Introductory Programming - Teaching

Sondre S. Bolland

April 19, 2024

1 Introduction

This paper aims to identify instances of the *Seven Deadly Sins* within programming languages [2], specifically in the context of teaching first-time programmers. We focus on Python (CS1) and Java (CS2) courses at the University of Bergen's Department of Informatics. Additionally, we discuss whether these languages align with the *seven principles towards more "teachable" languages* outlined by [2]. Finally, we provide our evaluation of whether Java and Python meet the criteria for introductory programming courses.

2 Seven Deadly Sins

The *Seven Deadly Sins* are undesirable features of a programming language in the context of teaching first-time programmers. Here we focus on two of the "sins" outlined by [2].

2.1 Less is More

Java, in comparison to Python, tends to be more verbose, requiring numerous modifiers and keywords to articulate desired functionalities. Python, on the other hand, adopts a more concise syntax, often perceived as simpler for beginners. Nonetheless, this "less is more" ethos in Python can be to the risk of program maintainability and readability. Notably, Python scripts often overlook the concept of encapsulation. Unlike Java, which employs access modifiers like *private*, *public*, *protected*, and *package private*, Python relies on annotations and conventions. Moreover, Java utilizes interfaces to restrict method exposure to relevant classes, a pivotal feature particularly in object-oriented programming. However, Python lacks such a mechanism for encapsulation as it does not have interfaces or anything equivalent.

2.2 Grammatical traps

Within grammatical traps, the notion of *syntactic synonym* emerges, wherein multiple syntaxes exist to define a singular construct. While [2] addresses this issue within individual languages, we underscore a comparable challenge when transitioning from introductory to secondary courses within our department, specifically the shift from Python to Java. Despite numerous syntactic similarities between these languages, Python distinguishes itself by employing unique terminology for certain keywords, deviating from the consistent conventions observed in many other programming languages. This includes:

In C++, C#, and Java, the main method serves as the entry point when executing a program. However, Python deviates from this convention by executing the first line of the script directly. To achieve a similar effect in Python, one can utilize the conditional statement *if __name__=="__main__":*.

	C++	Java	C#	Python
Object reference	this	this	this	self
Scope	{}	{}	{}	indentation
End line	;	;	;	New line
Datatype	string	String	string	str
Datatype	boolean	boolean	bool	bool

Another grammatical trap is the concept of *syntactic homonyms*, wherein constructs that appear syntactically identical possess two or more distinct semantics depending on the context. Both Java and Python exhibit this trait in the polymorphic behavior of certain operators, notably + and ∗. While + serves for both addition and concatenation, and ∗ denotes multiplication, in Python, it additionally signifies sequence repetition. In the Prior Knowledge Test in Programming [1], which assesses the programming proficiency of incoming higher education students, it was observed that many candidates adeptly applied + and ∗ in their conventional mathematical contexts. However, fewer accurately responded to questions concerning these operators when utilized in contexts involving strings and lists. Instead, many thought that expressions using + and ∗ with sequences would result in an error.

3 Seven Steps Towards More "Teachable" Languages

[2] also outlines seven principles for CS1 educators to follow to avoid the *seven sins*. Here we focus on two.

3.1 Provide a small and orthogonal set of features

To mitigate the challenges posed by *synonyms* and *homonyms*, it is advisable to opt for a concise set of distinct language features with non-overlapping syntactic representations. During the transition from Python to Java, several constructs become more intricate in the latter language. For instance, while Python features a single type of list data structure, Java distinguishes between static and dynamic lists. In the initial two weeks of the CS2 course, the emphasis is placed on mastering the Java programming language. Specifically regarding lists, our focus is directed towards dynamic lists as they bear closer resemblance to Python lists. This approach is intended to facilitates a smoother transition for learners accustomed to Python's simpler data structure paradigms.

3.2 Be especially careful with I/O

In the case of I/O Python follows more of the recommendations given by [2] than Java, but both fail at some. For instance, "The basic features of a good pedagogical I/O model as being a default idempotent I/O format for all data types", as both languages' I/O outputs the datatype string. As with lists, the transition period from CS1 to CS2 limits the functionality of I/O in Java by giving a method very similar to Python's *input*:

Regarding I/O operations, Python adheres more closely to the recommendations outlined by [2] compared to Java, although both languages exhibit shortcomings. For instance, neither language fully satisfies the requirement for "a default idempotent I/O format for all data types," as their I/O operations output data as strings.

Similar to lists, the transition from CS1 to CS2 in Java imposes limitations on the functionality of I/O operations by introducing a method akin to Python's *input*:

```
public static String input(String prompt) {
    System.out.println(prompt);
    String userInput = sc.nextLine();
    return userInput;
}
```

4 Do Java and Python Follow The Seven Criteria for Choosing an Introductory Language?

[2] gives seven criteria for choosing a programming language for an introductory course. Below these criteria are used to evaluate Python and Java **using the authors opinion**.

Criteria	**Java**	**Python**
Is the syntax of the language excessively complex or too sparse ?	May be too complex	Is too sparse
Are the control structures, operators and inbuilt functions of the language reasonably mnemonic?	Not for all	For most
Are the semantics of the language inconsistent, obscure, or unnecessarily complicated?	No	No
Are the error diagnostics clear and meaningful at a novice's level of understanding?	They are ok	A bit worse
Are parts of the language subject to unnecessary hardware dependencies or implementation-related constraints?	No	No
Is the language too big (over-featured) or too small (restrictive)?	No	Sometimes restrictive
Are the apparent virtues of the language equally "apparent" from the novice's perspective?	Absolutely not	Yes

References

[1] Nasjonal forkunnskapstest i programmering. https://programmeringstesten.no/. Hentet: 2024-04-19.

[2] Linda McIver and Damian Conway. Seven deadly sins of introductory programming language design. In *Proceedings 1996 International Conference Software Engineering: Education and Practice*, pages 309–316. IEEE, 1996.