Python vs Big Data

"*Python?? Why Python?*"

https://www.youtube.com/watch?v=VrutixOEtOM

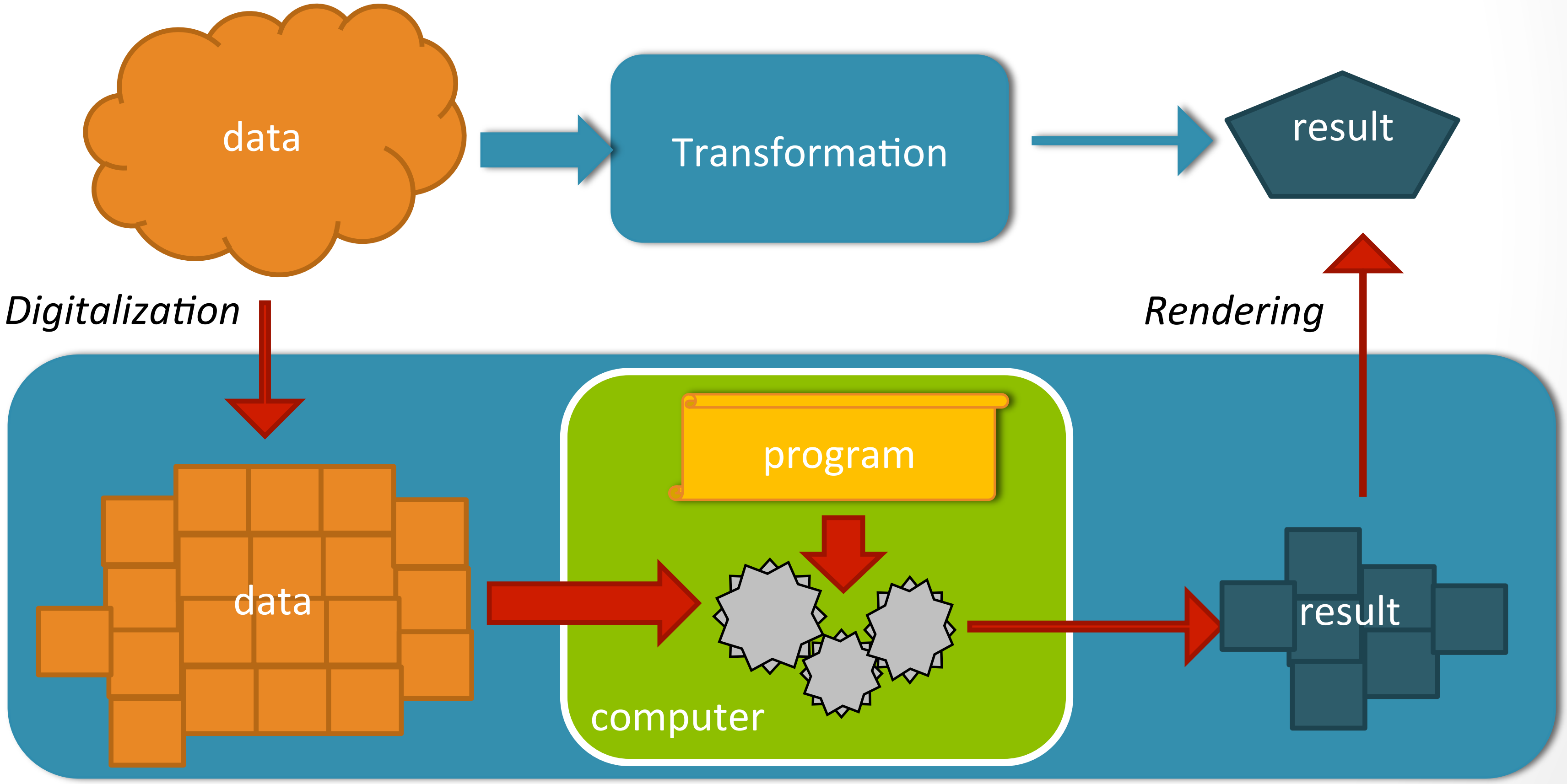
What Is Python

- Python is **a high-level, interpreted and general-purpose dynamic** programming language that focuses on **code readability**.
- The Python is widely used and have a large and active programmer **community**.
- It has a **comprehensive and large standard library** that has automatic memory management and dynamic features.
- It easily extensible by other programming language
  - https://www.python.org/
  - https://en.wikipedia.org/wiki/Python_(programming_language)

Why Python... some step back...

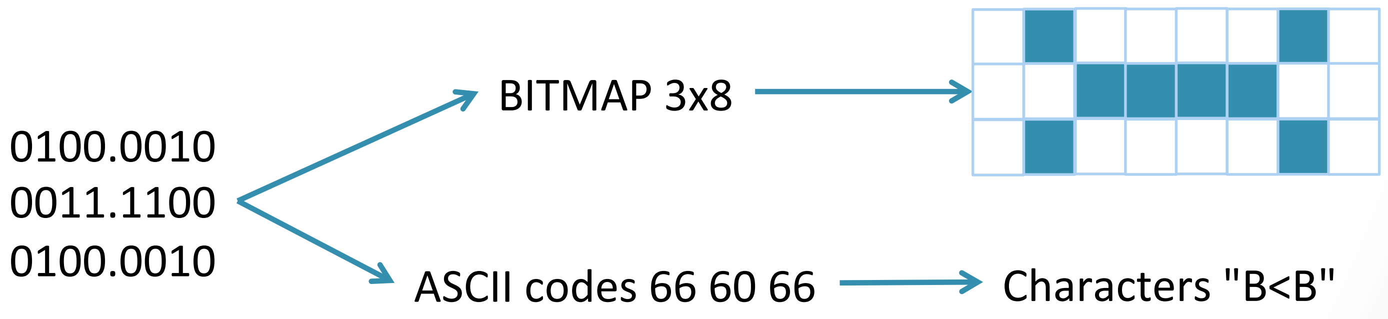
It's a dirty job, but someone have to do it

- People needs to elaborate **data** in order to extract **results**



Data coding

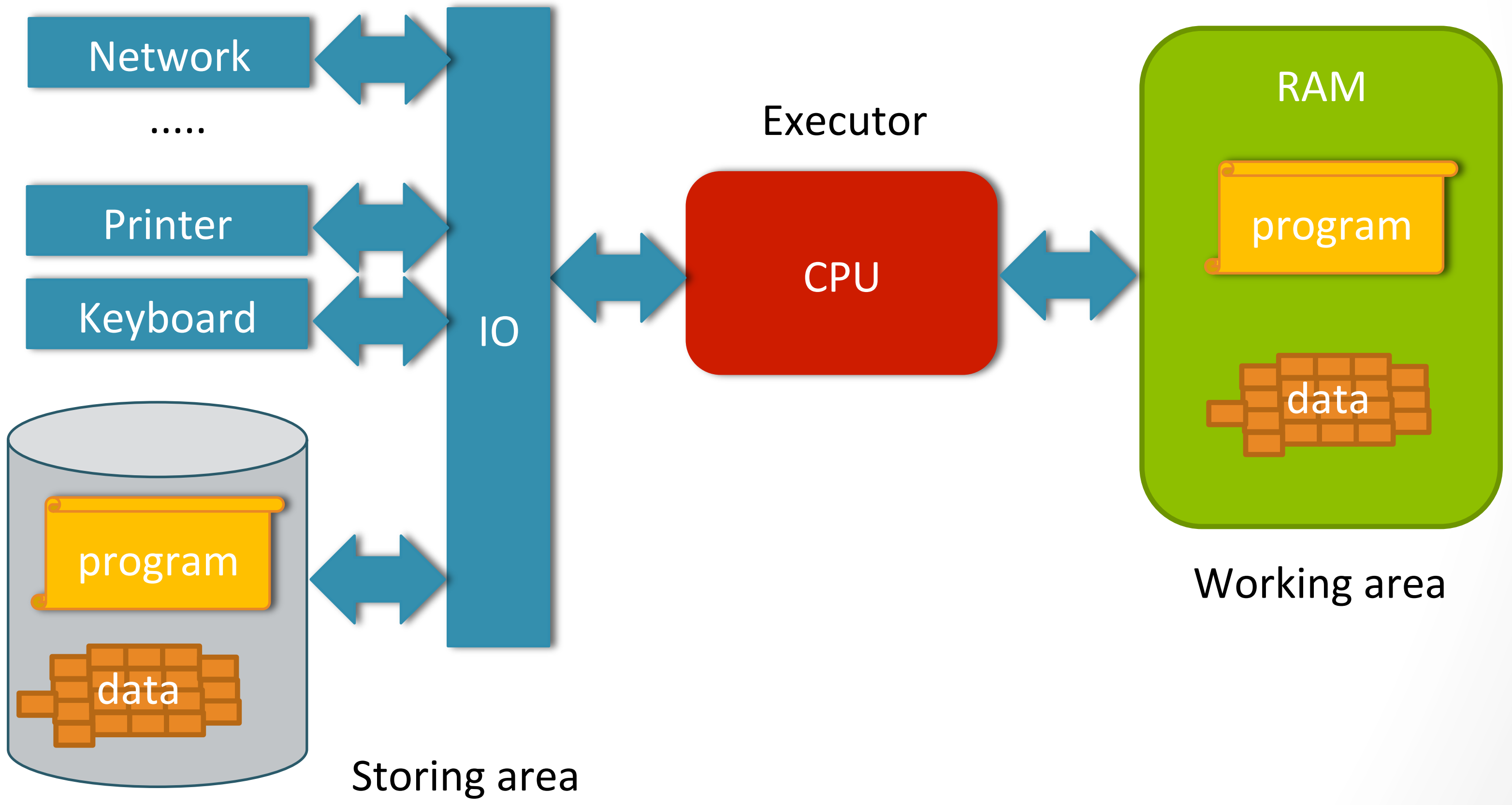
- Digital computers can handle only binary signals: sequences of 0 and 1 (bit = binary digit)
- In order to transform data by digital computers, it needs to **digitalize** data, i.e. transform real samples (images, sound, etc.) into sequences of bits, packed for technological and hostorical reasons into group of 8 bit, called bytes.
- The meaning of a sequence is given by the **format** used to code and interpreter the sequence, eg. ASCII, bitmap, mp3.



https://en.wikipedia.org/wiki/ASCII *https://en.wikipedia.org/wiki/BMP_file_format*

Computers at hardware level

A very schematic and simplified draft of a digita computer



Coding transformations

- A classical digital computer transforms digital data by following a **program**, i.e. a **sequence of commands** that describes the trasformations to be applied to data.
- A program can be written using variouse **Hi-Level** programming languages, i.e. language for humans, eg. *ADA, C, C++, Perl, Python, Java, Pascal, Basic*.
- Computers, at hardware level, understand only a very trivial set of commands, the *Assembly*, a **Low-Level** programming language, a language for CPUs.

Hi-Level languages

BASIC:

```
10 INPUT "Your name?: ", NAME$
20 PRINT "Hello "; NAME$
```

Python:

```
name=input("Your name?: ")
print("Hello",name)
```

C:

```
#include <stdio.h>
char * name[100];
int main() {
   printf("Your name?: ");
   scanf("%s",name);
   printf("Hello %s\n", name);
   return 0;
   }
```

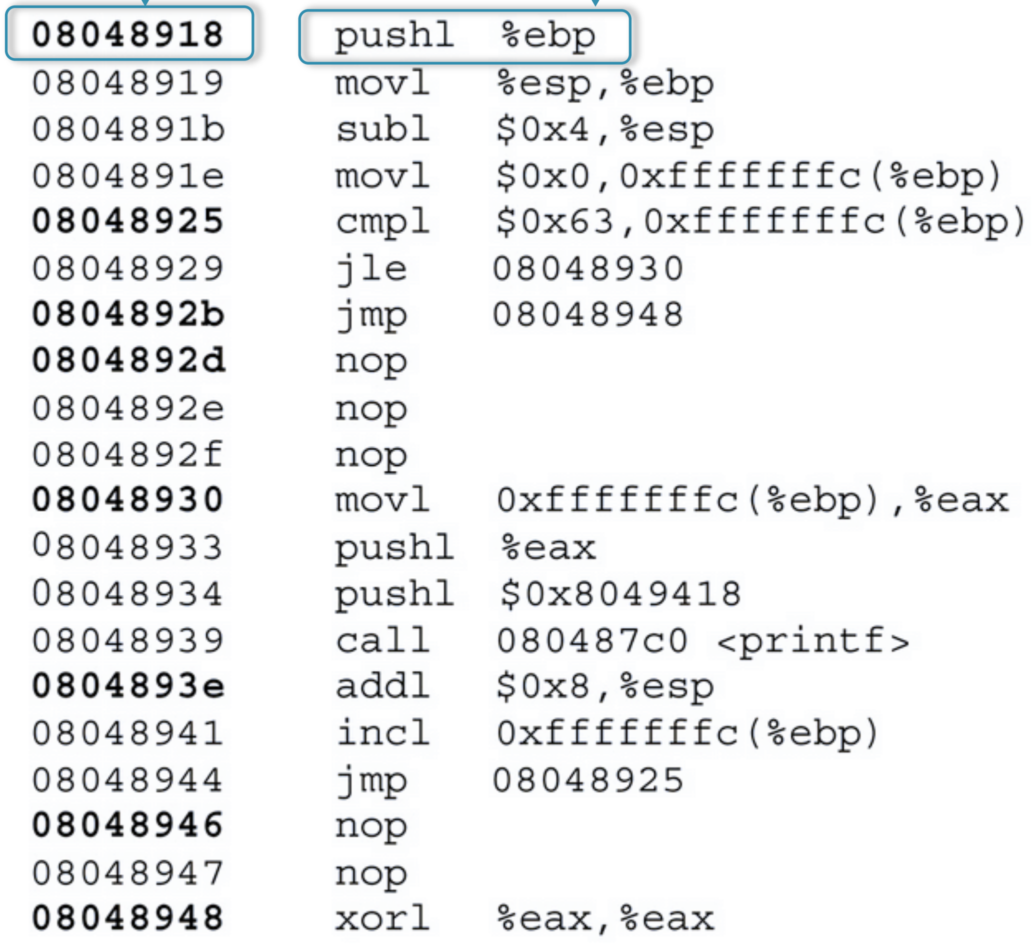
Java:

```
package stringvariables ;
import java.util.Scanner;
public class StringVariables {
   Scanner user_input = new Scanner( System.in );
   String name;
   System.out.print("Your name?");
   name = user_input.next( );
   System.out.print("Hello "+name);
   }
```

Assembly

instruction in memory used by CPU

instruction transliterated for humans



```
08048918    pushl   %ebp
08048919    movl    %esp,%ebp
0804891b    subl    $0x4,%esp
0804891e    movl    $0x0,0xfffffffc(%ebp)
08048925    cmpl    $0x63,0xfffffffc(%ebp)
08048929    jle     08048930
0804892b    jmp     08048948
0804892d    nop
0804892e    nop
0804892f    nop
08048930    movl    0xfffffffc(%ebp),%eax
08048933    pushl   %eax
08048934    pushl   $0x8049418
08048939    call    080487c0 <printf>
0804893e    addl    $0x8,%esp
08048941    incl    0xfffffffc(%ebp)
08048944    jmp     08048925
08048946    nop
08048947    nop
08048948    xorl    %eax,%eax
```

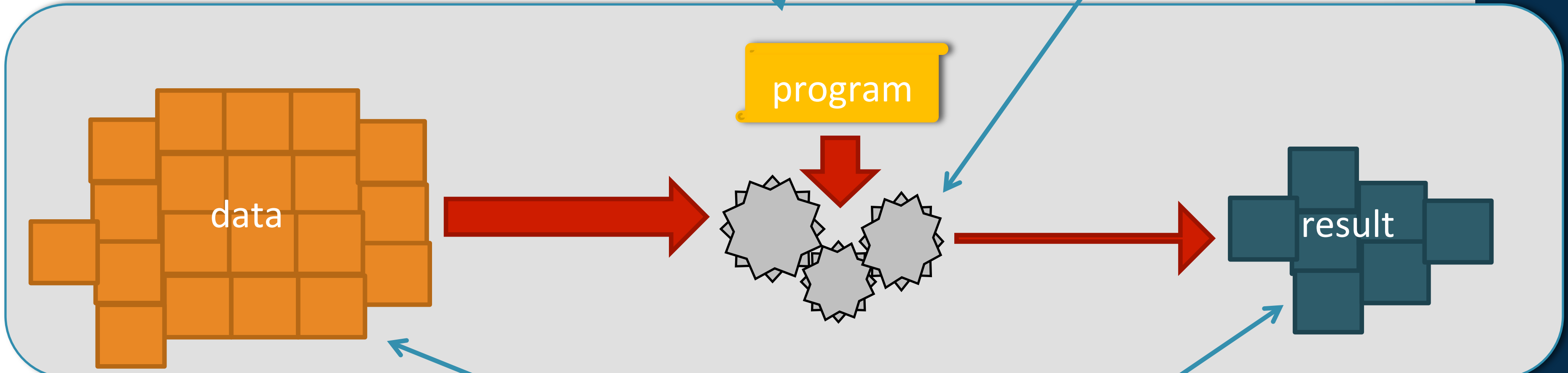
https://www.researchgate.net/figure/Assembly-instructions-of-an-x86-example-optimizing-frequently-executed-pieces-of-code_fig2_3881320

Use computers? start problems!

https://www.youtube.com/watch?v=tiq6v39YliQ

- Code readibility
- **Code maintenance**
- Code estensibility

- speed?
- cost!



- **Data management**
- Portability

Develope Code: a job for teams

- Code should (must?) be:
  - **readable**: projects pass thorugh many hands and may live, from change to change, for many years
  - **easy to develope**:
    - **easy syntax** → fast learning
    - **not error-prone**: syntax should aid good programming
  - **with a lot of already made wheels:** a wide library collection of good functions aid to build up good code rapidly (*dont reinvent the wheel*)
  - **Cool**: a large connected community of geeks that code with your programming language probably have already solved all of your possible problems.

Speed

- Speed generally conflicts with code maintenance.
- Fast codes in order to full control the flow of the istructions (usually):
  - is coded using a "raw" programming language (eg. C,C++) thus it result often unreadable.
  - it don't use "abstractions" for implementing algorithm and managing data thus it bacame easy to make mistakes and bugs
  - libraries are implemented from scratch in order to optimize code or remove unused part of code, thus "new code, new bugs".

"Don't run if there is not needs"

Interpreter vs Compiler

- The process of translate from HI to Low Level can be made in two way: translate the program with a **compiler** o execute the program with an **interpreter**
- Compilers:
  - take a lot of time for compile phase but the result*, the executable*, run fast on CPU.
  - Any new release of the code have to be compiled again
  - there no easy ways to run the code step by step for test (you have to use a *debugger*)
- Interpreters:
  - designed for interactive mode: easy to debug code
  - code is executed by an agent, not directly by CPU
    - easy to *port* to new kind of computer
  - Not so fast: each line have to be translated anytime is executed

Speed

C

```
char*
aword=malloc(typeof(char)*10);
scanf("%s",aword);
for (i=0;strlen(aword);i++){
        printf("%c\n",aword[i]);
}
free(aword);
```

+ fast: compiled for the running CPU
+ small binary
- unreadable
- memory mgmt is our duty
- easy to make mistakes on syntax

python

```
aword=input()
for c in aword:
        print(c)
```

+easy to undestand

+easy to find errors

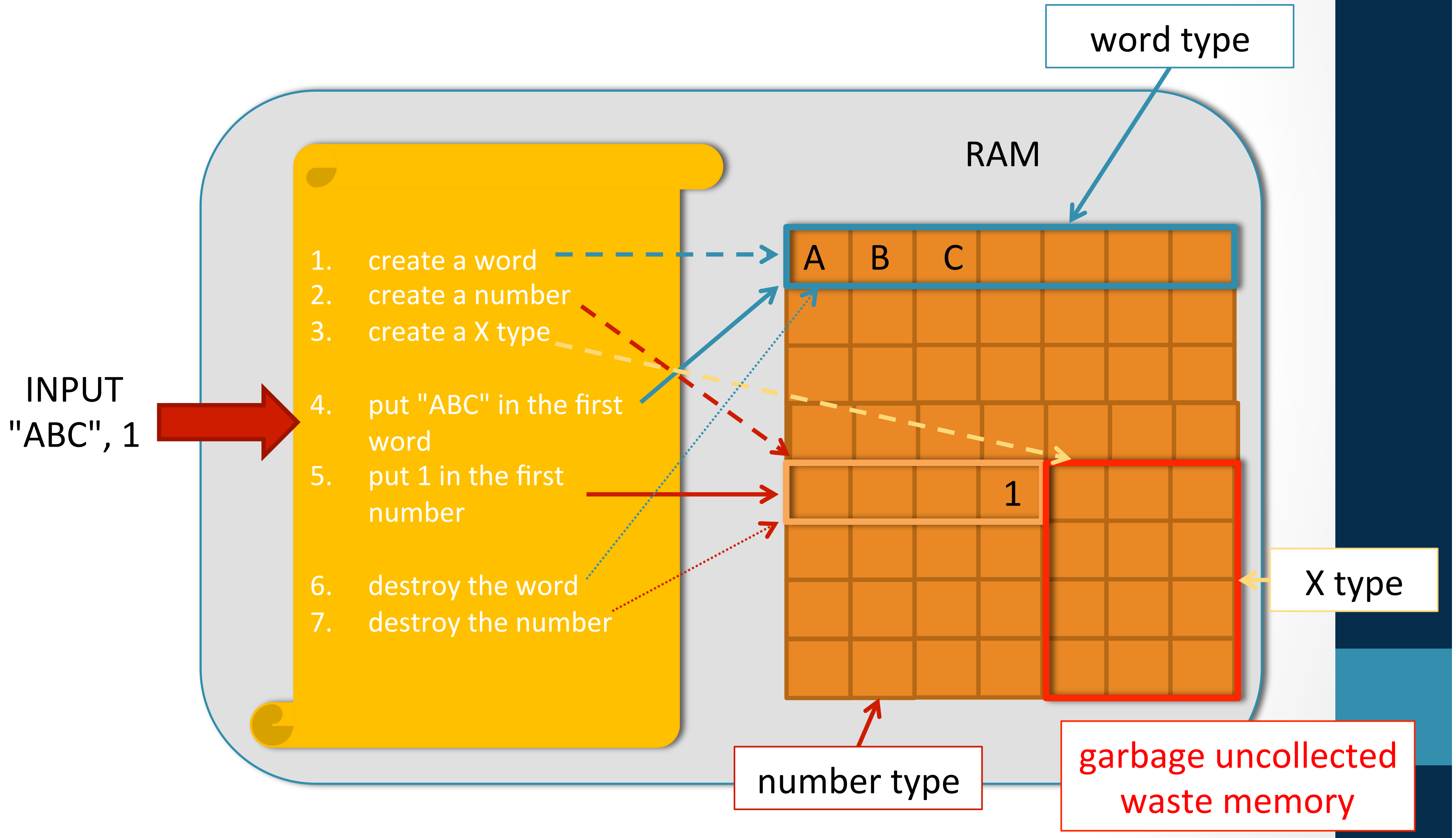
+memory mgmt is delagated to system

-not so fast: managing object requires a background process that sink some cpu time, it is interpreted.

Speed constrains

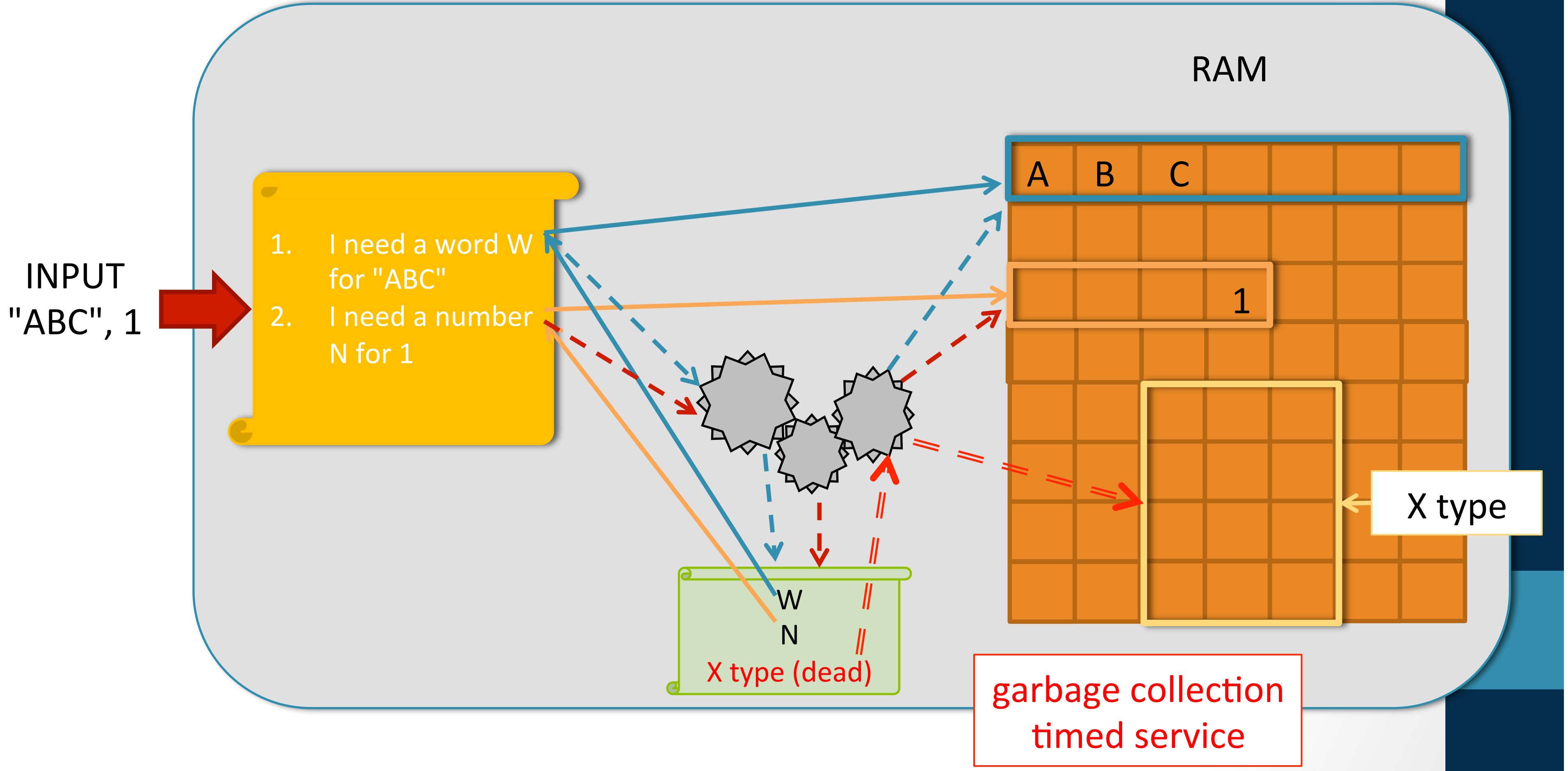
- Speed depends mainly:
  - data management:
    - how objects for data are create and, more importants, destroyed.
    - how access to data is made respect to the layered cached memory
  - CPU parallelism:
    - modern CPUs are **superscalar**: can do many steps at the same time, concurrently, if the code permits it.

Data management
a *do-it-yourself* view (*C style*)



Data management

a *data-as-service* view (*Java style*)



RAM
INPUT
"ABC", 1
1. I need a word W for "ABC"
2. I need a number N for 1
A
B
C
1
X type
W
N
X type (dead)
garbage collection timed service

Python spec

- General purpouse language
- Focused on readability
- Interpreted
- Modular
- Dynamic
- Object-oriented
- Portable
- Extensible in C++ & C

Snakify

- Snakify is a platform for e-Learning Python 3
    - Connect to https://snakify.org/
    - Sign up using
        - your @unimi.it email as username (dont use your private email,if possible)
        - a password **DIFFERENT** from the ona used for email
    - flag the option "**I have a teacher**"
    - put "massimo.marchi@unimi.it" in the field "**Teacher's email**"