PseuToPy: Towards a Non-English Natural Programming Language

Patrick Wang
patrick.wang@isep.fr
ISEP – Institut Supérieur d'Électronique de Paris
Issy-les-Moulineaux, France

ABSTRACT

Most text-based programming languages found in introductory programming courses use English words. This fact alone can deter non-English speakers who wish to learn to program: how can we expect them to learn a programming language if they do not even understand the meaning of the keywords they are manipulating? In addition, the syntax and semantics of programming languages are also known causes of learners' mistakes. In this paper, we highlight these difficulties and then present PseuToPy, a programming language which can be localized in several tongues on the one hand and produce instructions close to these natural languages on the other. PseuToPy is still a work in progress: we have developed a version in French and hope to study its use in an educational context to see whether or not programming beginners find it easier to learn programming by implementing algorithms in their native tongues.

CCS CONCEPTS

• **Theory of computation** → *Grammars and context-free languages*; • **Human-centered computing** → *Accessibility systems and tools*; • **Applied computing** → **Education**.

KEYWORDS

programming education, programming language, formal grammar, pseudocode

ACM Reference Format:
Patrick Wang. 2021. PseuToPy: Towards a Non-English Natural Programming Language. In *Proceedings of the 14th ACM Conference on International Computing Education Research (ICER 2021), August 16–19, 2021, VirtualEvent, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3446871.3469787

1 INTRODUCTION

Learning to program frequently relies on the introduction of algorithmic notions followed by their application using a programming language. The literature identifies difficulties originating both from the syntax and semantics of programming languages. For example, and regarding the syntactic aspect, Altadmri and Brown [1] have analyzed 37 million code compilations in Java and have found that the most common mistakes were mismatched parentheses. On another note, Portnoff [3] suggests that programming languages should be taught as a language course. However, and while introductory programming courses naturally emphasize on the vocabulary and semantics of a programming language, grammar rules are less likely to be presented to programming beginners due to their possibly overwhelming aspects [2]. To illustrate this last claim, the Python 3.8 grammar specification defines the following rule to characterize `if` statements. One would need to refer to the `namedexpr_text` and `suite` rules (and so on if other rules are used) to fully understand the intricacies of this `if_stmt` rule.

```
if_stmt: 'if' namedexpr_test ':' suite
        ('elif' namedexpr_test ':' suite)*
        ['else' ':' suite]
```

But one could also think of modifying this rule. For example and as suggested below, the first line could be edited to add more semantics to it. A potential benefit of this modification could be that this rule would now guide learners in what is expected from the programming language when writing `if` statements.

```
if_stmt: 'if' 'the' 'condition' namedexpr_test
         'is' 'true' ':' suite
```

However, an obstacle remains: most of the time, programming languages are written in English, which might exclude non-English speakers. This observation is striking for two reasons. The first reason, quite obviously, concerns the fact that a design decision potentially puts aside a large part of the world's population. The second reason is related to the conclusion drawn by Qian and Lehman [5] that English proficiency seems to be a strong predictor of students' performances in introductory programming courses.

In this paper, we introduce PseuToPy: a programming language aimed at producing *almost natural* instructions while also providing support for multiple different tongues. This is done by designing a formal grammar based on the Python grammar specification and editing its rules by adding keywords in the targeted languages. As a result, PseuToPy allows learners to write instructions that resemble natural language sentences in their native tongue if supported. The following sections detail the design and implementation of PseuToPy as well as leads for future work.

2 PSEUTOPY

2.1 Design of PseuToPy

The design of PseuToPy follows five aspects: (1) the formal grammar of PseuToPy is based on the grammar specification for Python, (2) this formal grammar is modified to add alternative instructions in another language, (3) the alternative instructions can help disambiguate symbols or emphasize on programming concepts, (4) the

```
?assign: (testlist_star_expr ("=" (yield_expr|testlist_star_expr))*)
        | ("assigner" "à" testlist_star_expr ("la" "valeur" (yield_expr | testlist_star_expr))*)
```

Figure 1: The assign rule with the specification for Python and for the French version of PseuToPy.

instructions produced resemble *almost* grammatically-correct natural language sentences, and (5) instructions written in PseuToPy can be converted into Python code and then ran to show learners the results of its execution.

The first two aspects present the benefits of offering a complete compatibility with Python while allowing for the construction of instructions in a language other than English. For example, if we consider an implementation in French, the two following instructions would be identical: "`x = True`" and "`assigner à x la valeur vrai`"[1]. With this example, we can also see the interest of the third aspect. The "=" symbol is a common source of misconception found in the literature [4] which, in PseuToPy, can be replaced to emphasize on the actual meaning of this statement: a variable assignment. As we can also see with the example above, PseuToPy can also produce instructions that are close to natural language sentences while complying to the strict rules of a formal grammar. And finally, these instructions can be converted into the equivalent Python program and then ran to allow learners to see the Python implementation and check the correctness of their algorithms.

2.2 Implementation of PseuToPy

The implementation of PseuToPy is a three-step process. First, PseuToPy specifies its formal grammar and implements a parser in order to produce a syntax tree. This is done thanks to the lark-parser Python package[2]. The second step consists in taking this syntax tree and generating the equivalent Python instructions. And finally, the last step consists in modifying this grammar specification by adding keywords in the targeted language. For example, the assign rule illustrated in Fig. 1 shows on its first line the Python rule and on the second line the equivalent rule with French words.

With this example, we can also easily see how other languages could be supported in PseuToPy, which we have done for Chinese and Arabic on the assign rule as proof of concept [2]. These two languages were selected to determine whether PseuToPy could work with non-ASCII characters and with right-to-left languages.

At the time of writing, PseuToPy is still under active development and a preliminary version of the French grammar has been implemented. In particular, we have targeted grammar rules which specify instructions that beginners would commonly write. This comprises arithmetic and boolean operations, comparisons, variable assignments, control structures, and function definitions and calls. In the future, we wish to provide support for more statements such as class definition and object instantiation.

PseuToPy is an open source project which the authors see as a community effort. It is only with the help of a diverse group of researchers and educators that more languages can be supported. More information on how to contribute to this project can be found here: https://github.com/PseuToPy/PseuToPy.

[1]In English, this instruction would translated into "`assign to x the value true`".
[2]https://lark-parser.readthedocs.io/

3 CONCLUSION AND FUTURE WORKS

PseuToPy wishes to offer the opportunity to any non-English speaker to learn to program, using their own native tongue. This task is ambitious, clearly. This is why this article is also a call to the community interested in broadening access to programming to contribute to this project by proposing a grammar specification in their native non-English language.

In parallel to this community effort, our next step consist in experimenting with learners in educational contexts with two research questions in mind. First, would the use of a *natural* programming language help learners to produce less syntax errors? And second, what would be the effects of using a *natural* programming language on the design and implementation of algorithms by programming beginners?

With the first research question, we hope to improve the design of PseuToPy and its underlying grammar specifications. As a first step towards providing an answer to this question, we plan to adopt a design-based research methodology to specify the French keywords to be used in the grammar with the help of researchers, experts, and practitioners. We then hope to conduct experiments in ecological settings to determine if using one's native tongue reduces the risk of making syntax errors when writing programs.

With the second research question, we then wish to study the possible effects of using PseuToPy to learn programming and algorithmic notions. In particular, we plan to conduct a comparative study with French high school students, whose English proficiency should be good enough to be introduced to either PseuToPy or Python, to evaluate how fast or how easily they convert their algorithms into written programs and if we can distinguish learning gains between the two configurations. Follow-up studies would then concern the learners' appropriation of PseuToPy: would a mastery of PseuToPy help learners when switching to Python? Or would they perhaps prefer to use both at the same time?

REFERENCES

[1] Amjad Altadmri and Neil C.C. Brown. 2015. 37 Million Compilations: Investigating Novice Programming Mistakes in Large-Scale Student Data. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)*. Association for Computing Machinery, New York, NY, USA, 522–527. https://doi.org/10.1145/2676723.2677258

[2] Yassine Gader, Charles Lefever, and Patrick Wang. 2021. PseuToPy: Vers Un Langage de Programmation Naturel. In *Atelier "Apprendre La Pensée Informatique de La Maternelle à l'Université", Dans Le Cadre de La Conférence Environnements Informatiques Pour l'Apprentissage Humain (EIAH)*, Julien Broisin, Christophe Declercq, Cédric Fluckiger, Yannick Parmentier, Yvan Peter, and Yann Secq (Eds.). Fribourg, Switzerland, 87–95.

[3] Scott R. Portnoff. 2018. The Introductory Computer Programming Course Is First and Foremost a *Language* Course. *ACM Inroads* 9, 2 (April 2018), 34–52. https://doi.org/10.1145/3152433

[4] Yizhou Qian and James Lehman. 2017. Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review. *ACM Transactions on Computing Education* 18, 1 (Oct. 2017), 1:1–1:24. https://doi.org/10.1145/3077618

[5] Yizhou Qian and James D. Lehman. 2016. Correlates of Success in Introductory Programming: A Study with Middle School Students. *Journal of Education and Learning* 5, 2 (2016), 73–83.