**Unit 3: Introduction to Programming Languages**

**Objectives:**

* Review the history and concepts of programming languages.
* Investigate language concepts using Python as an example.
* Describe best practices and methods to overcome common security issues.

**Outcomes:**

* Describe some key milestones in the development of programming languages.
* Outline some of the key paradigms that define the different types of languages.
* Explain the key concepts that determine the operation of programming languages.
* Discuss key programming challenges and recommended best practices.
* Explain what design patterns are and when to use them.

**Reflection:**

Programming languages are used to communicate with a computer. The accuracy and comprehensiveness required by programming languages set them apart from other forms of human expression.

Programs and programming languages can be presented in different frameworks or styles. Programming paradigms are defined by their structures, features, and beliefs about how to solve commonly used programming issues. Programming paradigms include:

1. In imperative programming, the desktop is given a series of specific commands, each of which must be carried out exactly as written. In imperative programming, each step of how a program works is described in detail.
2. It is motivated in procedural programming to break down the program into functions in order to increase modularity and organization
3. A third aspect of functional programming is that it extends the idea of functions.
4. Declarative programming is all about obscuring complexity and bringing programming languages closer to human speech and thought (Cocca, 2022).

The key programming challenges are:

1. Debugging
2. Keeping up with technology
3. Time estimation
4. Security threats
5. Working with another person code

Following are the best practices that overcome these challenges (K, 2021):

1. Perfect Your Indentations and Make Use of Comment Lines
2. Simplification of Code
3. Effective Debugging
4. Code everyday

It is common practice in the field of computer science to use "design patterns" to help solve recurring issues in software development. In contrast to a finished design, a pattern isn't something that can be translated into code right away. Description or template for how to address the problem that can be used in numerous varying situations (Maioriello, 2002).

**References:**

Cocca, G. (2022). *Programming Paradigms – Paradigm Examples for Beginners*. [online] freeCodeCamp.org. Available at: https://www.freecodecamp.org/news/an-introduction-to-programming-paradigms/ [Accessed 26 Jun. 2022].

Maioriello, J. (2002). *What Are Design Patterns and Do I Need Them?* [online] Developer.com. Available at: <https://www.developer.com/design/what-are-design-patterns-and-do-i-need-them/>.

K, B. (2021). *6 Best Programming Practices!* [online] Medium. Available at: https://towardsdatascience.com/6-best-programming-practices-f2605c5b469c.