Repeated Measures Study within Subjects with Randomizations (Python Codes)

ANA RITA TEIXEIRA[1,2], SÓNIA BRITO-COSTA[1,3], ANABELA GOMES[3,4]
[1]InED - Center for Research and Innovation in Education,
Polytechnic of Porto,
Rua Dr. Roberto Frias, 712, 4200-465 Porto,
PORTUGAL

[2]Porto Institute of Engineering (ISEP),
Polytechnic of Porto,
Rua Dr. António Bernardino de Almeida, 431, 42249-015 Porto,
PORTUGAL

[3]Polytechnic University of Coimbra,
Rua da Misericórdia, Lagar dos Cortiços, S. Martinho do Bispo, 3045-093 Coimbra,
PORTUGAL

[4]Centre for Informatics and Systems (CISUC),
University of Coimbra,
Polo II, Pinhal de Marrocos, 3030-290 Coimbra,
PORTUGAL

Abstract: - Learning a new programming language is challenging for essentially the entirety of our population that decides to try and pick up said skill even those who have previously learned another language find it very difficult. This study investigates the difficulties students face when learning "for" loops in the Python programming language. The research utilizes an eye-tracking device to analyze pupil dilation and blinking rates as participants attempt to solve Python code problems involving "for" loops. The study includes four different code scenarios, each with varying degrees of complexity, including nested "for" loops. The results show that a significant portion of the participants struggled with the tasks, achieving a low average success rate of approximately 28%. Consistent variations in pupil dilation and blinking patterns were observed, indicating high stress levels and potential confusion. The data revealed specific areas of the code where students commonly struggled, particularly with nested “for” loops and the “print()” function. Eye-tracking data revealed consistent variations in pupil dilation and blinking patterns, indicating high stress levels among participants. Teachers should be aware of the identified areas of confusion and design teaching strategies that address them directly. Leveraging eye-tracking data to inform the development of interactive programming exercises or tools that provide more effective visual representations of code concepts can significantly improve student understanding. Therefore, the paper ends with some incipient teaching recommendations and future research directions.

Key-Words: - Computer Science, Difficulties, Eye Tracker, “For” Loop, Introductory programming, Python, “For” Loop.

Received: March 29, 2024. Revised: October 4, 2024. Accepted: November 16, 2024. Available online: December 21, 2024.

1 Introduction

Learning programming is a complex problem in the varied contexts in which it is taught for the first time and for different reasons, [1], [2], [3]. Students in highly digital courses such as Communication, Design, and Multimedia have a lot of contact with programming and also face challenges with some programming languages, such as C or Python, [4], [5], on an almost daily basis. Among the diverse problems students face in programming learning for the first time, difficulty with ”for” loops is a common problem, [6], [7]. This difficulty is twofold: the syntax part of the loop – or, in other words, the functional and written aspect – and the

semantic part, which makes it reach the intended result.

The research presented in this paper will only cover the Python language and the ability to analyze and interpret ”for” loops. The use of ”for” loops is indispensable for creating complex, functional programs, and as such, finding ways to understand why people find it difficult is crucial to finding better teaching methods for the language.

This research is carried out using a GazePoint eye-tracker, which collects data on pupil dilation and the number of blinks per minute. The device’s software transmits a video with the study protocol (Figure 5), displaying four different Python codes, specifically containing ”for” loops. Two codes have only one loop, and the other two have a chained loop (a loop within another), sometimes nicknamed “for for”. By collecting this data, it is possible to analyze the lines of code and alternatives that the participants looked at for the longest time, as well as the blinkings, which is a strong indication of the comfort of the participant’s eyes, as well as how tired or bored they are. Many blinks far below or far above the average can indicate tiredness, discomfort, and drowsiness. Through this analysis, it is possible to identify the parts of the ”for” loop in which the participant experiences the most difficulty.

The rest of the paper is organized as follows. Section 2 describes the methodology, section 3 contains the research questions, section 4 contains the analysis of results, and section 5 contains the conclusions and some considerations for further work.

2 Methodology

2.1 Conditions of the Study

For the study analysis, 30 different versions of the same protocol were created, Table 1, each with the four ”for” loops organized in different, random ways without repetition to prevent participants from exchanging answers with each other. In this way, each participant had a specific order for each loop codification, which guaranteed greater fidelity in the sample of results. It is worth saying that the environment in which this experiment was carried out was highly controlled at both auditory and visual levels, avoiding distractions. In each protocol, four ”for” loops were presented, each with four response options, which participants had to observe and say out loud the option they thought was correct (Figure 1, Figure 2, Figure 3 and Figure 4). These same protocols were displayed and presented to the participants, who sat in front of the screen. At the same time, their gaze was tracked by the GazePoint eye-tracker. The main equipment required is an eye tracker, which can be mounted on a computer monitor. The eye-tracker device used for the study was the Gazepoint GP3 HD and the software provided by the same company in version 6.11.0 (August 31, 2023) running on a notebook with Windows 11 operating system connected to an external monitor to run the tests. The system uses a light source - usually infrared- for greater precision - aimed at the user’s eyes. A camera tracks the reflection of the light and the movement of the eye’s visible features, such as pupil dilatation. Through this process, we can access different data and metrics about the user being studied, including:

- Gaze direction;
- Number of fixations;
- Moment of the first fixation;
- Flashing rate;
- Blinking;
- Pupil dilation.

This data will be connected to user reactions. For example, the pupil's diameter corresponds to the user’s concentration level, allowing us to analyze said reactions and mindsets. On the other hand, blinking allows us to identify whether what the user sees is causing tiredness or lack of interest. This data collection form is much more realistic because we can confidently gauge the user’s reaction. In contrast, if they were to report their experience, it could be influenced and thus bias the results of the sample and, consequently, the study.

2.2 Data Collection

A total of thirty tests were carried out on university students, with 30 different individuals, 11 of whom were male and 19 female. The participants’ ages ranged from 18 to 28, and they were all students at the Superior School of Education of Polytechnic University of Coimbra. Each participant was called in turn in the laboratory. Before collecting data from each user, the eye-tracker was calibrated to keep the results as accurate as possible. The participant sits in front of the screen so that the eye tracker recognizes both eyes, and the calibration starts. A white dot appears in the center of the black screen, and the participant must follow it with their eyes as it moves to the edges of the screen. If the calibration is done correctly, data can then be collected. If not, the process is repeated until both eyes are correctly calibrated. After the calibration, the codification experiment begins, presenting the four codifications (Figure 1, Figure 2, Figure 3 and Figure 4) and the respective possible answers. After completing the

test, users were asked about their main struggles and tips for improving the protocol, in which the user listed various characteristics that helped them understand the weak and strong points of the test. The codifications used in the protocol are listed in (Figure 1, Figure 2, Figure 3 and Figure 4).

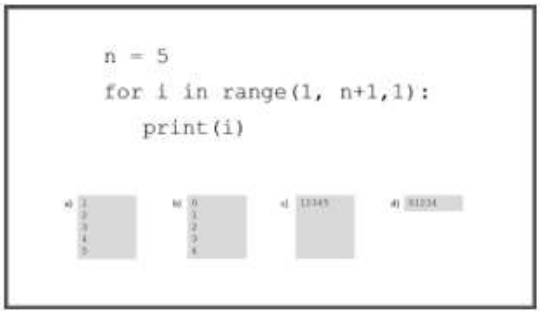

Fig. 1: Codification A

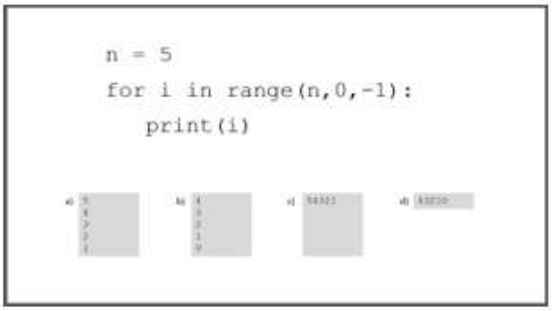

Fig. 2: Codification B

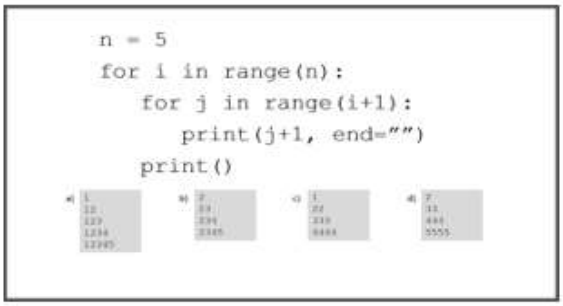

Fig. 3: Codification C

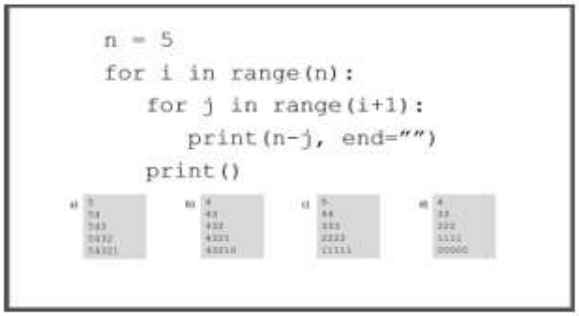

Fig. 4: Codification D

The procedure in which the Protocol proceeded is represented in Fig. 5.

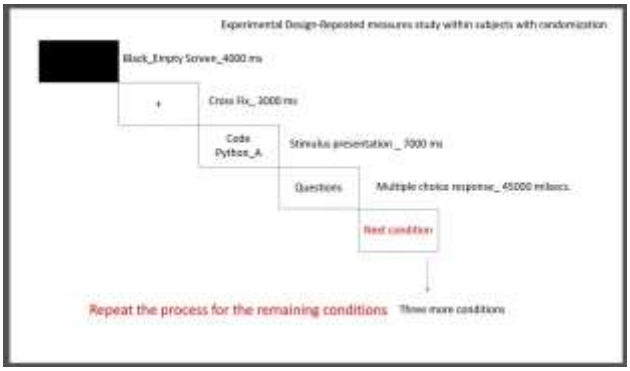

Fig. 5: Protocol Procedure

The order in which each condition was presented to each participant was different for each subject, as illustrated in Table 1.

Table 1. Randomization Table

Participant	Order of questions	Participant	Order of questions
P1	B, D, A, C	P16	C, D, B, A
P2	C, A, D, B	P17	D, C, A, B
P3	D, B, C, A	P18	A, D, C, B
P4	A, C, B, D	P19	C, A, B, D
P5	C, D, A, B	P20	B, C, D, A
P6	A, B, D, C	P21	A, B, D, C
P7	D, A, C, B	P22	D, A, B, C
P8	B, C, A, D	P23	C, B, A, D
P9	D, C, B, A	P24	A, C, D, B
P10	A, B, C, D	P25	B, A, C, D
P11	B, D, C, A	P26	D, C, A, B
P12	C, A, B, D	P27	A, D, B, C
P13	D, B, A, C	P28	C, B, D, A
P14	A, C, B, D	P29	B, A, D, C
P15	B, A, D, C	P30	D, C, B, A

3 Research Questions

This paper describes an experiment concerning students' primary problem when starting programming: the difficulty of using "for" loops in a programming language like Python. As such, we used some tests to obtain the gaze data necessary to identify which factors affect a user's capacity for code interpretation.

Several studies use eye tracking to study models of attention and decision-making in several situations and activities [8], [9]. In [10], the authors, besides concluding that eye-tracking software can directly inform the software tools and developers—letting them know which areas to change and clarify—it was also concluded that experts spend more time viewing complex statements and "jump around" more often, knowing exactly where to

gather information for certain aspects. At the same time, novices get easily distracted by comments and clumsily glance everywhere as they are not sure how the code itself works. This type of study can give meaningful insights to help students with difficulties. For instance, perhaps the code used should not include comments so as not to distract less experienced users from the central issue of the code. Additionally, the code should be made as simple as possible for this exact purpose.

This paper shall tackle three main questions that we plan to answer by the end:

1. Is it possible to predict the correctness of a multiple-choice answer based on the eye movement data?

- The movement data will be analyzed and compared to how accurate the answers are, taking into consideration the levels of proficiency of each user. According to this, it might be possible to identify patterns within correct answers.
- However, it is expected to simply predict the user's confidence, not precisely the correctness.

2. How does the complexity of a Python question influence the eye's behavior?

- Using "for" and nested "for" loops, it will be possible to test users' gaze when faced with differently composed code questions to see if their reactions change according to the complexity of the questions.
- It is expected that more complex questions will cause less experienced users to take more time analyzing the code before moving on to the answers, while more experienced users might have no trouble.

3. Can eye-tracking data identify areas in Python code that are commonly misunderstood or misinterpreted?

- As eye-tracking can detect confusion via constant revisits to the same areas, we might identify misunderstood areas that can later be fixed and polished.
- It is possible to expect that, yes, eye-tracking data might be able to identify areas that are commonly misunderstood.

Within this experience, there have been fifteen users who have reported not having any kind of experience in Python coding. On the remaining fifteen, however, it is possible to predict that the varying levels of Python programming proficiency that they said to have (from two to four, on a Likert scale of 5 points) will influence their eye movement patterns, as well as the complexity of the questions and how they interact with them. These results are assumed to later be used to predict how confident someone is in getting the correct answer to the questions, as well as help identify the areas in the code that might need alterations due to being commonly misunderstood.

4 Analysis

Before the analysis was started, there were defined "average" ranges for eye dilation (left and right eye) and the number of blinks per minute. These "standard" ranges were defined based on previous research on the subjects, [11], [12], [13], [14]:

- The human pupil is normally 2-4 mm in size.
- Human eyes usually blink on average between 14 and 17 times per minute.

Any deviation from these standards must be considered and analyzed to arrive at a more accurate result, taking into account the participant's context. For example, pupil dilation is not altered by light, as all the participants carried out the data collection in the same laboratory, with the same screen and lighting adjusted to this purpose.

In this study, three metrics were analyzed: LPD - Left Pupil Dilation, which represents the diameter of the pupil of the left eye in pixels; RPD - Right Pupil Dilation, which represents the diameter of the pupil of the right eye in pixels; and BKPMIN - Blinking Per Minute, which represents the number of blinking during 60 seconds (1 minute).

4.1 Codification A

Codification A consisted of just three lines of code, including a "for" statement and producing; as a result, the printing of a set of numbers in ascending order.

Although the correct answer was option A, the most answered option was C. Of all 30 participants, only nine answered correctly, making for a 30% success rate. In this condition, it can be seen that there was more significant pupil dilation in rows 2 and 3 compared to the first row. There was a lower blinking in the second row of code. Moving on to the answering options, both the pupils and the blinking level decreased in the most answered alternative, C. According to previous research, we can draw some interpretations from these deviations. The increase in pupil dilatation throughout the reading may indicate stress caused by the complexity of the code to be interpreted, while their decrease indicates greater focus. Less blinking indicates more attention/concentration needed for more complex instructions. These results are depicted in Figure 6, Figure 7, Figure 8, Figure 9, Figure 10 and Figure 11.

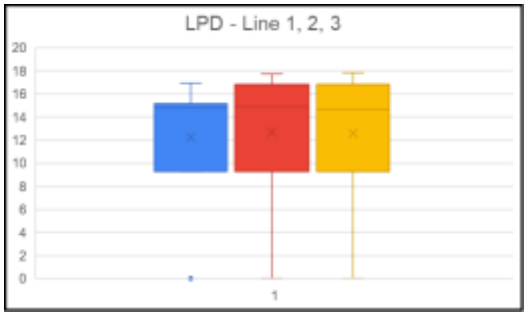

Fig. 6: LPD lines

Fig. 7: RPD lines

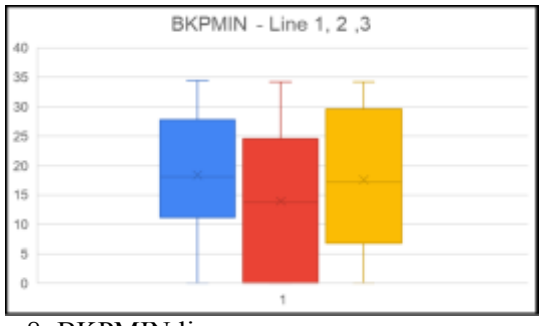

Fig. 8: BKPMIN lines

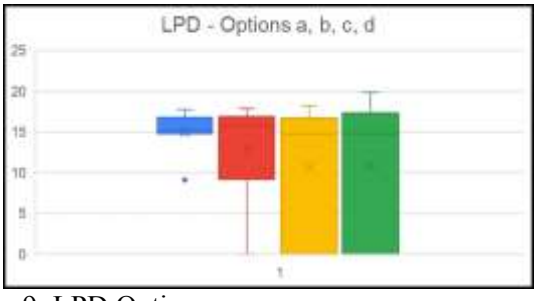

Fig. 9: LPD Options

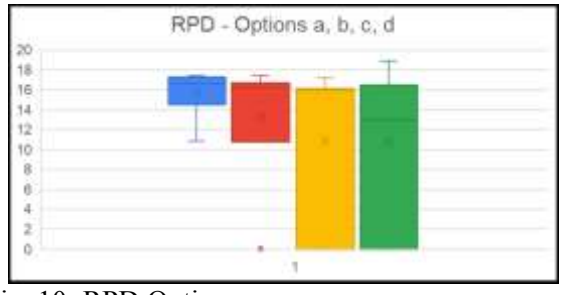

Fig. 10: RPD Options

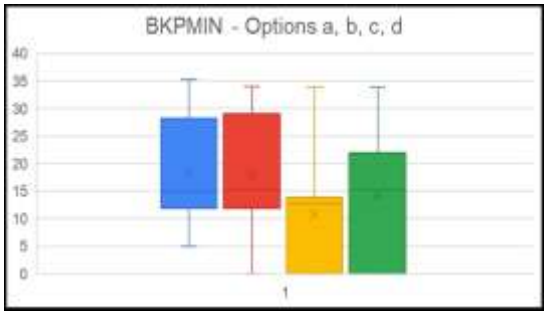

Fig.11: BKPMIN Options

4.2 Codification B

Codification B also consisted of just three lines of code, including a "for" statement but producing, as a result, the printing of a set of numbers in descending order. Although the correct answer was option A, most students answered option D. Of all 30 participants, only seven answered correctly, making for a success rate of roughly 23%. In this condition, it can be seen that the pupils of both eyes remained evenly dilated and constant along the lines of code; meanwhile, blinking increased in lines 2 and 3. Moving on to the answering options, pupil diameter decreased considerably in the most answered option, while blinking also decreased. A balanced pupil diameter can imply a lack of focus or understanding of the code to be interpreted, a sign of ”giving up” on understanding. Its increase in the most answered alternative is again associated with focus and trying to understand it. The increase in blinking in lines 2 and 3 can be interpreted as eyestrain and difficulty focusing. At the same time, the decrease in the options is a sign that the participants have focused on them, especially the most chosen ones. It is possible to observe that, although similar, this condition had fewer correct answers than the previous condition, which may mean that participants are having more difficulty interpreting codes that result in decreasing sequences. These results are depicted in Figure 12, Figure 13, Figure 14, Figure 15, Figure 16 and Figure 17.

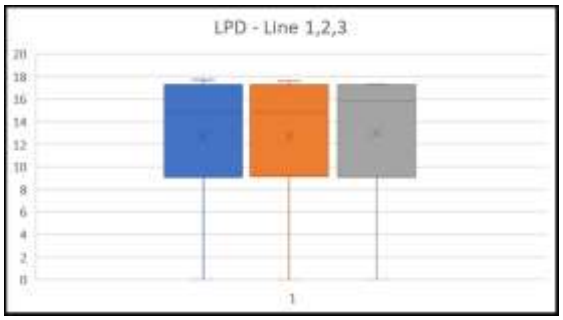

Fig. 12: LPD lines

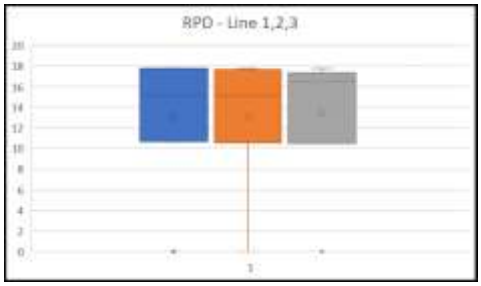

Fig. 13: RPD lines

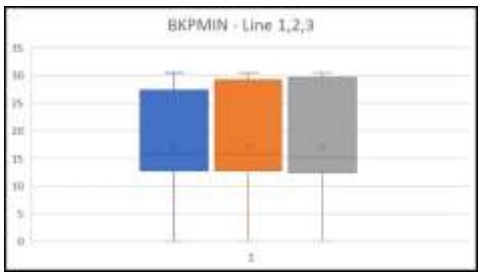

Fig. 14: BKPMIN lines

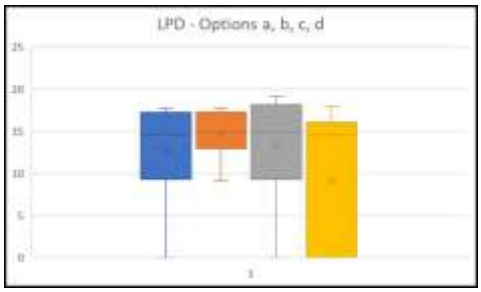

Fig. 15: LPD Options

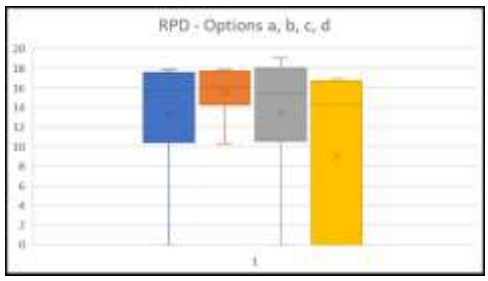

Fig. 16: RPD Options

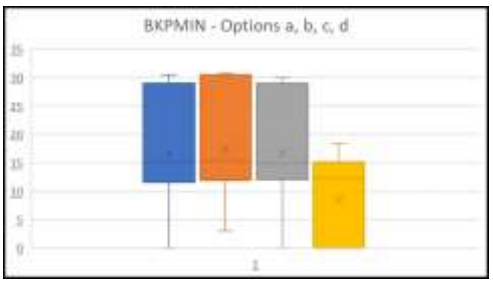

Fig. 17: BKPMIN Options

4.3 Codification C

Codification C consisted of five lines of code, including a chained "for" statement and producing, as a result, the printing of a set of numbers in ascending order. The correct answer to this condition is option A, which was also the one most answered by the participants. Of the 30, 11 chose option A, with a success rate of about 37%. The diameter of the pupils of both eyes remained stable along most of the lines, except line four, which is a challenging ”print()” function in which the diameter increased. Blinking increased on the first line and gradually decreased along the lines. Contrary to our expectations, blinking decreased more in option B, the second most answered option. The increased blinking in the first line could explain the participant’s ”shock” at seeing that it is a more complex code than the previous two. However, the stress decreases throughout the reading, and the participant stabilizes and focuses more. This could explain the decrease in blinking. These results are depicted in figures Figure 18, Figure 19, Figure 20, Figure 21, Figure 22 and Figure 23.

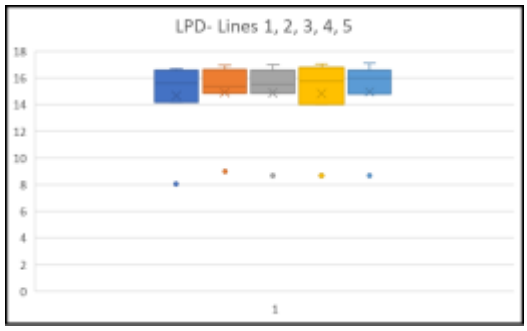

Fig. 18: LPD lines

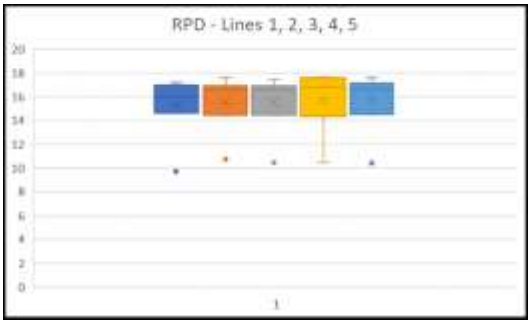

Fig. 19: RPD lines

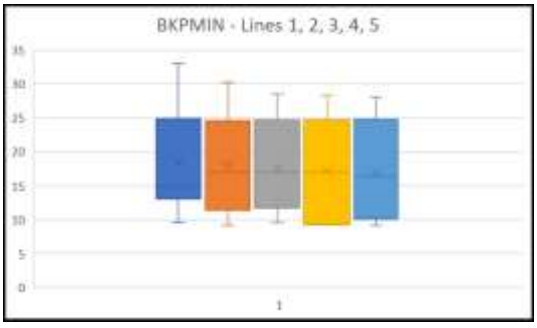

Fig. 20: BKPMIN lines

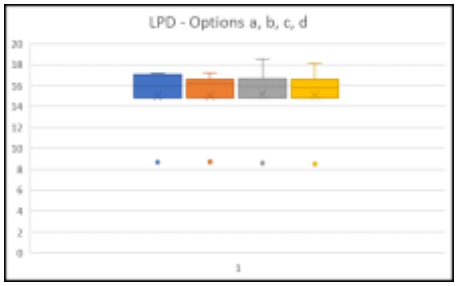

Fig. 21: LPD Options

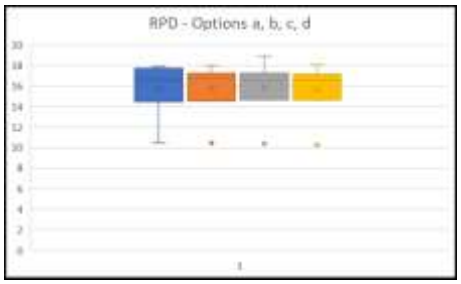

Fig. 22: RPD Options

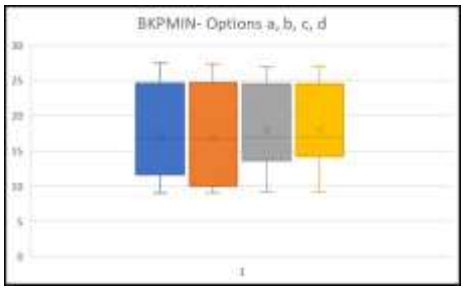

Fig. 23: BKPMIN Options

4.4 Codification D

Codification D also consisted of five lines of code, including a chained "for" statement but producing, as a result, the printing of a set of numbers in descending order. The correct answer to this condition is option A, but the one most answered by the participants was option C. Of all the participants, only 7 got it right – along with condition B, this was the question with the most errors, with a success rate of about 23%. On the first line and last lines, it can be seen that the diameter has a more significant variation, and it decreased considerably throughout the other lines of code. The blinking remained stable throughout the lines of code and answering options, increasing slightly in options C and D. The increase in pupil diameter throughout the reading of the condition shows that the participants had significant levels of stress with the interpretation of the code and the stable blinking may signal a high cognitive load, but little result due to a lack of knowledge. Although the participants answered alternative C most, they focused more on D, indicating indecision about which alternative to choose. These results are depicted in Figure 24, Figure 25, Figure 26, Figure 27, Figure 28 and Figure 29.

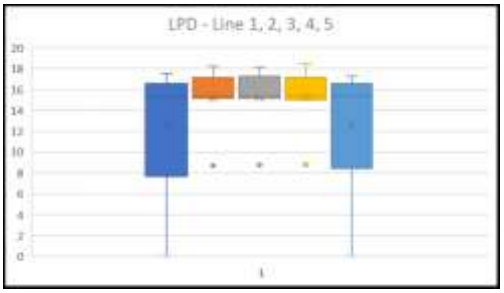

Fig. 24: LPD lines

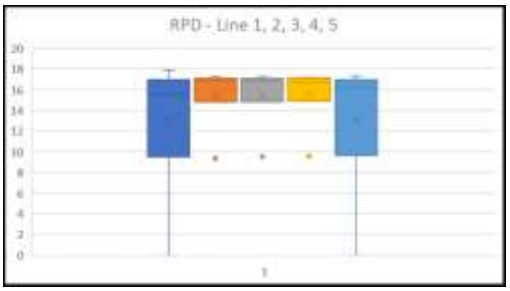

Fig. 25: RPD lines

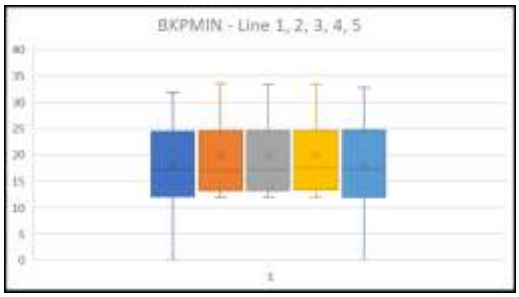

Fig. 26: BKPMIN lines

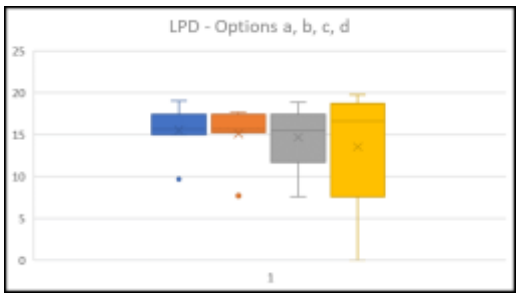

Fig. 27: LPD Options

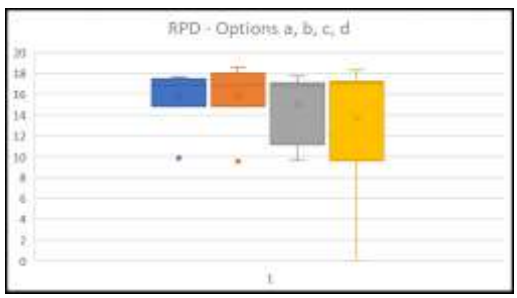

Fig. 28: RPD Options

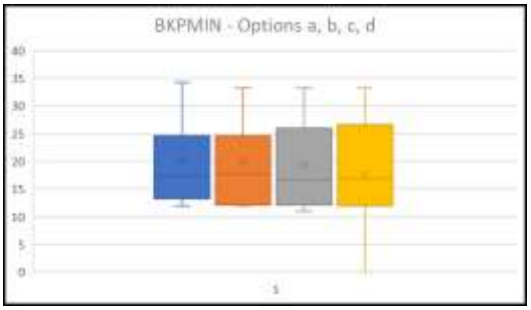

Fig. 29: BKPMIN Options

4.5 Post-Experiment Feedback

The results indicate that most of the participants had great difficulty responding to the questions, with a considerably low success rate at an average of only about 28%. The success rates by task type were 30% for Codification A, 23% for Codifications B and D, and 37% for Codification C. Additionally, there were constant variations in dilation and blinking, meaning that the participants were experiencing high-stress levels. With a post-study questionnaire, participants answered whether they had any difficulties and what they were. The most frequently cited struggle was not knowing or being confused with the syntax of the Python language, as reported by half of the participants (although all of them had already had some kind of contact with programming). Other struggles included difficulties with chained ”for” loops, difficulties with ”for” loops in general, and difficulties with loops that have decreasing results. Three participants claimed not to experience any difficulties. With these results, it will be possible to answer the three initial hypotheses. Concerning the first question, “Is it possible to predict the correctness of a multiple-choice answer based on the eye movement data?” the analyzed data enabled us to recognize the answering options on which the participants concentrated most, even if not the chosen option. Although it was impossible to predict the correct answer option, it is possible to know how confident the participant was in their response through their levels of stress and focus, analyzed through blinking and dilatation. Regarding the second question, “How does the complexity of a Python question influence the eye’s behavior?, the questions considered more ”complex” were those that had a chained loop, which were question C and question D. Despite being more complex, condition C received the most correct answers, while D received the fewest (tying with B). The main difference between the two questions was the sequence of their results - C presents the results in ascending order, while D presents them in descending order. In the most correct question, it is possible to notice less pupil dilation, which demonstrates greater concentration and understanding, while in the less correct question, there is a considerable increase in dilation. Therefore, the more complex the code, the higher the stress levels and the greater the pupil dilation.

Finally, and considering the hypothesis “Can eye-tracking data be used to identify areas in Python code that are commonly misunderstood or misinterpreted?”, we consider that the eye-tracker can, in fact, identify the areas of the code where participants gaze and fixate the most. Even though this data was collected, it was not explored in this study. The data analyzed only included the dilation of the right and left eyes and the number of blinks per minute. Although this data was not enough to identify the areas that are least understood, it was enough to identify the lines with higher rates of stress or tiredness.

4.6 Additional Considerations

To determine the difficulty levels of the code samples, we used a combination of structural complexity, participant performance, behavioral markers (pupil dilation and blink rate), and feedback. By the combination of all these factors, single "for" loops were rated as simpler than nested "for" loops. For programming tasks, single "for" loops typically present a lower difficulty due to their linear nature, while nested loops increase the difficulty by requiring additional layers of control flow to be managed simultaneously. This additional processing places greater demands, especially for novice learners lacking the structured knowledge to interpret complex constructs like nested loops efficiently.

The study's eye-tracking data provides evidence of this increased difficulty in more complex tasks. Heightened pupil dilation and reduced blinking rates observed during nested or descending loop codifications suggest higher stress and concentration levels, indicating that participants were nearing their cognitive processing limits. Complexity also increased with tasks requiring more challenging outputs, such as descending sequences. Eye-tracking data (notably pupil dilation and blink rates) showed higher stress and focus on specific lines in the more complex codifications, reinforcing these assessments. Lower success rates and longer completion times on nested loops (especially Codification D) further indicated higher difficulty levels. Participant feedback also confirmed that nested structures and descending outputs were particularly challenging.

These aspects together provided a well-rounded basis for categorizing code samples by difficulty.

4.7 Teaching Recommendations

We consider the findings in this study could give some suggestions on teaching strategies. For example, since the data suggest high stress during complex nested loops (Codification D), practical solutions, like dedicating more time to nested loop exercises or simplifying initial tasks to prevent overwhelm could be followed. With the difficulty of understanding "for" loops highlighted, curriculum changes or teaching techniques, such as visual aids, more gradual progression from simple to complex loops, or integrating eye-tracking data to create adaptive learning systems that detect and respond to student stress in real-time could also be used and integrated into the teaching practices.

Therefore, we consider potential broader applications of eye-tracking for programming education. For example, tracking eye behavior could allow educators to detect problem areas in other programming topics, like recursion or array handling, enhancing pedagogical approaches across different programming concepts. It can mainly be used to identify common misconceptions by explicitly stating the areas of the code that were consistently misunderstood by participants, as indicated by eye-tracking data. Based on the identified misconceptions, concrete teaching strategies that could be implemented to address those issues could be studied and proposed. For example, teachers could emphasize the importance of breaking down nested loops into smaller, more manageable steps, or integrate visual aids and animations to improve students' understanding of some concepts and structures.

5 Conclusions and Future Work

This study analyzed students' difficulties with the "for" loops of the Python language using an eye-tracker. The results indicate that a large proportion of students do experience difficulties when interpreting the loops for various reasons. The main difficulty pointed out is the language itself, going beyond the loop. Some students' lack of sufficient knowledge of the language's syntax could lead to inconclusive results. Among those who knew the syntax, the lack of practice and the difficulty of the codifications themselves were also problems pointed out by the participants. These results contribute to understanding the origin of this difficulty and reflecting on solutions. In this sense, it is essential to understand the reasons for the student's difficulties with the code in order to come up with new teaching strategies. In this case, the study was done with "for" loops, which are basic instructions in any programming language. From the data analyzed, it can be seen that the "for" loop causes high levels of stress in the participants, causing complications when it comes to their answers. This may, however, come from the lack of experience that many of the subjects demonstrated and reported, often coming from flaws in the teaching methods on such topics. However, we recognize the study's limitations. This is because all kinds of students were involved, aged between 16 and 28, and some didn't have a solid programming background. However, we found that everyone had the minimum knowledge to participate in the study in an intellectually honest way. This is an important subject and further research on the subject is strongly recommended in order to achieve a more effective way of teaching and educating about programming and, in particular, Python - as it is one of the most versatile programming languages used worldwide.

In conclusion, teaching practices can have some flaws that lead to minor errors on the part of students or even a general lack of knowledge of programming. In this sense, there is an urgent need to adopt new teaching strategies to make learning more efficient and effective. Frequent revision of basic programming concepts before moving on to more advanced conditions, the application of these concepts in exercises carried out in the classroom, and the appropriate monitoring by the teacher are fundamental for greater understanding and better results. With these changes, a new generation of programmers will be created who are much better trained and prepared for the evolution of technology.

Since this experiment opens up horizons for many more successors, it encourages the repetition of this experiment with a larger population and greater diversity, tests using other styles of alternative protocols and the inclusion of other Python concepts, or even other languages (C, C++, JavaScript...), enabling to compare the difficulties demonstrated by the participants observed.

Acknowledgment:
The authors would like to thank all the students who participated in the experiments.

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

During the preparation of this work, the authors didn´t use any AI service/tool and take full responsibility for the content of the publication.

References:

[1] C. S. Cheah. Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. *Contemporary Educational Technology*, vol. 12, no. 2, ep272, 2020. https://doi.org/10.30935/cedtech/8247.

[2] Yizhou Qian and James Lehman, Students' misconceptions and other difficulties in introductory programming. *ACM Transactions on Computing Education*, vol. 18, no 1, 1–24, October 2017. https://doi.org/10.1145/3077618.

[3] Prather, Raymond Pettit, Kayla McMurry, Alani Peters, John Homer, and Maxine Cohen. Metacognitive difficulties faced by novice programmers in automated assessment tools. I*n Conference on International Computing Education Research*, Espoo, Finland, pages 41–50. ACM, August 2018.

[4] Kadar, R., Mahlan, S. B., Shamsuddin, M., Othman, J. & Wahab, N. A. (2022). Analysis of Factors Contributing to the Difficulties in Learning Computer Programming among Non-Computer Science Students. In *Proceedings of the IEEE 12th Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, pp. 89-94, Penang Island, Malaysia. DOI: 10.1109/ISCAIE54458.2022.9794546.

[5] Kohn, Tobias. Teaching Python programming to novices: Addressing misconceptions and creating a development environment. ETH Zurich, 2017. I. Cetin, "Students' understanding of loops and nested loops in computer programming: An APOS theory perspective," *Canadian Journal of Science, Mathematics and Technology Education*, vol. 15, pp. 155-170, 2015. https://doi.org/10.3929/ethz-a-010871088.

[6] Gomes, A., Wei, K., Lam, C.-T., Teixeira, A., Correia, F., Marcelino, M. J. and Mendes, A. J. (2019). Understanding Loops a Visual Methodology. In *Proceedings of IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, Yogyakarta, Indonesia, Dezembro, 2019. DOI: 10.1109/TALE48000.2019.9225951.

[7] Chinedu Wilfred Okonkwo, Abejide Ade-Ibijola. Synthesis of nested loop exercises for practice in introductory programming. *Egyptian Informatics Journal*, vol. 24, no. 2, 2023, pp. 191-203, https://doi.org/10.1016/j.eij.2023.03.001.

[8] Pedro, Madeira. *Using eye-tracking data to study models of attention and decision-making*. Master theses in Biomedical Engineering, Universidade Nova de Lisboa, 2021.

[9] Jose Francisco. *Development of an eye-tracker for a HMD*. Master theses in Sciences, Universidade de Coimbra, 2017.

[10] Sharif, B., Shaffer, T. (2015). The Use of Eye Tracking in Software Development. In: Schmorrow, D.D., Fidopiastis, C.M. (eds) Foundations of Augmented Cognition. AC 2015. *Lecture Notes in Computer Science()*, vol 9183. Springer, Cham. DOI: https://doi.org/10.1007/978-3-319-20816-9_77.

[11] Spector RH. The Pupils. In: Walker HK, Hall WD, Hurst JW, editors. Clinical Methods: The History, Physical, and Laboratory Examinations. 3rd edition. Boston: Butterworths; 1990. Chapter 58, [Online]. https://www.ncbi.nlm.nih.gov/books/NBK381/ (Aaccessed Date: November 1, 2024).

[12] Layzer Yavin L, Shechter A, Rubinsten O. Mathematical and Negative Information Are Similarly Processed: Pupil Dilation as an Indicator. *J Intell.* 2022 Oct 3;10(4):79. DOI: 10.3390/jintelligence10040079.

[13] van der Wel, P., van Steenbergen, H. Pupil dilation as an index of effort in cognitive control tasks: A review. *Psychon Bull Rev*., 25, 2005–2015 (2018). https://doi.org/10.3758/s13423-018-1432-y.

[14] Paprocki R, Lenskiy A. What Does Eye-Blink Rate Variability Dynamics Tell Us About Cognitive Performance? *Front Hum Neurosci.*, 2017 Dec 19;11:620. doi:10.3389/fnhum.2017.00620.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed to the present research at all stages, from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received to conduct this study.

Conflict of Interest

The authors have no conflicts of interest to declare.