Compilers vs. Interpreters

Introduction

There are two different approaches about how to convert a program from source code into binary code that could be executed by the CPU.

Compiler

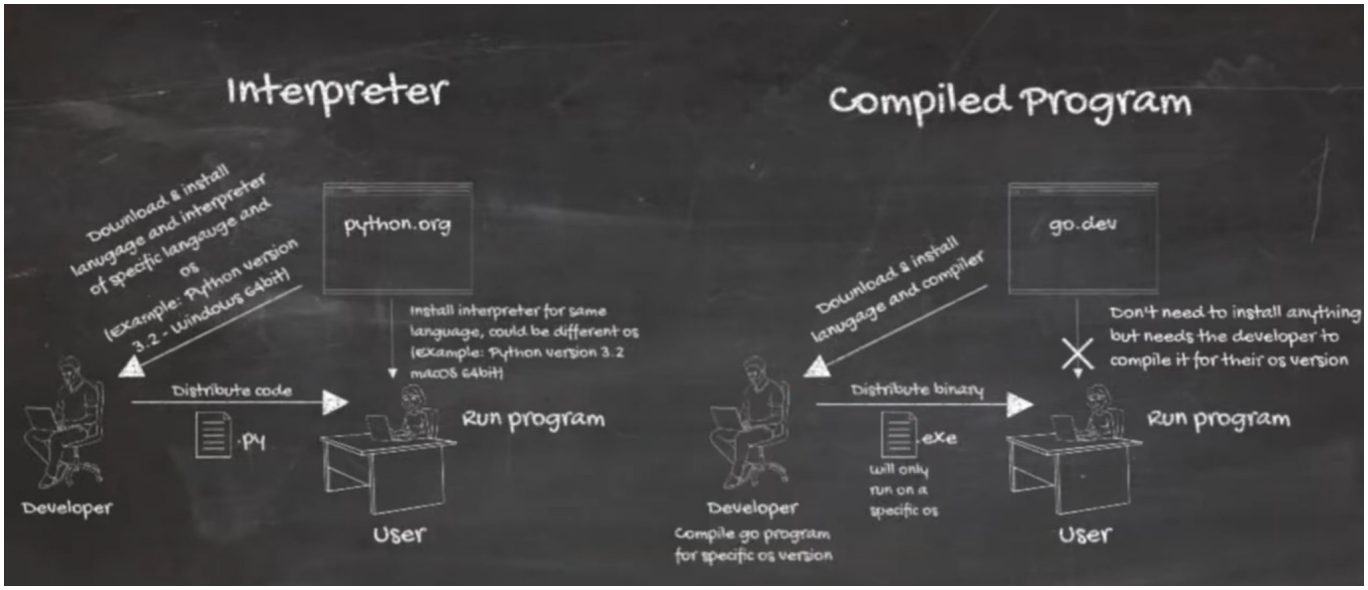
- Translates the entire code into machine language before execution, creating an independent executable file (like .exe on Windows).
- Errors must be fixed before the program can run.
- Compiled programs are typically faster because they are pre-translated.
- **Compiled Languages**: **C++**

Interpreter

- Translates and executes the code line-by-line, directly during runtime.
- If there is an error in one line, execution stops, and no further lines are processed.
- Typically slower than compiled languages because translation happens during execution.
- **Interpreted Languages: Javascript**

If you want to know more you can watch the following video.

Distribution of the program



	Compiled	Interpreted
Distribution	• Binary Code	• Source Code
Requirements in destination machine	• None • Each compiled version works only in one platform	• Runs in every platform, as long as it has the right interpreter installed.

You can also create .exe files using python as you can see in the following video.

Python: Compiled or Interpreted?

The short answer is both

1. **Source Code Compilation**
   - When you execute a Python program, the interpreter first compiles the source code (**written in .py files**) into bytecode. This is an intermediate, platform-independent representation of your code.
     - **When does this happen?**
       This compilation occurs automatically the first time the code is executed. If the code is run multiple times and there are no changes, Python will reuse previously generated bytecode
       - By default bytecode is stored in RAM but it could be stored in .pyc files in the __pycache__ directory.
     - **What is bytecode?**
       Bytecode is a low-level set of instructions that the Python Virtual Machine (PVM) can understand and execute. It is not human-readable and is specific to Python.
2. **Execution of Bytecode**
   - Once the source code is compiled into bytecode:
   - The **Python Virtual Machine (PVM)** interprets the bytecode and executes it step by step. This involves performing the operations described by the bytecode instructions, such as arithmetic calculations, variable assignments, function calls, etc.

In summary:

- Bytecode is generated before execution during the compilation phase.
- It serves as an intermediate representation that the PVM uses to execute the program efficiently.

	Advantages	**Disadvantages**
Compiled	• Speed of execution	• Poor portability • Long time of compilation
Interpreted	• Slow execution • Better with bytecode	• Better portability • Faster compilation

Read the following text

When writing computer software, programmers use **high-level programming languages** to tell the computer, or any other programmable device, what is to be executed. These programming languages adopt a high number of words from our everyday communication: “if”, “then”, “else”, “while”, “end”, and “break” are some, to mention a few. The result of what is written by programmers is called **source code. However**, computers do not understand this type of high-level programming language. So, for the machine to run the code, it must first be translated into something computers can **comprehend**. This is called **binary code**, or machine code, which is a language composed of zeros and ones, and this is the only language computers can understand, **so** to convert source code to binary code, a compiler or an interpreter is used.

A **compiler** will first read the entire source code and analyze it. This process may take some time, **although** once the code is analyzed and compiled, the execution is a lot faster.

If there is an error in the code, the compiler will output a message with information **regarding** the bug. If there are no **mistakes**, the result will be object code, also referred to as binary code, which can be executed **right away** by the machine for which it was generated. Because executable files don’t work on every **platform**, they must be generated for each one. Examples of programming languages that use compilers are C and C++.

One of the most important advantages of compilers is that the source code is no longer needed after the executable file is built. This protects the source code from unauthorized developers or hackers, who may wish to **tamper with** and **misuse** it. The **downside** is that if any changes need to be made, the programmer necessarily needs the source files.

An **interpreter** does things differently - it will not convert the source code into binary code all **at once**. **Instead**, it translates each **statement** one by one as the program is being executed. If an error is encountered, the code’s execution will stop abruptly. This can be considered a benefit, since the bug can easily be found. Interpreters are used by programming languages like Ruby and Python, for example.

Fortunately, you won’t have to decide which to implement. Depending on the programming language you choose to code in, the decision has already been made for you, since most languages have by now adopted one or the other. **However**, if you wish to keep **your skills sharp**, you should try programming in different languages: one that makes use of a compiler, and another that uses an interpreter.

Exercises

1. Translate to English the following terms
    a) Código fuente
    b) Código Binario
    c) Lenguaje de programación interpretado
    d) Lenguaje de programación compilado
2. What's the meaning of the following terms related to the video?
    a) Split them into two groups
    b) There are some caveats to it
    c) A programming language itself is neither compiled nor interpreted.
    d) To translate code on the fly.
    e) Drawbacks
    f) A brand new project
    g) high-level programming languages
    h) source code
    i) comprehend
    j) distinguish
    k) regarding
    l) mistakes
    m) right away
    n) platform
    o) tamper with
    p) misuse
    q) downside
    r) at once
    s) statement
    t) your skills sharp

3. Answer the following questions about the text
    a) Source code must be translated because:
        1. it is unintelligible to most humans.
        2. only computers can understand it.
        3. Neither
    b) Binary code is used because:
        1. a compiler says it is necessary.
        2. it is the only language which can be distinguished by computers.
        3. Neither
    c) The execution is a lot faster:
        1. once the interpreter has analyzed the source code.
        2. once the compiler has finished analyzing and compiling the source code.
        3. Neither
    d) The interpreter is not exactly the same as the compiler because:
        1. during execution, the interpreter translates statements one by one.
        2. during execution, the interpreter finds and solves errors without halting the process.
        3. Neither
    e) One benefit of using interpreters is:
        1. machine code is always easily stored.
        2. the source code does not need to be shared.
        3. Neither

4. Answer the following questions.

a) Which of the following languages is traditionally considered a compiled language?

a.1. Python

a.2. JavaScript

a.3. C++

a.4. Ruby

b) In the context of programming languages, what is the primary role of an interpreter?

b.1. To convert high-level code into machine code all at once before execution

b.2. To execute high-level code line by line at runtime

b.3. To manage memory allocation for programs

b.4. To optimize code for performance

c) What is a key advantage of compiled languages over interpreted languages?

c.1. Easier debugging

c.2. Platform independence

c.3. Faster execution speed

c.4. Dynamic typing

d) Which of the following languages is primarily interpreted?

d.1. C#

d.2. JavaScript

d.3. Rust

d.4. Go

e) In programming, what does the term "bytecode" refer to?

e.1. The binary code executed directly by the hardware

e.2. A high-level programming language

e.3. An intermediate code executed by a virtual machine

e.4. A markup language for web development

5. **Vocabulary.** Select the correct answer option that means the same as each word or phrase.

a) High-level programming:

a.1. An interpreted high-level general-purpose programming language.

a.2. The product of a compiler. A sequence of statements or instructions in a computer language.

a.3. A system using the digits 0 and 1 to represent a letter, digit, or other character in a computer or other electronic device.

a.4. A dynamic, open source programming language with a focus on simplicity and productivity.

a.5. A language that enables development of a program in a more user-friendly programming context and is generally independent of the computer's hardware architecture.

b) Binary code:

b.1. A dynamic, open source programming language with a focus on simplicity and productivity.

b.2. A system using the digits 0 and 1 to represent a letter, digit, or other character in a computer or other electronic device.

b.3. Interfere with something in order to cause damage or make unauthorized alterations.

b.4. The product of a compiler. A sequence of statements or instructions in a computer language.

c) Object code:

c.1. A dynamic, open source programming language with a focus on simplicity and productivity.

c.2. A system using the digits 0 and 1 to represent a letter, digit, or other character in a computer or other electronic device.

c.3. Interfere with something in order to cause damage or make unauthorized alterations.

c.4. The product of a compiler. A sequence of statements or instructions in a computer language.

d) tamper:

d.1. A dynamic, open source programming language with a focus on simplicity and productivity.

d.2. A system using the digits 0 and 1 to represent a letter, digit, or other character in a computer or other electronic device.

d.3. Interfere with something in order to cause damage or make unauthorized alterations.

d.4. The product of a compiler. A sequence of statements or instructions in a computer language.

e) Python:

e.1. An interpreted high-level general-purpose programming language.

e.2. A system using the digits 0 and 1 to represent a letter, digit, or other character in a computer or other electronic device.

e.3. A language that enables development of a program in a more user-friendly programming context and is generally independent of the computer's hardware architecture.

Connectors

Clauses of contrast

- **However** (but)
  - However is normally used at the beginning of a sentence, before a comma (,) and after a full stop (.) or a semicolon (;).
    - We didn't like the hotel. **However**, we had a good time.
    - I would like to have a dog; **however**, my husband is allergic to dogs.
- **Although** ('despite the fact that' or 'but')
  - Although can be used at the beginning or in the middle of a sentence. We do NOT use a comma after although; we use although + subject + verb.
    - **Although** he had a bad leg, he still won the game.
    - I passed the exam, **although** I hadn't studied.

Clauses of result

- **so**
  - So is the most common connector to express result. It is normally used in the middle of a sentence after (,).
    - We worked hard all morning, **so** I am very tired now.
    - The TV is very expensive, **so** I don't think I'll buy it.

Instead

- Instead is an adverb meaning 'alternatively' or 'as an alternative.' We can use it at the beginning or at the end of a clause, though it is normally used at the end.
  - As we had run out of coffee, I had a cup of tea **instead**.
  - We didn't go to the beach. We went to the waterpark **instead**.
- When used at the beginning of a sentence, instead must be followed by a comma.
  - I don't drink coffee. **Instead**, I drink tea.
  - I didn't have a sandwich for lunch. **Instead**, I had soup.

Exercise:

6. Choose the right connector in each sentence.
    a) We're not going abroad. We're going camping _____
    b) We couldn't find a taxi, _____ we walked home
    c) _____it was very cold, she wasn't wearing a coat.
    d) We need to buy a new car. _____, we can't afford it right now.

Programming related Vocabulary

- Scope
- Nested Loops
- iteration
- IDE
- Snippet
- Library
- Compiler
- Interpreter

Exercise:

7. Match the words from above with the following definitions
    a) is a loop inside another loop.
    b) Translates source code into machine code.
    c) small piece of reusable code that performs a specific task. Snippets are often used to save time and avoid repetitive coding.
    d) The process of repeating a set of instructions in a loop until a specific condition is met
    e) Executes code line by line
    f) A collection of pre-written code
    g) Refers to the visibility and accessibility of variables within a program. It determines where a variable can be used
    h) A software application that provides tools for coding, debugging, and testing programs