**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 23 Jun. 2022].

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

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/ [Accessed 23 Jun. 2022].

Need to read chapters 2, 6, 7, 8 of the course file (Pillai, 2017) and Cifuentes & Bierman (2019) and then answer the questions:

**What factors determine whether a programming language is secure or not?**

Developers may write readily legible code since Python is an open-source language and has a wealth of documentation (Pillai, 2017). Other languages, on the other hand, may produce illegible, difficult-to-modify code that raises maintenance costs and may introduce security risks each time the code is changed. This is because they lack adequate documentation. A highly rich environment is provided by Python for creating reusable design and pattern systems (Pillai, 2017). Additionally, Python pattern implementation is frequently quite simple and straightforward.

**Could Python be classed as a secure language? Justify your answer.**

If properly developed, Python may qualify as a secure language.

The security of it, nevertheless, relies on how the programmer uses it, just like with any other language.

It's possible for the code to be badly written, use complicated functions, have significant coupling, use unsafe libraries, etc.

In this case, Python would be equally insecure as any other language.

**Python would be a better language to create operating systems than C. Discuss.**

C appears to operate more swiftly. In contrast to Python, variables in C must be declared explicitly to avoid the program misinterpreting them. Python's syntax and error debugging are easier to understand. Python also uses an automatic garbage collector, in contrast to C, where memory management must be done manually by the programmer. Additionally, Python makes building data structures easier and has more built-in functions (Sidana, 2021).

According to a study (Prechelt, 2000) that also examined six other applications made in different programming languages, Python is a very effective replacement for C for managing large volumes of computation and data. It should be made clear that the main advantage of this language is that it makes programmers more productive.

**References**

Pillai, A.B. (2017) Software Architecture with Python. Birmingham, UK. Packt Publishing Ltd.

Prechelt, L. (2000). An empirical comparison of c, c++, java, perl, python, rexx and tcl. IEEE Computer, 33(10), 23-29. [Accessed 24 Jun. 2022].

Cifuentes, C. & Bierman, G. (2019) What is a Secure Programming Language? 3rd Summit on Advances in Programming Languages (SNAPL).136(3): 1 - 15. [Accessed 24 Jun. 2022].

Sidana, U. (2021) Python vs C: Know what are the differences. Available from: https://www.edureka.co/blog/python-vs-c/ [Accessed 24 Jun. 2022].