Principles & Architecture

```
86          self.names = names
87          self.name2index = dict(zip(names, range(len(names))))
88
89
90      def __del__(self):
91          # free memory created by C to avoid memory leak
92          if hasattr(self, '__createfrom__') and self.__createfrom__ == 'C':
93              if pointer(self) is not None:
94                  libbigfile.free_file(pointer(self))
95
96      def read(self, requested, isname=True):
97          if isname:
98              index_name_array = [(self.name2index[x], x) for x in requested]
99          else:
100             assert(min(requested)>=0)
101             assert(max(requested)<len(self.names))
102             index_name_array = [(x, self.names[x]) for x in requested]
103         index_name_array.sort()
104
105         npoints = len(index_name_array)
106         c_index = (c_ulonglong * npoints)()
107         for i in range(npoints):
108             c_index[i] = index_name_array[i][0]
109
110         size = self.ndims * npoints
111         pdata = (c_float * size)()
112         res = libbigfile.seq_read_memory(self, npoints, c_index, pdata)
113         assert(res)
```

EPITA Bachelor of Science

Principles and Architecture of Information Systems

Chapter #3

Software

Olivier BERTHET

Principles & Architecture

Structure

- Chapter 1 : Introduction and Organisations
- Chapter 2 : Hardware
- Chapter 3 : Software
- Chapter 4 : Database Systems
- Chapter 5 : Network
- Chapter 6 : Internet and E-Commerce
- Chapter 7 : Major Information Systems
- Chapter 8 : Systems Development
- Chapter 9 : Security,Privaca and Ethical issues

Why Learn About Software?

- **Software is indispensable for any computer system and the people using it**
- **Applications software:**
  - **Key to helping you achieve your career goals and enrich your life**
  - **Stock trading, scientific, accounting, tax, etc.**

Principles & Architecture

Wooclap : List several programming languages

Principles & Architecture

Evolution of programming language

https://youtu.be/Og847HVwRSI

Principles & Architecture

Energy Efficiency across Programming Languages

Article here

Total

	Energy
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

	Time
(c) C	1.00
(c) Rust	1.04
(c) C++	1.56
(c) Ada	1.85
(v) Java	1.89
(c) Chapel	2.14
(c) Go	2.83
(c) Pascal	3.02
(c) Ocaml	3.09
(v) C#	3.14
(v) Lisp	3.40
(c) Haskell	3.55
(c) Swift	4.20
(c) Fortran	4.20
(v) F#	6.30
(i) JavaScript	6.52
(i) Dart	6.67
(v) Racket	11.27
(i) Hack	26.99
(i) PHP	27.64
(v) Erlang	36.71
(i) Jruby	43.44
(i) TypeScript	46.20
(i) Ruby	59.34
(i) Perl	65.79
(i) Python	71.90
(i) Lua	82.91

	Mb
(c) Pascal	1.00
(c) Go	1.05
(c) C	1.17
(c) Fortran	1.24
(c) C++	1.34
(c) Ada	1.47
(c) Rust	1.54
(v) Lisp	1.92
(c) Haskell	2.45
(i) PHP	2.57
(c) Swift	2.71
(i) Python	2.80
(c) Ocaml	2.82
(v) C#	2.85
(i) Hack	3.34
(v) Racket	3.52
(i) Ruby	3.97
(c) Chapel	4.00
(v) F#	4.25
(i) JavaScript	4.59
(i) TypeScript	4.69
(v) Java	6.01
(i) Perl	6.62
(i) Lua	6.72
(v) Erlang	7.20
(i) Dart	8.64
(i) Jruby	19.84

Principles & Architecture

Intermediate Quiz

- **Please enter your full name : first and last no nickname, no superman…**
- **WOOCLAP : GINREQ**

Principles & Architecture

Principles

- **Systems and application software are critical in helping individuals and organizations achieve their goals**
- **Organizations use off-the-shelf application software for common business needs and proprietary application software to meet unique business needs and provide a competitive advantage**
- **Organizations should choose programming languages with functional characteristics that are appropriate for the task at hand and well suited to the skills and experience of the programming staff**
- **The software industry continues to undergo constant change; users need to be aware of recent trends and issues to be effective in their business and personal life**

Principles & Architecture

An Overview of Software

- **Computer programs**
  - **Sequences of instructions for the computer**
- **Documentation**
  - **Describes program functions to help the user operate the computer system**
- **Types of software**
  - **Systems software**
  - **Application software**

Systems Software

- **Set of programs that coordinates the activities and functions of hardware and other programs**
- **Types of systems software**
  - **Operating systems**
  - **Utility programs**
  - **Middleware**

Application Software

- **Helps users solve particular problems**
- **In most cases, resides on the computer's hard disk**
- **Can be stored on CDs, DVDs, and even USB flash drives**

Principles & Architecture

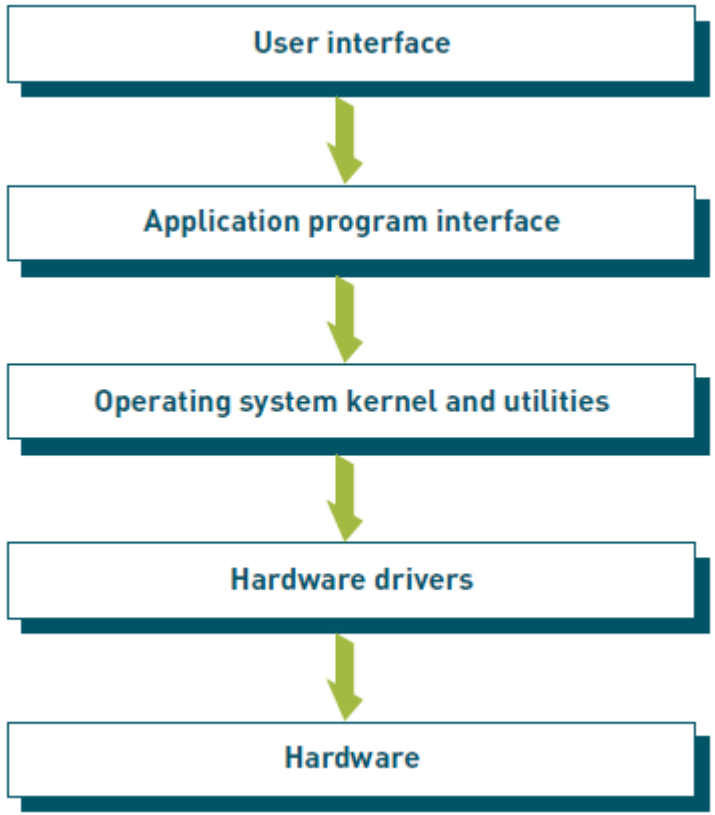
Software supporting individuals, workgroups and enterprises

Software Type	Personal	Workgroup	Enterprise
Systems software	Smartphone, tablet, personal computer, and workstation operating systems	Network operating systems	Server and mainframe operating systems
Application software	Word-processing, spreadsheet, database, and graphics programs	Email, group-scheduling, shared-work, and collaboration applications	General-ledger, order-entry, payroll, and human-resources applications

Principles & Architecture

Operating System

- **Set of programs that controls computer hardware and acts as an interface with application programs**
- **Kernel:**
  - **Ties all components of the OS together and regulates other programs**
- **Combinations of OSs, computers, and users:**
  - **Single computer with a single user**
  - **Single computer with multiple simultaneous users**
  - **Multiple computers with multiple users**
  - **Special-purpose computers**



Principles & Architecture

Activities performed by the operating system

- **Perform common computer hardware functions**
- **Provide a user interface and input/output management**
- **Provide a degree of hardware independence**
- **Manage system memory**
- **Manage processing tasks**
- **Provide networking capability**
- **Control access to system resources**
- **Manage files**
- **Get input from keyboard or another input device**
- **Retrieve data from disks**
- **Store data on disks**
- **Display information on a monitor or printer**

Principles & Architecture

Operating systems by sphere of influence

Personal	Workgroup	Enterprise
Microsoft Windows	Microsoft Windows Server	Microsoft Windows Server
Mac OS X, iOS	Mac OS X Server	
Linux	Linux	Linux
Google Android, Chrome OS	UNIX	UNIX
HP webOS	IBM i and z/OS	IBM i and z/OS
	HP-UX	HP-UX

Principles & Architecture

History of Microsoft Windows

Year	Version	Highlights
1985	Windows 1.0	Ran as a graphical, 16-bit multitasking shell on top of an existing MS-DOS installation, providing an environment that could run graphical programs designed for Windows as well as existing MS-DOS software
1987	Windows 2.0	Introduced more sophisticated keyboard shortcuts as well as the ability to minimize and maximize Windows
1988	Windows 2.03	Allowed application Windows to overlap each other
1990	Windows 3.0	Introduced a multitasking capability with a protected/enhanced mode, which allowed Windows applications to use more memory; first widely successful version of Windows
1992	Windows 3.1	Introduced improved system stability and expanded support for multimedia, TrueType fonts, and workgroup networking
1995	Windows 95	Introduced numerous important features and functions, such as the taskbar, the Start button, and a new approach to user navigation; moved from a 16-bit architecture to a 32-bit architecture
1998	Windows 98	Introduced many features, such as the Quick Launch toolbar, the Active Desktop, single-click launching, Back and Forward navigation buttons, favorites, and the address bar in Windows Explorer, and image thumbnails; heavily criticized operating system, with major compatibility issues
1999	Windows 98 Second Edition	Included fixes for many Windows 98 problems and replaced Internet Explorer 4.0 with Internet Explorer 5.0; improved audio, modem, and USB support

Principles & Architecture

History of Microsoft Windows

Year	Version	Highlights
2000	Windows 2000	An operating system for use on both client and server computers; marketed as the most secure Windows version ever, but it became the target of a number of high-profile virus attacks, such as Code Red and Nimda
2000	Windows ME	Rated Microsoft's worst OS by many industry observers; exhibited stability and compatibility issues; included Internet Explorer 5.5, Windows Media Player 7, and Windows Movie Maker software, which provided basic video editing functions that were designed to be easy for consumers to use
2001	Windows XP	Offered a major advance from the MS-DOS–based versions of Windows in terms of security, stability, and efficiency; introduced a significantly redesigned graphical user interface
2007	Windows Vista	Focused primarily on improving security; offered an updated graphical user interface and visual style dubbed "Aero" and a new search component called Windows Search; provided redesigned networking, audio, print, and display subsystems, as well as new multimedia capabilities, including Windows DVD Maker
2009	Windows 7	Provided an incremental upgrade to the operating system; intended to address Windows Vista's performance issues, while maintaining hardware and software compatibility; provided support for touch displays and 64-bit processors
2012	Windows 8	Introduced major changes to the operating system's platform and user interface to improve user experience on tablets; included a touch-optimized Windows shell, a Start screen that displays programs and dynamically updated content on a grid of tiles, the ability to sync apps and settings between devices, and the Windows Store for downloading and purchasing new software

Principles & Architecture

History of Microsoft Windows

Year	Version	Highlights
2013	Windows 8.1	Included an improved Start screen, additional bundled apps, tighter OneDrive integration, Internet Explorer 11, a Bing-powered unified search system, and restoration of a visible Start button on the taskbar
2015	Windows 10	Brought back the familiar Start menu and desktop; introduced the Edge browser and the Cortana assistant, which responds to natural language and can perform a variety of organizational tasks for the end user, including setting reminders, scheduling calendar events, calculating math problems, and converting measurements and money

Principles & Architecture

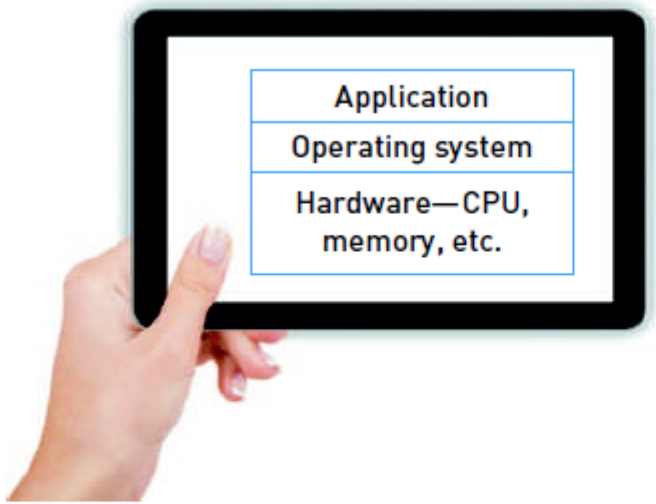
User interface and input/output management

- **User interface:**
  - Allows individuals to access and command the computer system
- **Command-based user interface:**
  - Requires that text commands be given to the computer to perform basic activities
- **Graphical user interface (GUI):**
  - Uses icons and menus displayed on screen to send commands to the computer system
- **Networking capability:**
  - Allows computers in a network to send and receive data and share computing resources
- **Access to system resources and security:**
  - Protection against unauthorized access
  - OS establishes a logon procedure
- **File management:**
  - Ensures that files in secondary storage are available when needed and that they are protected from access by unauthorized users

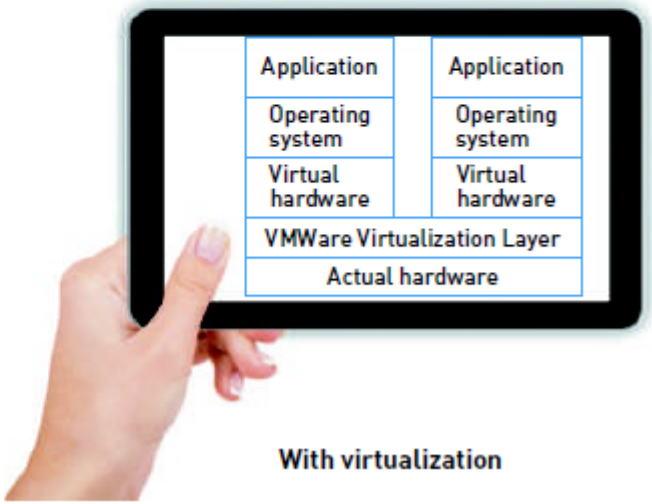
Principles & Architecture

Virtualization

- **Virtual servers that separate a physical computing device into one or more “virtual” servers, each of which can be easily used and managed to perform computing tasks.**



Without virtualization



With virtualization

Julia Ivantsova/Shutterstock.com

Utility Programs

- **Help to perform maintenance or correct problems with a computer system**
- **Common types of utilities:**
  - **Hardware utilities**
  - **Security utilities**
  - **File-compression utilities**
  - **Spam-filtering utilities**
  - **Network and Internet utilities**
  - **Server and mainframe utilities**

Other Utilities

- **Key logging software allows a manager to see every keystroke a worker makes on a computer system**
- **Monitoring software can catalog the Internet sites that employees visit**
- **Keyboard shortcut utilities allow users to map common tasks to defined keyboard combinations**

Examples of Utility Programs

Personal	Workgroup	Enterprise
Software to compress data so that it takes less hard disk space	Software to provide detailed reports of work-group computer activity and status of user accounts	Software to archive contents of a database by copying data from disk to tape
Screen saver	Software that manages an uninterruptible power supply to do a controlled shutdown of the workgroup computer in the event of a loss of power	Software that compares the content of one file with another and identifies any differences
Antivirus and antispyware software	Software that reports unsuccessful user logon attempts	Software that reports the status of a particular computer job

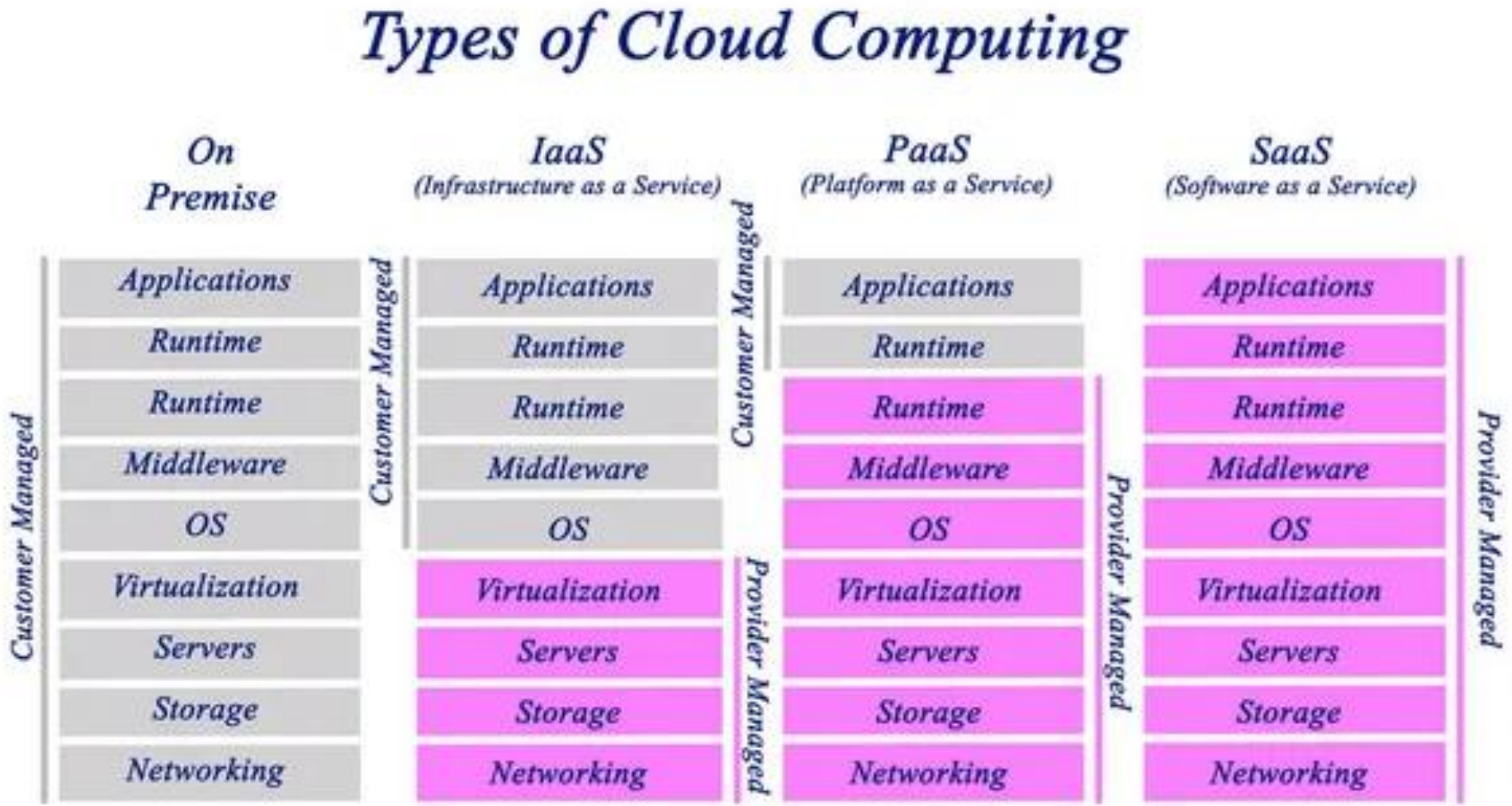
Principles & Architecture

One example PRTG Network monitoring

Middleware

- Software that allows different systems to communicate and exchange data
- This systematic tying together of disparate applications, often through the use of middleware, is known as enterprise application integration (EAI).
- Can also be used as an interface between the Internet and older legacy systems
- Service-oriented architecture (SOA)
  - Uses modular application services to allow users to interact with systems, and systems to interact with each other

Principles & Architecture



Application software

- **The primary function of application software is to apply the power of a computer system to enable people, workgroups, and entire enterprises to solve problems and perform specific tasks.**
- **Millions of software applications have been created to perform a variety of functions on a wide range of operating systems and device types**

Business	Genealogy	Personal information manager
Communications	Language	Photography
Computer-aided design	Legal	Science
Desktop publishing	Library	Simulation
Educational	Multimedia	Video
Entertainment	Music	Video games

Principles & Architecture

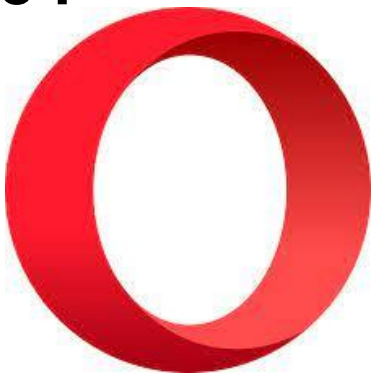
Wooclap ETFIAT : List several Web navigators

Principles & Architecture

One example : Web browser

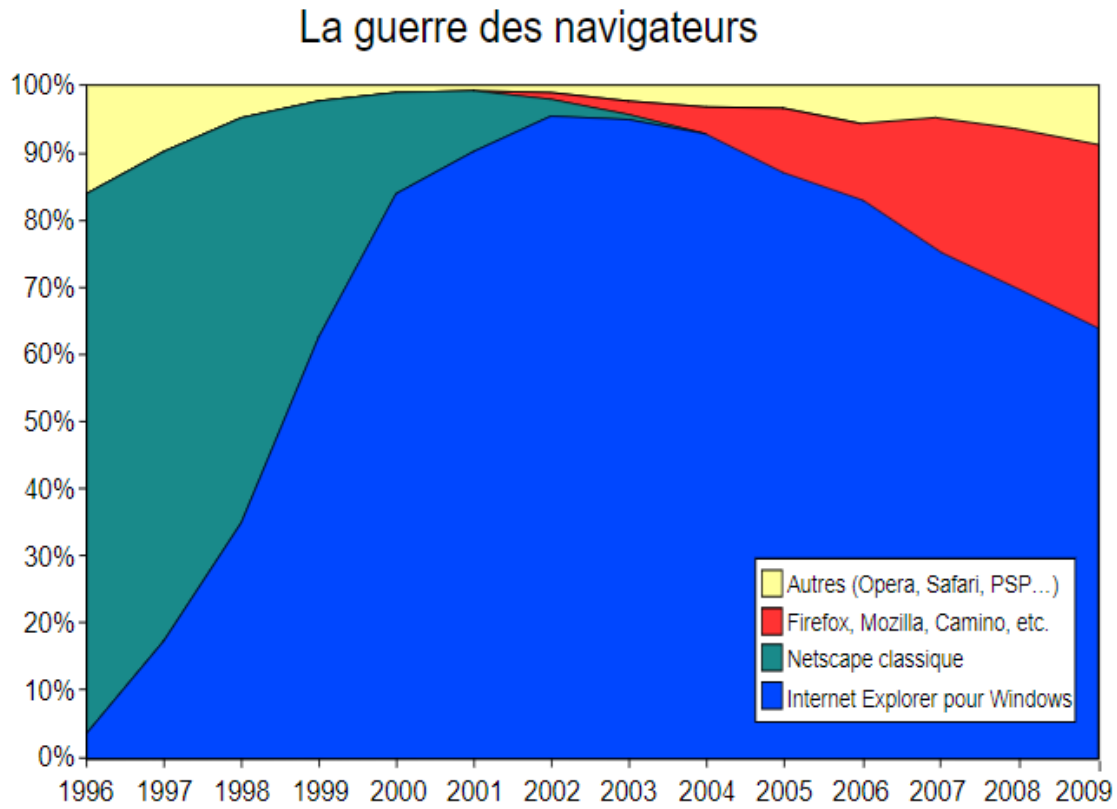


You have the choice !

Principles & Architecture

One has disappeared



Overview of Application software

- **Proprietary software:**
  - One-of-a-kind program for a specific application, usually developed and owned by a single company
- **Off-the-shelf software:**
  - Existing software program that is purchased
- **Application service provider (ASP):**
  - Company that can provide software, support, and computer hardware on which to run the software from the user's facilities over a network

Principles & Architecture

Proprietary versus Off-the-shelf

Proprietary Software		Off-the-Shelf Software	
Advantages	**Disadvantages**	**Advantages**	**Disadvantages**
You can get exactly what you need in terms of features, reports, and so on.	It can take a long time and a significant amount of resources to develop required features.	The initial cost is lower because the software firm can spread the development costs across many customers.	An organization might have to pay for features that it does not require and never uses.
Being involved in the development offers more control over the results.	In-house system development staff may be hard-pressed to provide the required level of ongoing support and maintenance because of pressure to move on to other new projects.	The software is likely to meet the basic business needs. Users have the opportunity to more fully analyze existing features and the performance of the package before purchasing.	The software might lack important features, thus requiring future modification or customization, which can be very expensive, and because users will eventually be required to adopt future releases of the software, the customization work might need to be repeated.
You can more easily modify the software and add features that you might need to counteract an initiative by competitors or to meet new supplier or customer demands.	The features and performance of the delivered software may fail to meet evolving business and end user needs.	The software is likely to be of high quality because many customer firms have tested the software and helped identify its bugs.	The software might not match current work processes and data standards.

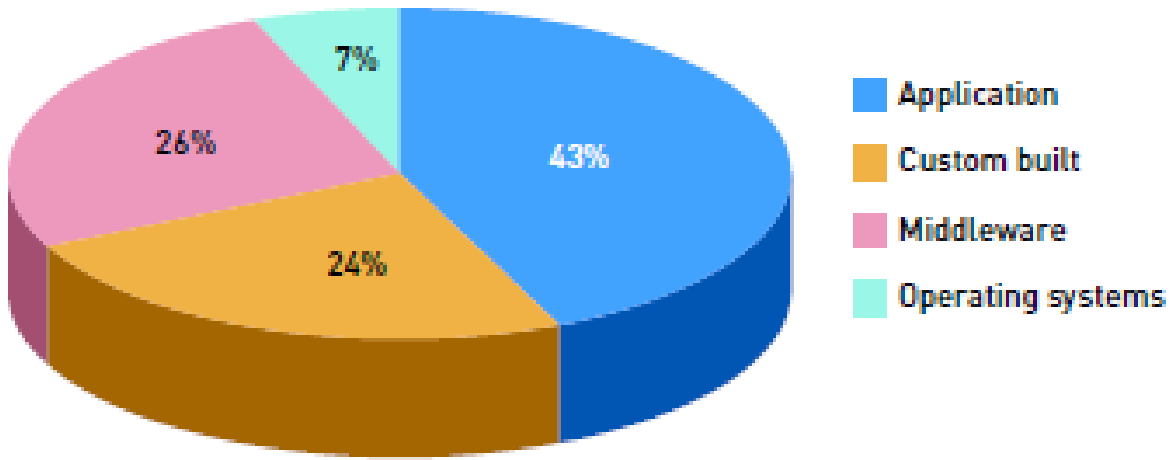
Principles & Architecture

Personal Application software

Type of Software	Use	Example
Word processing	Create, edit, and print text documents	Apache OpenOffice Writer Apple Pages Corel Write Google Docs Microsoft Word WordPerfect
Spreadsheet	Perform statistical, financial, logical, database, graphics, and date and time calculations using a wide range of built-in functions	Apache OpenOffice Calc Apple Numbers Google Sheets IBM Lotus 1-2-3 Microsoft Excel
Database	Store, manipulate, and retrieve data	Apache OpenOffice Base Microsoft Access IBM Lotus Approach
Graphics	Develop graphs, illustrations, drawings, and presentations	Adobe FreeHand Adobe Illustrator Apache OpenOffice Impress Microsoft PowerPoint
Personal information management	Helps people, groups, and organizations store useful information, such as a list of tasks to complete or a set of names and addresses	Google Calendar Microsoft Calendar Microsoft Outlook One Note
Project management	Plan, schedule, allocate, and control people and resources (money, time, and technology) needed to complete a project according to schedule	Microsoft Project Scitor Project Scheduler

Enterprise Application software

- **Worldwide spending on enterprise software was estimated to be about $310 billions in 2015. Most software spending goes to application software.**



Principles & Architecture

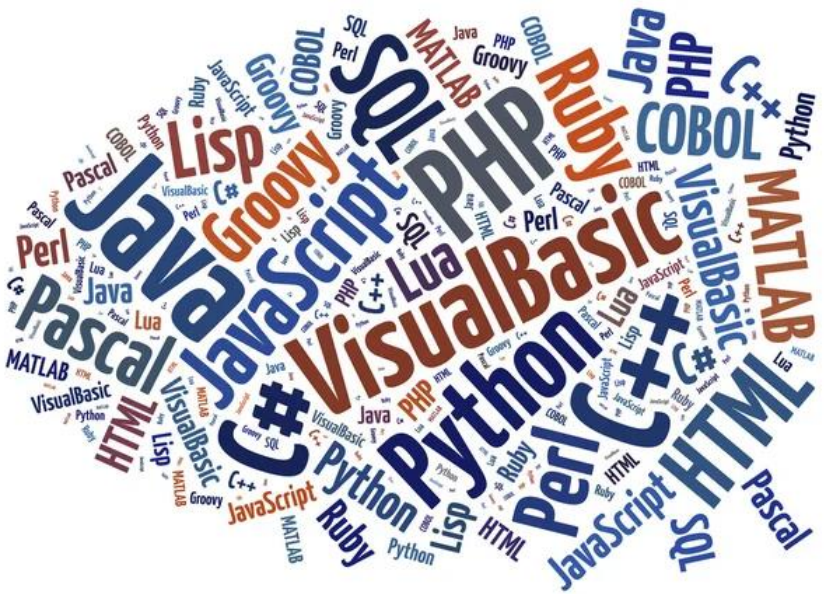
Programming languages

- **The primary function of a programming language is to provide instructions to the computer system so that it can perform a processing activity.**
- **Information systems professionals work with different programming languages, which are sets of keywords, commands, symbols and rules for constructing statements that people can use to communicate instructions to a computer.**
- **Programming involves translating what a user wants to accomplish into a code that the computer can understand and execute**

Principles & Architecture

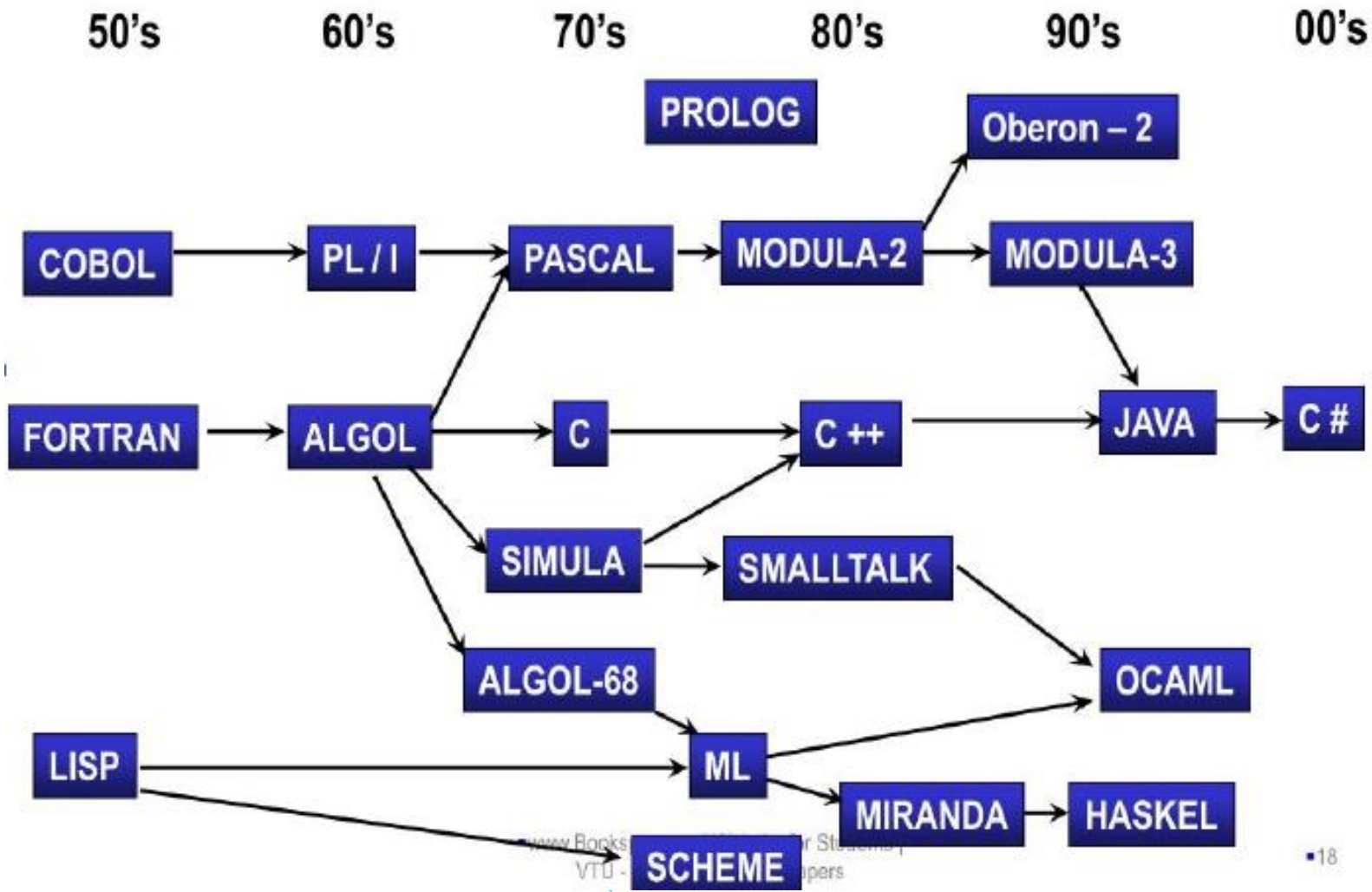
Why so many programming languages ?

- **The wide range of computational needs has prompted the creation of hundreds of programming languages.**
- **Different classes of problems may demand different levels of abstraction, and different programmers have different ideas on how abstraction should be done**
- **Language influences perception and perception creates new language**
- **New languages have in turn introduced new way of thinking about programming & improved upon old ones**



Principles & Architecture

Evolution of Programming Languages



Principles & Architecture

IBM APL

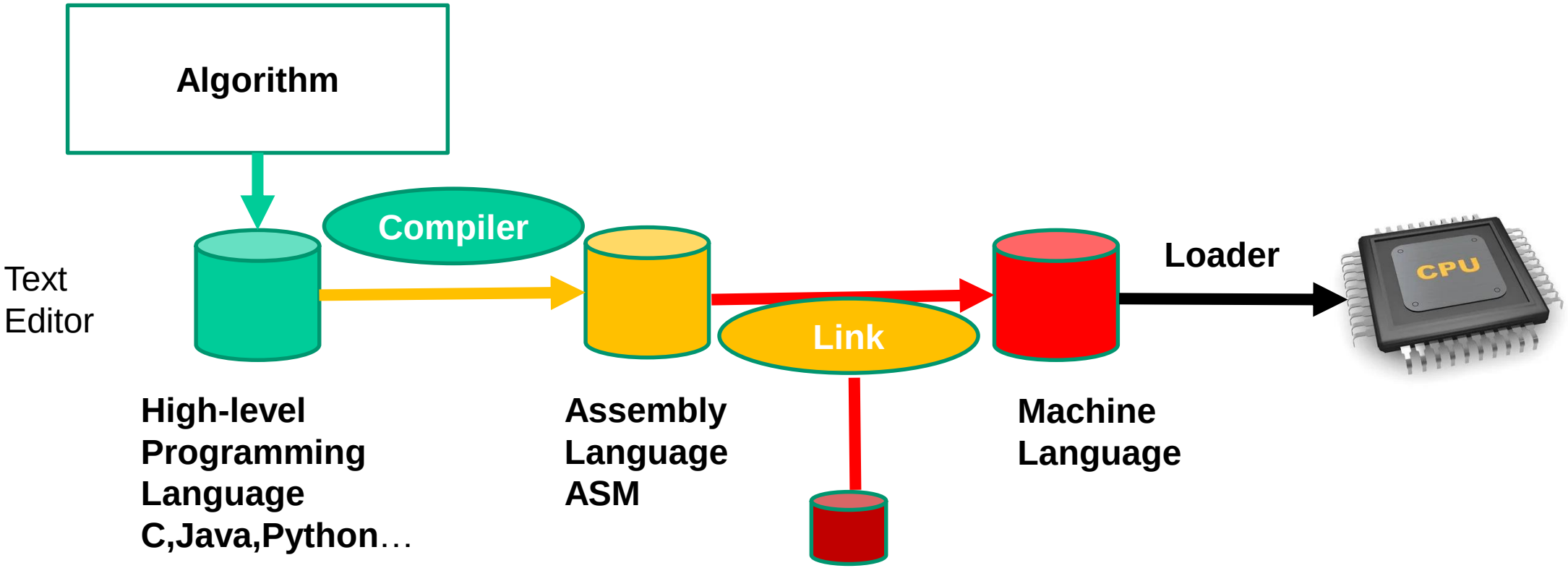
```
      ∇DET[⎕]∇
     ∇ Z←DET A;B;P;I
[1]    I←⎕IO
[2]    Z←1
[3]   L:P←(|A[;I])⍳⌈/|A[;I]
[4]    →(P=I)/LL
[5]    A[I,P;]←A[P,I;]
[6]    Z←-Z
[7]   LL:Z←Z×B←A[I;I]
[8]    →(0 1 ∨.=Z,1↑⍴A)/0
[9]    A←1 1 ↓A-(A[;I]÷B)∘.×A[I;]
[10]   →L
[11]  ⍝EVALUATES A DETERMINANT
      ∇
```

Three levels of languages

- **High level programming language**
  - **the level of programming the more used today. It is a level of programming independent of the physical structure of the machine with syntax and grammar**
- **Assembly language**
  - **Assembly language or ASM is any low-level programming language with a very strong correspondence between the instructions in the language and the architecture's machine code instructions. It corresponds to a symbolic form of the machine language associated with the processor**
- **Machine language**
  - **machine language is a compound language on a binary alphabet. It is the only language executable directly by the processor.**

Principles & Architecture

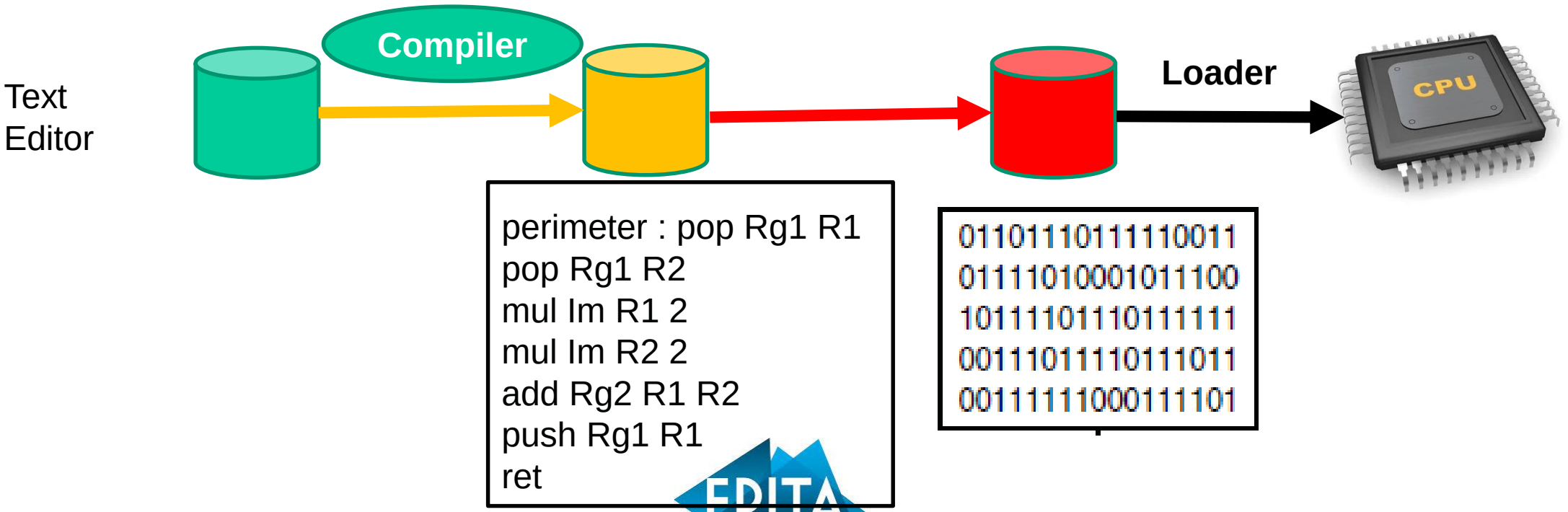
Production of programs



Principles & Architecture

One example

```
function perimeter (a, b : in integer) return integer is
begin
perimetre=(2*a)+(2*b);
end;
```



Principles & Architecture

What do Compilers do ?

- **A compiler acts as a translator transforming human-oriented programming languages into computer-oriented machine languages**
- **Converts the programmer's source code into machine-language instructions**
- **Ignore machine dependent details for programmer**
- **The compiler is an application**

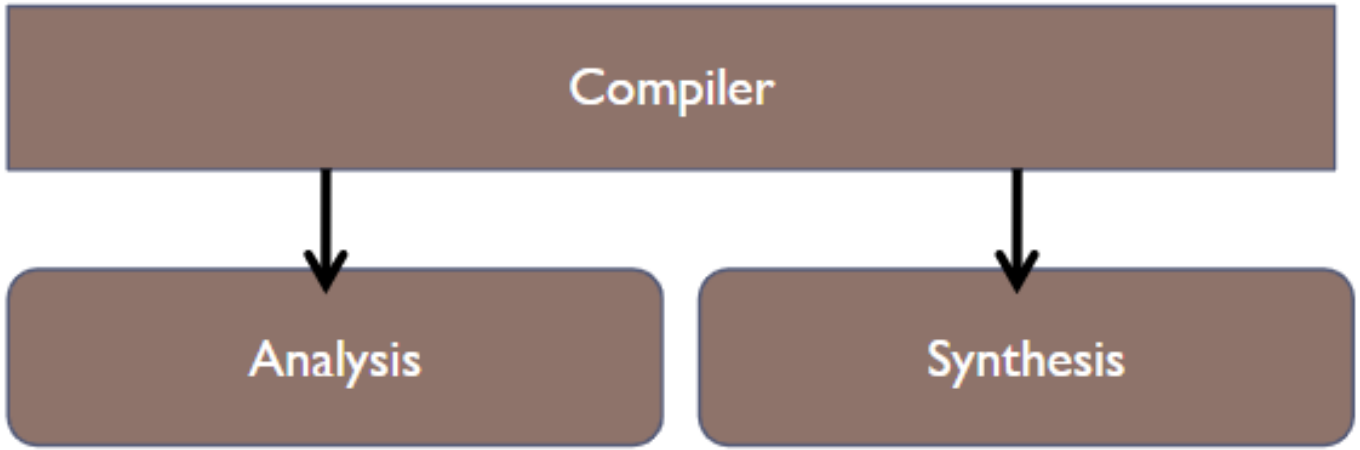
Principles & Architecture

Phases of a compiler

- **Lexical analysis (recognition of the words of the language, i.e. apprehension of the vocabulary)**
- **Parsing (syntax checking , i.e. understanding grammar)**
- **Semantic analysis (verification of semantics, i.e. apprehension of the meaning)**
- **Code optimization and generation object**

Phases of a compiler

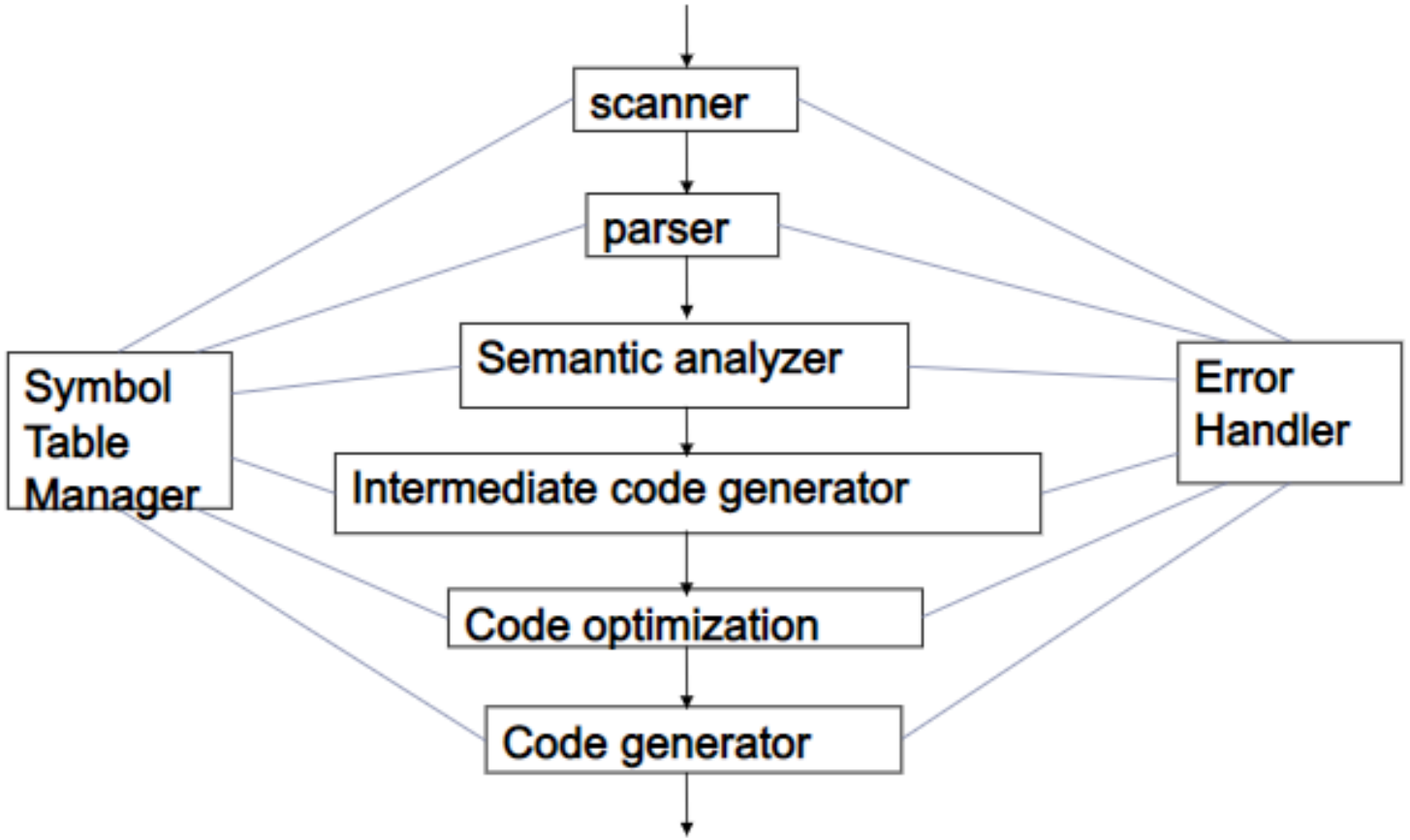
- Any compiler must perform two major tasks



- *Analysis* of the source program
- *Synthesis* of a machine-language program

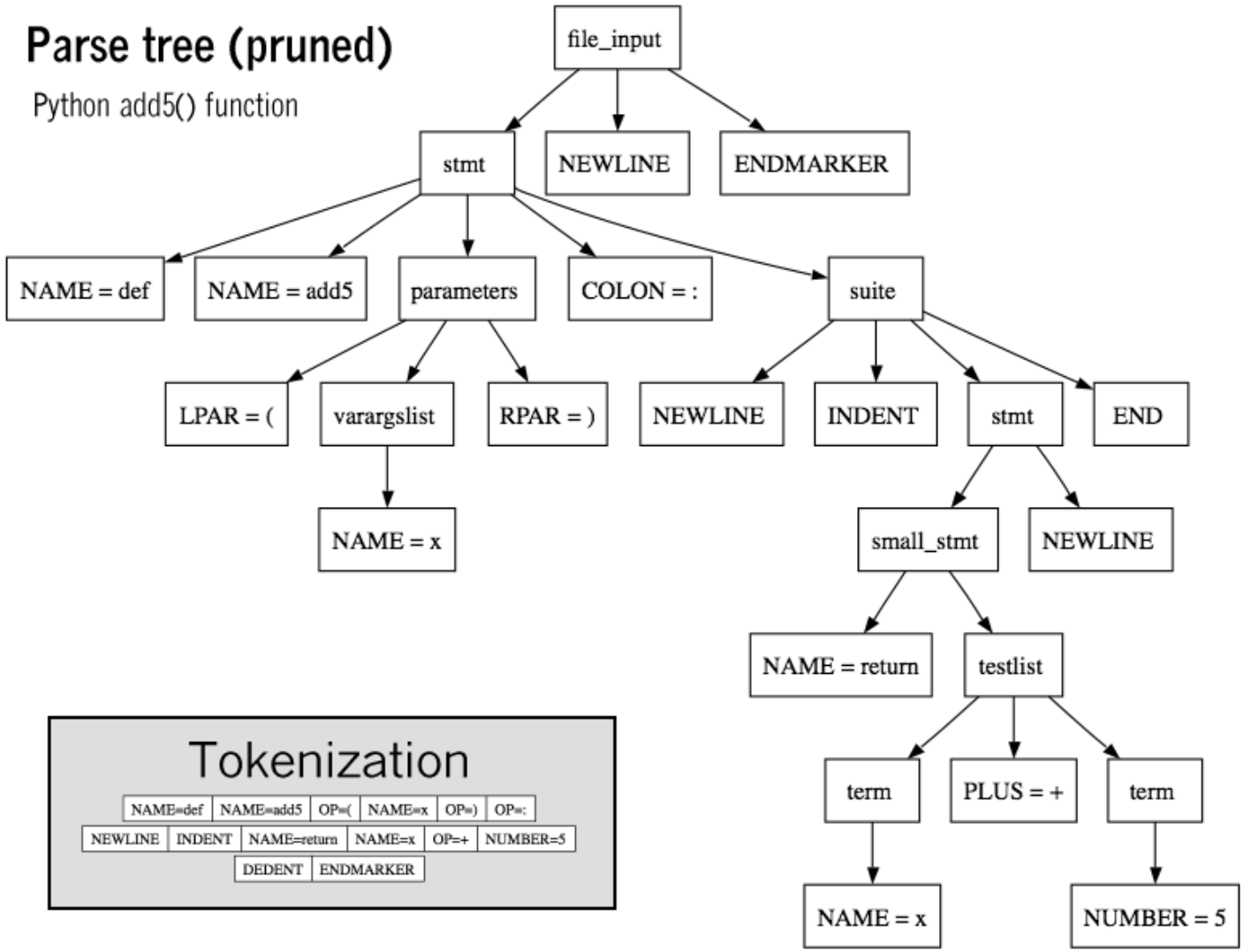
Principles & Architecture

Process of a compiler



Principles & Architecture

Example of a parser

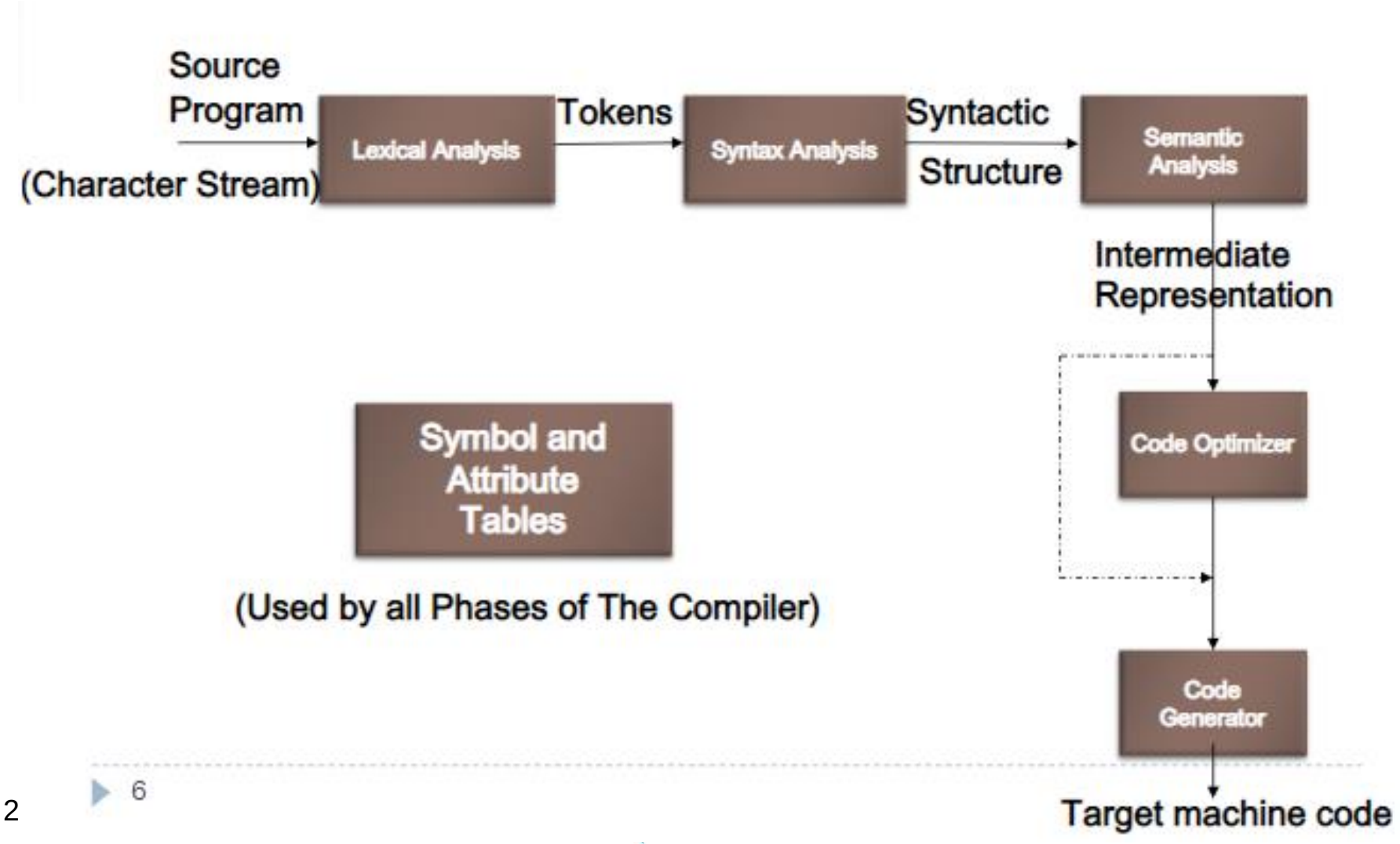


Tokenization

NAME=def	NAME=add5	OP=(	NAME=x	OP=)	OP=:
NEWLINE	INDENT	NAME=return	NAME=x	OP=+	NUMBER=5
DEDENT	ENDMARKER				

Principles & Architecture

Phases of a compiler



Interpreter , AOT versus JIT compiler

- **AOT**
  - **When all the code is transformed at one time before it reaches the platforms that run it, the process is called ahead-of-time (AOT) compilation.**
- **Interpreter**
  - **Interpreted code executes instructions in a program without compiling them into machine language.**
  - **The interpreted code parses the source code directly, is paired with a virtual machine that translates the code for the machine at the time of execution, or takes advantage of precompiled code.**
  - **Javascript is usually interpreted.**

Interpreter , AOT versus JIT compiler

- **JIT**
  - **Just-in-time compilers are a combination of AOT compilers and interpreters.**
  - **Java and C# use just-in-time compilers.**
  - **After a Java program is written, the JIT compiler turns the code into bytecode rather than into code that contains instructions for a specific hardware platform's processor.**
  - **The bytecode is platform independent and can be sent and run on any platform that supports Java. In a sense, the program is compiled in a two-stage process.**

Pros and Cons of AOT and JIT Compilation

- **Ahead-of-time (AOT) compilation delivers faster startup time, particularly when much of the code executes at startup. However, it requires more memory and more disk space.**

- **Just-in-time (JIT) compilation profiles the target platform while it runs and re-compiles on the fly to deliver improved performance. JIT generates improved code because it targets the current platform, although it usually takes more time to run than AOT compiled code.**

Principles & Architecture

Same program in different languages

- **A loop counting downward from 10 to 1**

```
#include <stdio.h>
main() {
 int k;
 for (k=10; k>=1; k--) {
  printf("%d\n",k);
 }
}
```

C

Same program in different languages

- **A loop counting downward from 10 to 1**

```
using namespace std;
#include <iostream>
main() {
 int k;
 for (k=10; k>=1; k--) {
  cout << k << endl;
 }
}
```

C++

Same program in different languages

- **A loop counting downward from 10 to 1**

```
public class revl {
 public static void main(String []
args) {
  int k;
  for (k=10; k>=1; k--) {
   System.out.println(k);
  }
 }
```

Java

Principles & Architecture

Same program in different languages

- **A loop counting downward from 10 to 1**

```
<?php
for ($k=10; $k>=1; $k--) {
 print ("$k\n");
 }
?>
```

PHP

for k in range(10,0,-1):
 print k

Principles & Architecture

Same program in different languages

- **A loop counting downward from 10 to 1**

```
for k in range(10,0,-1):
 print k
```

Python

Principles & Architecture

Another Example

- https://www.geeksforgeeks.org/hello-world-in-30-different-languages/

Software issues and trends

- **Software bug:**
  - **Defect in a program that keeps it from performing as it should**
- **Some tips for reducing impact of software bugs:**
  - **Register all software**
  - **Check read-me files for workarounds**
  - **Access support area of the manufacturer's Web site for patches**
  - **Install latest software updates**

Copyrights and Licenses

- **Most software products are protected by law using copyright or licensing provisions:**
  - **In some cases, you are given unlimited use of software on one or two computers**
  - **In other cases, you pay for your usage; if you use the software more, you pay more**
- **Some software now requires that you register or activate it before it can be fully used**

Copyrights and Licenses

License	Description
Single-user license	Permits you to install the software on one computer, or sometimes two computers, used by one person.
Multiuser license	Specifies the number of users allowed to use the software, and can be installed on each user's computer. For example, a 20-user license can be installed on 20 computers for 20 users.
Concurrent-user license	Designed for network-distributed software, this license allows any number of users to use the software, but only a specific number of users to use it at the same time.
Site license	Permits the software to be used anywhere on a particular site, such as a college campus, by everyone on the site.

Freeware and Open-Source Software

- **Freeware**
  - **Software that is made available to the public for free**
- **Open-source software**
  - **Distributed, typically for free, with the source code**
- **General Public License GPL grants you the right to**
  - **Run the program for any purpose**
  - **Study how the program works and adapt it to your needs**
  - **Redistribute copies so you can help others**
  - **Improve the program and release improvements to the public**