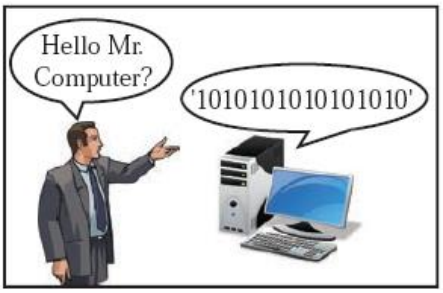
Compares and Contrasts Different Programming Paradigms

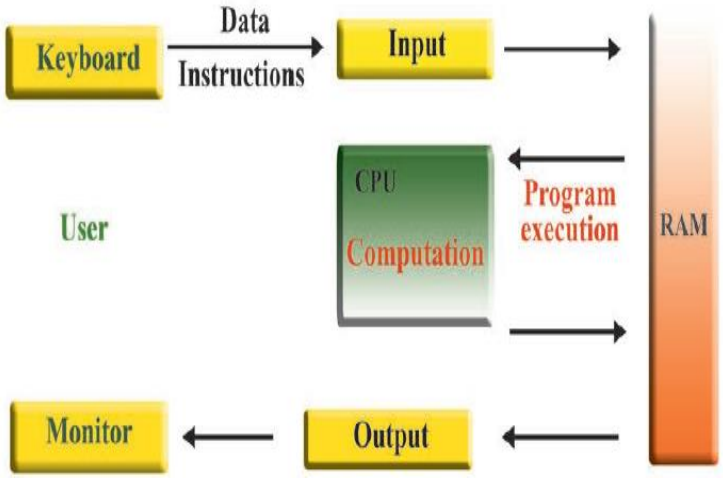
- Evolutions of Programming Languages
- Programming Paradigms
    - Imperative Languages
    - Declarative Languages
    - Object Oriented Languages

Evolutions of Programming Languages

Need of a Programming Language

- A program is a sequence of instruction which performs a certain task using the computer.
- A computer language is needed to provide the instructions





A Programming Language

- A computer needs to be given instructions in a programming language that it understands.
- A programming language is an artificial language that can be used to control the behavior of computer.
- Programming languages, like human languages, are defined through the use of syntactic and semantic rules, to determine structure and meaning respectively. The programming language has Syntax and language elements have Semantics.
- Programming languages are used to facilitate communication about the task of organizing and manipulating information, and to express algorithms precisely.

How it starts?

- In 1822, the first version of a programming language arose from work by Ada Lovelace, the benefactor and business partner of Charles Babbage.
- Unfortunately, her work went mostly ignored since Babbage never built a completed Analytical Engine so there was no public deployment.
- She is remembered in the programming language still in use on military-grade projects, Ada.

What is a Programing?

- **Programming is a Science**
  Because it implements the algorithm describe by mathematics and science.
- **Programming is a Skill**
  Because it requires design efforts
- **Programming is an Engineering**
  Because it requires tradeoffs between program size, speed, time (required for development and debugging) and maintainability among many solutions.
- **Programming is an Art**
  It requires creativity and employ imagination

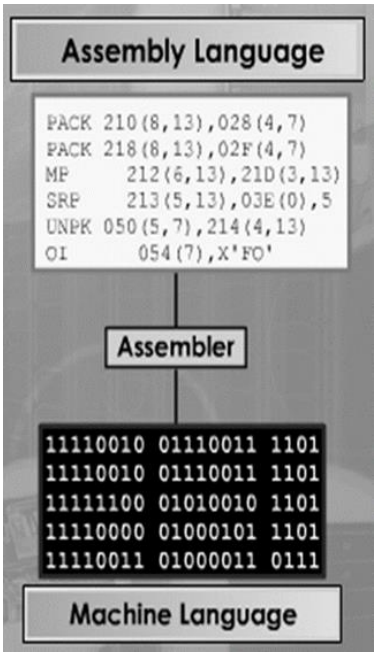
programming Languages mainly categorizes as :

Low Level Programming Languages

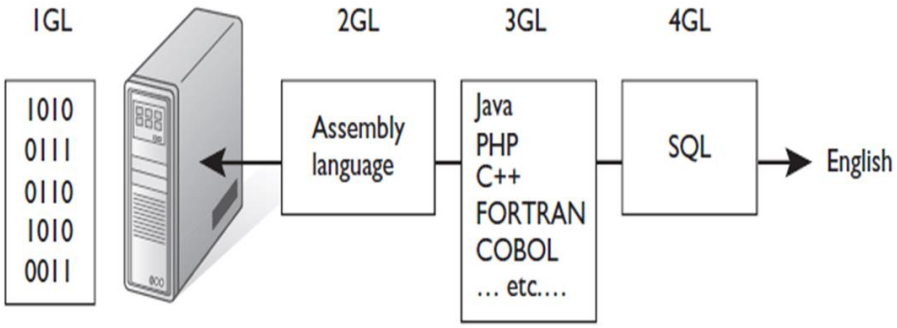
Machine Language
Assembly Language

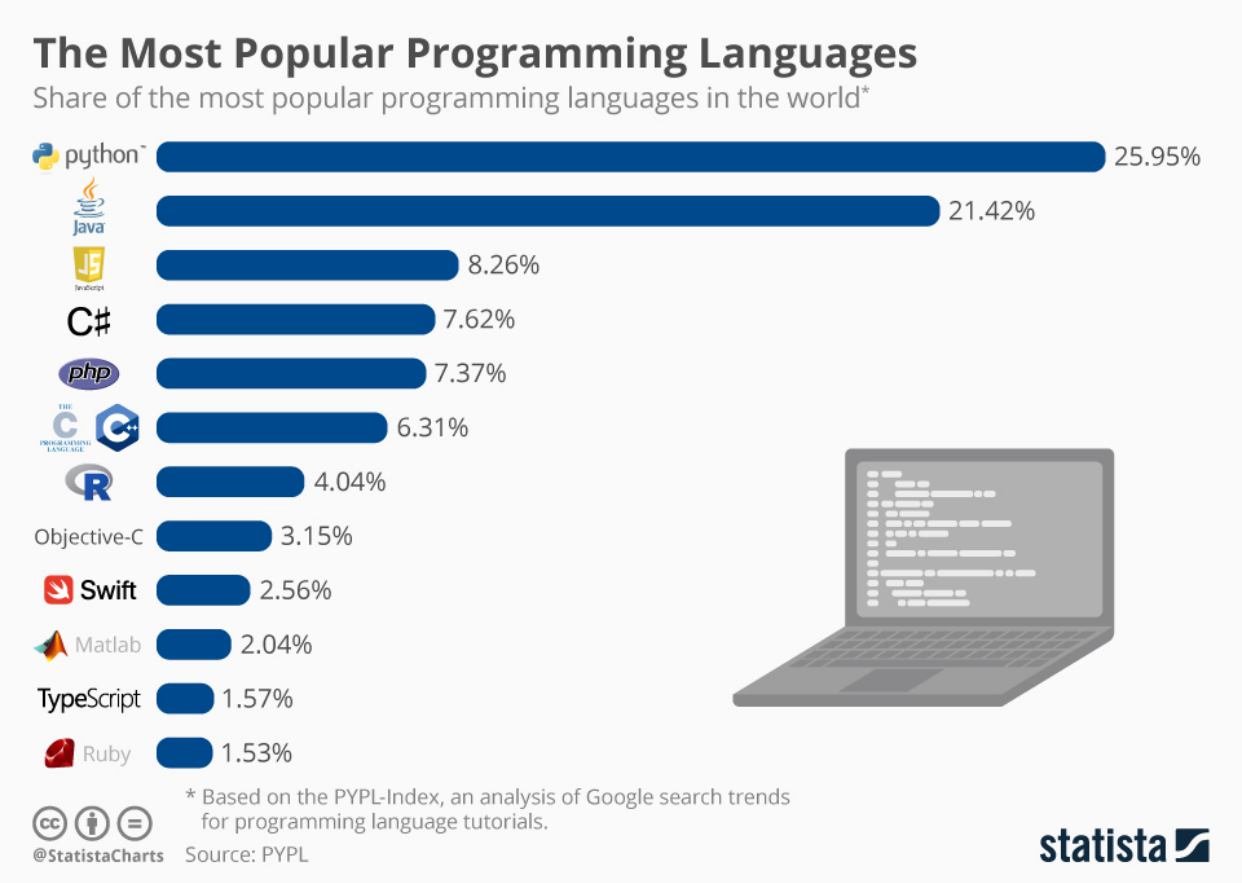
High Level Programming Languages

FORTRAN
BASIC
COBOL
PASCAL
PYTHON
C etc…



Levels Generations of Programming Languages





Here is a table that shows the dispersion of Programming Languages across various popular sites that we are very much familiar with.

Websites	Popularity (unique visitors per month)[1]	Front-end (Client-side)	Back-end (Server-side)	Database	Notes
Google.com[2]	1,600,000,000	JavaScript	C, C++, Go,[3] Java, Python	BigTable,[4] MariaDB[5]	The most used search engine in the world
Facebook.com	1,100,000,000	JavaScript	Hack, PHP (HHVM), Python, C++, Java, Erlang, D,[6] Xhp,[7] Haskell[8]	MariaDB, MySQL,[9] HBase Cassandra[10]	The most visited social networking site
YouTube.com	1,100,000,000	JavaScript	C, C++, Python, Java,[11] Go[12]	BigTable, MariaDB[5][13]	The most visited video sharing site
Yahoo	750,000,000	JavaScript	PHP	MySQL, PostgreSQL[14]	Yahoo is presently[when?] transitioning to Node.js[15]
Amazon.com	500,000,000	JavaScript	Java, C++, Perl[16]	Oracle Database[17]	Popular internet shopping site
Wikipedia.org	475,000,000	JavaScript	PHP, Hack	MySQL[citation needed], MariaDB[18]	"MediaWiki" is programmed in PHP, runs on HHVM; free online encyclopedia
Twitter.com	290,000,000	JavaScript	C++, Java, Scala, Ruby on Rails[19]	MySQL[20]	140 characters social network
Bing	285,000,000	JavaScript	ASP.NET	Microsoft SQL Server	
eBay.com	285,000,000	JavaScript	Java,[21] JavaScript,[22] Scala[23]	Oracle Database	Online auction house
MSN.com	280,000,000	JavaScript	ASP.NET	Microsoft SQL Server	An email client, for simple use. Mostly known as "messenger".
Microsoft	270,000,000	JavaScript	ASP.NET	Microsoft SQL Server	Software company
Linkedin.com	260,000,000	JavaScript	Java, JavaScript,[24] Scala	Voldemort[25]	World's largest professional network
Pinterest	250,000,000	JavaScript	Django (a Python framework),[26] Erlang	MySQL, Redis [27]	
WordPress.com	240,000,000	JavaScript	PHP, JavaScript [28] (Node.js)	MariaDB, MySql	

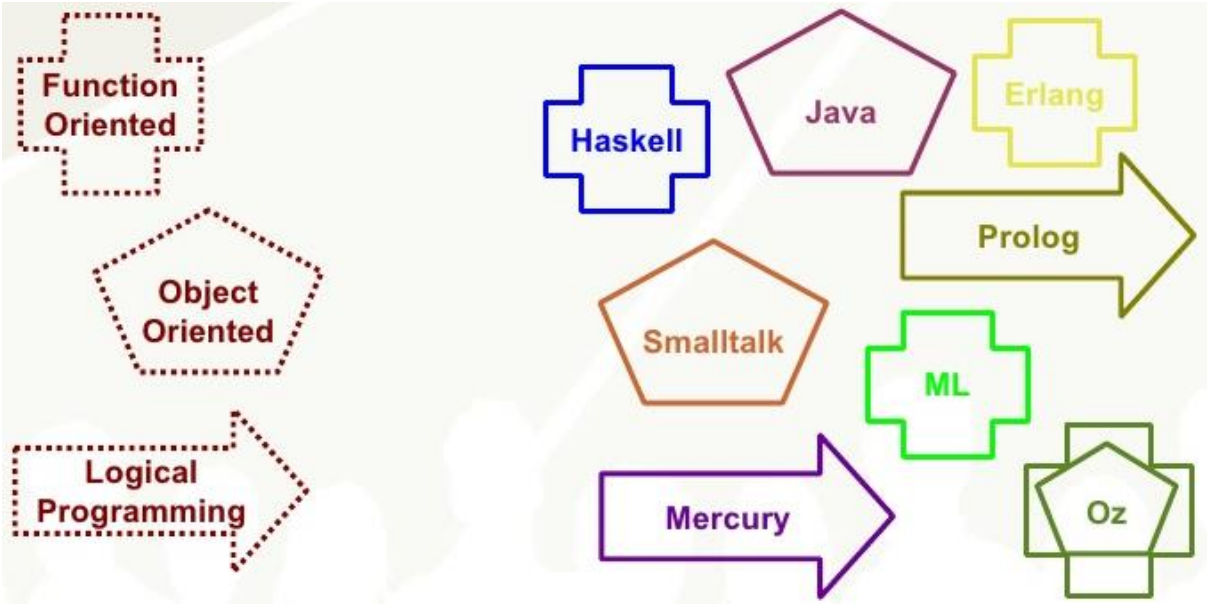
Programming Paradigms

Is there any Best Programming Language?

- Programming Language is probably used most efficient if it is well suited for a specific task.
- For example
  - Business applications are often written in *COBOL.*
  - Beginners to programming use *BASIC.*
  - Scientific programming is often undertaken with either *FORTON, PASCAL or C.*

What are Programming Paradigms?

- Programming languages can be categorized into programming paradigms.
- Programming paradigms are the result of people's ideas about how computer programs should be constructed.
- Programming Paradigm is a fundamental style of computer programming and it serves as a pattern or model for a programming language



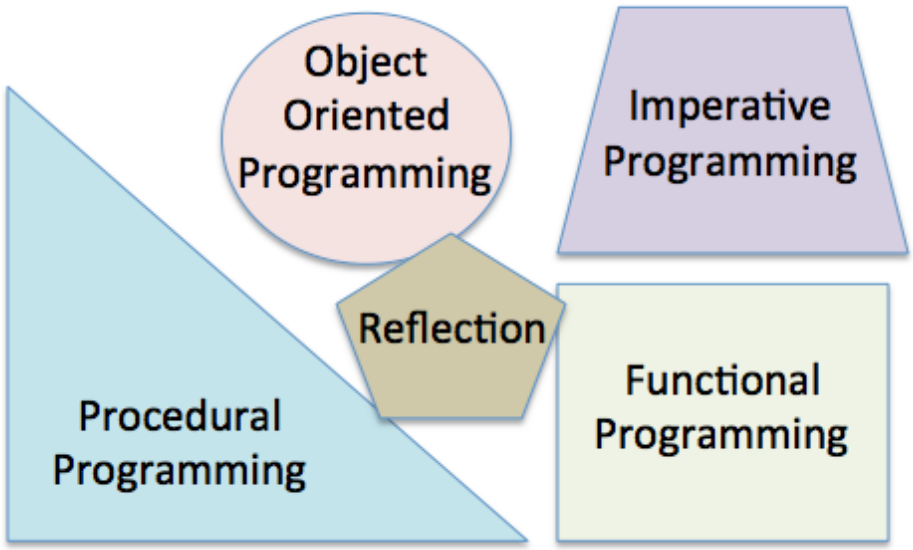
A number of programing paradigms are:

- **Procedural / Imperative Languages**
- **Declarative Languages**
- **Object Oriented Languages**

Different programming languages belongs to different programming paradigms.

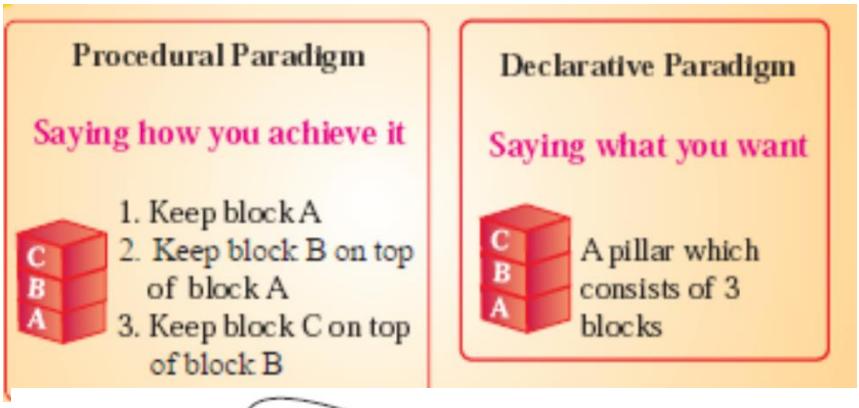
Imperative/ Algorithmic	Declarative		Object-Oriented
	Functional Programming	Logic Programming	
Algol Cobol PL/1 Ada C Modula - 3	Lisp Haskell ML Miranda APL	Prolog	Smalltalk Simula C++ Java

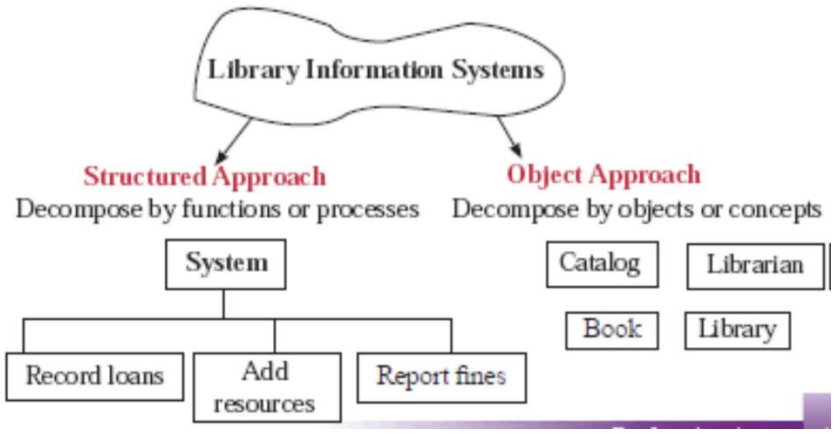
Programming paradigms supported by Python

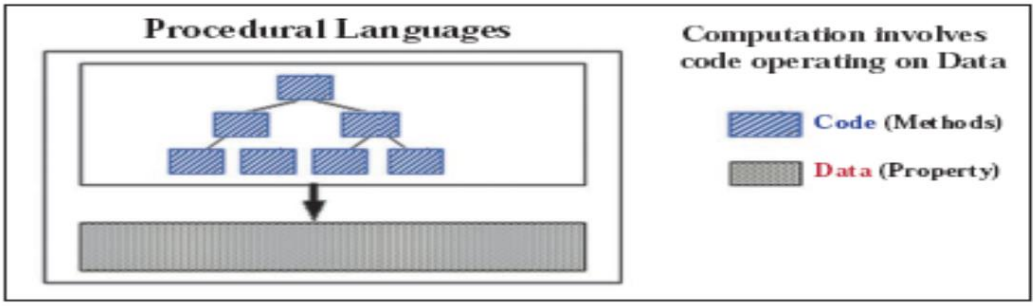


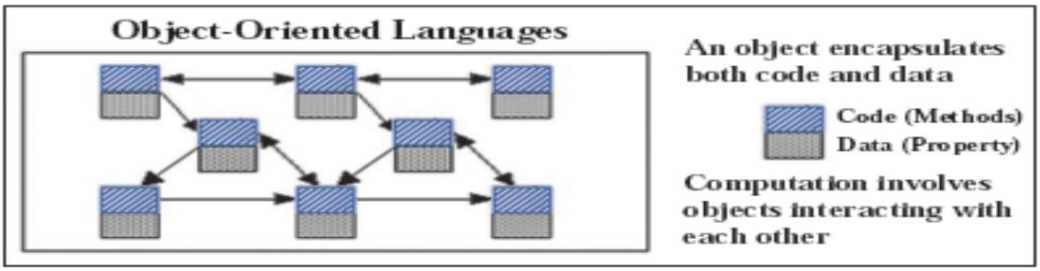
Why we bother about Programming Paradigms?

Bridge Example		
Requirement	**Tools one knows**	**Better approach**
The bridge should be - Earthquake resistant at x level - Should be erosion resistant - So on...	Pros and cons of building it with either wood or iron	One knowing pros & cons of wood, iron (& possibly concrete) to satisfy the requirements.






Procedural Languages
Computation involves code operating on Data
Code (Methods)
Data (Property)


Object-Oriented Languages
An object encapsulates both code and data
Code (Methods)
Data (Property)
Computation involves objects interacting with each other

The Need of Program Translation and The Type of Program Translators

- Need of Program Translation
- Source Program
- Object Program
- Program Translators
    - Interpreters
    - Compilers
    - Hybrid Approach
- Linkers

Need of Program Translation

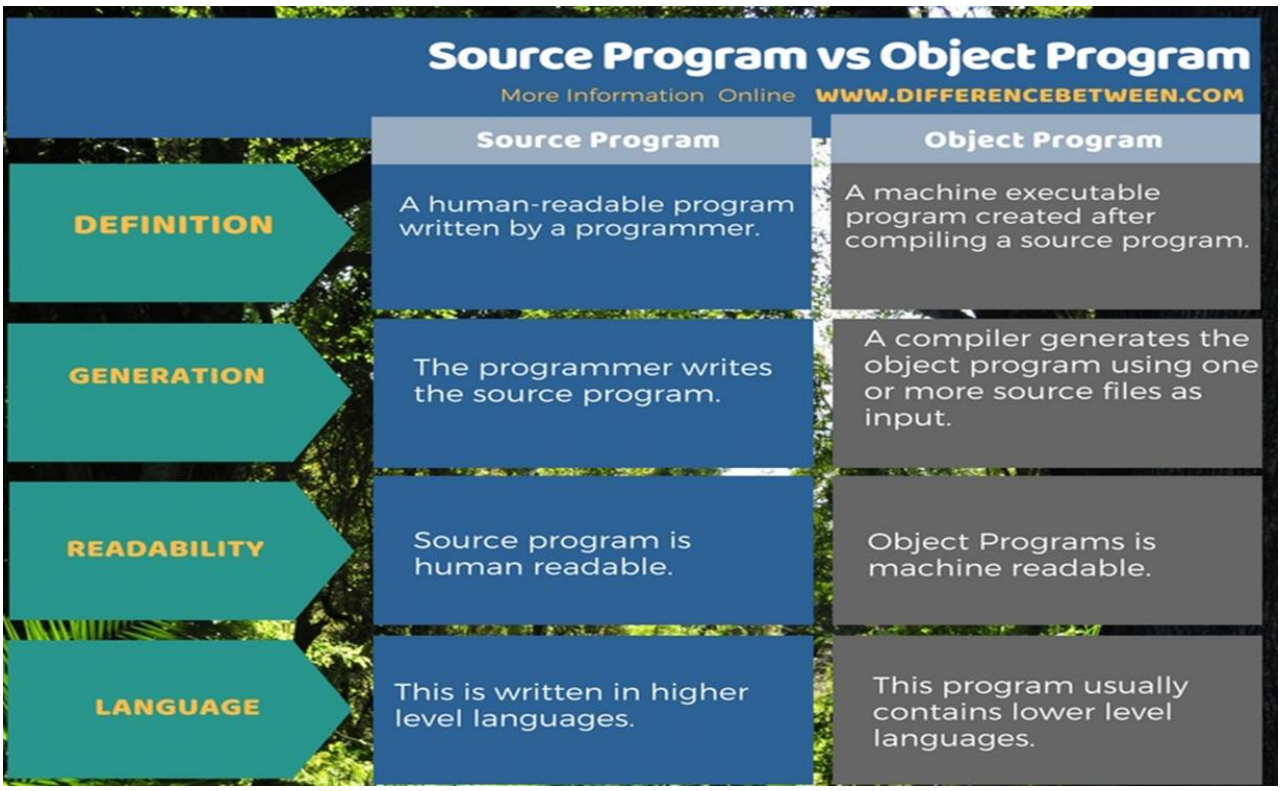
- Computers only understand machine code (binary), this is an issue because programmers prefer to use a variety of high and low-level programming languages instead.
- To get around the issue, the high-level and low-level program code (source code) needs to pass through a translator. A translator will convert the high-level and low-level program code (source code) into machine code (object code).

Source Program

- The programmer writes the source program using higher level language.
- Therefore, it is easily readable by the humans.
- Source programs usually contain meaningful variable names and helpful comments to make it more readable.
- A machine cannot directly execute a source program.
- A compiler helps to transform source program to executable code to execute by the machine.
- Alternatively, is to use an interpreter. It executes a source program line by line without pre-compilation.

Object Program

- Object program is usually a machine executable file, which is the result of compiling a source file using a compiler.
- Apart from machine instructions, they may include debugging information, symbols, stack information, relocation, and profiling information.
- Since they contain instructions in machine code, they are not easily readable by humans.



What is Program Translator?

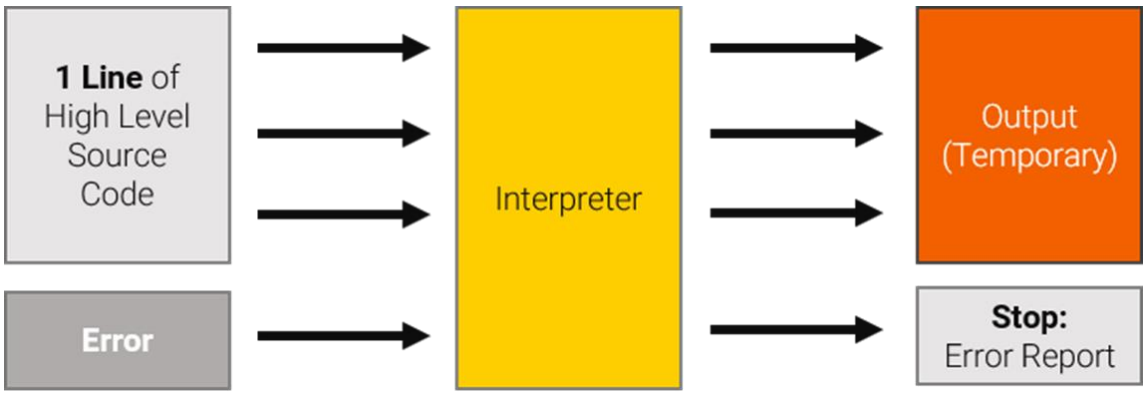
- A translator will convert the source code into machine code (object code).
- There are several types of translator programs, each able to perform different tasks.
  - Assembler
  - Interpreters
  - Compilers
  - Hybrid Approach

Assembler

- Assemblers are used to translate a program written in a low-level assembly language into a machine code (object code) file.
- so, it can be used and executed by the computer.
- Once assembled, the program file can be used again and again without re-assembly.

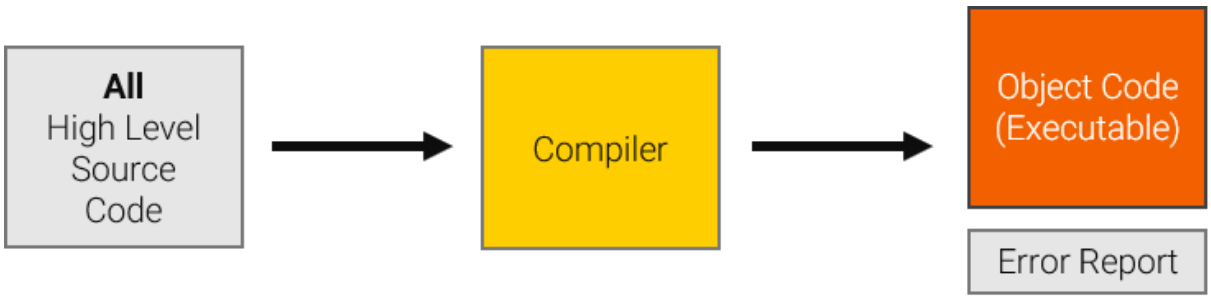
Interpreter

- Interpreter programs are able to read, translate and execute one statement at a time from a high-level language program.
- The interpreter stops when a line of code is reached that contains an error.
- Interpreters are often used during the development of a program. They make debugging easier as each line of code is analyzed and checked before execution.
- Interpreted programs will launch immediately, but your program may run slower than a complied file.
- No executable file is produced. The program is interpreted again from scratch every time you launch it.



Compiler

- Compilers are used to translate a program written in a high-level language into machine code (object code).
- Once compiled (all in one go), the translated program file can then be directly used by the computer and is independently executable.
- Compiling may take some time but the translated program can be used again and again without the need for recompilation.
- An error report is often produced after the full program has been translated.
- Errors in the program code may cause a computer to crash.
- These errors can only be fixed by changing the original source code and compiling the program again.



Hybrid Approach

- some interpreting, some compiling. Java is a well-known example. Java source is compiled to byte codes (.class files). For execution, byte codes are interpreted directly by the Java Virtual Machine (JVM). Any Java application will run on every machine for which a Java virtual machine has been implemented.

Here are some examples of translators per type:

Translator	Examples
Compiler	Microsoft Visual Studio GNU Compiler Collection (GCC) Common Business Oriented Language (COBOL)
Interpreter	OCaml List Processing (LISP) Python
Assembler	Fortran Assembly Program (FAP) Macro Assembly Program (MAP) Symbolic Optimal Assembly Program (SOAP)

Program Translators - Comparison

Compiler	Interpreter	Assembler
Translate high – level languages into machine code	Temporarily executes high – level languages, one statement at a time	Translate low-level assembly code into machine code
An executable file of machine code is produced (object code)	No executable file of machine code is produced (no object code)	An executable file of machine code is produced (object code)
Compiled program no longer need the compiler	Interpreted programs cannot be used without the interpreter	Assembled programs no longer need the assembler
Error report produced once entire program is compiled. These errors may cause your program to crash	Error message produced immediately (and program stops at that point)	One low-level language statement is usually translated into one machine code instruction
Compiling may be slow, but the resulting program code will run quick (directly on the processor)	Interpreted code is run through the interpreter (IDE), so it may be slow, e.g. to execute program loops	
One high-level language statement may be several lines of machine code when compiled		

Linker

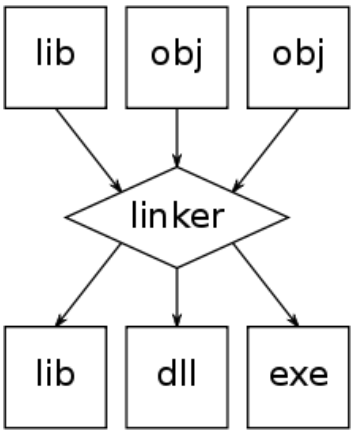
A **linker** is a computer program that takes one or more object files generated by a compiler and combines them into one, executable program.

- Computer programs are usually made up of multiple modules that span separate object files, each being a compiled computer program.
- The program as a whole refers to these separately compiled object files using symbols.

- The linker combines these separate files into a single, unified program; resolving the symbolic references as it goes along.

Functions of Linker

- For most compilers, each object file is the result of compiling one input source code file.
- When a program comprises multiple object files, the linker combines these files into a unified executable program.



Reference

- https://www.differencebetween.com/difference-between-source-program-and-vs-object-program/
- https://www.computerscience.gcse.guru/theory/translators
- https://teachcomputerscience.com/translators/
- https://www.computerhope.com/jargon/l/linker.htm