**Unit 6: The Security of Programming Languages Debate**

**Objectives:**

* Review the security capabilities of a selection of programming languages, which include F#, Python, Rust, and Swift.
* Engage in a debate about the above languages.
* Explore some of the advances needed to support secure programming.

**Outcomes:**

* Recommend which programming language is the optimum to adopt to support a secure system.
* Provide evidence to support your recommendation.
* Make recommendations on the way(s) in which programming languages might be extended to meet the security needs of our modern world.

**Reflection:**

In the 21st century, modern systems programming languages like Rust have emerged. For writing clarity, reliable, and performant code, # is a universal programming language. F# allows you to write clean, self-documenting code instead of getting bogged down in the nitty-gritty of the programming language.

In my opinion, Rust's safety guarantee is one of its most compelling features. Because Swift is essentially Objective-C with a different syntax, it is vulnerable to many of the same threats as Objective-C code. Programs are protected and can be written in any syntax, so no language is secure.

The speed of Rust is superior to that of Python, but the ease of use of Python is superior as well. Even so, the construction costs are higher for Rust than Python. Compared to Rust's development costs, Python's are lesser, but it's less secure. In contrast to Python's dynamic typing, Rust uses static typing. F# addresses several of the issues faced by Python developers. static kinds don't give up Python developers' love of shortness despite the fact that they're more complex (Barguzar, 2022).

A safe system programming language is what we're talking about here (i.e., a language that can build systems other software runs on, like OS kernels). In order to meet these demands, you'll want to use C, C++, or Rust (H, 2022).

A future programming language should be able to handle parallel computations better, in my opinion. For synchronization processes in today's languages, dealing with locks, threads, or callback functions can be challenging. The goal of a future programming language is to make it easier for programmers to express parallel operations without encountering race conditions, side effects, or deadlocks.

**References**

Barguzar, A. (2022). *Python versus Rust: What Are the Differences?* [online] www.netguru.com. Available at: https://www.netguru.com/blog/python-versus-rust#:~:text=As%20an%20interpreted%20language%2C%20Python [Accessed 15 July. 2022].

H, L. (2022). *Python vs Swift: Which Language is Better to Learn*. [online] Cleveroad Inc. - Web and App development company. Available at: https://www.cleveroad.com/blog/python-vs-swift/ [Accessed 15 July. 2022].