The Way of the Program

CSC 1200 - Principles of Computing

Overview

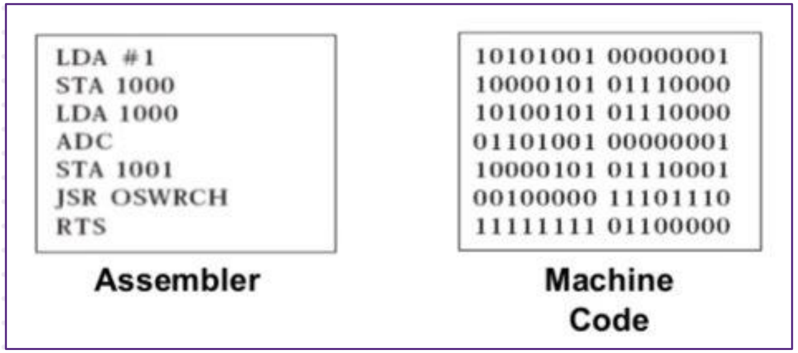
- Programming Languages
    - High-level vs. low-level
- Interpreters & Compilers
- Programs
- Errors & Debugging
- Formal and Natural Languages

Programming Languages

- The programming language we will use in this course is Python.

- Python is an example of a **high-level programming language**.

- Low-level programming languages provide little or no abstraction from a computer's architecture -- commands or functions in the language map closely to processor instructions.



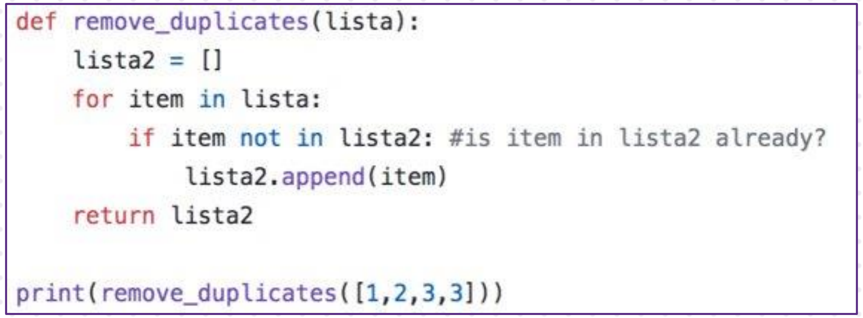
```
LDA #1
STA 1000
LDA 1000
ADC
STA 1001
JSR OSWRCH
RTS
```

Assembler

```
10101001 00000001
10000101 01110000
10100101 01110000
01101001 00000001
10000101 01110001
00100000 11101110
11111111 01100000
```

Machine Code

- High-level programming languages are independent from a particular type of computer -- use English words and mathematical symbols that make it closer to human language and further from machine language.



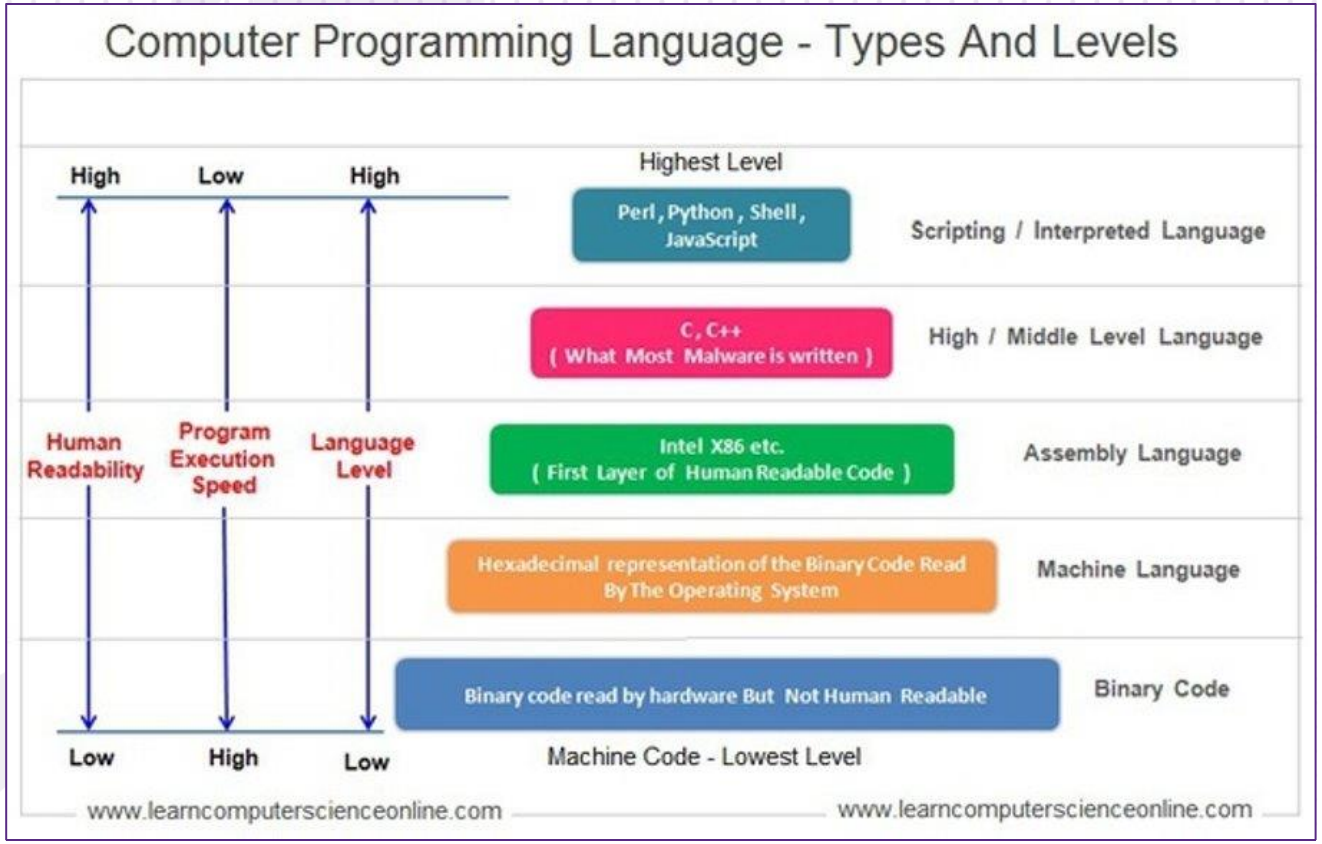
```
def remove_duplicates(lista):
    lista2 = []
    for item in lista:
        if item not in lista2: #is item in lista2 already?
            lista2.append(item)
    return lista2

print(remove_duplicates([1,2,3,3]))
```

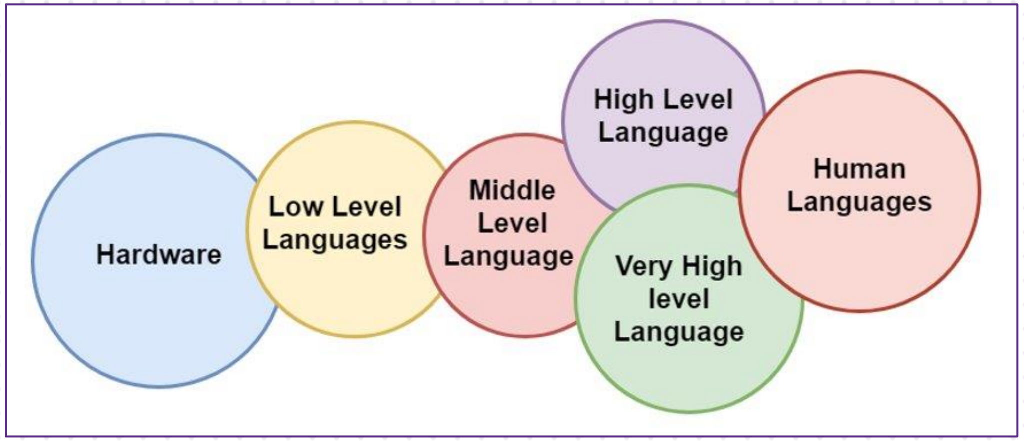
Advantages of High-Level Languages

- It is much **easier to program** in a high-level language.
  - Takes less time to write.
  - Shorter and easier to read (modify, maintain).
  - More likely to be correct.

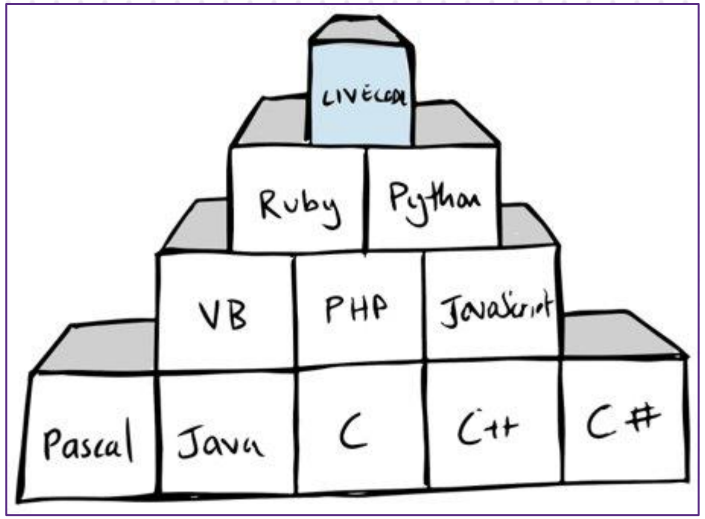
- Programs are **portable** (can run on different kinds of computers)

A Spectrum of Languages


Computer Programming Language - Types And Levels
High
Low
High
Highest Level
Perl, Python, Shell, JavaScript
Scripting / Interpreted Language
C, C++ (What Most Malware is written)
High / Middle Level Language
Human Readability
Program Execution Speed
Language Level
Intel X86 etc. (First Layer of Human Readable Code)
Assembly Language
Hexadecimal representation of the Binary Code Read By The Operating System
Machine Language
Binary code read by hardware But Not Human Readable
Binary Code
Low
High
Low
Machine Code - Lowest Level
www.learncomputerscienceonline.com
www.learncomputerscienceonline.com

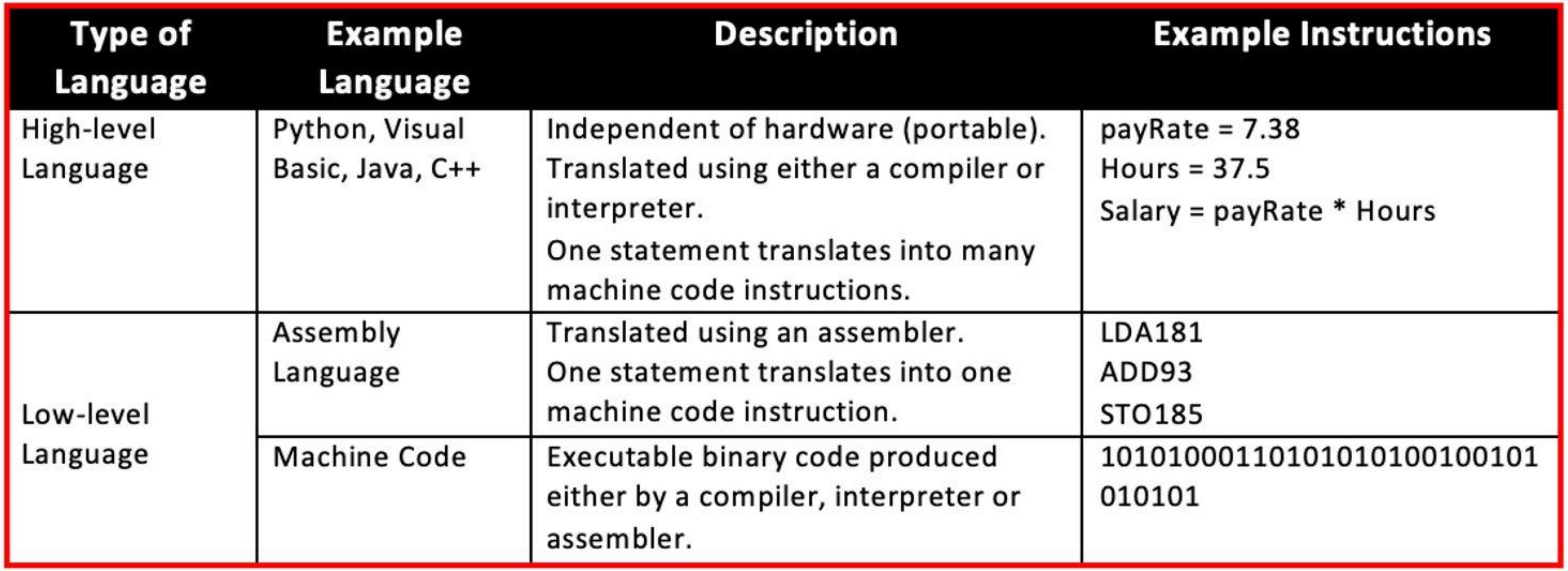


https://www.tutorialandexample.com/middle-level-language



https://livecode.com/core-benefits-of-livecode/

High-Level and Low-Level Language Summary



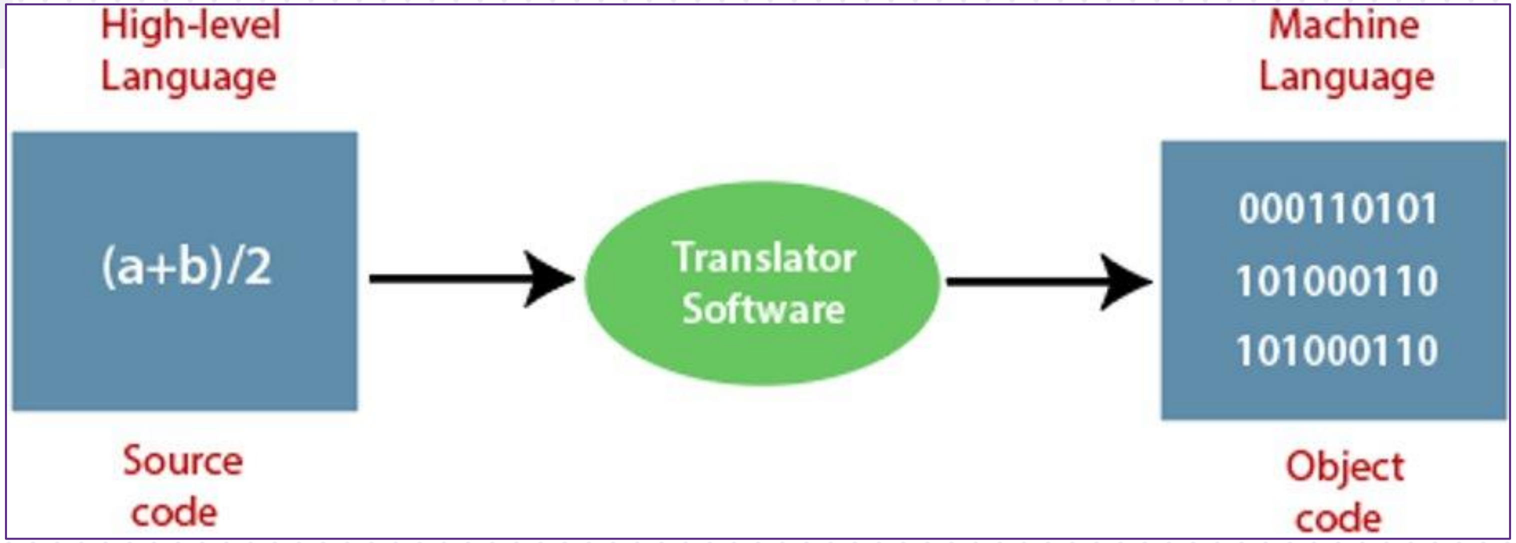
Type of Language	Example Language	Description	Example Instructions

High-level Language	Python, Visual Basic, Java, C++	Independent of hardware (portable). Translated using either a compiler or interpreter.
		One statement translates into many machine code instructions.
			payRate = 7.38
			Hours = 37.5
			Salary = payRate * Hours

Low-level Language	Assembly Language	Translated using an assembler. One statement translates into one machine code instruction.	LDA181 ADD93 STO185
	Machine Code	Executable binary code produced either by a compiler, interpreter or assembler.	10101000110101010100100101 010101

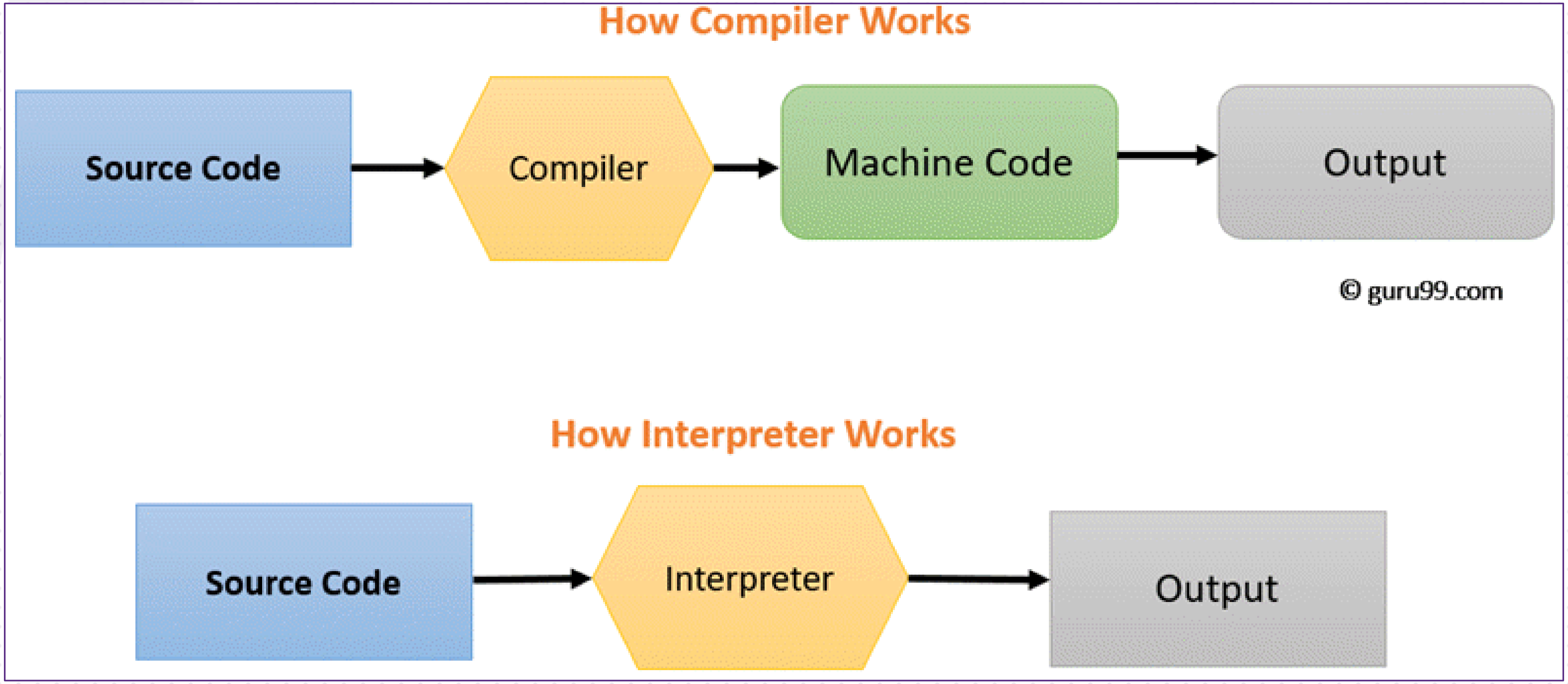
https://bournetocode.com/projects/GCSE_Computing_Fundamentals/pages/3-2-9-class_prog_langs.html

Translating From High-Level to Low-Level

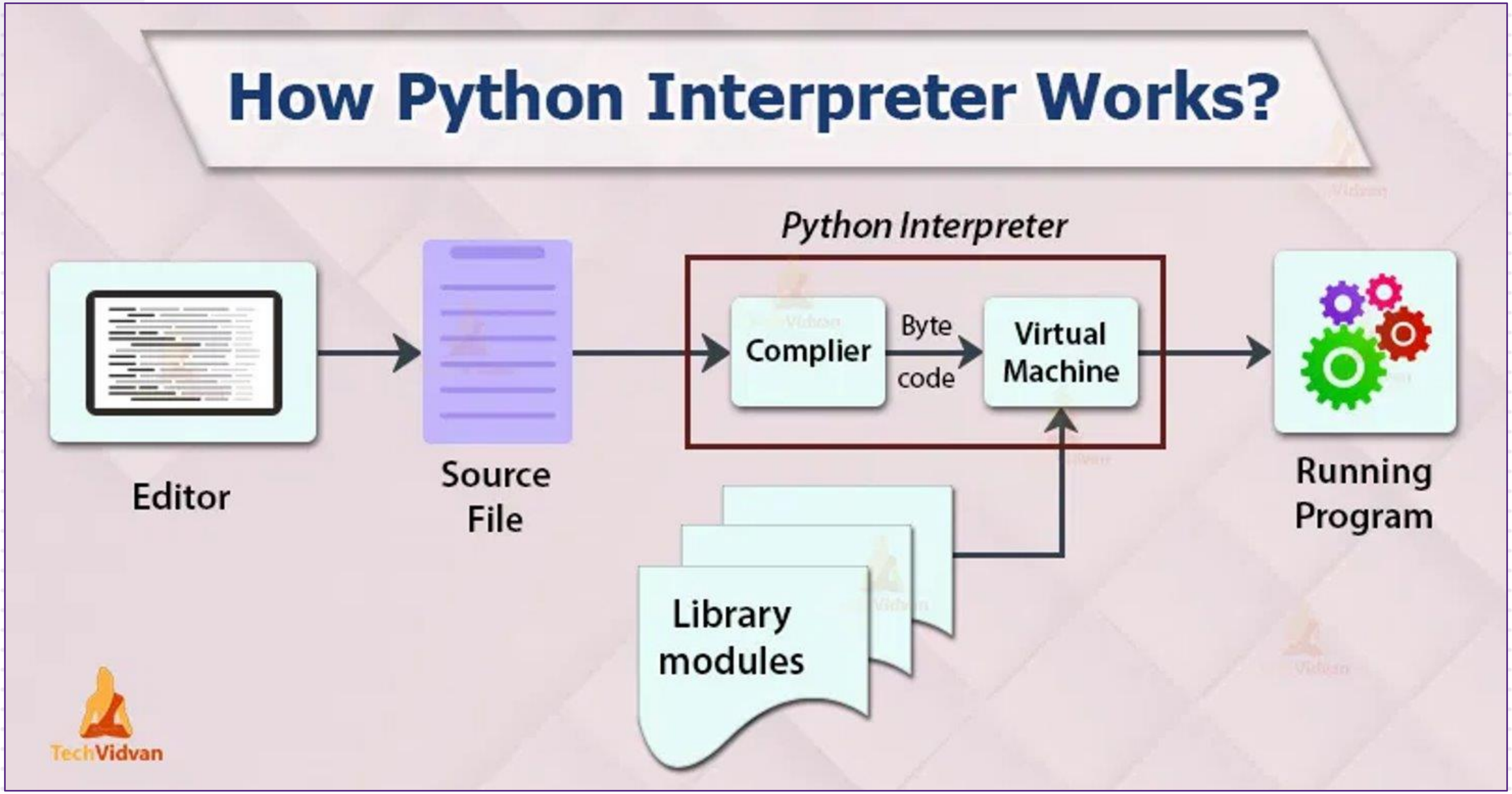


- Two kinds of programs process high-level languages into low-level languages:
  - **Interpreters:** reads and executes the program (a line at a time)
  - **Compilers:** reads and translates the program completely before execution.

Interpreters vs. Compilers



The Python Interpreter



https://techvidvan.com/tutorials/python-interpreter/

Python

Python is considered an interpreted language because Python programs are executed by an interpreter.

There are two ways to use the interpreter:

- **Interactive Mode**
  - A command line shell gives immediate feedback for each statement entered.
  - All previously fed statements and their results are kept in active memory.
  - As new lines are fed into the interpreter, the fed “program” is evaluated in the context of previous statements.
  - When the shell is closed the “program” disappears (can’t “save” it).
  - Interactive mode is a good way to play around and try variations on Python statements.
- **Script Mode**
  - You have to create a file and give it a name with a .py extension
  - The file must contain all the statements of the program.
  - The interpreter will read the statements from the .py file and execute them.
  - Script mode must be used for any programs that you want to save.

What Is A Program?

- **Program:** a sequence of instructions that specifies how to perform a task.
- Instructions look different, depending on the language, but a few types/categories of instructions are common to all languages.
- Types of instructions:
  - **Input**: Get data from the keyboard, a file, or some other device
  - **Output**: Display data on the screen or send data to a file or other device
  - **Math**: Perform mathematical operations
  - **Conditional Execution**: Check for certain conditions to make decisions about appropriate actions
  - **Repetition**: Perform some action repeatedly
- Programming is the process of breaking a large complex task into smaller and smaller subtasks until the subtasks are simple enough to be performed with these basic instructions.

Types of Errors

When writing computer programs, errors are inevitable.

Types of errors:

- **Syntax Errors:** computer programs must follow strict syntax (well-defined structure) to compile correctly; any aspects of the code that do not conform to the syntax of the programming language will produce a syntax error; these errors are detected when the program is compiled.
- **Runtime Errors:** an error that occurs when the program is running; also called exceptions; these cannot be detected by the compiler.
- **Semantic Errors:** logic error; the program will run, but it does not do what you want it to do.

Example: calculator

1 // 2

2/0

6!/4!2!

Debugging

Debugging is the process of finding and correcting errors in a computer program.

- The compilation process will detect syntax errors -- the programmer must correct the syntax
- Rigorous testing with expected and Unexpected input can reveal runtime errors. They can be difficult to detect, but once found are not too hard to correct.
- Semantic errors are usually the most difficult to correct.

Formal and Natural Languages

- **Natural Languages:** languages that people speak; not designed by people
- **Formal Languages:** languages designed by people for a specific purpose; e.g. mathematicians use a formal language to denote relationships among numbers ($(x+a)^n = \sum_{k=0}^{n} \binom{n}{k} x^k a^{n-k}$) and chemists use a formal language to represent chemical structure of molecules and reactions (2C2H6 + 7O2 → 4CO2 + 6H2O)
- Formal languages have strict **syntax** (rules to be followed to create well-formed statements).
- Syntax rules come in two flavors:
  - **Tokens:** the basic elements of the language (words, punctuation, numbers, operations, chemical elements, etc.)
  - **Structure:** how tokens are correctly arranged

Why Don’t We Program in Natural Languages

- Syntax - what a statement looks like
- Semantics - what a statement means
- Natural languages:
  - Ambiguous
  - Employ redundancy to make up for ambiguity; often verbose
  - Full of idioms and metaphors
- Formal languages:
  - Not ambiguous
  - Concise
  - Literal (always mean exactly what they say)