A micro-research on the materialities of programming languages

Using the example of an array in Java versus Python

by Anaïs Siebers & R. Melis Baydag

Introduction

This paper deals with the question of the materiality of programming languages. Based on Paul Dourish's book "The Stuff of Bits" (2017), key concepts of materiality in the digital world will be analyzed in regard to programming languages. Different programming languages are designed for different use-cases which influence their materiality. Furthermore, the materiality of a programming language (time and place of invention, programming style, communication system) influences the possibilities of working with these languages. As Dourish (2017) points out "an algorithm as a program requires the programmer to incorporate not just the algorithmic procedure itself but also the ancillary mechanisms needed to make it operate" (p. 214). The materiality will be outlined based on a comparison of arrays, or data structures, in the programming languages Java and Python.

A data structure in a programming language is used to store data to be able to work with this data in the course of the program. "An array (more specifically) is a data structure that holds similar, related data."[1] It can be compared with a cupboard where you have multiple shelves to put things inside and get them back. Arrays can be one-dimensional or multidimensional. For reasons of simplicity, we will focus on one-dimensional arrays.

What does an array look like? What does it do? An array saves data and returns the data at a given position. Graphically, it could look like the table in Figure 1. In a programming language, it could look like this: my_array = ["element1", "element2", "element3"]. This array has three fields and stores one value in each field. One can access an element by using the index, i.e., the position of the element: my_array[1] refers to "element2". Furthermore, most programming languages are zero-based, thus the index starts with zero, not one: my_array[0] = "element1" (*see* Figure 1).

Figure 1 An array (my_array) with three fields, that contains an element each.

index	**0**	**1**	**2**
value of field with index	element1	element2	element3

Figure 2 illustrates two possibilities to create an array in Java and Python[2]. On the left, a Java array is created. Here, the first line initializes the array by stating that it is an array –

[1] BBC. (n.d.). *Programming techniques: Arrays*. Retrieved on 11.07.2022 from: https://www.bbc.co.uk/bitesize/guides/zfnny4j/revision/7

[2] Python does not have built-in support for arrays. Here we use Python lists as arrays, because they act similar and are often used analogously. Java has built-in arrays, but no built-in lists. Lists in Java are more complex data structures and require a

indicated by the squared brackets (int[]), which can only be filled with up to five integers (new int[5]). This array is called javaArray1. In the next five lines, the fields are filled with integers by accessing each field via its index (e.g. javaArray1[3] = 8;). Another possibility to create such an array is to assign the values in order by using curly brackets: {1,5,3,8,1023}. On the right, a python array is initialized. By assigning the squared brackets ([]), it is declared that python_array_1 is an array. In the next five lines, the array is filled with a boolean value, an integer, text and a decimal. The same array can be created by assigning the values in order using squared brackets: [True,5,“text",10.23].

Figure 2 Arrays in Java and Python

Java	Python

int[] javaArray1 = new int[5];
javaArray1[0] = 1;
javaArray1[1] = 5;
javaArray1[2] = 3;
javaArray1[3] = 8;
javaArray1[4] = 1023;
	python_array_1 = []
	python_array_1.append(True)
	python_array_1.append(5)
	python_array_1.append("text")
	python_array_1.append(8)
	python_array_1.append(10.23)

int[] javaArray2 = {1,5,3,8,1023};	python_array_2 = [True,5,“text",10.23]

Although arrays might seem very abstract, they can be used for a variety of use-cases. For instance, imagine a marathon: the names of the participants competing can be saved in the order of arrival. If John, Max and Julia participated in a race and Max would be the first, Julia second and John the third person to arrive, Max would be saved at the first position (participants[0] = “Max”), Julia at the second (participants[1] = “Julia”) and John at the third (participants[2] = “John”). Using the index, one can retrieve the name of the person in the third-place (participants[2]), which would be John (*see* Figure 3 The race scenario).

This is only a small example, but sometimes there are multiple large arrays containing information that belong together. Imagine that there are two arrays: One array saves the constellation of football matches (matches) and the other saves the corresponding outcomes (results). Then, the index can help get corresponding values from two arrays: matches[i] will retrieve information about the match i from the array matches and results[i] will retrieve the results of the match i. In the given scenario, England and France ended in a draw (1:1) and Germany lost against England (0:2) (*see* Figure 3 The match scenario).

Each programming language exhibits different material properties and different trade-offs that will affect processing, storage, transmission, error detection, malleability and the range of characteristics that might matter in different applications or use (Dourish, 2017, p. 17). The next sections comparatively analyze particular materialities of programming languages through the cases of Java and Python, and attempt to answer three main questions Paul

special import.We will further refer to Python lists as arrays in this context, because we concentrate the usage of these lists as arrays.

Dourish focuses on in his book: The question of knowledge (*What can you know? What must you know?*), the question of practice (*How can you use it?*) and the question of cooperation (*How does it make you cooperate?*). To make the restrictions and possibilities imposed by the digital representation (thus the materiality) of the array accessible, we will compare the realization of the array in Java and Python, two widely known and distributed yet very different programming languages.

Figure 3 Using arrays in two different scenarios that belong together

The race scenario[3]

```
String[] participants = new String[3];
participants[0] = "Max";
participants[1] = "Julia";
participants[2] = "John";
```

The match scenario[4]

```
String[] matches = new String[12];
matches[0] = "ENG vs. FR";
…
matches[11] = "GER vs. ENG";

String[] results = new String[12];
results[0] = "1:1";
…
results[11] = "0:2";
```

Questions of Knowledge

What can you know?

Programmers can directly access certain information from the implementation and usage of an array. The following example shows Python-code. An empty array is created and a boolean value (`True` or `False`) is inserted. Thus, the array apparently has one element so far, which is a boolean value and has to be at index 0:

```
python_array = []
python_array.append(True)
```

Now, we look at an example of an array in Java. Here the programmer directly sees that an array with a size of five is created, so only five values can be saved. Furthermore, the programmer knows which kind of data (data type) is saved in the array. In this case it is `int` which stands for integer. We can further see that only the first value is assigned with the number 1.

```
int[] javaArray = new int[5];
javaArray[0] = 1;
```

For programmers both examples are easy to understand. In cases where no values are directly assigned to the array, it is crucial that the name of the array is self-explanatory. Imagine an online shop. It is possible to add articles to the shopping basket, but initially it is empty and it might also be empty after usage. Now, an array could save the items of your basket, but it will only be created and cannot directly be filled with elements (`shopping_basket = []`). The elements will only be added later in the implementational procedure, if a customer

[3] For both scenarios, Java will be used.

[4] In this case String is used for reasons of simplicity – normally, dedicated classes would be used.

adds something to it. If multiple developers work on the same project not, it is important that the name is self-explanatory, so that it is understandable, what the array is, what it is used for, and so on. If, for example, another function such as "Delete all items from the shopping basket" is added, the programmer has to know which array has to be emptied.

Looking at a computer program closely, what is visible or seems to be material to non-programmers is quite different from the materiality of arrays. For instance, a person who knows about functions or has pre-knowledge on basic math would find it easier to grasp the structure of a programming language, such as Python, compared to a person that has no idea about the subject. Similarly, a certain level of English language is helpful to understand commands in a programming language. For non-programmers programming languages still possess a certain visually recognizable structure in terms of how numbers, symbols, or colors are organized, and they all together follow a pattern. Nevertheless, such visible features are also broadly shaped by human expertise or organizational implementation processes. Dourish's conceptualization of materiality, on the other hand, proposes not to attend exclusively to the sociality of what is virtual, as it actually hinders our understanding of their materialities.

What must you know?

In order to work with programming languages, one must at least know how programming and the underlying concepts work. Code represents a computer program. A *computer code* refers to a set of rules or instructions defined by a programming language. These are made up of words and numbers, and tell what the computer has to do, when they are put in the right order.[5] Therefore, the basic knowledge one has to possess is, among other things, the syntax of the programming language, the semantics of the programming language, programming conventions like naming of variables and functions, programming paradigms, and so on. The following example can be given for naming conventions: In Java the name of the array starts with a small letter and a new word is written without a separator (javaArray) but with a capital letter, whereas in Python '_' is used as a separator and no capital letters are used (python_array). This example also shows how programming languages are used differently which will be the topic of the next section.

Question of Practice – How can you use it?

In order to work with programming languages, one must at least know how to use the language, which is already outlined before: "properties – reversibility, robustness, directness, correspondence, and so forth – are essentially properties of specific digital materials" (Dourish, 2017, p. 22). In the given example, there are various implications concerning the

[5] BBC. (n.d.). *What is code?* Retrieved on 11.07.2022 from: https://www.bbc.co.uk/bitesize/topics/z3tbwmn/articles/zykx6sg

usage of an array depending on the programming language used. There are simple differences as to the way you can address data of an array. In Python it is possible to retrieve the last element of an array by using -1: given the array `my_array = ["element1", "element2", "element3", "last_element"]`, `my_array[-1]` thus returns `"last_element"`, an operation, which is not possible in Java. Furthermore, in comparison with Java, Python also allows retrieval of a subarray: `my_array[1:3]` returns `["element2", "element3", "last_element"]`. Another difference, which is very important in practice, is the size of an array and the data it contains. In Java the size of an array is predefined and the data type of all elements in the array is given (in the examples above it is `int`(eger)), which means that the developer does not need to test the values retrieved from the array to know which methods and operations he can apply. To give an example, if the value is an integer, numbers can be added, but adding a number to text (String) would result in appending the number to the text. On the other hand, in a String, words can be searched, yet this method leads to an error if performed on integers (*see* Figure 4).

Figure 4 Different methods and operations in an array

Addition with an integer-array Java[6]

```
int[] numbers = new int[2];
number[0] = "3";
number[1] = "1";

int add_res = number[0] + 2;
// add_res=5 after this operation

boolean cont_res =
      number[1].contains("Hello");
// this will lead to an error
```

Addition with a text-array Java

```
String[] texts = new String[2];
texts[0] = "Hi!";
texts[1] = "Hello World!";

String add_res = texts[0] + 2;
// add_res="Hi!2" after this operation

boolean cont_res =
      texts[1].contains("Hello");
// cont_res=True after calling this function
```

To change the size of a Java array, developers have to create a new array that fits more elements (in the example illustrated in Figure 4, the array was of size it was 2: `new int[2]`), if the array should contain 7 elements, a new array has to be created and all values have to be moved to the new array). But in Python, the size of the array is not predefined which means that the number of elements can always be increased or decreased. "Studies of students learning to write software suggest that the types of problems they encounter are coordinated with the lexical properties of the programming languages they are using" (Dourish, 2017, p. 9). This can also be found in the given example. Whereas students using Java might expect and encounter accessing a nonexistent index (for example: an array has a size of 4 and they try to access `my_array[1000]`) or an empty field, Python students will not experience and care about accessing empty fields. Accessing a nonexistent index means that an array of for

[6] The text behind the "//" is called a comment and allows programmers to write texts and comments in code which do not belong to the executed code.

example 5 was created in Java and then at a later position in the code, it is attempted to access the element at index 10, which does not exist, because there are only 5 values.

Python students encounter the same problem. But they also encounter an additional error concerning arrays. Compared to working with Java, in Python, they might not know the data type of the value they retrieved. Because values of different data types can be added to one array in Python (see Figure 2), accessing the first element may return an integer and the second one might be text. This means python students cannot know how to operate on it: If you, for instance, retrieve a number, then you can multiply, subtract or divide other numbers. But if you retrieve a text, you will be able to do different things, such as searching in the text (*see* Figure 4). Therefore, different operations are possible in Python depending on the element retrieved and this influences the source of errors. If one does not know what kind of element it is, one will check or assert in the code that the planned operation will work: "An algorithm as a program requires the programmer to incorporate not just the algorithmic procedure itself but also the ancillary mechanisms needed to make it operate" (Dourish, 2017, p. 214). This is the case in Python, where developers usually nevertheless only use one datatype in one array by convention. Moreover, this is why Python has an in-built method to check for the data type or assert that the data type is the required data type.

The last aspect of this section is that Java allows you to randomly fill in values in an array. Therefore, if one has 100 marathon participants, one can create an array for the order of arrival at the finishing line. This means we create an empty array of the size 100. Now some people have a health problem and quit during the race. In Java you can assign these persons to the end of the array. But in Python, you have to manually create an array with 100 empty objects (None) to be able to fill in the last or use a specific method called insert, which you can first tell the position of the value to be added and then the value (*see* Figure 5).

Figure 5 Filling in values in an array

Java	Python
String[] participants = new String[100]; participants[100] = "Max Mustermann"; participants[99] = "Maria Musterfrau";	participants = [None, None, None, None, None, None, None, …, None, None, None] participants[-1] = "Max Mustermann" participants[99] = "Maria Musterfrau"
	participants = [] participants.add("Max Mustermann") participants.insert("Maria Musterfrau")

If values are added in order, using the insert method in Python is comparable to Java. But when adding values randomly (e.g., if each participant draws a ticket to get his starting

position and is saved at his starting position in the array), you have to create the empty array in Python[7] (*see* Figure 6).

Figure 6 Filling in values in an array randomly

Java	Python
String[] participants = new String[100]; participants[42] = “Max Mustermann”; participants[77] = “Maria Musterfrau”;	participants = [None, None, None, None, None, None, None, …, None, None, None] participants[42] = “Max Mustermann” participants[77] = “Maria Musterfrau”

Question of Cooperation – How does it make you cooperate or work with others?

With regards to the question of cooperation, Java and Python allow a limited group of people (mainly the developers, who work in these languages) to both communicate and work with each other, even if they do not speak the same natural language. In this respect, programming languages create some form of social interaction between people, who would not be able to interact otherwise. In addition, different programming languages and paradigms influence how people work on the code, how it is maintained and how the workflow looks if bugs are fixed. Furthermore, how a programming language is used depends on its purpose. With some programming languages, such as Python, it is possible to work with large amounts of data, whereas Java is often used for business applications. Different usages mean differences in the way programmers code. For instance, data analysis of a single researcher in Python requires less cooperation than a big business application in Java, where up to hundreds of developers have to cooperate, and the people starting the implementation probably are not even in the team anymore.

It is the developers of programming languages (or committees), who define how programming languages should work, e.g., what one has to write to create a variable. Moreover, although there are certain things up to a programmer, the conventions (also mentioned above) sort of create informal practices among developers, e.g., the naming conventions of variables mentioned above. The following example from Oracle can be given for naming conventions[8]:

> “Except for variables, all instance, class, and class constants are in mixed case with a lowercase first letter. Internal words start with capital letters. Variable names should not start with underscore _ or dollar sign $ characters, even though both are allowed.
>
> Variable names should be short yet meaningful. The choice of a variable name should be mnemonic- that is, designed to indicate to the casual observer the intent of its use. One-character variable names should be avoided except for temporary "throwaway" variables.

[7] Therefore, Python has another data type called dictionary, which would be preferred in this case. But dictionaries (dict) are out of the scope of this article.

[8] Oracle. (n.d.). 9 - Naming Conventions. Retrieved on 30.01.2023 from: https://www.oracle.com/java/technologies/javase/codeconventions-namingconventions.html

> Common names for temporary variables are i, j, k, m, and n for integers; c, d, and e for characters."

The question of cooperation can also concern as to whether programming languages possess a "performative aspect", as illustrated by the author in the example of the spreadsheets. Dourish (2017) focuses on how material forms perform as organizational tools, how they are incorporated into our routines, practices and planning, or whether they create new ways of working (pp. 82-83). Concerning their role in social organization, programming languages themselves are not necessarily organizational tools, as we understand from the example of the spreadsheets. They are rather used to create tools for, such as, managing work, increasing efficiency and productivity, or enabling collaboration. In a way, they define what organizational tools can do, how they do it and they create new ways of working, which then determine organizational routines, practices and planning more indirectly, and most importantly limit social interactions to the extent of the language's capacities.

There has currently been an open-source movement, which is about publishing the code of programs. The intention behind this is to keep the transparency of underlying algorithms and reduce redundant programming by enabling the communities of developers to use it, while at the same time contributing to the code (e.g., increasing robustness, fixing bugs and realizing new features). Opposing this movement, there also are enterprises using certain programming languages that are, for example, private (so, one has to pay licenses to be able to use them). They might even encrypt their executable program code so that one cannot access their algorithms. Returning to our example of Python and Java, both are free programming languages and are used for proprietary as well as open-source projects. Python is a script-language and as such directly executable. Java on the other hand needs to be *compiled* first. This means that the human-readable code is translated to machine-readable code, which also improves the running-time. But this can be encrypted so that you are not able to *decompile* the code to retrieve the original code.

Conclusion

In this research we explored arrays in programming languages Java and Python, and raised key questions on materiality of these two programming languages based on what Dourish (2017) poses in his descriptive analysis on the materiality of the digital: *What can you know by looking at arrays in both Java and Python?* and *How can you use arrays (differently)?* Our comparative case study highlighted several findings on what the two programming languages allow one to know or do within the framework of their materialities.

On materiality, reflecting on Goody (1977), Dourish (2017) argues that "... what can be set out on the page—with the representational capacities of different lexical devices—shapes what can be known. These lexical devices become information technologies with associated material properties. While lists suggest hierarchies, columns create paths for addition and subtraction; lines and arrows enable categories and groupings. Such devices provide written texts with logics of manipulability and preconceived relations" (p. 9). As we illustrated in our

examples, the usage of a programming language depends on the possibilities and limitations the implementation of an array imposes. Relatedly, materiality of each programming language is restrained by differences in terms of the syntax, the semantics and how conventions of the language are named. For example, one can create an empty array of a specific size and already enter values for high indices in Java, whereas one can therefore put data with different data types in a Python array.

It should be noted that materialities of programming languages come about in our social organizational life as well. In his work, Dourish (2017) does not necessarily overlook "[the]complex and evolving relationship between forms and possibilities of digital materials and the shape of human practices that create and respond to them" (p. 25). Both Java and Python, as digital materials, have a degree of sociality depending on the knowledge of the human users or the social organization that they are embedded in. The purpose of a programming language might define how working among developers should be like in the sense of whether it requires an intensive cooperation among large groups of developers or a single researcher should be enough. Nonetheless, following Dourish's conception, the main point of our research was to inquire what the programming language allows one to know, *or* make one know and do, that is, how their materiality comes about.

The micro research on the arrays in Java and Python concludes that these languages themselves make you able to use them in certain ways and not others. It exemplified, as Dourish (2017) points out, "ways in which representational forms, their associated material properties and knowledge practices are coupled" (p. 9). It emphasized that material properties of programming languages are beyond those that are immediately visible to people when they interact with or use them. They rather possess their "internal representations" as digital information systems - that is, their own materialities.

References

BBC. (n.d.). *Bitesize*. https://www.bbc.co.uk/bitesize

Dourish, P. (2017). *The Stuff of Bits. An Essay on the Materialities of Information*. Cambridge, MA: MIT Press.

Goody, J. (1977). *The Domestication of the Savage Mind*. Cambridge, UK: Cambridge University Press.