-computing students. The findings obtained were critically analysed and presented. Thus, the contribution of this paper is summarized as follows.

1) We designed a module layout for introductory Python programming and developed an interactive learning environment, PyLe, to introduce non-computing students to Python. It includes activities to assess students' ability to learn and master syntax, visualizations to enhance understanding of basic Python concepts, and resources focused on everyday problems with minimal mathematical challenges.
2) Conducted experiments to assess the impact of the proposed PyLe on the teaching and learning of programming and its usability, respectively.
3) Provided discussion on the effectiveness of PyLe and the comparison with other existing web-based learning environments.

This document is organized as follows: Section II presents the method used and Section III presents the results. In addition, Section IV presents the paper discussion and Section V concludes the paper.

2. METHOD

This section covers the literature review, research method, and procedure.

2.1. Literature review

This study completed a comprehensive review to identify the gaps in the existing tools and provide some background information. Therefore, this section presents some of the relevant background information and related works, including the tools developed to improve beginners' programming skills. We reviewed the development of web-based tools integrating several types of intelligent learning content for IP courses for first-year students, such as visual programming environments, visualization environments, and educational environments for programming.

2.1.1. Programming and Syntax Errors

Computer programming is the process that formulates instructions for a computer to execute to solve a problem. To effectively solve a problem, these instructions must adhere to certain rules and semantics that are specific to the programming language employed. These programming languages' syntax rules include norms for words, symbols, and punctuation. In the programming context, the journey of learning to program is often riddled with mistakes, and beginners invariably commence coding with syntactical errors, and as they progress, semantical errors become an increasingly prevalent aspect of their programming experience [19]. Syntax errors, therefore, emerge as one of the most common types of errors in IP modules, which often lead to student frustration, high failure rates, and even module abandonment [17], [20]. The challenge is particularly intense for non-computing students, who tend to struggle more compared to their computing counterparts [21]. In such scenarios, tailored support can prove to be immensely beneficial, especially if the students are spending a significant amount of time grappling with specific mistakes. This support can help them overcome their challenges more efficiently and improve their learning experience.

2.1.2. Visual programming environments

Visual programming environments are web platforms that enable learners to create programs graphically. Known as block-based or drag-and-drop environments, they help students build programming logic without the complex syntax of programming languages. They bridge the gap between learning syntax and computational thinking. However, most of these environments are designed for children, and transitioning to text-based programming can be challenging. Consequently, researchers pay consideration to this because block-based environments often lack a meaningful programming experience. Many researchers

concur that students could be attracted by the graphics without comprehending the underlying meaning they express [22]. Furthermore, visual programming tools can prove challenging to use when tackling large-scale projects. Kyfonidis et al. [2] introduced a block-based programming teaching tool designed to simplify learning C IP courses for beginners. The tool progressively translates visual models into programming concepts, allowing students to build programs by dragging and dropping blocks. However, it emphasizes logic over syntax and provides the option to convert block-based code to C text code, and vice versa. In the same vein, Jung et al. [3] built an interactive block-based environment to support program creation using the visual programming language and code generation in the C programming language. Concurrently, Moussa et al. [4] designed OOPVisual, an interactive 3D visualization tool, to enhance novice females' understanding of Object-Oriented Programming (OOP) concepts, particularly polymorphism. OOPVisual employs the drag-and-drop technique, which ultimately helps students disregard syntax errors. It incorporates tutorials detailing the concept of polymorphism, quizzes, and exercises to aid students in practice.

2.1.3. Visualization environments

The visualization environments in this context are also web-based platforms that enable learners to graphically visualize the various stages of program execution. The aim is to improve code comprehension, code construction, and understanding of most core programming concepts. Most of these environments offer some level of interaction with students and integrate virtual compilers. Thus, Rowe and Thorburn [5] introduced Visual Instruction for Novices in a C Environment (VINCE), a Java-based web tool that visually demonstrates the execution of a correct C program. It enables students to observe or write a C program and examine its detailed execution. It covers basic programming concepts, pointers, structs, arrays, function calls, and dynamic memory allocation. Similarly, Hijón-Neira et al. [6] introduced the Visual Execution Environment (VEE), a tool for teaching essential Java programming concepts to CS1 students. It uses visual metaphors to guide Java tasks and provides integrated applications with preloaded scripts. These scripts offer a variety of scripting practice options for learners during their courses. Furthermore, Yan et al. [7] created PROgramming Visualization Tool (PROVIT) for Web, a Java-based e-learning platform for C programming. PROVIT allows users to write, run, verify, and visualize C programs. Unlike many self-study tools like VILLE, VINCE, WADEIn, Jeliot, and VIP, PROVIT is also suitable for lectures. Each of these environments, with their unique features, significantly improves the programming learning experience.

2.1.4. Educational Environments for Programming

These environments provide a plethora of interactive web-based content and activities that can be fully leveraged for teaching and learning IP. Some of these

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

are extensions to the Moodle platform, a renowned free, open-source learning management system. The existence of these environments is often justified by the insufficiency of educational environments for practical programming languages such as C/C++ and Java. In addition, educators often express a desire to integrate several types of iterative learning content from a single source, rather than using multiple types from various sources or servers. Equally, Brusilovsky et al. [8] developed an architectural framework to unify different intelligent content systems into one system, leading to the creation of the Python Grids training system. It operates on servers across two continents, provides a non-mandatory learning environment for Python, and focuses on basic programming concepts. It caters to diverse student needs, offering a robust and versatile learning experience. Similarly, Mutiawani and Juwita [9] discussed the creation of an e-learning application tailored for IP courses that have content with a variety of activities, code practice, images, sounds, animations, and text. The code practice section uses EditArea, a free JavaScript editor, to improve code readability through syntax highlighting. On the same note, Samat et al. [10] investigated the creation and implementation of a constructivist multimedia learning environment to improve the programming skills of computer science students. It focuses on learning content, developing programming skills, and using technology. It also covers both basic and advanced programming concepts. Evaluations showed its effectiveness for students. Similarly, Virvou and Sidiropoulos [11] introduced a new e-learning system with collaborative tools for Python programming instruction. It facilitates group management, and student interaction and acts as an intelligent tutor, creating a unique student model based on interactions and recommending lessons tailored to each student's profile.

Furthermore, Stupina and Paniotova [12] introduced Chatbox, an interactive tool used in a blended learning environment for programming education. Chatbox improves student engagement, supports learning goals, and promotes mobile and interactive learning. It adapts to the student's learning needs and style, allows self-paced learning, and offers feedback. Similarly, Kakeshita and Murata [13] utilized pgtracer, a fill-in-the-blank tool, as a homework helper in programming classes. Pgtracer, a Moodle plug-in, helps beginners grasp programming concepts like loops, functions, and pointers. Students fill in the blanks, and pgtracer visualizes the step-by-step execution of the program, compares the response to the correct answer, and automatically grades the student. Pgtracer also has data collection features. Ferreira et al. [14] also discussed and evaluated SICAS2, a Moodle plug-in based on constructivist theory. SICAS2 enables students to create programs with flowcharts and visualize program execution. It is not limited to any specific programming language and helps students understand basic programming concepts like read and write, for and while loops, and if and if/else statements.

The discussed studies provide some of the digital tools to improve beginner IP skills, highlighting the importance of this in CSE. However, their application to

non-computing students is often not explicitly addressed. While these tools generally bypass syntax issues, an earlier study [21] indicates that syntax remains a significant challenge for these students [17] while teachers also find it difficult to adapt these tools to their needs. Moreover, large-scale learning management systems lack flexibility and dynamic content [23]. This study proposes an interactive learning environment to improve non-computing students' understanding of basic programming concepts and syntax, aiming to fill existing gaps and offer a more customized and effective learning experience.

2.2. Research Method

This section presents the methodology employed in this paper to design, implement, and evaluate PyLe, an interactive tool aimed at enhancing Python syntax proficiency among non-computer science students. We follow a structured approach, incorporating literature review, design, surveys, and experimentation.

1) Literature review: We conducted a comprehensive literature review to identify existing deficiencies in instructional programming tools as shown in Section 2.1. This step informed our design process for PyLe.
2) Surveys on non-computing students: Surveys were administered to non-computing students in two separate universities and results can be found in [21]. These helped in the understanding of the challenges these students face in IP courses. The findings from this phase guided the formulation of PyLe's requirements.
3) Requirements definition: Based on the outputs from the literature review and surveys, we outlined the requirements for PyLe. These requirements served as the foundation for the subsequent design and development phases.
4) Design and development: We followed the IIAPDIE framework [18], specifically tailored for educational software design. PyLe was designed and developed with a focus on addressing the identified deficiencies and meeting the requirements.
5) Evaluation: To assess PyLe's effectiveness, we conducted experiments with non-computer science students from two institutions. The evaluation metrics included:
    a) Time Taken (TT): The duration required to complete a programming task using PyLe.
    b) Solution Quality (Grade): The quality of solutions produced by students using PyLe.
    c) Sample Size: We randomly selected 70 participants, dividing them into control and experimental groups.
    d) Comparison: We compared the performance (grades) and TT of the control group with those of the experimental group.
    e) Utility assessment: A quantitative survey was administered to control group participants to gauge the utility of PyLe.

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

f) Instructor influence: Interviews were conducted with instructors to understand the impact of PyLe on their teaching experience.

Thus, PyLe's design, development, and evaluation were guided by a rigorous methodology, resulting in an interactive tool that aims to enhance Python syntax mastery for non-computer students.

3. RESULTS AND DISCUSSION

3.1. The Proposed Interactive Learning Environment

This sub-section presents the proposed interactive learning environment, referred to as PyLe, which serves as a response to RQ1. PyLe is designed and implemented following the various stages of the educational software development framework known as IIADPIE discussed in [18]. IIADPIE is an acronym that stands for the seven phases of the framework, detailed as follows: Initial; Instructional orientation; Analysis; Design; Production; Integration &Implementation; and Evaluation. This framework amalgamates certain practices of agile techniques, such as Scrum and the dynamic systems development method (DSDM), with instructional design methods such as ADDIE [24] and ASSURE [25]. ASSURE and ADDIE is instructional design models that provide a framework for developing and delivering learning content that utilizes technology. ASSURE is an acronym that stands for: Analyse learner characteristics; State Objectives, Select, modify or design materials; Utilise materials; Require learner response; Evaluation. ADDIE is an acronym that stands for Analyse; Design; Develop; Implement; Evaluate. PyLe is grounded in the principles of cognitivism, which emphasizes the development of the learner's memory. The primary objective of PyLe is to assist students in acquiring a thorough understanding of the syntax of basic programming concepts in Python. Thus, our main focus is on syntax comprehension, as it is intended to facilitate the learning process and enhance the students' programming skills. The processes involved are discussed as follows:

1) Design Goals

PyLe aims to provide a comprehensive set of learning materials and a robust software infrastructure to support courses covering the fundamental concepts of Python programming. It empowers teachers to select from available and relevant materials to incorporate into their courses, making the content dynamic and organized into modules. These modules, managed by instructors, are further organized into a series of lessons. Each lesson comprises a set of sections or parts and includes resources and activities. These resources and activities are presented straightforwardly to enhance learning effectiveness. Moreover, activities are specifically designed to engage and motivate students, while teachers can assign homework to assess the quality of student learning. Thus, PyLe materials meet the following specifications and features:

a) Platform independence - can be accessed from various platforms and devices.
b) Non-proprietary format - materials are not restricted to a specific proprietary format.
c) Scalability and reusability - modules can be scaled and reused as needed.
d) Dynamic content - content is not static and can adapt to different learning scenarios.
e) Interactive Simulation - facilitates in-depth understanding.
f) Interactive and automated feedback evaluation activities - activities provide instant feedback to students.
g) Storage of student exercise attempts and grades - allows tracking of students' progress.
h) Learner's profile building - PyLe tracks students' progress and builds learner's profiles.
i) Feedback mechanism - provides the instructor with the ability to give feedback to students and vice versa.
j) Student progress management and display - individual student reports and progress are managed and displayed.
k) Time-bound lessons - lessons must be completed within a given timeframe, otherwise, students will not be able to move on to the activities section of the lesson.

These features collectively make PyLe a comprehensive and effective learning environment for Python programming.

2) PyLe Infrastructure

PyLe is a web application that operates on a client-server architecture implemented using the PHP CodeIgniter Framework, with a MySQL database serving as the data storage medium. Additionally, PyLe employs the Model View Controller design pattern, which separates the application logic into three interconnected components, enhancing its manageability and scalability. The overall behaviour of the system, as depicted in Figure 1, is governed by the interactions of different users. Each client request is processed through the application programming interface (API). The system initially verifies the authenticity of the user, following which it directs the processing of the request to the system core. Depending on the nature of the user's request, the system kernel routes it to the appropriate module. This module then processes the request and generates the corresponding response. This response can sometimes necessitate access to the database or an external service. While the system predominantly uses the API for most requests, when a user is logged into their administration panel, they are granted a session. This session permits them to access certain modules directly for the duration of the session. This architecture ensures a secure, efficient, and user-friendly interaction with the PyLe environment. The components of the PyLe architecture as shown in Figure 1 are as follows:

a) Administrative tools: This provides teachers and learners with essential workspace management functions, allowing them to interact with the system. For instance:
   - Module management: Teachers can create courses from scratch or use pre-existing modules. The default course module is already available in the environment.
   - Lesson updates: Teachers can add and update lessons, incorporating various resources and activities.
   - Assignments: Teachers can add, update, grade, and export assignment grades. Learners can complete assignments and view their grades.
   - Personalization: Learners can update profile information, including learning style, colour preferences, and prior programming experience.

b) Client layer: This enables users (teachers and learners) to access PyLe via any mobile device or computer.

c) PyLe server: At the core of the system, this handles security, authentication, authorization, routing, and interaction with other system elements. It constructs different learner profiles based on their interactions with the system.

d) Database: This organizes data storage within the database.

e) Third-party: The content delivery or distribution network (CDN) facilitates rapid resource transfer for loading internet content. It includes HTML pages, JavaScript files, style sheets, images, and videos. In PyLe, we utilize Bootstrap, jQuery, and the TINYMCE editor.

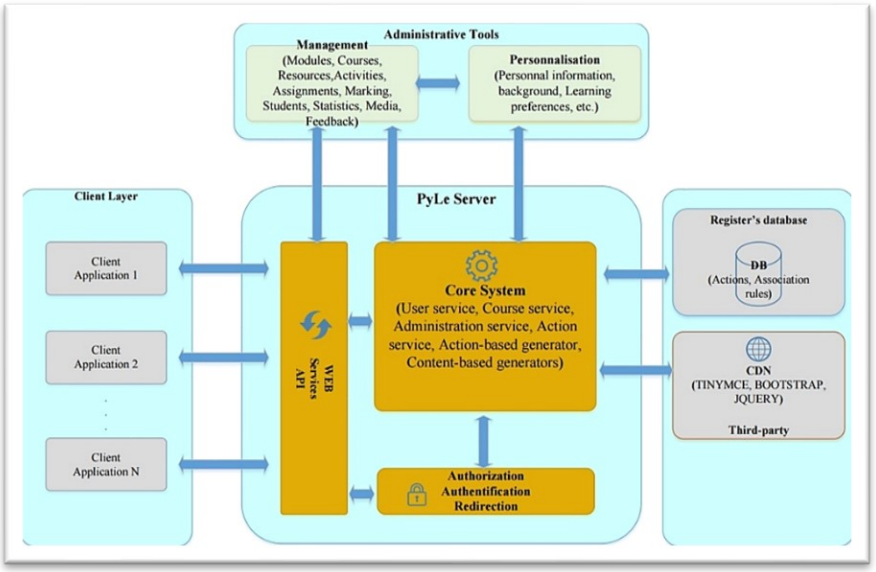

Figure 1. PyLe Architecture

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

Table 1. Student Profile in PyLe

Categories	Elements	Type/Values	Use
Personnal Information	Nom	String	Identification
	Gender	String	Identification
	Age	Scale	Identification
	Country	String	Identification
	School/University	String	Identification
	Study level	{Undergraduate, PostGraduate, Pupils, Other,}	Identification
	Prior programming experiences	{Yes, No}	Initialization
	Prior mathematics experiences	{Yes, No}	Initialization
	Prior English experiences	{Yes, No}	Initialization
	UserName/ Password/ Email	String	Identification
Learner's Features	Cognitive capacity	{Slow/Fast working memory, Poor/Good memory person}	Initialization (Time assigned to each lesson might depend)
	Learning styles	{Visual, Auditory and Reading/Writing}	Initialization; Choice of content type and activities
	Learning goals	{Acquire skills, Train for Exams, Obtain a good Grade, other,}	Initialization; Choice of concepts and examples
	Self-confidence level	{High, Medium, Low}	Inference of learner's understanding
	Grade expectation	{A, B, C, A+, etc.}	Check if the desire is in adequation of the time dedicated to learning
	Learner color preference	{Set of defined colors}	Personalization of the learner's environment
Learning states	State of Section/lesson	{Acquired, To be Acquired, To be revised, Not to be suggested}	Statistics
	State of the Activities	{Acquired, To be Acquired, To be revised, Not to be suggested}	Statistics
Interaction between learner and system	Number of lessons read	Number	Statistics
	Number of activities done	Number	Statistics
	State of the Lesson	{Completed, Not completed}	Statistics
	Last connexion	datetime	Statistics

Understanding of the activity	{Not Understood, Not well understood, Well understood, All understood}	Statistics
Duration of the connexion	In minute	Statistics

3) Student profile in PyLe

In PyLe, profiles are constructed by amalgamating the following norms: PAPI, IMS LIP, and IMS RDECEO [18]. Learners' profiles are created based on the interactions of the students with the system, as shown in Table 1. In essence, each student can have multiple profiles within the system. These diverse profiles are utilized to compile the student's progress report. This approach ensures a comprehensive understanding of each student's learning journey, facilitating personalized instruction and feedback.

4) Resources in PyLe

The current version of PyLe encompasses nine (9) lessons, as detailed in Table 2. The resources provided by PyLe place a strong emphasis on the syntax of the fundamental concepts of the Python programming language. These resources utilize a variety of mediums, including text, code, images, sound, video, and visualization. This multi-modal approach caters to diverse learning styles, thereby enhancing the learning experience.

Table 2. A suggestion for the order of PyLe's lessons

Lesson	Description	Number of activities
Lesson 1	Output function	09
Lesson 2	Some Basic Programming Notions	09
Lesson 3	Variables	14
Lesson 4	Simple Data types	04
Lesson 5	Input Function	09
Lesson 6	List	37
Lesson 7	Conditionals	09
Lesson 8	Loops	15
Lesson 9	Functions	10

The code type resource in PyLe provides the instructor with the flexibility to add or modify a piece of code in the section. To alleviate students' frustration and enhance their debugging skills, the lessons incorporate, wherever feasible, the most common mistakes or misconceptions made by novices. These misconceptions, sourced from the literature, are accompanied by the corresponding compiler's error message, providing students with practical insights into error handling. Furthermore, the lessons include visualizations designed to bolster students' understanding of programming concepts. A distinctive feature of PyLe visualizations, as shown in Figure 3 is the ability for students to provide the

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

input data before execution. This feature contrasts with most existing visualizations, where the user cannot modify the input data, thereby offering a more interactive and engaging learning experience. The significant challenge in developing the content was to avoid problems based on mathematics, which are prevalent in traditional teaching method and most learning environments. This approach ensures that the focus remains on understanding programming concepts rather than mathematical problem-solving.

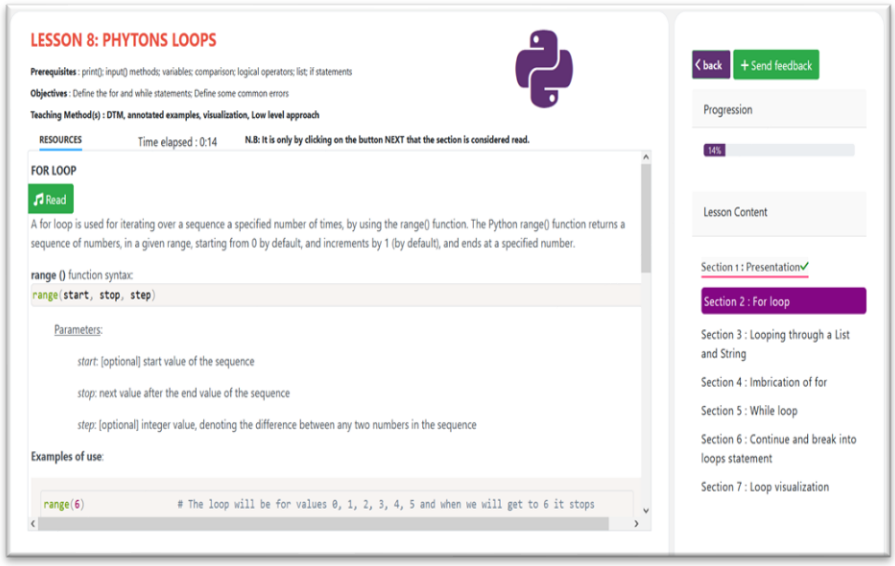

Figure 2. PyLe lesson overview

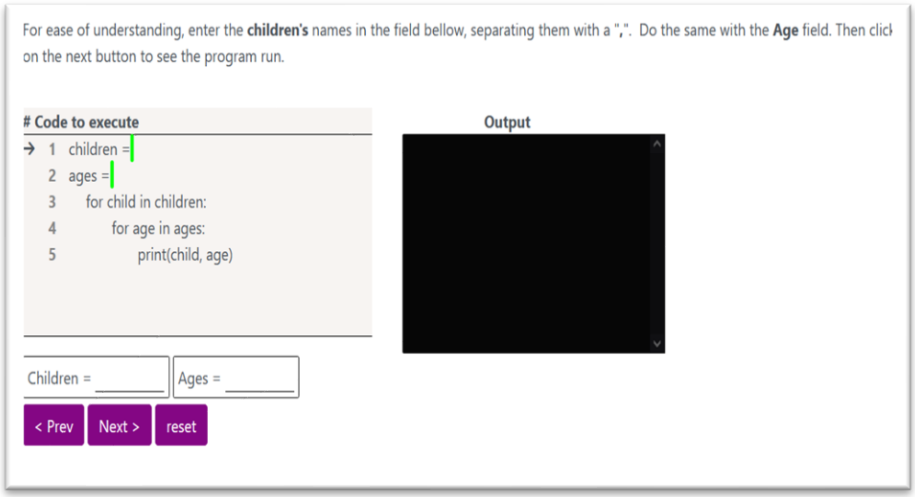

Figure 3. Sample of interactive visualization

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

5) Activities in PyLe

PyLe activities encompass a broad range of exercises designed to verify whether the student has comprehended the lessons and can consequently identify and correct syntax errors. Most of the activities are based on misunderstandings of programming students discovered in the literature and previously covered in the course content. These activities, as shown in Figure 4, are categorized into the following types:

a) Concepts inventories: These are a series of quizzes based on students' misconceptions that can help the instructor determine whether or not students have mastered a concept [18].
b) Pearson problems: Unlike most Pearson problems that use the drag-and-drop system, the Pearson problem in PyLe requires students to type their answers in the correct order. This ensures that students can enter the corresponding answer without making syntax errors.
c) Matching questions: These use the drag-and-drop mechanism to test the students' understanding.
d) Fill-in-the-Blank Questions.

Moreover, the study has resulted in a series of activities for all these categories, ready to be used for the aforementioned concepts in Python. These diverse and interactive activities aim to reinforce learning and enhance the student's understanding of Python programming syntax.

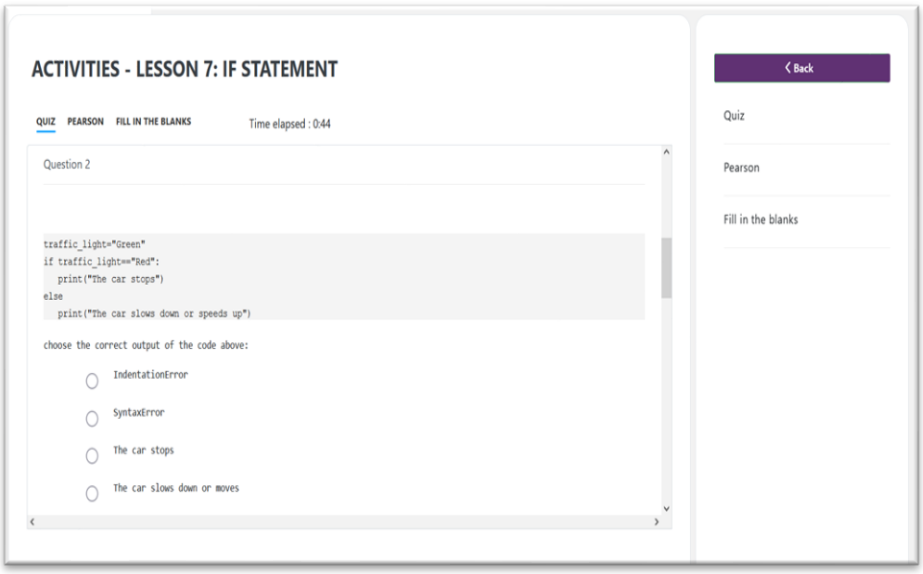

Figure 4. Overview of activities

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

As discussed above, PyLe is an educational environment that incorporates students' misconceptions to enhance motivation and performance. It focuses on improving students' Python programming syntax and offers user-friendly content on Python basics. It is suitable for use in both blended and online learning settings for an IP course or self-directed learning. Its rationale stems from the fact that most lecturers struggle to integrate programming syntax into teaching, and current research often overlooks syntax issues and their relevance to non-computer science students. PyLe addresses these concerns by considering students' misunderstandings and introducing features for module reuse and content updates. Unlike other existing tools and given the target population and challenges discussed in [18], its content is based on real-life situations, not just mathematical skills, making it unique in its objective and development process.

3.2. Evaluations

This sub-section presents the evaluation of the proposed PyLe to establish its effectiveness in enhancing the programming skills of non-computing students. The assessments carried out in this section serve to address RQ2 and RQ3. This aligns with the final phase of the IIADPIE framework [18], which was employed to construct the proposed learning environment. An overview of the strategy implemented in this evaluation is presented in Figure 5 The strategy ensures a thorough understanding of the effectiveness and usability of the proposed PyLe.

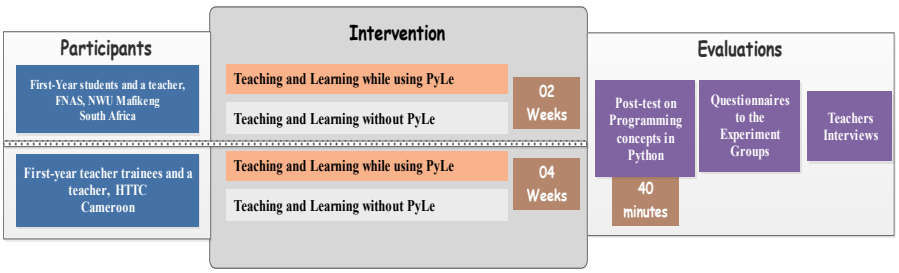

Figure 5. Evaluation workflow

1) PyLe's effectiveness

To evaluate the effectiveness of PyLe, this study conducted an experiment to address RQ2. It aims to verify whether the TT to solve a problem has an impact on the grade or quality of the solution and whether using PyLe is more effective in learning program syntax than not using PyLe.

Participants: The study involved non-computing students, including first-year trainee teachers from the Higher Teacher Training College (HTTC) in Cameroon

and first-year students from the Faculty of Natural and Agricultural Sciences (FNAS) in South Africa. At both institutions, control and experimental groups were formed, with 15 students at HTTC and 20 students at FNAS. All 70 students were enrolled in introductory Python programming courses, with those in Cameroon studying in the second semester of 2023-24 and those in South Africa in the first semester of 2024. The course at FNAS is part of the CMPG111 module, while at HTTC, it serves as preparation for the INFO2122 unit on OOP with Python.

Experiment and task description: The experimental group at HTTC used the PyLe environment in a blended learning setup, and the students engaged with the content and activities across nine lessons. The in-person sessions we employed allowed the instructor to address student concerns and provide additional explanations. Also, a WhatsApp group was established for ongoing support and engagement. Meanwhile, the FNAS experimental group (EG) used PyLe in a fully online setting, with a similar WhatsApp group for support. The focus of the experiment was on conditional statements and loops. The control groups (CG) at both universities underwent traditional courses on the same concepts. In addition, conditionals and loops were the programming concepts implemented during the experiment. Concurrently, CGs from both universities took traditional courses on the same programming concepts. The investigation was to determine the potential effect of TT to complete a task on the final GRADE achieved. In particular, the GRADE is related to the quality of the final product submitted by the students after a specific time TT spent. Also, the tasks given to the students included quizzes and problems identifying and correcting syntax errors in given programs. The quizzes aimed to enable students to predict the result of executing a given program, containing or not syntax errors, by choosing the correct answer from several propositions.

Measuring instruments and variables: There were no pre-tests conducted on the groups. The experiments used post-tests to assess students' ability to learn and grasp program syntax in Python. Hence, two variables, students' GRADE (0-100%) and TT for a 40-minutes test were collected. Thus, the GRADE is a dependent variable while TT is the independent variable. In addition, the EG participants were coded as E01 to E015 for HTTC and E101 to E120 for the FNAS while the CG participants were coded as C01 to C015 for HTTC and C101 to C120 for FNAS.

Hypotheses formulation: The null hypotheses tested in these experiments aim to gauge the significance of adopting the PyLe environment as a crucial learning material for improving students' ability to learn and master program syntax in the introductory Python programming courses. The specific hypotheses are outlined as follows:

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

H_{01}: The ability to learn and master program syntax in Python is the same for both EG and CG in terms of grade.

H_{01}: $\mu_{GRADEEG} = \mu_{GRADECG}$

H_{02}: The time spent performing on PyLe is the same for both EG and CG.

H_{02}: $\mu_{TTEG} = \mu_{TTCG}$

Additionally, based on the test, the H_{01} and H_{02} will be rejected if their respective p-values $\varrho < 0.05$.

Statistical techniques: The selection of statistical techniques is contingent upon the distribution of the variables in question. Given the sample size of over 50 participants, we conducted the Kolmogorov-Smirnov test on the GRADE and TT variables to test the normality of the data. The results indicated that both GRADE and TT were not normally distributed, as their p-values were less than 0.05. Consequently, we used the independent-sample Mann-Whitney U-test to test hypotheses H_{01} and H_{02}. Moreover, we carried out Pearson's correlation test to check whether there was a relationship between TT and GRADE.

3.3 Results and analysis

This subsection presents the results of the evaluation. As presented in Figure 6 and Figure 7, the participants' grades, reflecting their ability to learn and master problem syntax in Python, are highly satisfactory for the experimental groups (EGs), with the top score reaching 100%. Furthermore, the analysis of means and standard deviations reveals that EGs generally surpass CGs in terms of grades. Specifically, EGs mean grade is 61.5%, as shown in Table 3, which is significantly higher than CGs mean grade of 37%. Conversely, when it comes to the average time spent on the test, CGs slightly exceed EGs, with a percentage of 23.6% compared to EGs at 21.6%.

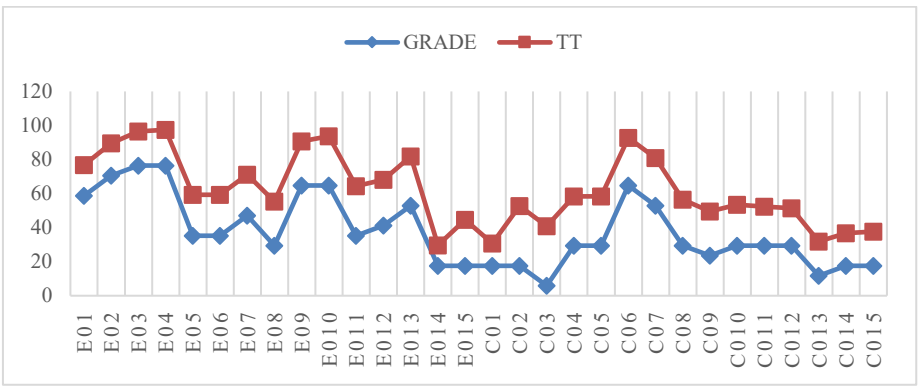

Figure 6. Post-test results for Cameroon

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

Table 3. Descriptive analysis of Grade and TU.

		Mean	SD
Experimental Group	GRADE	61.53	19.59
	Time Taken (TT)	21.63	6.41
Control Group	GRADE	37.03	19.31
	Time Taken (TT)	23.34	6.79

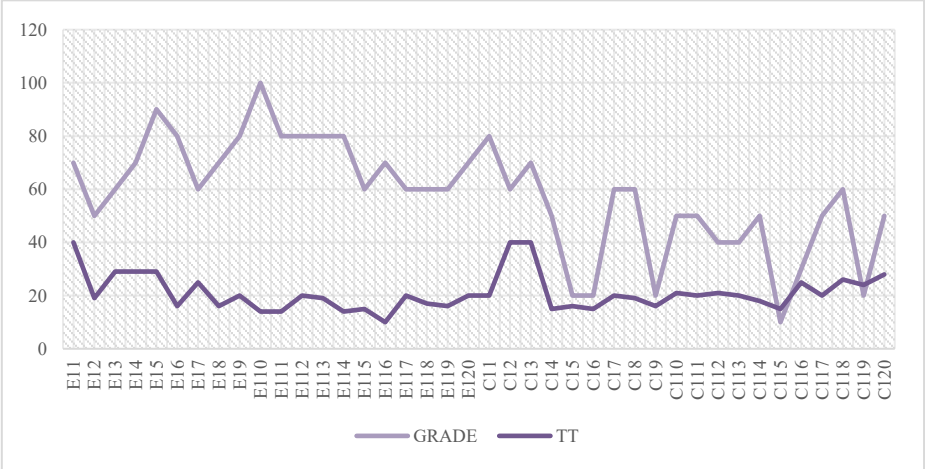

Figure 7. Post-test results for South Africa

In addition, to further determine the significance of the above results, we tested the formulated hypotheses using the independent-samples Mann Whitney U-test as shown in Figure 8, and Figure 9. The findings indicate that the p-value for the paired variable $GRADE_{CG}$ - $GRADE_{EG}$ is 0.000, which is less than 0.05. This leads to the rejection of the H_{01}. Consequently, the alternative hypothesis that $GRADE_{CG}$ is superior to $GRADE_{EG}$ is accepted, given that the mean $\mu GRADE_{EG}$ = 61.53 significantly exceeds the mean $\mu GRADE_{CG}$= 37.03. Moreover, the p-value for the paired variables TT_{CG} - TT_{EG} is 0.361, which is greater than 0.05. This implies that the H_{02} cannot be rejected. Therefore, we can conclude that the difference between μTT_{EG} and μTT_{EG} is not statistically significant due to several factors that guide the experiment. As a result, the variance in the average time students take to complete the test between the two groups is not significant. Furthermore, a Pearson's correlation test was conducted, revealing that the p-value for the variables $GRADE_{EG}$ and TT_{EG} is 0.191, and for the variables $GRADE_{CG}$ and TT_{CG} is 0.233, both of which are greater than 0.05. This suggests that the student's grades are not dependent on the time taken to complete the test for participants in either group.

In light of these findings, while the use of PyLe did not consistently lead to students completing exams more quickly, it did significantly enhance their ability to learn and master Python's syntax. This is a reasonable outcome, considering that most of the students have not previously taken exams designed to identify and correct syntactic errors.

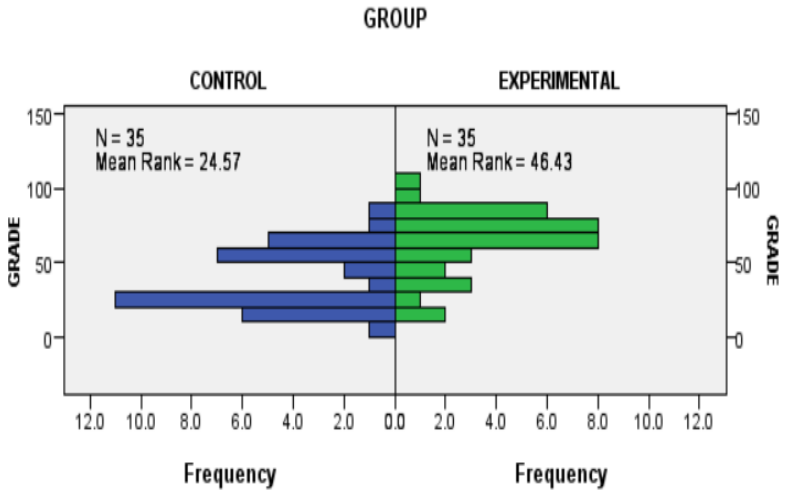

Figure 8. Independent-samples Mann-Whitney U Test on GRADE

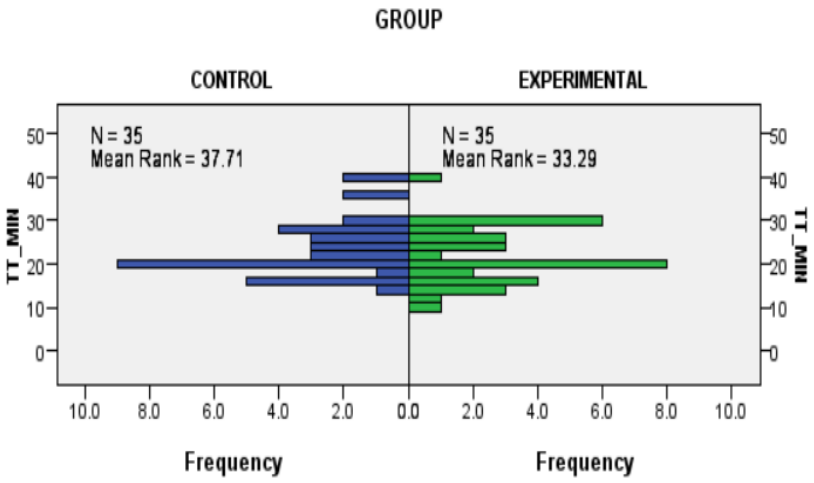

Figure 9. Independent-samples Mann-Whitney U Test on TT

1) PyLe's usability assessment and instructional effect

This subsection presents PyLe usability evaluation and its instructional effect on the parts of the instructors. We employed post-experiment procedures that adhered to the quantitative design technique to address RQ3. The study participants are all CG students discussed above, totalling 35 students. In this case, we designed and distributed questionnaires after the experiment about the usability of the PyLe environment based on several usability assessment criteria. The questionnaire, adapted from the one referenced in [26], consists of six sections with a total of 23 questions. The participants were asked to respond based on an ordinal scale with values ranging from Strongly Disagree (1) to Strongly Agree (5). The questionnaires were administered to the students, and the data was subsequently collected and analyzed using descriptive statistics in SPSS software. Moreover, before data collection from NWU teachers and students, an ethics certificate was obtained from the FNAS ethics committee. To ensure or affirm the reliability and validity of the results, the Cronbach's alpha value obtained was 0.80, indicating a high degree of internal consistency in the questionnaire responses.

The usability of PyLe for students was assessed across several categories or criteria, such as visual design, navigation, accessibility, interactivity, self-assessment and learnability, and motivation to learn. The findings are detailed in Table 4. Findings analysis shows that students strongly endorse PyLe's navigation, interactivity, self-assessment, and learnability features. However, they concur that PyLe enhances their motivation to learn and that its visual design and accessibility are satisfactory. Moreover, most student responses are concentrated around the corresponding means. Despite the acceptable responses, there is room for PyLe to enhance its accessibility. Particularly, the question regarding whether PyLe has any technical issues frequently receives a score of 2, which corresponds to 'Disagree' on the Likert scale. This suggests a need to address potential technical issues in PyLe to improve the overall user experience.

Table 4. Descriptive analysis result of PyLe's Usability

Criteria	Mean	Median	Mode	Std. Deviation
Visual Design	4.11	4	4	0.79
Text, images, and visualization are easy to understand	3.91	4	5	1.12
Fonts (style, colour, etc.) are easy to read	4.03	4	4	0.86
Relevant information is placed in areas that catch your attention.	3.94	4	4	0.91
Navigation	4.60	5	5	0.49
You can decide which sections of the lesson you want to view	4.29	4	4	0.71
Lesson content is only a few clicks away	4.26	4	4	0.78

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

Criteria	Mean	Median	Mode	Std. Deviation
You can track your progress on PyLe	4.34	5	5	0.90
The menus for accessing content are organized	4.17	4	4	0.82
Accessibility	3.85	4	4	0.80
PyLe pages and other elements were easily accessible in a reasonable time	4.06	4	4	0.80
PyLe is easy to access from any platform	3.66	4	5	1.18
PyLe has no technical problems	3.17	3	2	1.24
Interactivity	4.40	5	5	0.73
PyLe offers facilities to make the learning process more engaging and motivating	4.20	4	4	0.75
PyLe provides access to a set of resources adapted to the learning context	4.31	4	4	0.67
PyLe engages learners in tasks that are closely linked to learning goals and objectives	4.20	4	4	0.90
Self-Assessment and Learnability	4.57	5	5	0.55
You can predict the overall result of clicking on each button or link	3.77	4	4	0.91
You can easily understand the purpose of using PyLe in the learning process	4.46	4	4	0.56
Each lesson offers you a set of activities to check your level of understanding	4.49	5	5	0.70
Several types of activities are offered in Pyle	4.51	5	5	0.50
PyLe activities prepare you to avoid a range of syntax errors	4.49	5	5	0.85
Motivation to learn	4.14	4	4	0.84
You've found the content very useful for understanding and avoiding syntax errors	4.20	4	4	0.86
PyLe's lessons encourage you to deepen your knowledge	4.26	4	4	0.78
You found PyLe's content enjoyable and interesting	4.03	4	5	1.07
PyLe content provides lessons that match your life experience	3.40	4	4	1.00
PyLe content provides frequent and various learning activities that set you up for success	4.17	4	4	0.89

Moreover, still, in response to RQ3, we also conducted post-experimental interviews with the two teachers responsible for the modules. This was aimed at providing feedback on PyLe's content, its utility, and their willingness to incorporate it into their teaching materials. The analysis of their responses revealed that the content was found to be factual, current, and comprehensive. The system layout was intuitive and easy to navigate, with real-world examples. The tools were

deemed superior to comparable ones, with appreciated features like student progress tracking, content adaptability, relevant objectives, and logical task sequencing. The tool aligned with their ethical standards and values, and the content was consistent with the curriculum. Following the experiments, additional students were enrolled in PyLe, confirming its seamless integration into teaching and learning practices. However, suggestions for improvements included more visualizations, inclusion of other Python concepts, and expansion to other programming languages like C, C++, and Java.

2) Validity Threats

To ensure the veracity of the results presented above, we have implemented various measures. We carefully recorded the reactions of PyLe users, and only those students who actively engaged with the environment by reading information and participating in activities were included in the post-test and the subsequent questionnaire. Moreover, the teachers who were interviewed were exclusively those who participated in the trials. The post-test questions were carefully curated to only include subjects that were studied using the PyLe environment. To accurately estimate the consent of participants and prevent measurement bias, the responses to the questionnaire were evaluated on a 5-point Likert scale. Furthermore, to ensure the quality of the data, a reliability test was conducted using the data obtained from the students' usability survey. It is important to note that every participant completed the questionnaire and took part in the post-test. The analysis of these results has bolstered our confidence in the accuracy of the previously reported findings. Consequently, we are confident in the quality of the outcomes presented above.

3.4 Discussion And Comparison

In this paper, we have designed, implemented, and evaluated a novel learning environment for programming in Python called PyLe. PyLe is an educational environment designed for introducing Python programming, grounded in the IIADPIE framework. This framework integrates pedagogical and agile approaches to structuring and managing educational software design [18]. PyLe is specifically designed for non-computing students, focusing on syntactic issues in Python. It offers easily digestible content on the fundamentals of Python programming. In blended or online learning settings, instructors can employ PyLe for Python courses, while students can use it for self-directed learning and assessment. Unlike most existing tools [2]–[14], PyLe incorporates students' misconceptions into its content development to alleviate widespread frustration during coding. Moreover, the key features of PyLe include module reuse across different classes and the ability for teachers to update existing content, aspects that are often overlooked in other tools. In addition, PyLe stands out due to its specific objectives and

development process, which prioritize real-life scenarios, contrasting with tools that are primarily tailored to mathematically skilled students. Table 5 provides important details on the differences between PyLe and some existing tools.

Furthermore, PyLe underwent implementation and assessment on two modules across Cameroon and South Africa to determine its effectiveness in teaching and learning. The post-experimental evaluations conducted revealed that students who used PyLe outperformed their peers in assessing their ability to learn and solve syntactic programming problems. Interestingly, the TT by students to complete the task given did not significantly impact their final scores (Grade). This suggests that PyLe's unique assessment design effectively helps students learn, master, identify, and correct Python syntax. Furthermore, post-experiment surveys indicated high usability for PyLe. Most lessons were fully read, and participants completed the activities as shown in Figure 10 and Figure 11. In addition, an analysis of interviews with teachers confirmed their positive perception of PyLe, leading to its incorporation into their teaching practices.

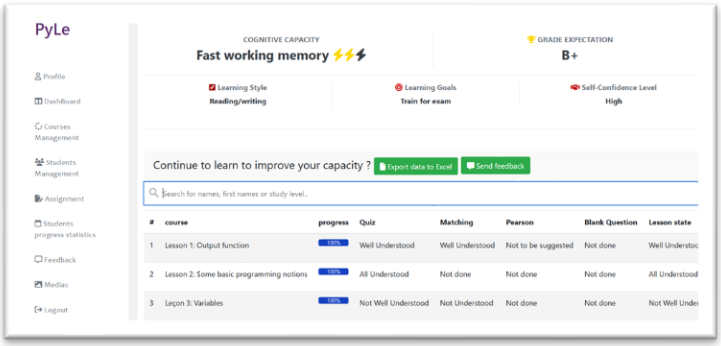

Figure 10. Overview of a Student Progress Report

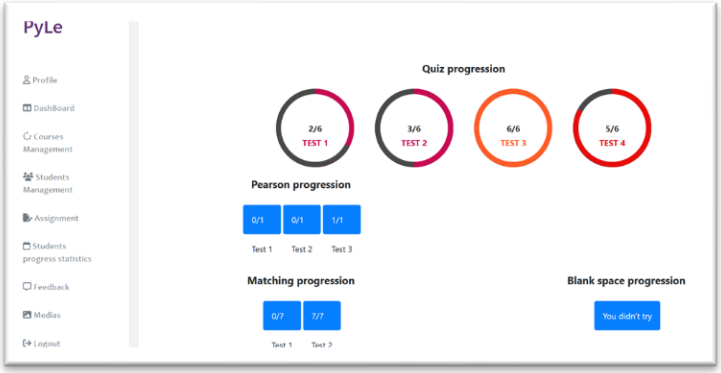

Figure 11. Student's activities report

Table 5. Theoretical comparison of PyLe with some existing learning programming environments

Learning Tools	Education theory	Target population	Aims and learning context	Nature of the Content and Other Features
PyLe	Cognitivism	Non-computer students	Learning and understanding of programming syntax. Use in blended and e-learning contexts.	Resources (text, images, video, dynamic visualization, annotated examples, etc.), and activities that can be updated by the teacher. Based on the IIADPIE framework [18] and Python programming language. Includes output, comparison operators, logic operators, simple data types, input, lists, variables if statements, loops, and functions.
Practice Grids training system [8]	N/A	N/A	Code comprehension, construction, and non-mandatory practice. Can be used in blended situations.	Focus on Python programming language. It includes variables, if statements, loops, and logical operators.
Multimedia Learning Environment [9]	Constructivist	Computer education students	Code comprehension	Defined its development design method in four main points. Includes basic and advanced programming concepts.
Constructivist multimedia learning environment [10]	Constructivist	N/A	Support student interaction and group formation	No content.
Chatbox [12]	N/A	N/A	Facilitate self-paced learning and can be used in a blended context.	A development design method based on 10 principles for PHP and others not mentioned. Includes variable and data types, conditionals, loops, arrays, strings, and functions.
Pgtracer [13]	N/A	N/A	Moodle plugin for Code comprehension. It is used as a homework aid in programming.	Focus on C programming language and includes loops, functions and pointers.

Learning Tools	Education theory	Target population	Aims and learning context	Nature of the Content and Other Features
SICA2 [14]	Constructivist	N/A	Moodle plugin for the construction of program flowcharts.	Programming concepts included variables and basic control structures.

4. CONCLUSION

This paper presented PyLe, a novel tool designed to help non-computing students in mastering Python syntax. PyLe's significance lies in its focus on improving syntax error detection and correction skills, an area often overlooked by existing programming learning tools. Our PyLe, designed using the IIAPDIE framework and cognitivism principles, offers dynamic resources and activities. It can be utilized for self-learning or in blended and online educational settings, providing various features beneficial for educators. The effectiveness of PyLe was evaluated through post-test studies involving first-year students from NWU, South Africa, and the HTTC, University of Yaoundé I, Cameroon. The students were divided into control and experimental groups. The results obtained show that the control group who used PyLe in their tasks, outperformed the experimental group with a mean score of 61.53 in the learning and mastery of Python syntax. Furthermore, PyLe's usability and instructional impacts were assessed using questionnaires for the control group and interviews with the module instructors. The feedback indicated that students found several aspects of PyLe's usability effective, and teachers deemed PyLe a useful and acceptable teaching tool. This underscores the need to integrate PyLe into teaching and learning to help non-computing students grasp and master the syntax of programming more effectively. However, it was observed that PyLe did not significantly reduce the time students spend on coding tasks. This is an area we aim to improve in future iterations of the tool. Our future work will focus on enriching PyLe's content and creating more visualizations to enhance the learning experience. We also plan to extend PyLe's capabilities to other programming languages such as C and C++, thereby broadening its scope and utility. These aim to further boost PyLe's effectiveness as a learning tool for non-computing students.

REFERENCES

[1] S. Grover, 'Designing an assessment for introductory programming concepts in middle school computer science', in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 2020, pp. 678–684.

[2] C. Kyfonidis, N. Moumoutzis, and S. Christodoulakis, 'Block-C: A block-based programming teaching tool to facilitate introductory C programming courses', in *2017 IEEE Global Engineering Education Conference (EDUCON)*, IEEE, 2017, pp. 570–579.
[3] I. Jung, J. Choi, I.-J. Kim, and C. Choi, 'Interactive learning environment for practical programming language based on web service', in *2016 15th International Conference on Information Technology Based Higher Education and Training (ITHET)*, IEEE, 2016, pp. 1–7.
[4] W. E. Moussa, R. M. Almalki, M. A. Alamoudi, and A. Allinjawi, 'Proposing a 3d interactive visualization tool for learning OOP concepts', in *2016 13th Learning and Technology Conference (L&T)*, IEEE, 2016, pp. 1–7.
[5] G. Rowe and G. Thorburn, 'VINCE—An on-line tutorial tool for teaching introductory programming', *Br. J. Educ. Technol.*, vol. 31, no. 4, pp. 359–369, 2000.
[6] R. Hijón-Neira, C. Pizarro, J. French, P. Paredes-Barragán, and M. Duignan, 'Improving CS1 Programming Learning with Visual Execution Environments', *Information*, vol. 14, no. 10, p. 579, 2023.
[7] Y. Yan, H. Nakano, K. Hara, T. Kazuma, and A. He, 'A Web Service for C Programming Learning and Teaching', in *2016 10th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*, IEEE, 2016, pp. 414–419.
[8] P. Brusilovsky, L. Malmi, R. Hosseini, J. Guerra, T. Sirkiä, and K. Pollari-Malmi, 'An integrated practice system for learning programming in Python: design and evaluation', *Res. Pract. Technol. Enhanc. Learn.*, vol. 13, pp. 1–40, 2018.
[9] V. Mutiawani and others, 'Developing e-learning application specifically designed for learning introductory programming', in *2014 International Conference on Information Technology Systems and Innovation (ICITSI)*, IEEE, 2014, pp. 126–129.
[10] C. Samat, S. Chaijaroen, I. Kanjug, and P. Vongtathum, 'Design and development of constructivist multimedia learning environment enhancing skills in computer programming', in *2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, IEEE, 2017, pp. 1023–1026.
[11] M. Virvou and S. C. Sidiropoulos, 'Collaborative tools in learning a programming language', in *2012 International Conference on E-Learning and E-Technologies in Education (ICEEE)*, IEEE, 2012, pp. 162–165.
[12] M. Stupina and V. Paniotova, 'An Educational Chatbot in a Blended Learning Environment', in *2023 3rd International Conference on Technology Enhanced Learning in Higher Education (TELE)*, IEEE, 2023, pp. 276–279.
[13] T. Kakeshita and M. Murata, 'Application of programming education support tool pgtracer for homework assignment', *Int. J. Learn. Technol. Learn. Environ.*, vol. 1, no. 1, pp. 41–60, 2018.

[14] A. Ferreira, A. Gomes, and A. J. Mendes, 'SICAS2: Interactive Tool to Support Programming Learning', in *2022 International Symposium on Computers in Education (SIIE)*, IEEE, 2022, pp. 1–5.
[15] S. B. Yusupova, O. R. Sultanov, R. S. Baltayev, and F. A. Bekchanov, 'The advantage of using e-learning in teaching students programming languages', in *2022 IEEE International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*, IEEE, 2022, pp. 1910–1913.
[16] J. Figueiredo and F. García-Peñalvo, 'Teaching and learning tools for introductory programming in university courses', in *2021 International Symposium on Computers in Education (SIIE)*, IEEE, 2021, pp. 1–6.
[17] M. A. Sana'a, T. A. Dousay, and C. L. Jeffery, 'Integrated learning development environment for learning and teaching C/C++ language to novice programmers', in *2020 IEEE Frontiers in Education Conference (FIE)*, IEEE, 2020, pp. 1–5.
[18] A. K. Mbiada, B. Isong, F. Lugayizi, and A. Abu-Mahfouz, 'Towards integrated framework for efficient educational software development', in *2023 IEEE/ACIS 21st International Conference on Software Engineering Research, Management and Applications (SERA)*, IEEE, 2023, pp. 53–60.
[19] A. Ahadi, R. Lister, S. Lal, and A. Hellas, 'Learning programming, syntax errors and institution-specific factors', in *Proceedings of the 20th Australasian Computing Education Conference*, 2018, pp. 90–96.
[20] A. K. Veerasamy, D. D'Souza, and M.-J. Laakso, 'Identifying novice student programming misconceptions and errors from summative assessments', *J. Educ. Technol. Syst.*, vol. 45, no. 1, pp. 50–73, 2016.
[21] A. Mbiada, B. Isong, and F. Lugayizi, 'A Comparative Study of Computer Programming Challenges of Computing and Non-Computing First-Year Students', *Indonesia. J. Comput. Sci.*, vol. 12, no. 4, 2023.
[22] D. De Silva, S. Vidhanaarachchi, K. Siriwardana, S. Gunasekara, U. Piyumantha, and S. Thilakaratne, 'RookieScript: Constructive Programming Learning Space for Beginners', 2023.
[23] G. Rößling *et al.*, 'Enhancing learning management systems to better support computer science education', *ACM SIGCSE Bull.*, vol. 40, no. 4, pp. 142–166, 2008.
[24] N. M. Seel, T. Lehmann, P. Blumschein, and O. A. Podolskiy, *Instructional design for learning: Theoretical foundations*. Springer, 2017.
[25] R. Heinich, M. Molenda, and J. D. Russell, *Instructional media and the new technologies of instruction*. Macmillan, 1989.
[26] I. S. Junus, H. B. Santoso, R. Y. K. Isal, and A. Y. Utomo, 'Usability evaluation of the student centered e-learning environment', *Int. Rev. Res. Open Distrib. Learn.*, vol. 16, no. 4, pp. 62–82, 2015.