The Future Of Programming Languages And The Language Of The Future

Adapted from: https://www.techrepublic.com/article/the-future-of-programming-languages-what-to-expect-in-this-new-infrastructure-as-code-world/

Read the text and answer the questions:

Developers are moving away from managing physical servers to calling APIs that touch storage, compute, and networking resources. In turn, developers are trying to automate everything as code through static configurations, scripts, and files. Such automation would be easier if developers had programming languages that matched the task at hand, but they don't. So, using a general purpose language like Java, a developer might invest thousands of lines of code to try to express business logic and mostly fail.

To solve this, we're seeing companies like HashiCorp (HCL) and oso (Polar) release special-purpose declarative languages. Even at the risk of programming language proliferation, this feels like the right way forward: Purpose-built instead of general-purpose languages. However, we're likely to see many of these programming languages rise and fall before we settle into a useful set of standard declarative languages.

The irony is that the "novel" approach taken by special-purpose declarative languages really isn't very novel. Years ago, programming languages split between functional (declarative) programming languages like Lisp and imperative programming languages like C. While the latter dominated for decades, functional declarative languages are making a comeback, said Jared Rosoff in an interview, a software executive who has built product at VMware, MongoDB, and more.

Even Polar, a declarative logic programming language specialized for making authorization decisions and tightly integrating with an application's native language, really isn't new. As Sam Scott, co-founder and CTO of oso, suggested in an interview, Polar has its roots in Prolog, which was developed way back in 1972, yet has the feel of imperative languages like Python. This is important because it's difficult to encode authorization logic in traditional, general-purpose programming languages. Doing so in a declarative language like Polar is more expressive and concise--think "tens of lines of code" instead of "thousands of lines of code."

And yet, many will question whether creating new programming languages is the right approach. How many do we really need? The short answer is one: The Language of The Future.

The following prediction is offered up for the near future (the next 10-20 years or so). Beyond that and we start getting into the realm of ScienceFiction - which isn't bad, but that's extrapolating the curve well beyond the dataset.

- The Language Of The Future will strongly borrow from both Lisp and Smalltalk; but will be neither of these.

- The LanguageOfTheFuture will not be C/C++, Java, or any of the DotNet languages. The CommonLanguageRuntime and the JavaVirtualMachine will likely be important targets in the near future, but not in the longer term.

-The Language Of The Future will use Dynamic Typing by default; with Type Inference performed as a performance optimization. In other words, it will use Soft Typing.

-The Language Of The Future will be suitable both for Large Applications and small applications (like scripts). It will be (relatively) easy to do Meta Programming, and write Domain Specific Languages.

-Implementations of the Language Of The Future will be capable of both ahead-of-time compilation and interpretation; including the compilation of code at runtime.

-The Language Of The Future will be Open Source -- meaning that its specification will not be controlled by any one vendor, the specification will be published and freely available, and an Open Source implementation will be available.

-A theorem-proving system such as Prolog Language will be available; and used by the language itself to assist in proving programs correct. This won't be foolproof, of course, this is still an active area of research.

-Relational features, tables and records as data types, as well as a database type will be first-class elements of the language.

-There will not be a multitude of identical types distinguished by their storage (table, array, file, stream, etc). It will simply be an attribute optimizable on the fly. The runtime can choose to store an object in memory, across the wire, or in a certain file format. All of the code that translates and map an object to a file to a table to a widget will be gone.

That said, there will be other ways (in the further distant future) of communicating algorithms, etc. to a computer besides text. Heck, we have this today with visual GUI-builders of all descriptions, spreadsheets, and numerous other such technologies. As things progress, more and more will be programmed by means other than humans typing on a keyboard.

Instructions:
Choose the correct option for each sentence

1. The Language Of The Future will be Open Source means that ___.

A. It will be of free access
B. It will come from an open origin

2. The most important elements of the language will be ___.

A. Tables and code
B. Database types and data types

3. According to the text, what are other ways of communicating algorithms to a computer?

A. Besides text of all descriptions
B. GUI builders and spreadsheets

4. What tool will be used to help testing programs?

A. A theorem-proving system
B. Ahead-of-time compilation and interpretation

5. What kind of typing will the language of the future have?

A. Soft typing
B. Hard typing

6. Why are companies like HashiCorp (HCL) and oso (Polar) releasing special-purpose declarative languages?

A. Because developers invest thousands of lines of code to program.
B. Because developers need programming languages that match the task at hand.

7. The novel approach taken by special-purpose declarative languages is completely new.

A. True
B. False

8. Imperative programming languages dominated the programming world for decades.

A. True
B. False