**Compiled vs Interpreted programming concept:**

A compiler takes as input a source program and generates as output another (equivalent) program, called a target program, which can then be executed (Sestoft, 2012)

In this example, the host machine (It can be native machine code or virtual, if it is intermediate language) compiles the code, and during this time it performs several checks to identify possible errors within the code (syntax, style-checking, semantic checking). The main advantage of compiled languages is, that once the code gets compiled it works as desired.

On the other hand, Interpreted languages offer interactivity and portability. This means, that the code is converted to a blocks at a time (line by line). The main advantage of using these languages hides in faster debugging, while on the other side, these languages tend to be slower.

Many modern languages (Java, C#) compile source code into an intermediate language (CIL, or byte code) that runs on a virtual machine. The benefit of this approach is that it combines best of the both worlds. The code gets compiled at first execution and for each next run it is held in memory (JIT compilation). That way the executables run almost as fast as native code (Lecturecasts, 2023).

References:

Sestoft, P. (2012). Programming Language Concepts. Undergraduate Topics in Computer Science. Springer-Verlag, London. [Accessed 28 May 2023]

University of Essex Online. (2023). Lecturecasts [Accessed 28 May 2023]