v4

Penetration Testing Student

Introduction to Programming

Section 02 | Module 01

Table of Contents

Module 01 | Introduction to Programming

Learning Objectives

By the end of this module, you should have a better understanding of:

- Basic concept of programming
- Typical programming constructs

1.1

What is Programming

1.1 What is programming

Programming basically creates a set of instructions that a computer may follow.

It can be used to automate tasks, leaving specific things to be done by a machine instead of a human.

1.1 What is programming

In order to make a computer do specific tasks, such tasks should be set-up in a way that the machine can understand them.

There are various programming and scripting languages that can be used to achieve this. They may have a different **syntax** and **usage requirements**, but their purpose is the same - to help humans by automating tasks.

1.2

Low and High Level Languages

1.2 Low and high-level languages

Programming languages can be divided into two main groups:

- **Low-level languages**
- **High-level languages**

The level can tell how close they are to the hardware.

1.2 Low and high-level languages

Low level languages are interpreted directly by the computer. Their advantage is that you can do almost everything in them, while their disadvantage is their complicated nature that can lead to a **vulnerability**, if the developer doesn't have a deep understanding of the language's capabilities.

1.2 Low and high-level languages

High level languages offer ease of development, but they are less flexible. If there are no available **libraries** that hold certain functionality, writing custom functionality from scratch might be a very difficult task.

1.2 Low and high-level languages

An example of a low-level language is **assembly**, which consists of instructions for the processor itself.

On a side note, hackers and penetration testers use the assembly language during their (advanced) exploit development activities.

1.2 Low and high-level languages

Modern processors execute billions of operations per second, so writing a program using processor instructions might not seem to be worthy of any effort. There are only a few modern software types where such a language might be used, for instance, device drivers.

1.2 Low and high-level languages

On the other hand, **high-level languages** offer much more development convenience and support broader operations. Lots of single hardware operations are invisibly managed by the language engine.

Java, **Python**, and **Visual Basic** are examples of high-level languages.

1.2 Low and high-level languages

However, programs created using high-level languages cannot reside on a bare operating system and will need some software already installed on the system to run.

For example, in order to run Java programs, you need to install the Java Runtime Environment.

https://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html

1.2 Low and high-level languages

If you are about to create code that will operate directly on computer memory, you should use a lower level language.

1.2 Low and high-level languages

If you are about to create a fancy graphical interface, you should use a higher level language and its graphical libraries that are available.

Programming vs. Scripting

1.3 Programming vs scripting

High level programming languages consist of:

- **Programming languages**, and
- **Scripting languages**

1.3 Programming vs scripting

Programming languages require a **compiler**. What does that do?

After writing code in your language of choice, you need to use a **special piece of software** called a compiler to convert your plain-text program file into something unreadable by a human, but readable by the language environment. You can program in **Java** or **C++** to experience this.

1.3 Programming vs scripting

On the other hand, scripting languages are usually **interpreted**, which means that the software environment installed on your computer can read a plain-text program file the same way that you can, and it can execute the instructions without changing the file in any way.

Some examples of such scripting language are **Visual Basic,** an partially, **Python.**

Basic Concepts

1.4 Basic Concepts

Regardless of the programming language level, or whether we are dealing with a scripting language or not, all programming languages share some basic concepts that help in planning and executing the tasks that they are told to perform.

1.4 Basic Concepts

Next slides will shortly present some similarities between programming languages.

1.4 Basic Concepts

Each programming language has its own **syntax.** You can think of it like grammatic rules in human languages.

Syntax may require some instructions to use certain characters; for example, „;” at the end of each **statement**, while in another language this may not be needed.

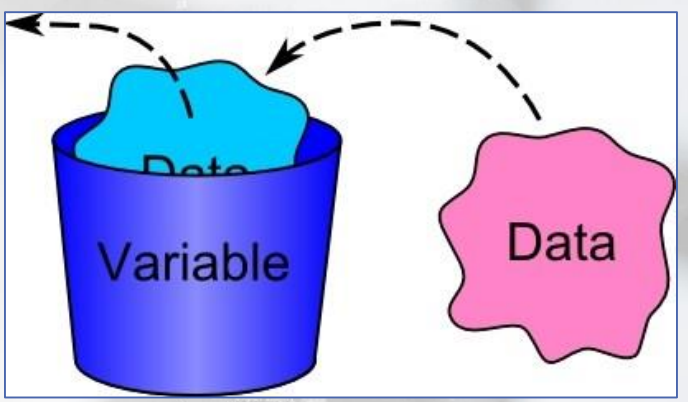
1.4.1 Variables

Programming also uses **variables**, which are containers that can be named and filled with data.

Depending on the programming language, you may also have to choose the correct variable **type**. Numbers, words, or single characters are usually stored in different variable types.

1.4.1 Variables

If you want to collect the name of a program user, you need to create a **variable;** for example, a variable named **„username”** in which the name will be held and used throughout the program’s execution.



https://i0.wp.com/www.fatosmorina.com/wp-content/uploads/2015/03/Variable-in-Java.jpg

1.4.2 Functions

Programming languages allow users to create and use their own **functions.**

In short, functions are pieces of code responsible for some repeatable tasks.

1.4.2 Functions

Functions use **arguments** and might **return** a value.

You can think of a **function** as a box that takes **arguments** (some input), processes them somehow, and then **returns** (throws out) some value – that is, the outcome of **arguments'** transformation inside this box.

1.4.3 Conditional statements

Every programming language contains **conditional statements**.

Conditional statements means that there is a condition to be checked (i.e., is the user's name Bob?) and that there is at least one instruction defined on how to proceed further.

1.4.3 Conditional statements

Conditional statements are something that we deal with in everyday life.

The following image is a good example of just that!

hhttps://www.excelcampus.com/wp-content/uploads/2015/06/The-IF-Function-in-Excel-Traffic-Light-Example.png

1.4.4 Loops

Loops are also common on every programming language and are set of instructions that need to be executed numerous times. They are often paired with conditional statements in order to check if they should stop, or if they should repeat its instructions again.

1.4.4 Loops

For example, when shopping, you want to buy five apples. You take one off the shelf and put it in the basket. When you notice you do not have 5 apples yet, you take another one until there are 5. You just executed a **loop**!

1.4.5 Understanding the code

When dealing with a new, unknown programming language that you do not understand, you should always start from checking all the aforementioned basic constructs in the language's manual. This will greatly help you understand what the inspected code does.

Conclusion

1.5 Conclusion

All programming languages consist of the same building blocks. When learning your first programming language, you may find the process a bit challenging. Don’t worry though, coding proficiency will come with time and you will find learning new programming languages easier as time progresses.

1.6

References

References

Java Runtime Environment

https://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html

Variable in Java

https://i0.wp.com/www.fatosmorina.com/wp-content/uploads/2015/03/Variable-in-Java.jpg