Computer Science Basics

Introduction to Computer Programming
Bachelor in Data Science

Roberto Guidi

roberto.guidi@supsi.ch

September 2021

SUPSI University of Applied Sciences and Arts of Southern Switzerland

Content created in collaboration with: L.Grossi, F.Landoni, G. Landolfi, T. Leidi, A.E. Rizzoli, S. Pedrazzini

What is Computer Science?

Computer science is the discipline of rational treatment of information, by means of machines or automatic procedures, considered as a support of knowledge and communications in technical, economical and social fields.

— *Académie française, 1966*

Informatics: the study of processes for storing and obtaining information.

— *Oxford Learner's Dictionaries, 2020*

Information is a set of facts which knowledge allows to take action or decision.

Automatic elaboration

The automatic electronic computers manage large amount of data exploiting a remarkable processing power.

The efficacy of the computer is mostly highlighted when it is needed to execute, a large number of times repetitive operations.

The computers are only blazing fast and precise executors of commands.

The task of solving the real problem is, at least for the moment, in the hands of men.

Automatic elaboration

The first tools created to help to run calculations exists for thausend of years. The working principle is that of hands fingers.

The abacus is a tool created probably in China in order to facilitate mathematical calculations. It date back to Babylon around 2400 b.C.

First computers

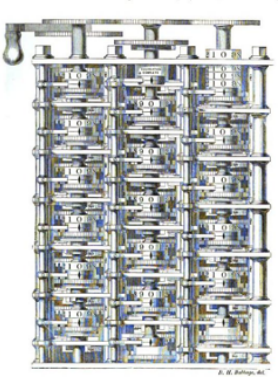
Many mechanical calculators were invented in ancient times and during the Middle Age to perform astronomical calculations.

Among them for example we can find the ancient Greek Anticitera machine (150 - 100 b.C)

First computers

The first programmable computers were invented only at the beginning of the 19th century and were able to arise, as we know them today, beginning from the 40s.

Charles Babbage, a British mechanical engineer is considered the father of the computer. He invented a mechanical device capable of executing calculations.



Some history

Most of the computer's development steps were kept secrets for long time, often because they were related to military projects. Nowadays it is possible to retrace history transparently.

Useful link: https://en.wikipedia.org/wiki/History_of_computing_hardware

Turing and Von Neumann

Alan Turing is considered the father of computer science and artificial intelligence. In the 30s he theorised the first real computer thanks to his formalisation of the concepts of algorithm and computational calculus by means of the Turing machine.

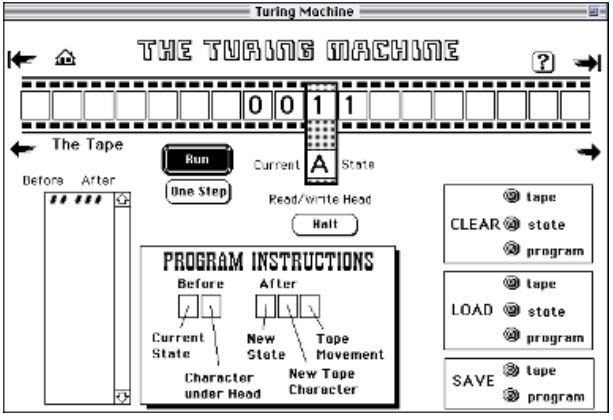
John Von Neumann was the American researcher that, urged by the American army during WWII, devoted his studies to the realisation of the first electronic calculators. In 1945 he developed a sequential architecture still in use today.

The Turing Machine

The principle of functioning of modern computers was introduced by Alan Turing, who described the idea in a 1932 publication.

Turing reformulated the results of Kurt's Gödel's 1931 study on the limits of computability, replacing the formal language used by Gödel with a hypothetical machine known as the Turing machine, capable of performing any mathematical operation that can be represented in the form of an algorithm.

The Turing Machine



A Turing machine is an ideal machine that manipulates data stored on a tape of infinite length, according to a predetermined set set of well-defined rules. In other words, it is an abstract model that defines a machine able of executing algorithms.

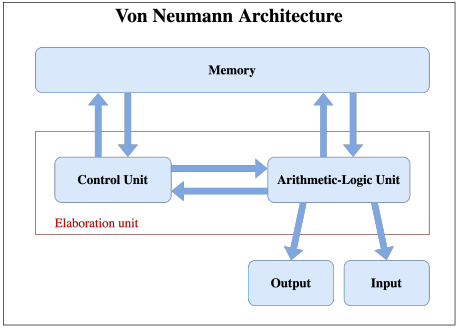
Digital Computers

The first digital computers were of the electromechanical type. Electric circuits piloted relays (switches) used to perform the calculations. These devices were very slow and were replaced by fully electric computers that used valves instead of relais.

The Von Neumann architecture

In 1945, Von Neumann developed a sequential architecture, still in use today, with the following features:

- the instructions are loaded into memory,
- the data are also put into memory,
- the instructions are executed one by one by the Central Processing Unit (CPU).



Computers with transistors

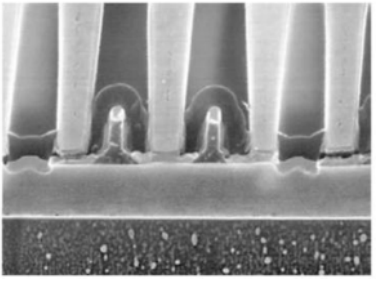
Transistors were invented in 1947. Starting in 1955 they replaced the tubes inside computers, giving rise to the second generation of computers.

Transistors are semiconductors capable of amplifying and switching electrical signals.

Compared to tubes, transistors are smaller, consume less power, dissipate less heat and fail less frequently.

Integrated circuitry

The next step in the development of computers came with the rise of integrated circuitry: complex electronic components capable of containing one or more miniaturised electronic circuit.



Two transistors out of the 780 milions present inside the IBM POWER6 processor

Microprocessors

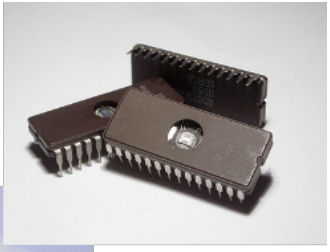
Starting from the 70s the evolution of computers has been characterised by the introduction of microprocessors.

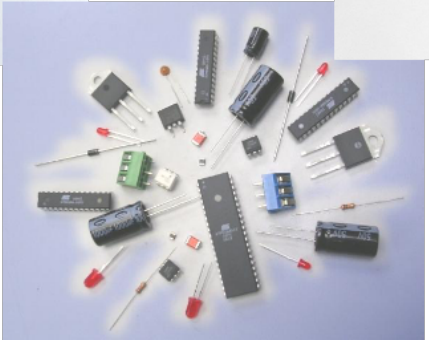
The microprocessors allowed the integration of functionalities of the CPU in a single integrated circuitry.

The programmable multifunctional microprocessors accepts digital data that, by means of instruction present in the memory, can be processed to produce results.

In the 70s and 80s, the microprocessors allowed the development of the first personal computers.

Integrated circuitry and microprocessors



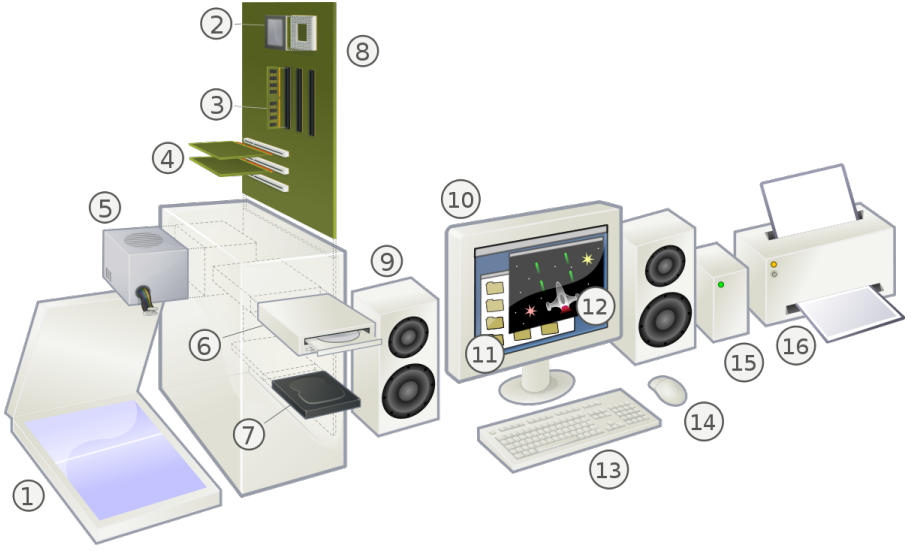


Computer

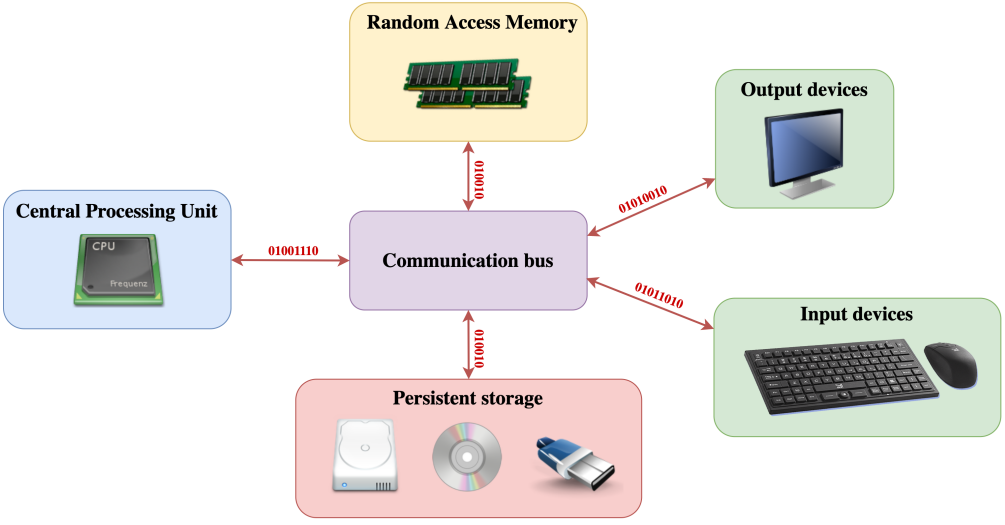
The computer is made up by two parts:

- Hardware: the physical part of the computer. Keyboard, mouse, monitor, cables, electric and electronic circuitry and mechanical parts are an example.
- Software: the set of all the programs and functions that are needed by the computer to manage the data and himself.

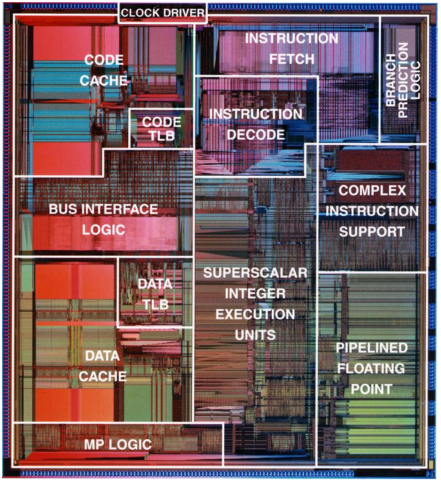
Hardware



Conceptual schema of a computer

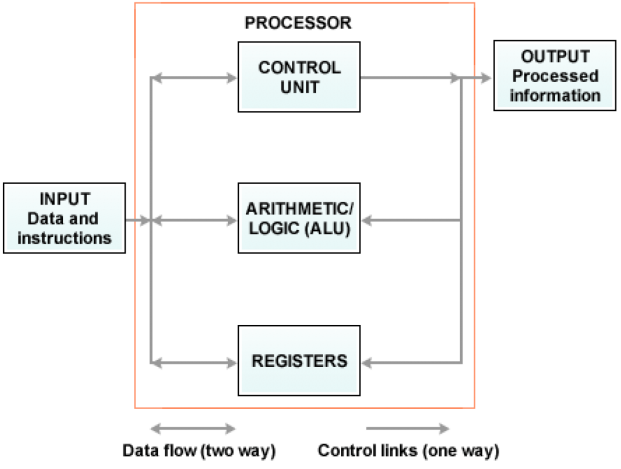


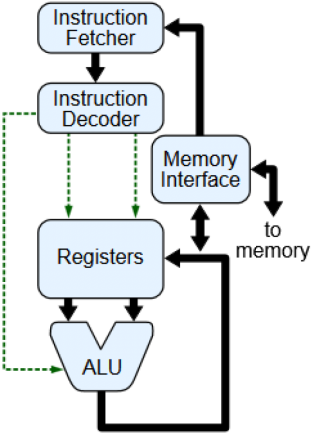
CPU



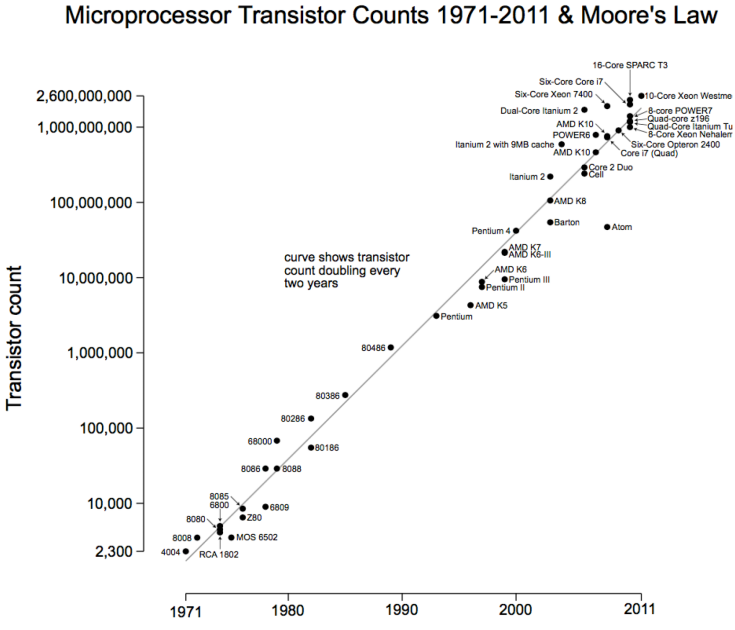
Intel Pentium

Conceptual schema of a CPU

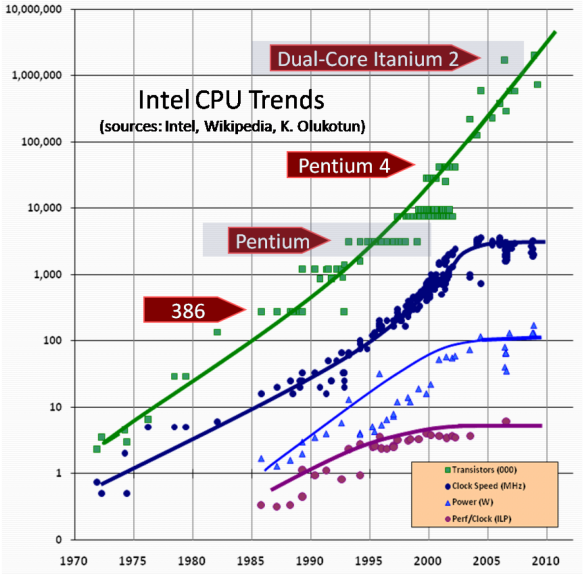


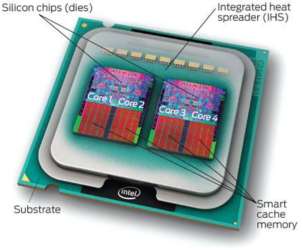


Moore's Law


Microprocessor Transistor Counts 1971-2011 & Moore's Law
Transistor count
2,600,000,000
1,000,000,000
100,000,000
10,000,000
1,000,000
100,000
10,000
2,300
1971
1980
1990
2000
2011
curve shows transistor count doubling every two years
16-Core SPARC T3
Six-Core Core i7
Six-Core Xeon 7400
10-Core Xeon Westme
Dual-Core Itanium 2
8-core POWER7
Quad-core z196
Quad-Core Itanium Tu
AMD K10
POWER6
8-Core Xeon Nehalem
Itanium 2 with 9MB cache
Six-Core Opteron 2400
AMD K10
Core i7 (Quad)
Core 2 Duo
Cell
Itanium 2
AMD K8
Barton
Atom
Pentium 4
AMD K7
AMD K6-III
AMD K6
Pentium III
Pentium II
AMD K5
Pentium
80486
80386
80286
68000
80186
8086
8088
8085
6800
6809
Z80
8080
MOS 6502
8008
4004
RCA 1802

Multi-core processors





Fonte grafico: "The Free Lunch Is Over" di Herb Sutter

Fastest computer on the planet

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442,010.0	537,212.0	29,899
2	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
3	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
4	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
5	**Perlmutter** - HPE Cray EX235n, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Slingshot-10, HPE DOE/SC/LBNL/NERSC United States	706,304	64,590.0	89,794.5	2,528
6	**Selene** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, Nvidia NVIDIA Corporation United States	555,520	63,460.0	79,215.0	2,646

— *www.top500.org*

Software

Software is divided in two main groups:

- System software: operating system, utilities, compilers and interpreters.
- Application software: user-oriented programs for word processing, graphics, multimedia, specific programs (accounting, mathematical models, . . .), games, . . .

Operating System

An Operating System (OS) is a software component that manages the hardware resources providing their features to the applications.

An OS is also in charge to host the applications while they execute.

The main tasks of an OS are:

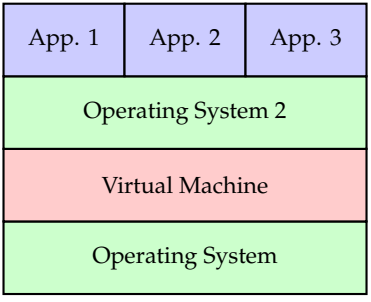
- Management of the execution of applications.
- Managements of the hardware interrupts (e.g. keyboard key-press).
- Management of the memory.
- Management of the users (e.g. file access).
- Provision of a graphical or textual user interface.
- Management of the file system.

Virtual machine

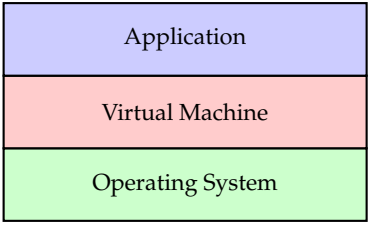
A Virtual Machine (VM) is a software application that emulates the features of a real computer and is therefore able to execute program like a real computer.

The VMs can be classified in two groups:

System VMs



Process VMs



Virtual Machine

System VMs:

- Complete emulation of the features of a real computer.
- They allow the installation and execution of a full OS (e.g. VMWare, VirtualBox, Parallels Desktop, . . .).

Process VMs (or Application VMs):

- They offer enough features to execute an application compatible with the emulated system (e.g. Java Virtual Machine).
- They are commonly used with high level programming languages such as Java & .Net.
- Benefits:
  - usage of interpreted and portable ByteCode + Just In Time (JIT) compilation,
  - garbage collector for automatic memory management.

What is a language?

A language is a set of words and methods to combine words used and understood by a comunity of people.

A language is characterised by:

- Alphabet: set of symbols.
- Syntax: set of grammatical rules to define correct sentences composed by words obtainable from the language alphabet.
- Semantics: meaning of the sentences of the language.
- Pragmatics: which sentences to use depending on the context.

Programming languages

A programming language is a language built to communicate instructions to a machine. They can be used to create programs that control the computer behavior.

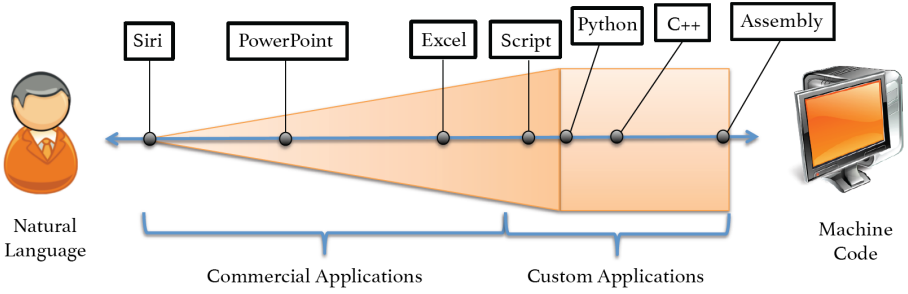
Programming languages, in contrast to natural languages, cannot be ambiguous and must be formalized in no uncertain terms.

They are composed by:

- Syntax: set of rules that define what is valid to write.
- Semantics: meaning of what is valid to write.
- Pragmatics: ability to know which sentences (instructions) it is best to use depending on the context.

Programming languages

- First generation: based on machine code (binary).
- Second generation: assembly code (symbols with usage of mnemonics).
- Third generation: structured and hardware independent (e.g. C, Python, Java).
- Fourth generation: simple syntax and application specific (e.g. SQL).



Examples

C language

```
void swap(int v[ ], int k)
{
  int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```

Assembly

```
swap:
        muli    $2, $5, 4
        add     $2, $4, $2
        lw      $15, 0($2)
        lw      $16, 4($2)
        sw      $16, 0($2)
        sw      $15, 4($2)
        jr      $31
```

Machine code (binary)

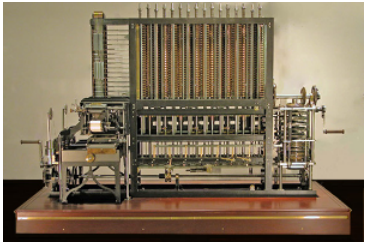
```
00000000101000010000000000011000
00000000100011100001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Who invented programming?

The first programmable computer was Babbage analytic machine.

The first programmer was Lady Ada Byron (1851-1852), countess of Lovelace, daughter of the famous poet Lord Byron.

Babbage invented the mechanical notation, maybe the first programming "language".



Short history: the 50s

1950: Maurice Wilkes uses at Cambridge University the Assembler (symbolic assembly language).

1954: John Backus from IBM creates the Fortran (Formula Translation) language.

1959: The "conference on data system languages" (codasyl) defines the COBOL (common business oriented language), which introduce the use of code blocks.

Short history: the 60s

1960: Definition of the Algol 60 (algorithmic language), which introduces the concept of recursion.

1964: John Kemeny and Tom Kurtz create the Basic (Beginners All-Purpose Symbolic Instruction Language).

1969: Niklaus Wirth from ETH writes the Pascal compiler, a language with strong data typing.

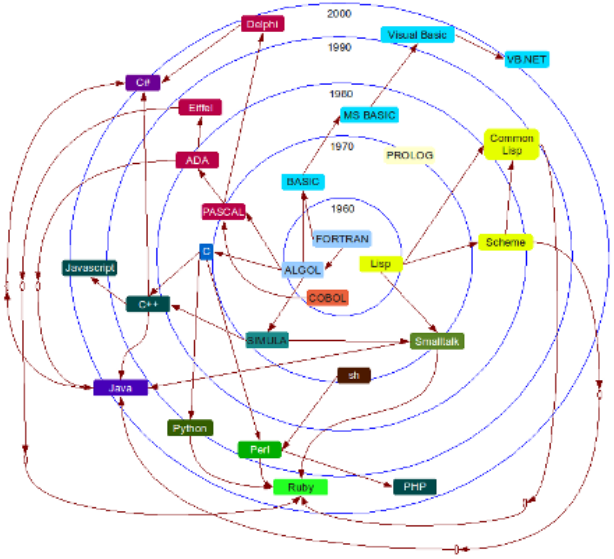
Short history: the 70s

1970: Dennis Ritchie creates the C language to write UNIX.

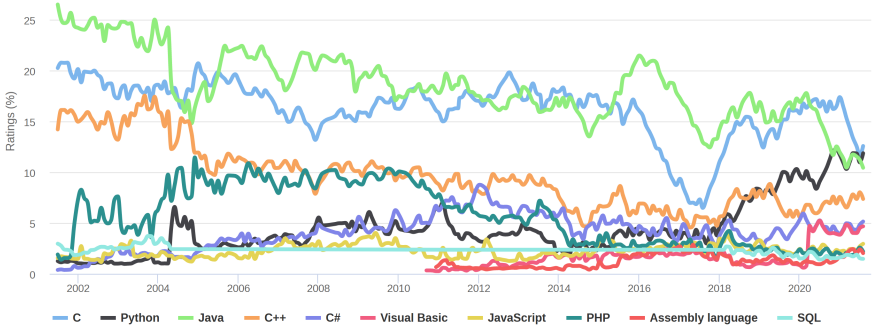
1972: Gary Kildall (1942-1994) writes the PL/I the first language for the Intel 4004 microprocessor.

1973: Alain Colmerauer develops for the french university of Marseilles-Luminy the Prolog(programmation en logique).

Third generation languages



Most popular languages



Aug 2021	Aug 2020	Change	Programming Language	Ratings	Change
1	1		C	12.57%	-4.41%
2	3	▲	Python	11.86%	+2.17%
3	2	▼	Java	10.43%	-4.00%
4	4		C++	7.36%	+0.52%

— *www.tiobe.com*

Compiled vs. Interpreted languages

The expression "compiled language" indicates a programming language implemented usually by means of a compiler instead of an interpreter.

Compiler: converts source code to machine code.
Interpreter: directly executes source code.

Theoretically every language can be implemented both with a compiler or an interpreter.

Compiler

A compiler is a program that translates instructions written in a certain programming language (source code) to instruction in another language, typically a lower level one.

This process of translation is named compilation.

The most common compilers translate third generation languages in assembly for a specific processor architecture.

Interpreter

An interpreter is a program that directly executes instruction written in a programming or scripting language, without requiring them to be compiled into low level machine language.

An interpreter is typically implemented using one of the following strategies:

- Parse (analyze) the source code and perform instructions directly
- Translate source code into an intermediate representation and execute that
- Executes precompiled code generated by an embedded compiler

Programming paradigms: Procedural programming

Each programming language support different programming paradigms.

Procedural Programming is a programming paradigm that consists in the creation of blocks of source code, identified by a name and surrounded by delimiters.

These are known as subrutines, procedures or functions.

Programming paradigms: Object oriented programming

Object Oriented Programming (OOP) is a programming paradigm that allows to define software objects that are able to interact with each others.

Object oriented programming provides a natural support to reproduce in software objects from the real world or from abstract models.

It is particularly suited in contexts where it is possible to define interdependent relations between the concepts to model.

C and C++

C

- Dennis Ritchie, 1969.
- Created to rewrite UNIX in C.
- "Low" level language used in "high" level applications

C++

- Bjarne Stroustrup, 1983.
- Adds the object oriented programming to C.

Python

Python is an high level, interpreted, general purpose, dynamically typed programming language. It supports both procedural and object oriented approaches.

It is available for all the main OS distributions. It is therefore possible to write a program once and run it anywhere (WORA)

The availability of a large set of data manipulation libraries makes it the best choice for a data scientist.

Python is free, it doesn't cost anything to be used and it is open source. It is available to be freely modified or re-distributed under a GLP-compatible licence.

Python has been created in the 1990s by Guido van Rossum at Stichting Mathematisch Centrum. The core of the language is still developed by its creator and by a team at the Python Software Foundation(PSF).

Python main features

High level language

- basic data types, variables and operators,
- control flow and loop instructions (`if`, `for`, `while`, ...).
- functions,
- lists, dictionaries, tuples, objects, classes,
- exceptions management,
- modules and packages.

Object oriented

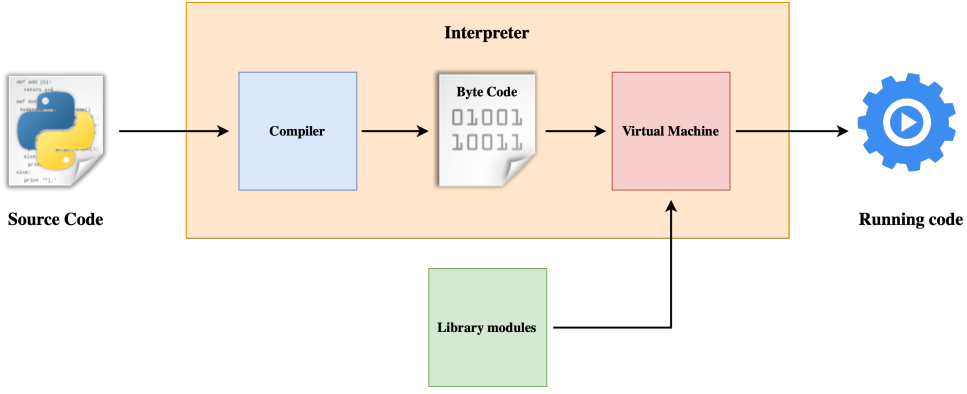
- classes, inheritance, abstraction, polymorphism

Input and output features

Parallel and concurrent programming

Python Interpreter

The default and most widely used implementation of the Python Interpreter is CPython. CPython includes also a compiler, used to convert code to bytecode before the interpretation step.



Python Interpreter

The byte code is usually generated in memory and discarded when the program ends.

However, if importing a python module, a *.pyc* file including its byte code is generated.

When the module is imported the next time the byte code is directly loaded from the pyc, skipping the compilation step.

Integrated Development Environments

An Integrated Development Environment (IDE), provides a graphical user interface that allows to perform all the activities necessary for software developments.

Among the most used for Python:

- Visual Studio Code: https://code.visualstudio.com
- PyCharm: https://www.jetbrains.com/pycharm/
- Jupyter: https://jupyter.org
- Spyder: https://www.spyder-ide.org