Coding in Python

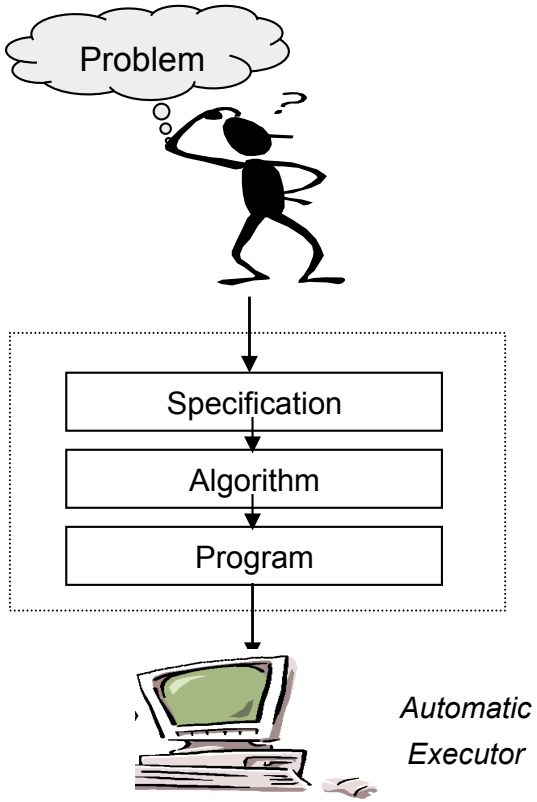
Gianluca Palermo – gianluca.palermo@polimi.it

Target

- To Learn the basic concepts behind computer programming
  - using Python as programming language
- To Learn how to solve simple problems using a computer

- **To develop a simple Videogame!**

Coding, What's this?



- Use of the computer to simplify some data manipulation tasks

- Solution of the problem in a way that has been adapted to an automatic solution

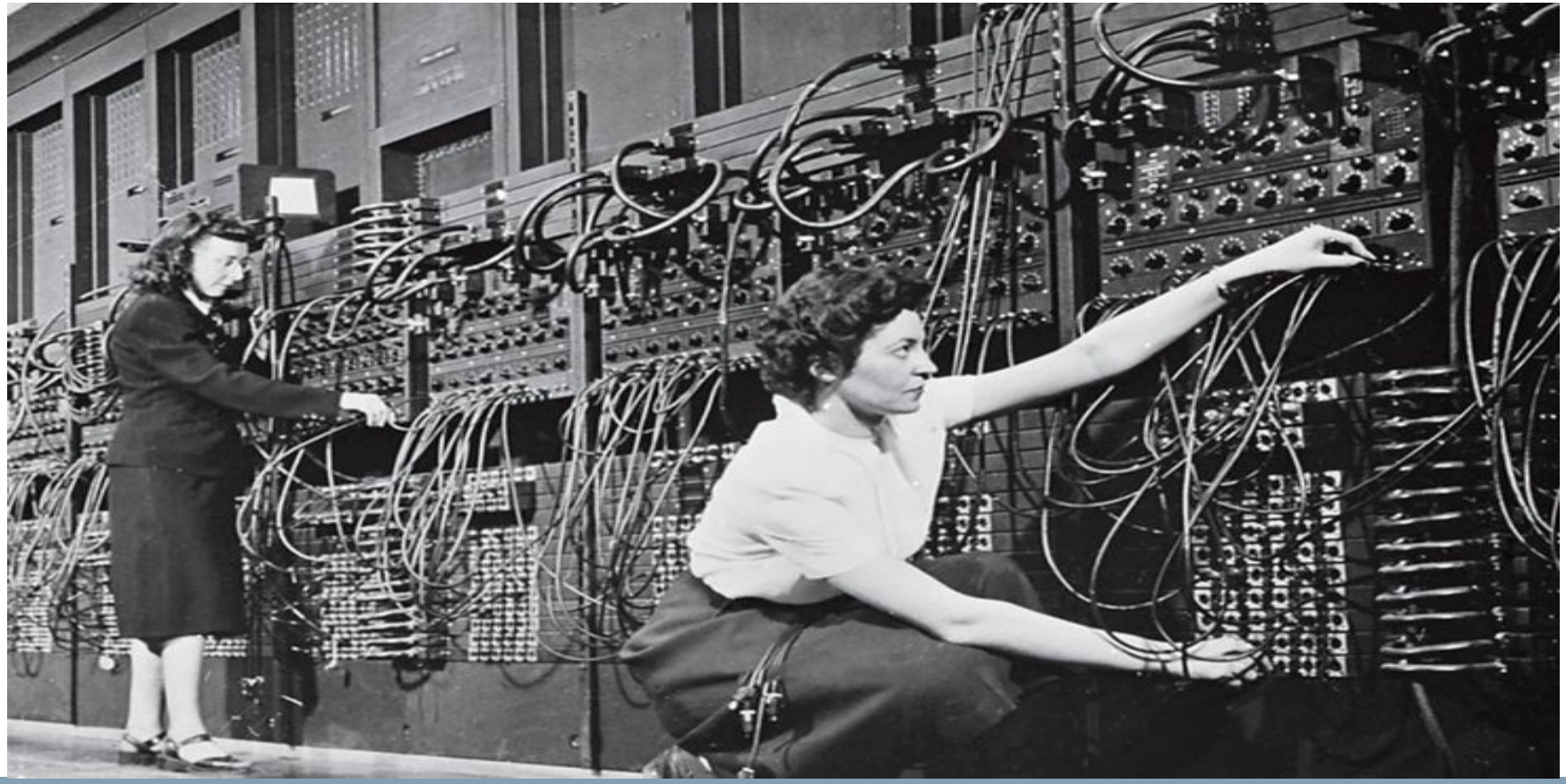
- Description of the algorithm in a way that can be understood by the automatic executor (Computer)

Computer History: From application specific machines… (one Task)

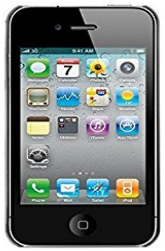
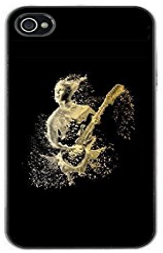
By Alessandro Nassiri - Museo della Scienza e della Tecnologia "Leonardo da Vinci",
CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=47910919

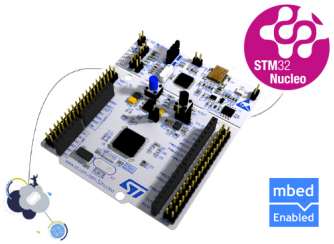
The Enigma machine is an encryption device developed and used in the early- to mid-20th century to protect commercial, diplomatic and military communication. It was employed extensively by Nazi Germany during World War II, in all branches of the German military.

…to programmable machines …

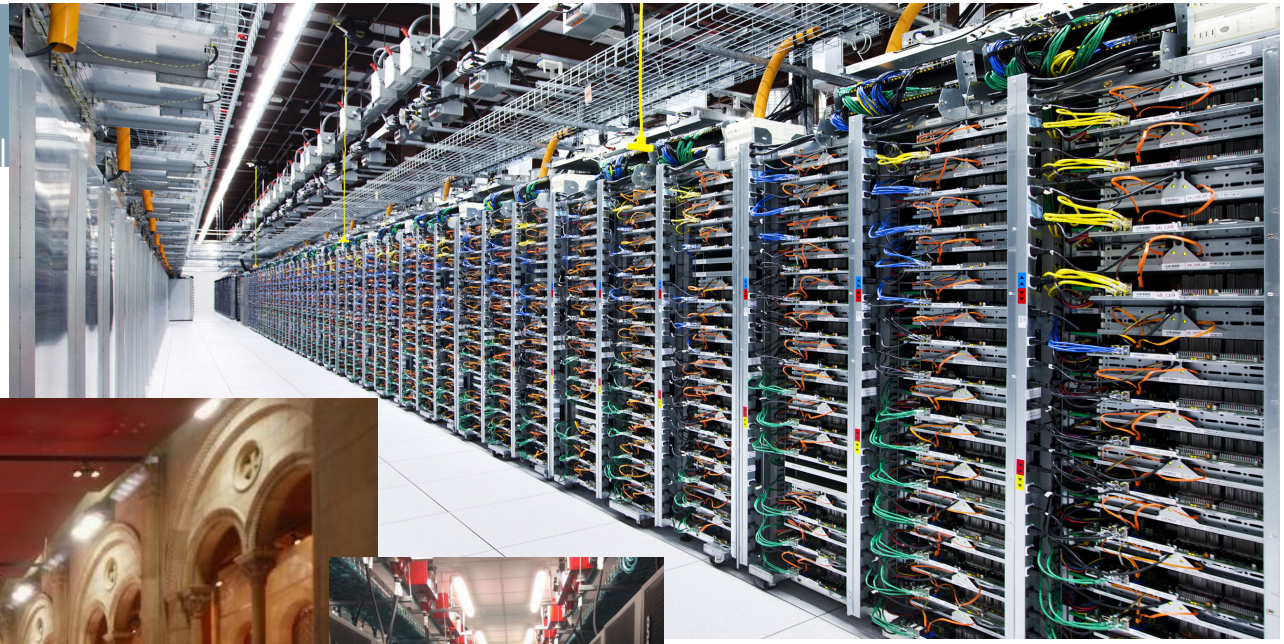


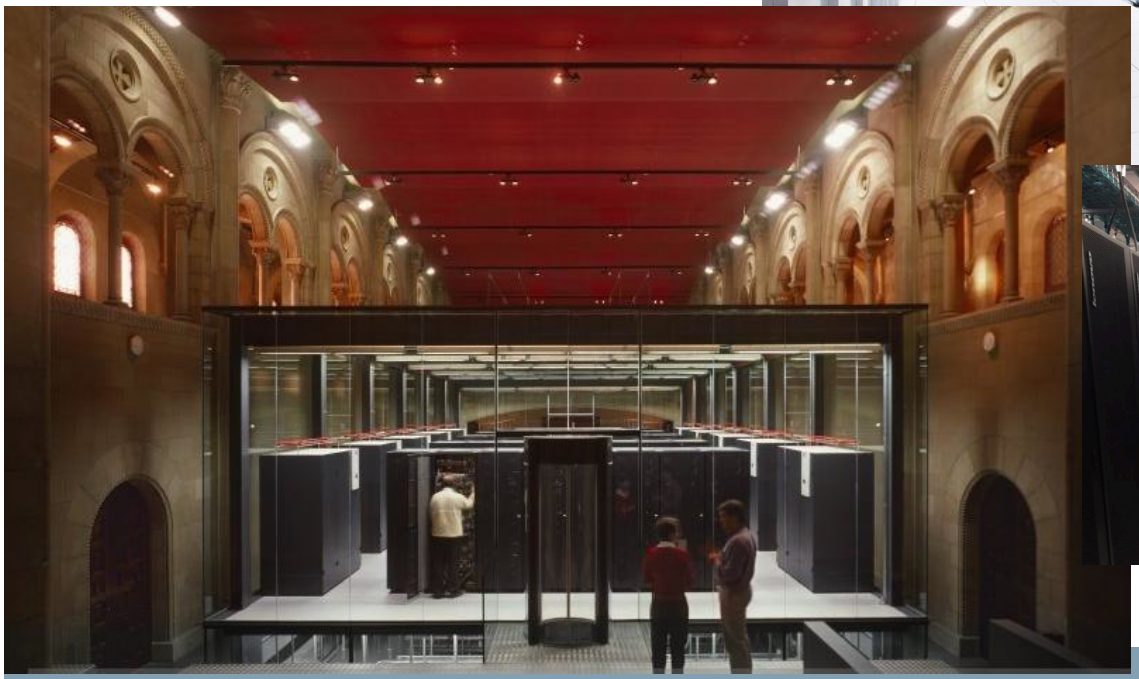
… easier to be managed …



… or not ☺





https://www.youtube.com/watch?v=avP5d16wEp0

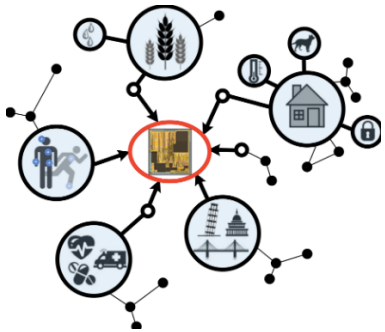
Connected and changing world…

We use every day “connected” computers.
However, The **USER** is not the **PROGRAMMER**

Why be a Programmer?

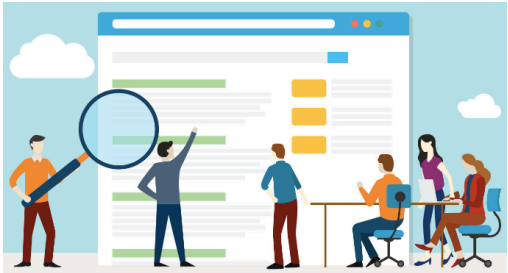
To get some task done automatically

- Acting as user and programmer
- E.g. Processing collected sensor data

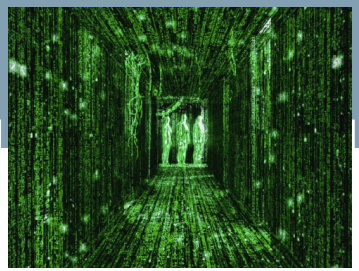


To produce something for others to use

- This is the task for a programmer as a job
- Fix a performance problem in a videogame
- Add a guestbook to a web site



What is Code? Software? A Program?



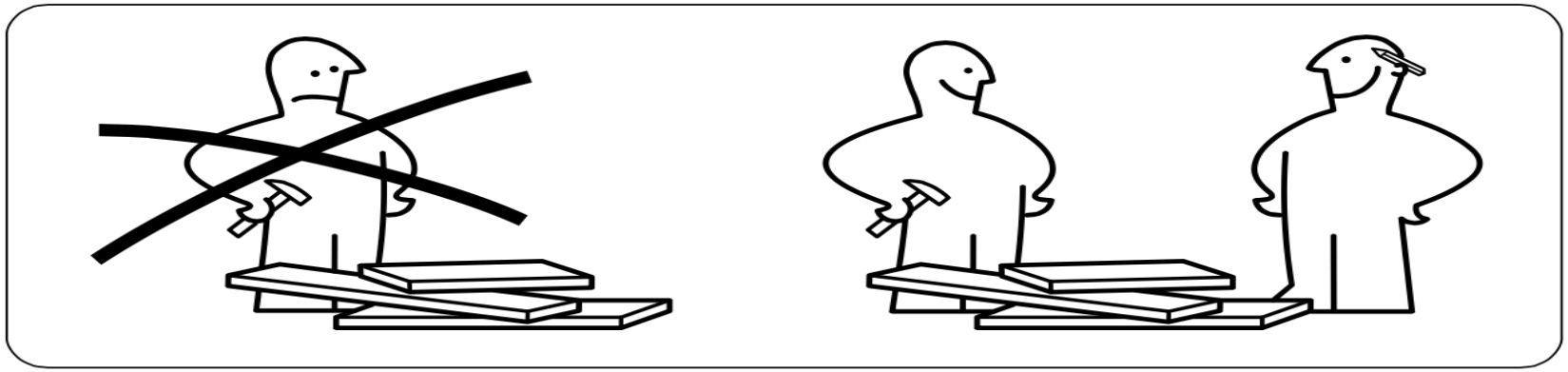
A sequence of instructions

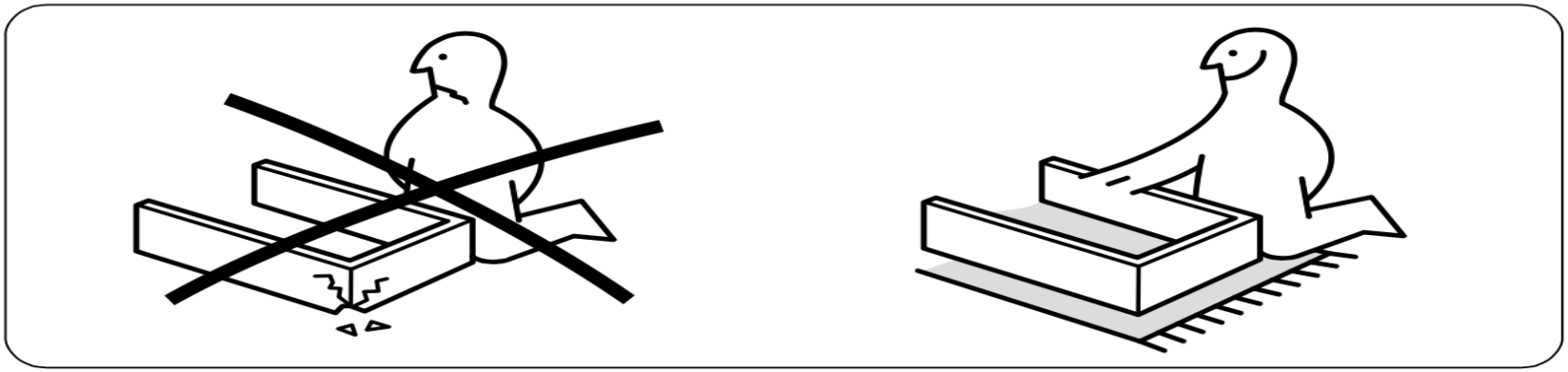
- It is a little piece of our intelligence in the computer
- We figure something out and then we encode it and then give it to someone else to save them the time and energy of figuring it out

A piece of creative art - particularly when we do a good job on user experience

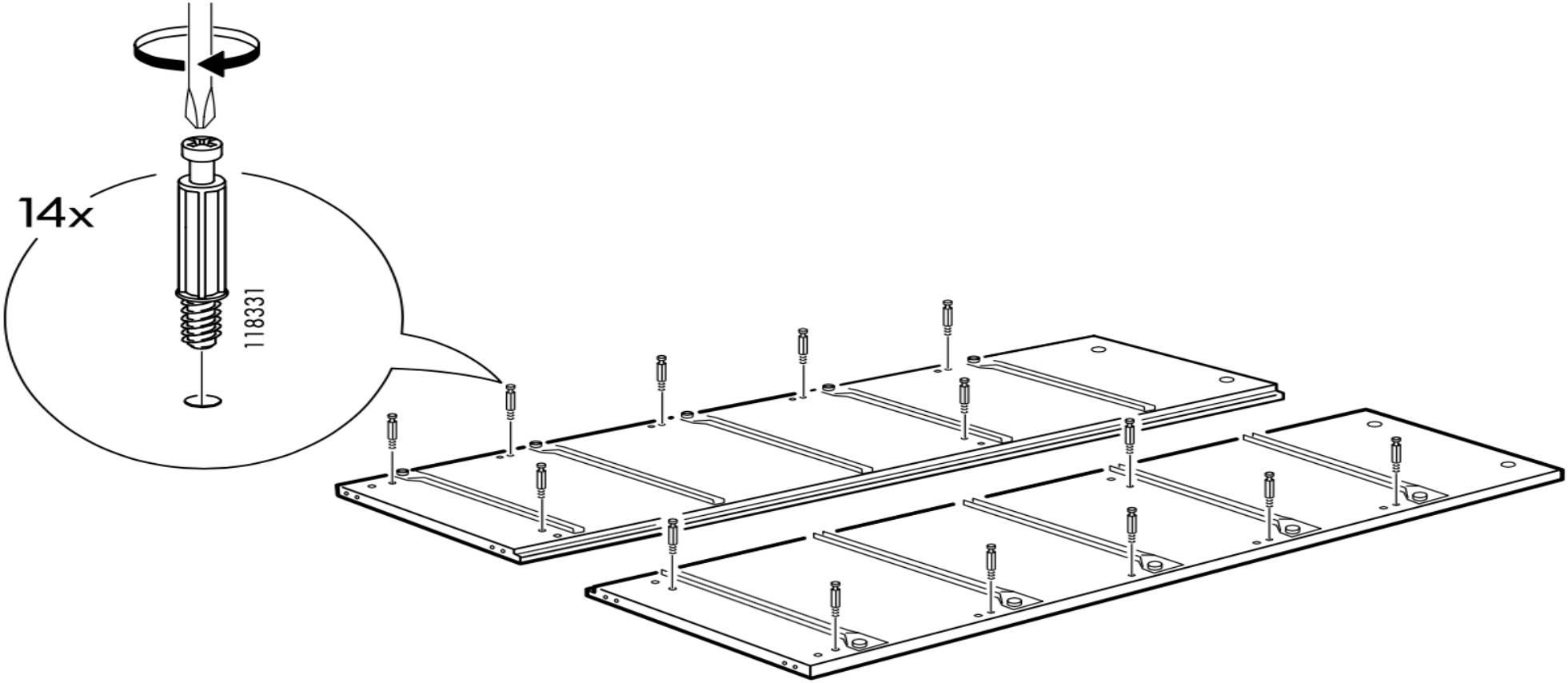
- Unfortunately the creativity of a programmer is not always exposed to the user. However, we know that it is there!

Programs for Humans





Programs for Humans



Programs for Humans

Multiplication by successive sums

ACC= M1*M2

```
Read M1 and M2
Initialize an accumulator to zero (ACC=0)
Repeat |M2| times
     ACC=ACC+M1
If M2 is < 0
     change the sign to ACC
Write ACC
```

Long Polynomial Division

Here are the steps required for Dividing by a Polynomial Containing More Than One Term (Long Division):

Step 1:	Make sure the polynomial is written in descending order. If any terms are missing, use a zero to fill in the missing term (this will help with the spacing).
Step 2:	Divide the term with the highest power inside the division symbol by the term with the highest power outside the division symbol.
Step 3:	Multiply (or distribute) the answer obtained in the previous step by the polynomial in front of the division symbol.
Step 4:	Subtract and bring down the next term.
Step 5:	Repeat Steps 2, 3, and 4 until there are no more terms to bring down.
Step 6:	Write the final answer. The term remaining after the last subtract step is the remainder and must be written as a fraction in the final answer.

Python programs … an example: “Is it a PANGRAM?”

```
text = input('Write a sentence: ')

Alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',
            'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r',
            's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

notAPangram = False

for letter in Alphabet:
    if letter not in text:
        notAPangram = True

if notAPangram:
    print('the string is not a pangram')
else:
    print('the string is a pangram')
```

A PANGRAM is a sentence including all the alphabet letters

Exercise

Let’s write a program/algorithm that using the following material puts together a sandwich

Algorithms, programs, executors and programming languages

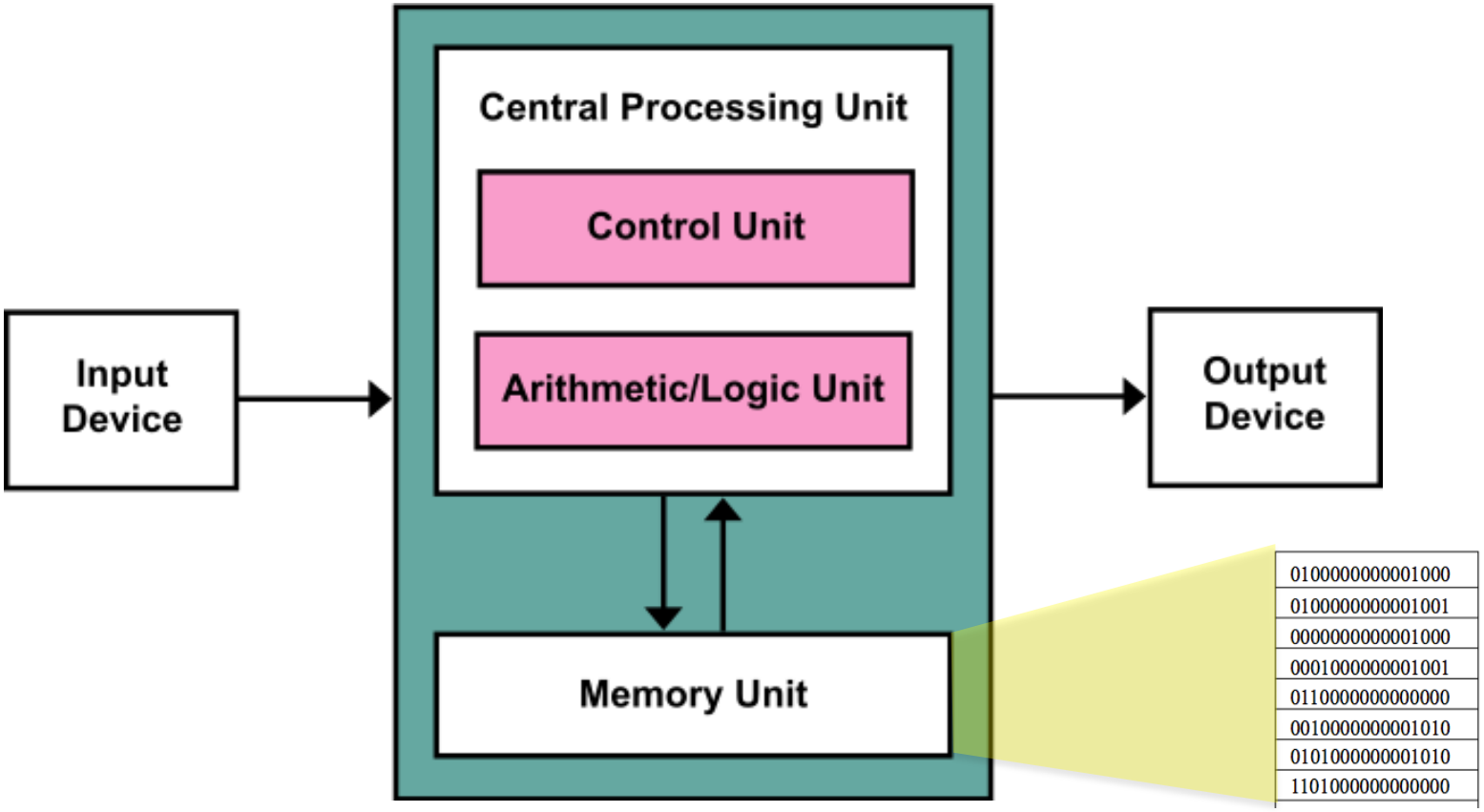
An algorithm is a sequence of **unambiguous instructions** for solving a problem in a finite amount of time

A program is the encoding of the algorithm by using a programming language with the goal of being understood by the executor (the computer)

What is the language that can be understood by the computer?

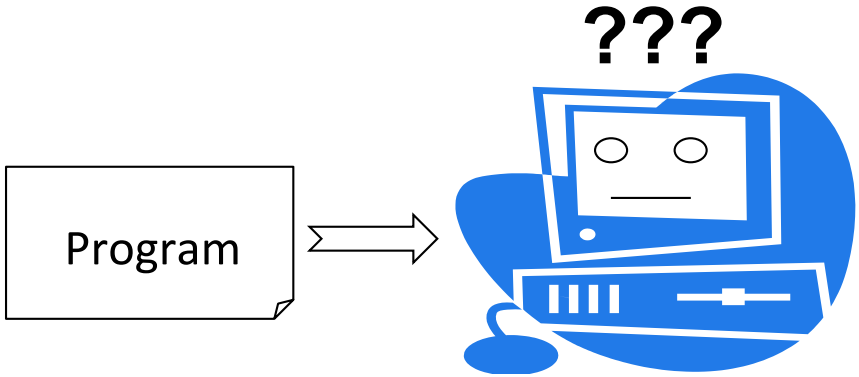
First of all, what do we mean as a computer?

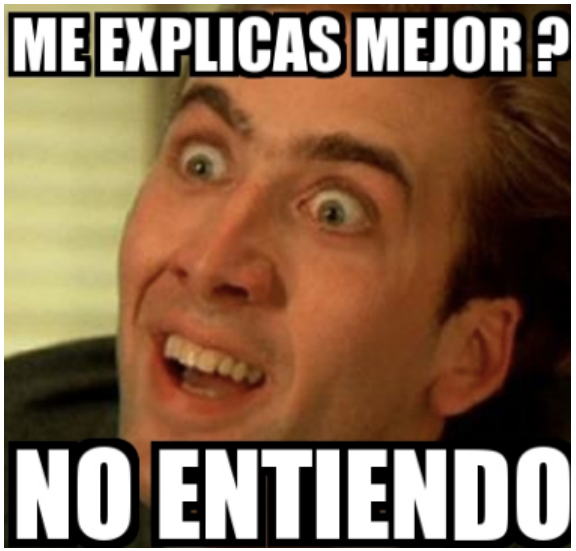
Von Neumann architecture



Including Data and Instructions

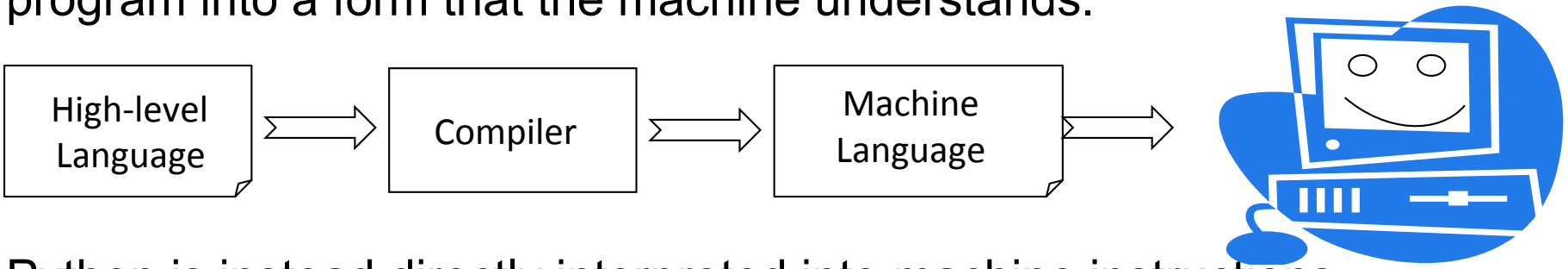
Compiling and interpreting



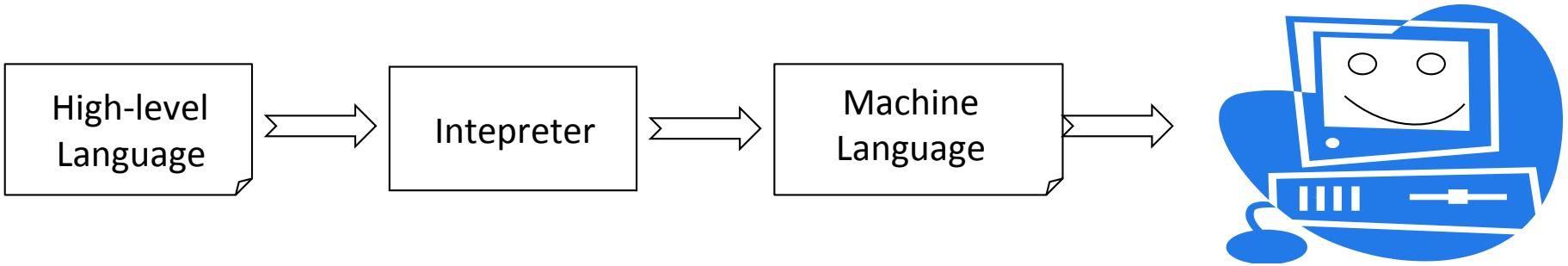


Compiling and interpreting

Many languages (e.g. C/C++) require you to compile (translate) your program into a form that the machine understands.



Python is instead directly interpreted into machine instructions.



Programming environments

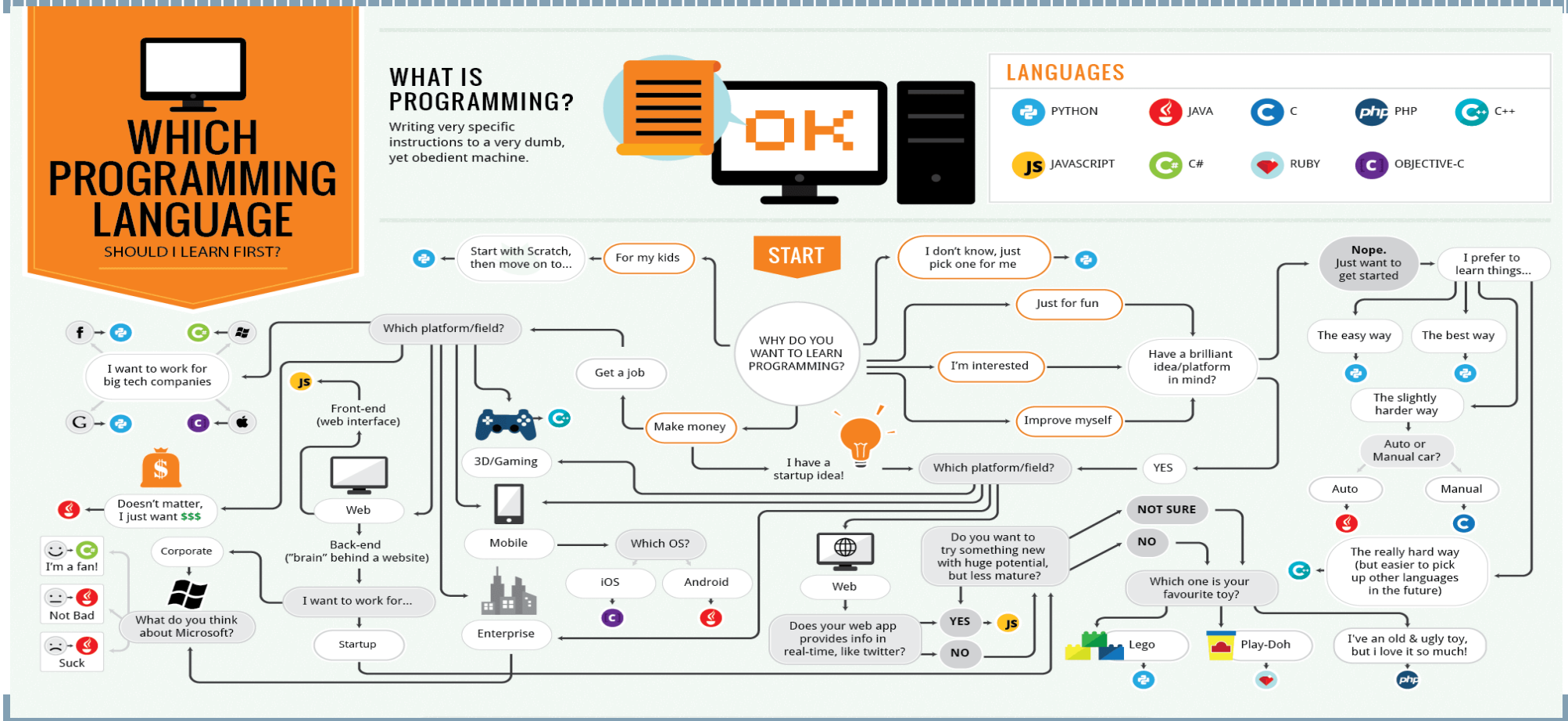
An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development

An IDE normally consists of:

- A source code editor to easily write code
- A compiler or interpreter, depending on the target language
- A syntax checker (including syntax highlighting)
- A debugger to control the correctness of the program execution

Python as Programming Language

Languages


WHICH PROGRAMMING LANGUAGE
SHOULD I LEARN FIRST?
WHAT IS PROGRAMMING?
Writing very specific instructions to a very dumb, yet obedient machine.
OK
LANGUAGES
PYTHON
JAVA
C
PHP
C++
JAVASCRIPT
C#
RUBY
OBJECTIVE-C
START
WHY DO YOU WANT TO LEARN PROGRAMMING?
For my kids
Start with Scratch, then move on to...
I don't know, just pick one for me
Just for fun
I'm interested
Improve myself
Have a brilliant idea/platform in mind?
Nope. Just want to get started
I prefer to learn things...
The easy way
The best way
The slightly harder way
Auto or Manual car?
Auto
Manual
The really hard way (but easier to pick up other languages in the future)
YES
Which platform/field?
I have a startup idea!
Make money
Get a job
Which platform/field?
I want to work for big tech companies
Front-end (web interface)
Web
Back-end ("brain" behind a website)
I want to work for...
Startup
Doesn't matter, I just want $$$
3D/Gaming
Mobile
Which OS?
iOS
Android
Enterprise
Corporate
What do you think about Microsoft?
I'm a fan!
Not Bad
Suck
Web
Does your web app provides info in real-time, like twitter?
YES
NO
Do you want to try something new with huge potential, but less mature?
NOT SURE
NO
Which one is your favourite toy?
Lego
Play-Doh
I've an old & ugly toy, but i love it so much!

Languages

THE LORD OF THE RINGS ANALOGY TO PROGRAMMING LANGUAGES

Python
The Ent

DIFFICULTY ★☆☆☆☆

Help little Hobbits (beginners) to understand programming concepts

Help Wizards (computer scientists) to conduct researches

Widely regarded as the best programming language for beginners

Easiest to learn

Widely used in scientific, technical & academic field, i.e. Artificial Intelligence

You can build website using Django, a popular Python web framework

POPULARITY ★★★★½

USED TO BUILD YouTube, Instagram, Spotify

AVG. SALARY $107,000

Java
Gandalf

DIFFICULTY ★★★☆☆

Wants peace & works with everyone (portable)

Very popular on all platforms, OS, and devices due to its portability

One of the most in demand & highest paying programming languages

Slogan: write once, work everywhere

POPULARITY ★★★★★

USED TO BUILD Gmail, Minecraft, Most Android Apps, Enterprise applications

AVG. SALARY $102,000

C
One Ring

DIFFICULTY ★★★☆☆

The power of C is known to them all

Everyone wants to get its Power

Lingua franca of programming language

One of the oldest and most widely used language in the world

Popular language for system and hardware programming

A subset of C++ except the little details

POPULARITY ★★★★★

USED TO BUILD Operating systems and hardware

AVG. SALARY $102,000

C++
Saruman

DIFFICULTY ★★★★☆

Everyone thinks that he is the good guy

But once you get to know him, you will realize he wants the power, not good deeds

Complex version of C with a lot more features

Widely used for developing games, industrial and performance-critical applications

Learning C++ is like learning how to manufacture, assemble, and drive a car

Recommended only if you have a mentor to guide you

POPULARITY ★★★★½

USED TO BUILD Operating systems, hardware, and browsers

AVG. SALARY $104,000

JavaScript
Hobbit

DIFFICULTY ★★☆☆☆

Frequently underestimated (powerful)

Well-known for the slow, gentle life of the Shire (web browsers)

"Java and Javascript are similar like Car and Carpet are similar" - Greg Hewgill

Most popular clients-side web scripting language

A must learn for front-end web developer (HTML and CSS as well)

One of the hottest programming language now, due to its increasing popularity as server-side language (node.js)

POPULARITY ★★★★☆

USED TO BUILD Paypal, front-end of majority websites

AVG. SALARY $99,000

C#
Elf

DIFFICULTY ★★★☆☆

Beautiful creature (language), but stays in their land, Rivendell (Microsoft Platform)

A popular choice for enterprise to create websites and Windows application using .NET framework

Can be used to build website with ASP.NET, a web framework from Microsoft

Similar to Java in basic syntax and some features

Learn C# instead of Java if you are targeting to work on Windows platform only

POPULARITY ★★★★½

USED TO BUILD Enterprise and Windows applications

AVG. SALARY $94,000

Ruby
Man (Middle Earth)

DIFFICULTY ★★☆☆☆

Very emotional creature

They (some Ruby developers) feel they are superior & need to rule the Middle Earth

Mostly known for its popular web framework, Ruby on Rails

Focuses on getting things done

Designed for fun and productive coding

Best for fun and personal projects, startups, and rapid development

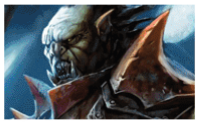
POPULARITY ★★★½☆

USED TO BUILD Hulu, Groupon, Slideshare

AVG. SALARY $107,000

PHP
Orc

DIFFICULTY ★★☆☆☆



Ugly guy (language) and doesn't respect the rules (inconsistent and unpredictable)

Big headache to those (developers) to manage them (codes)

Yet still dominates the Middle-earth (most popular web scripting language)

Suitable for building small and simple sites within a short time frame

Supported by almost every web hosting services with lower price

POPULARITY ★★★★☆

USED TO BUILD Wordpress, Wikipedia, Flickr

AVG. SALARY $89,000

Objective-C
Smaug

DIFFICULTY ★★★☆☆

Lonely and loves gold

Primary language used by Apple for Mac OS X & iOS

Choose this if you want to focus on developing iOS or OS X apps only

Consider to learn Swift (newly introduced by Apple in 2014) as your next language

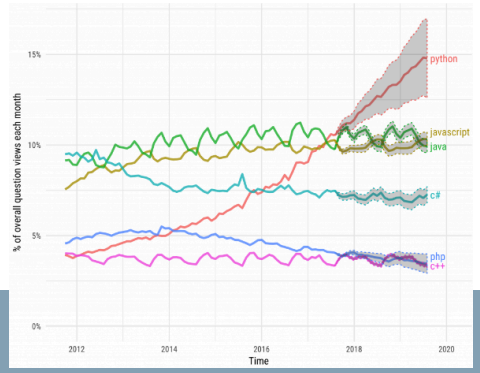
POPULARITY ★★★☆☆

USED TO BUILD Most iOS Apps and part of Mac OS X

AVG. SALARY $107,000

Why Python…

- Python is an ***easy-to-learn*** programming language that has some really useful features for a beginning programmer
- The code is quite *easy* ***to read and more natural to write*** when compared to other programming languages
- Python has some characteristics that make the learning process easier, giving the possibility to develop an animation or a simple ***videogame in less than a week***!
- Python is the ***fastest-growing programming language***



Early Learner: Advises

We need to learn the Python language, thus

- We will “speak” gibberish like small children
- We will make lots of mistakes

When you make a mistake

- The computer says “syntax error”
- It seems like Python is cruel and unfeeling

You must remember

- You are intelligent and can learn
- The computer is simple and very fast, but “cannot learn”
  - at least for the moment ;-)

It is easier for you to learn Python than for the computer to learn English...

Elements of Python

Vocabulary / Words

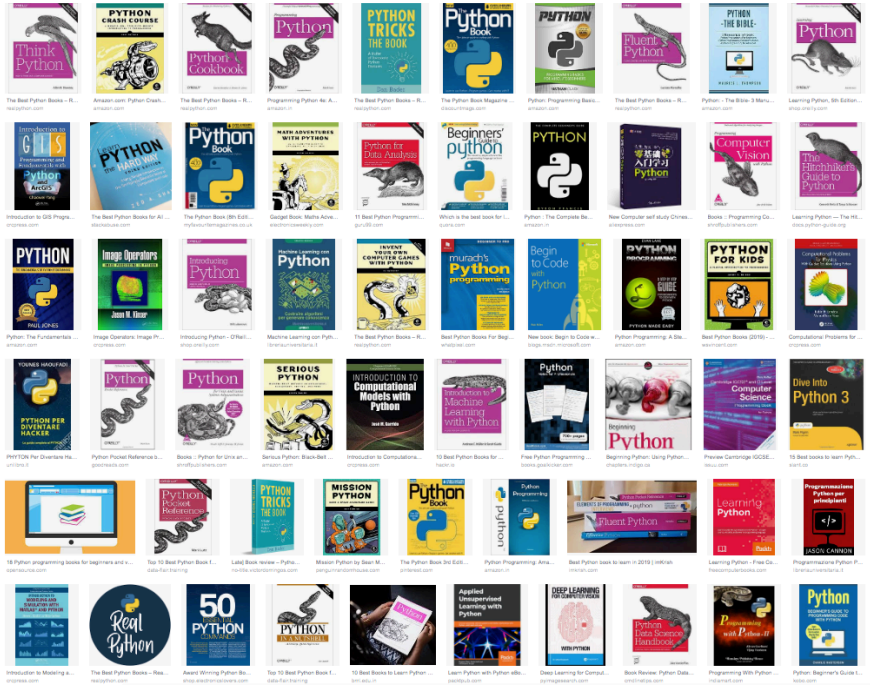
- Variables
- Reserved words

Sentence structure

- Valid syntax patterns

Story structure structure

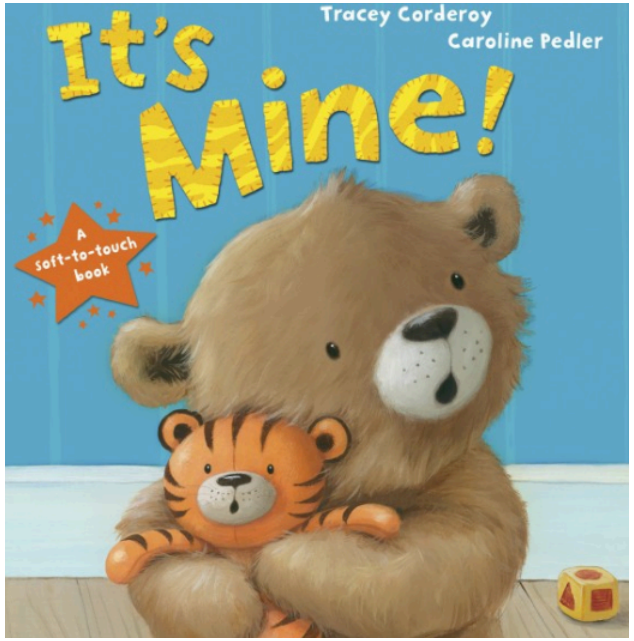
- constructing a program for a purpose



Vocabulary: Reserved Words

Reserved words are words with a specific mean in Python

```
False   class   return  is      finally
None    if      for     lambda  continue
True    def     from    while   nonlocal
and     del     global  not     with
as      elif    try     or      yield
assert  else    import  pass
break   except  in      raise
```



Vocabulary: Variables

- A variable is a named place in the memory where a programmer can store data and later retrieve the data using the variable “name”
- Programmers get to choose the names of the variables
- You can change the contents of a variable in a later statement

```
x = 12.2
y = 14
```

x	12.2
y	14

Vocabulary: Variables

- A variable is a named place in the memory where a programmer can store data and later retrieve the data using the variable “name”
- Programmers get to choose the names of the variables
- You can change the contents of a variable in a later statement

```
x = 12.2
y = 14
x = 100
```

x	~~12.2~~ 100
y	14

Sentences: Instructions

```
x = 2
x = x + 2
print(x)
```

← Assignment statement

← Assignment with expression

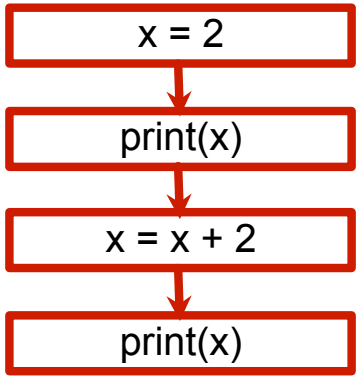
← Print statement

Variable Operator Constant Function

Program Steps or Program Flow

- Like a recipe or installation instructions, a program is a sequence of steps to be done in order.
- Some steps are conditional - they may be skipped.
- Sometimes a step or group of steps is to be repeated.
- Sometimes we store a set of steps to be used over and over as needed several places throughout the program (these are functions).

Sequential Steps



Program:

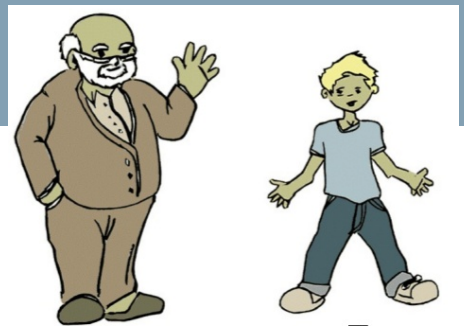
```
x = 2
print(x)
x = x + 2
print(x)
```

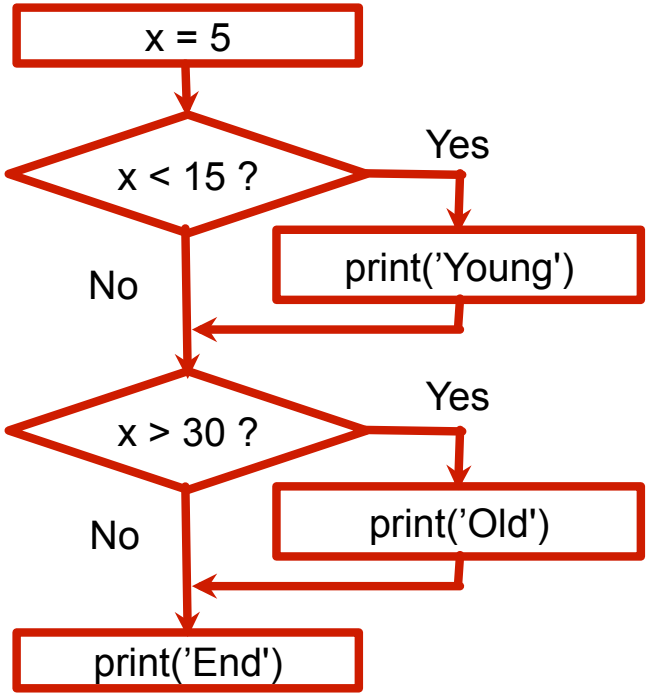
Output:

```
2
4
```

When a program is running, it flows from one step to the next. As programmers, we set up “paths” for the program to follow.

Conditional Steps





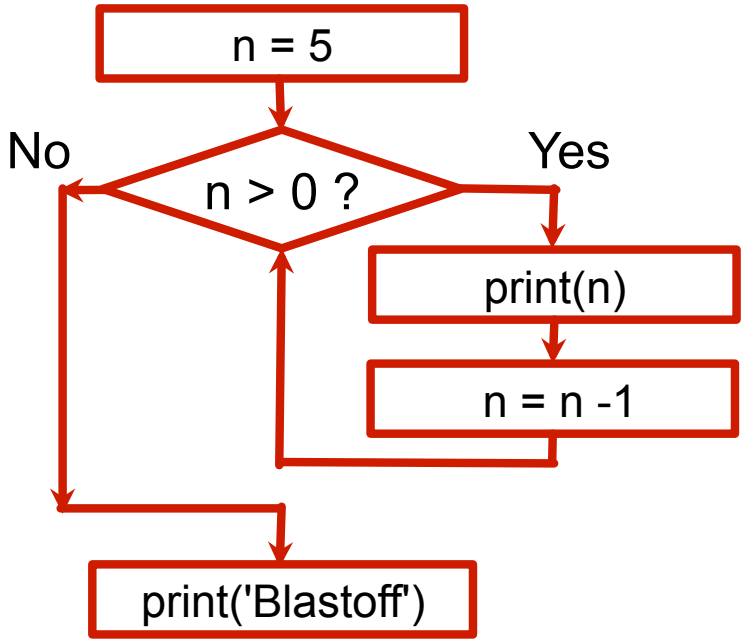
Program:

```
x = 5
if x < 10:
    print('Young')
if x > 20:
    print('Old')

print('End')
```

Output:

Young
End

Repeated Steps



Program:

```
n = 5
while n > 0 :
    print(n)
    n = n – 1
print('Blastoff!')
```

Output:

5
4
3
2
1
Blastoff!

Loops (repeated steps) have iteration variables that change each time through a loop.

Classifying the instructions of our first example…

```
text = input('Write a sentence: ')

Alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',
            'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r',
            's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

notAPangram = False

for letter in Alphabet:
    if letter not in text:
        notAPangram = True

if notAPangram:
    print('the string is not a pangram')
else:
    print('the string is a pangram')
```

Sequential

Repeated

Conditional

Words, sentences and paragraphs

```
text = input('Write a sentence: ')

Alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',
            'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r',
            's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

notAPangram = False

for letter in Alphabet:
    if letter not in text:
        notAPangram = True

if notAPangram:
    print('the string is not a pangram')
else:
    print('the string is a pangram')
```

A short Python “Story” about how identifying pangrams

A word used to read data from a user

A sentence about considering all alphabet letters

A paragraph about informing the usage of the program outcome

… We have almost done!!!

CODING - TechCamp@POLIMI

Prof. Gianluca Palermo - Tutors: Davide Gadioli, Andrea Damiani

Introduction

Like humans, computers use multiple languages (we call them programming languages). A programming language is a particular way to talk to a computer, with the goal of describing the task to do. There are a lot of programming languages, probably too many ;-), each one with a different peculiarity. It does not exist the best programming language. During the CODING course at the techCamp, we are using Python.

Python is an easy-to-learn programming language that has some really useful features for a beginning programmer. The code is quite easy to read and more natural to write when compared to other programming languages. Moreover, Python has some characteristics that make the learning process easier, giving the possibility to develop an animation or a simple videogame in less than a week (This is great for the TechCamp!).

Even if you are attracted from building immediately your first videogame, it is always the best approach to start with the basics. I prepared some short lesson that you have to attend, read and elaborate, before going to more complicated (and nicer) stuff. Try to solve the proposed exercise, to improve your confidence with the language and with the algorithmic way to solve problems. Always remember that the better you understand the basics, the easier it will be to use them in complicated stuff later on, such as videogames.

Don't think of these lessons as something boring that you have to work on. Try always to have fun in what you are doing. Try, Try and Try. Try, whatever you have in mind by using what you are learning day-by-day. Think of programming as a way to create something. Be creative, and remember that your applications and the final videogame can be shared with your friends.

Remember, ask the professor and the tutors. We are here to help you. Sometimes, it can happen that we do not have immediately the answer. However, we will find a way to find it.

Installing Python

Installing Python is generally easy, and nowadays many Linux and MacOS distributions include a recent Python version. Even some Windows computers now come with Python already installed. However, if you do not have python installed it is enough to go to the official Python webpage https://www.python.org/downloads/ (https://www.python.org/downloads/) and download the most recent version. In this course we are using Python 3.x, and not Python 2.x. If you are not confident about the installation process, ask the Professor and the tutors.

POLITECNICO
MILANO 1863

Input and Output in Python

print(SOMETHING) … SOMETHING

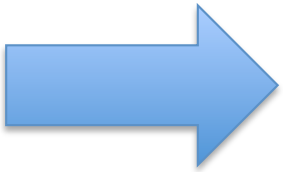
```
print("CIAO")
```

```
CIAO
```

```
print(5)
```

```
5
```

```
print( 5 + 10 )
print( 47 - 12 )
print( 15 / 2 )
print( 9 * 4 )
```



```
15
35
7.5
36
```

print(SOMETHING) … SOMETHING (2)

```
print( 2 ** 16 )
print( 37 // 3 )
print( 37 % 3 )
```

```
65536
12
1
```

```
# The following code prints the result of 3*9
print(3*9)
#print("CIAO!")
```

```
27
```

This is a comment

Variables

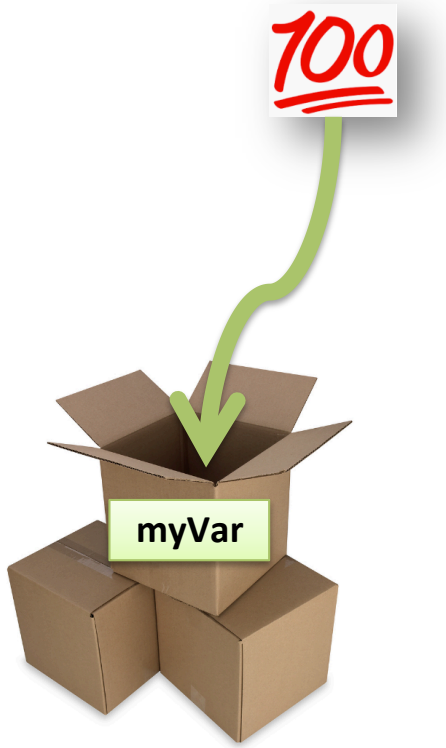
Variable name

Assignment operator

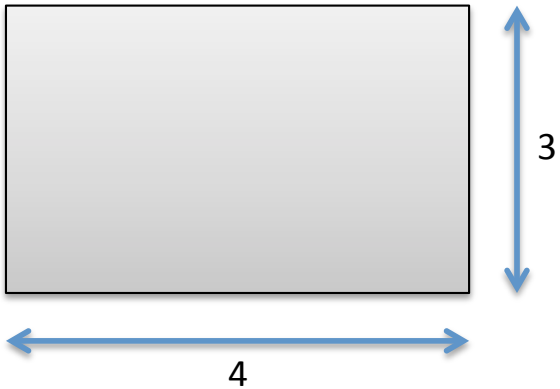
Constant value

```
myVar = 100
print(myVar)
```

```
100
```



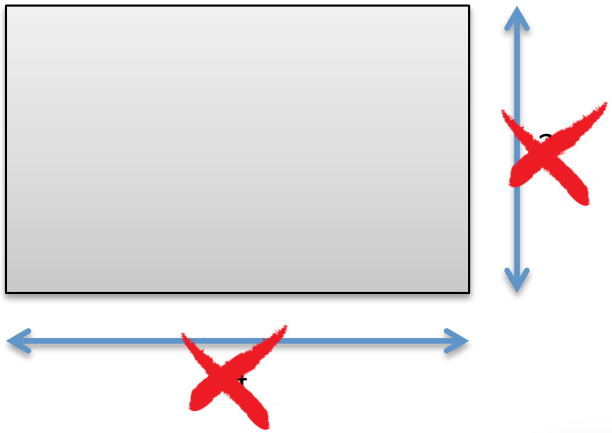
Area of a rectangle! Area = Length * Width



How can I solve the problem by using what I just learnt in Python?

```
length = 4
width = 3
area = length * width
print(area)
```

Area of a rectangle! Area = Length * Width (2)



How can I solve the problem by using what I just learnt in Python?

```
length = 
width = 
area = length * width
print(area)
```

How can I get those data as input?

Obviously… with input( )!!!

Input( )

print( )

```
text = input()
print(text)
```

The program works with data unknown to the programmer

The problem to solve is the same but with different data!

Input(SOMETHING)

This text will be shown while "asking" for the input data

```
name = input("What is your name? ")
print("Ciao " + name + "! Welcome to the CODING course!")
```

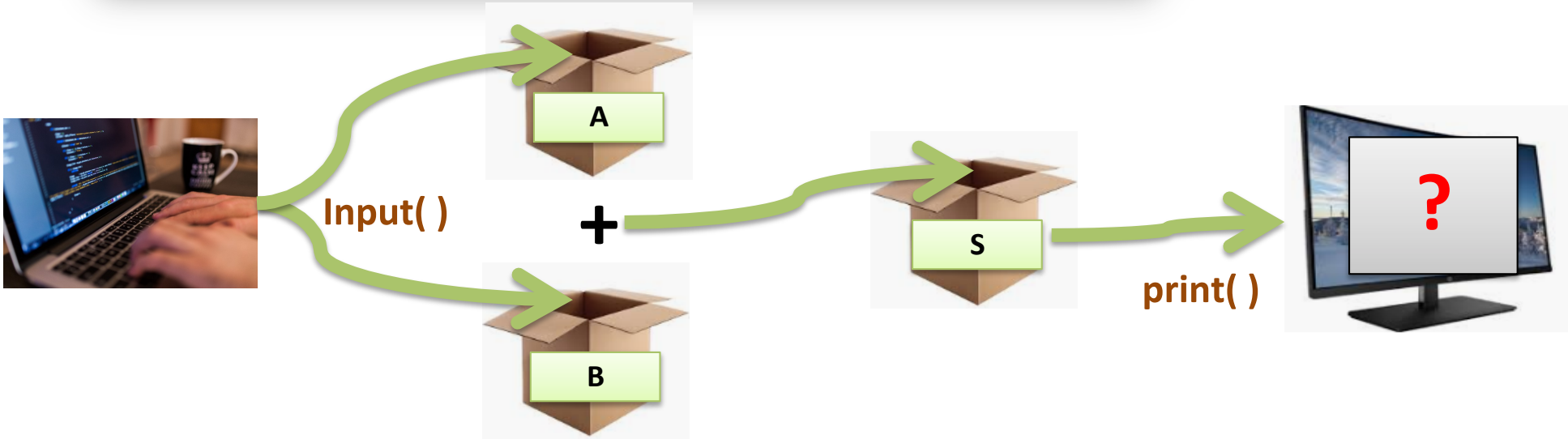
"+" when used with STRINGS of characters means concatenation

```
"Gian" + "luca" => "Gianluca"
```

```
subject = input("What is your preferred school subject? ")
print(subject + " is nice. However, CODING is much much better")
```

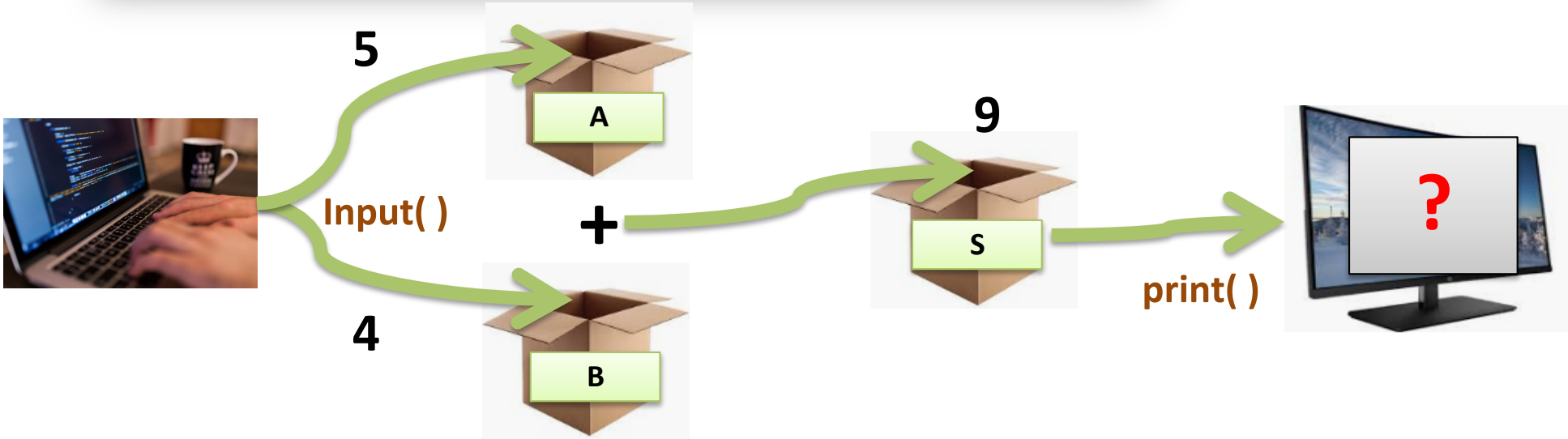
Sum A+B

```
a = input("Give me the value for 'a': ")
b = input("Give me the value for 'b': ")
s = a + b
print(s)
```



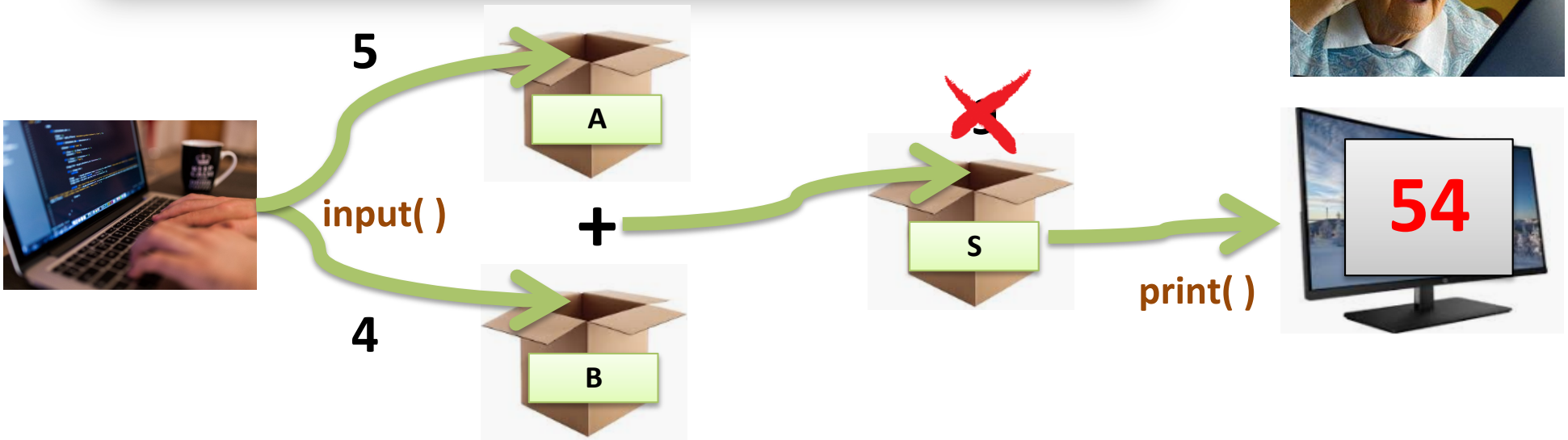
Sum A+B

```
a = input("Give me the value for 'a': ")
b = input("Give me the value for 'b': ")
s = a + b
print(s)
```



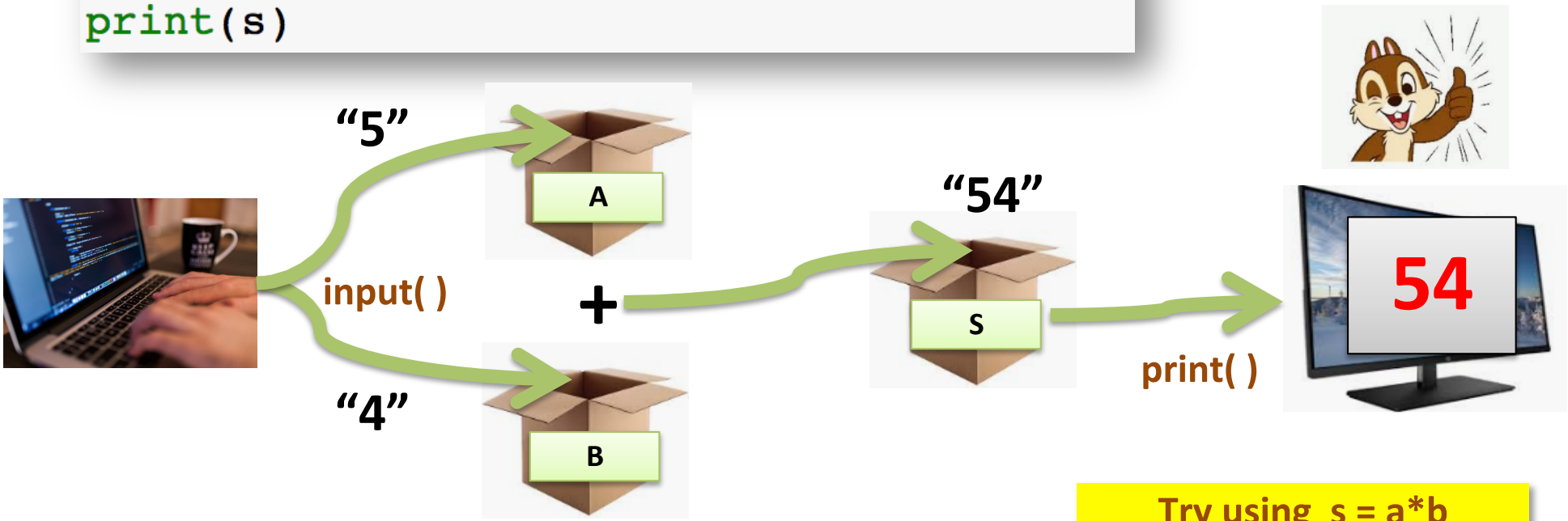
Sum A+B

```
a = input("Give me the value for 'a': ")
b = input("Give me the value for 'b': ")
s = a + b
print(s)
```



Input( ) reads STRINGS of characters!

```
a = input("Give me the value for 'a': ")
b = input("Give me the value for 'b': ")
s = a + b
print(s)
```



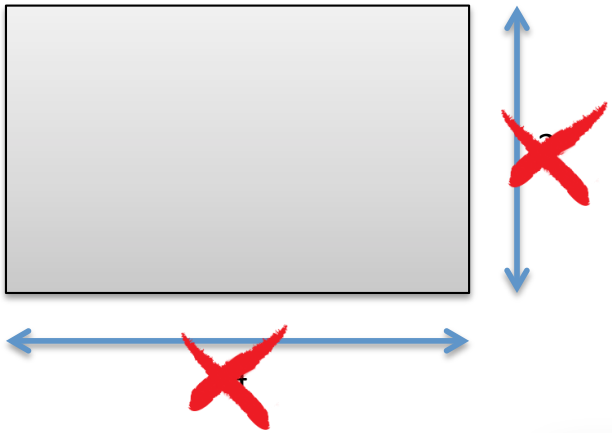
Try using s = a*b

Data conversion functions: cast operations

```
a = input("Give me the value for 'a': ")
b = input("Give me the value for 'b': ")
s = int(a) + int(b)
print("The result is: " + str(s))
```

```
Give me the value for 'a': 5
Give me the value for 'b': 4
The result is: 9
```

Area of a rectangle! Area = Length * Width (3)

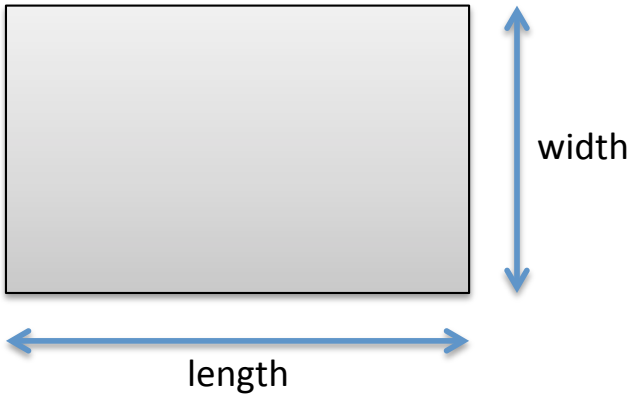


How can I solve the problem by using what I just learnt in Python?

```
length = ?
width = ?
area = length * width
print(area)
```

How can I get those data as input?

Area of a rectangle! Area = Length * Width (Final)



How can I solve the problem by using what I just learnt in Python?

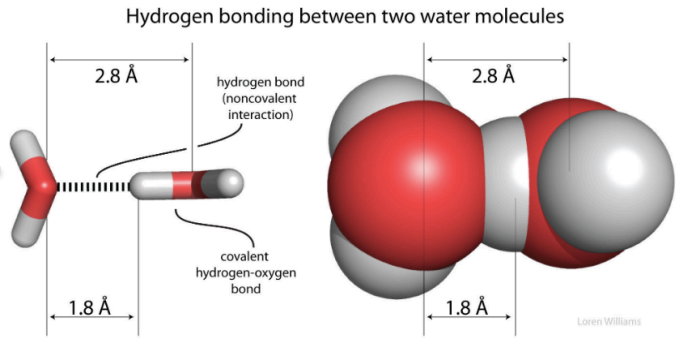
```
print(" *** This program calculates the area of a rectangle ***")
lenght = int(input("Rectangle lenght: "))
width = int(input("Rectangle width: "))
area = lenght * width
print("The area of the rectangle is: " + str(area))
```

POLITECNICO
MILANO 1863

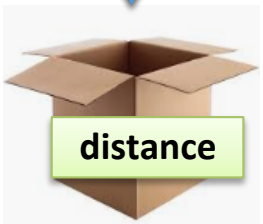
Floating Point Numbers and Modules

Variable types

Distance in meter



$$1.2345 = \underbrace{12345}_{\text{mantissa}} \times \overbrace{10^{-4}}^{\text{exponent}}$$



Can't be simply an integer number

Real numbers – float( )

Python can also manipulate real numbers.

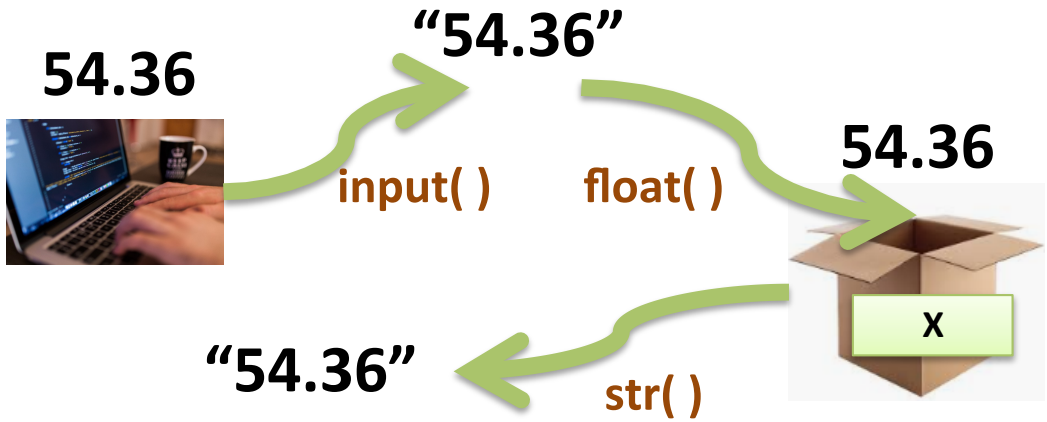
– Examples: `6.022` `-15.9997` `42.0` `2.143e17`

Decimal Separator

```
print(17 / 2)
print(2 ** -2)
```

```
8.5
0.25
```

```
x = float(input())
print("x = " + str(x))
```



Some useful built-in functions

5.3 —int(…)→ 5

5 —float(…)→ 5.0

5.6 —int(…)→ 5

In addition to conversion and I/O functions there are other useful built-in functions in Python

3, 7 —min(…)→ 3

3, 7 —max(…)→ 7

-5 —abs(…)→ 5

5 —abs(…)→ 5

5.3 —round(…)→ 5

5.8 —round(…)→ 6

Python Modules

Not all the functions are built-in…

import module_name

python™

module_name.function_name( )

Useful Math functions

Command name	Description
`ceil(`***value***`)`	rounds up
`floor(`***value***`)`	rounds down
`log(`***value***`)`	logarithm, base *e*
`log10(`***value***`)`	logarithm, base 10
`sin(`***value***`)`	sine, in radians
`cos(`***value***`)`	cosine, in radians
`sqrt(`***value***`)`	square root
`[…]`	

```
import math

average_grade=5.6
good_mood = math.ceil(average_grade)
bad_mood = math.floor(average_grade)
print("If good mood: " + str(good_mood))
print("If bad mood: " + str(bad_mood))
```

```
If good mood: 6
If bad mood: 5
```

Constant	Description
`e`	2.7182818...
`pi`	3.1415926...

```
import math
print(math.e)
print(math.pi)
print(math.sin(math.pi/2))
```

```
2.718281828459045
3.141592653589793
1.0
```

Useful “Random” functions

```python
import random

print(random.randint(1,100))
print(random.random())
print(random.uniform(-1,1))
```

```
58
0.2543450307434296
-0.6826947939469641
```

POLITECNICO
MILANO 1863

Conditional Statements

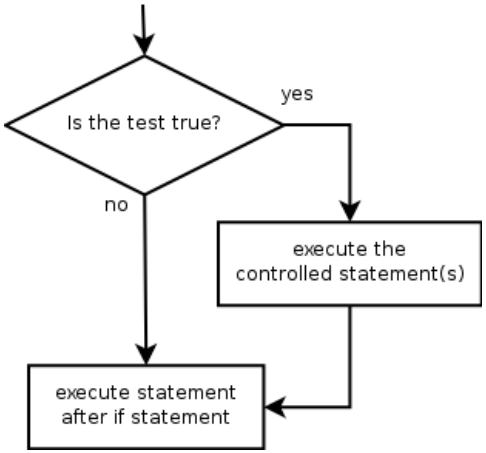
Conditions

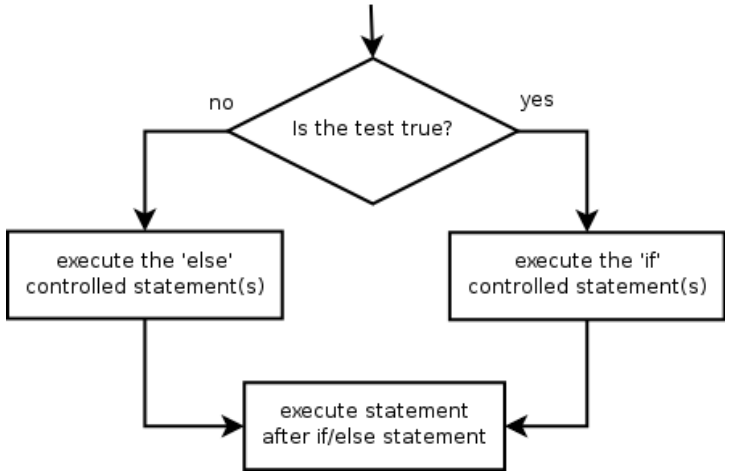
```
walk()
person = meet_someone()
say("Hello!")
walk()
```

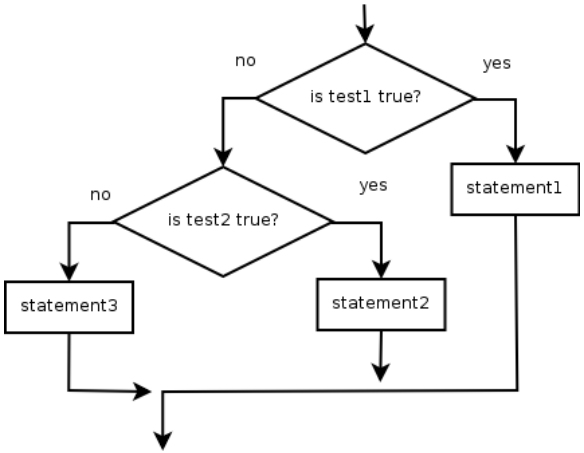
Do you really want to say "hello" to everyone?

```
walk()
person = meet_someone()
if person is nice:
        say("Hello!")
walk()
```

if(-elif)-else








```
if condition:
    statements
```

4 spaces or 1Tab

```
if condition:
    statements
else:
    statements
```

```
if condition:
    statements
elif condition:
    statements
else:
    statements
```

Conditions

Many logical expressions use relational operators:

Operator	Meaning	Example	Result
`==`	equals	`1 + 1 == 2`	`True`
`!=`	does not equal	`3.2 != 2.5`	`True`
`<`	less than	`10 < 5`	`False`
`>`	greater than	`10 > 5`	`True`
`<=`	less than or equal to	`126 <= 100`	`False`
`>=`	greater than or equal to	`5.0 >= 5.0`	`True`

Logical expressions can be combined with logical operators:

Operator	Meaning	Example	Result
`and`	True if both True, False otherwise	`9 != 6 and 2 < 3`	`True`
`or`	True if at least one True, False otherwise	`2 == 3 or -1 < 5`	`True`
`not`	True if False, False otherwise	`not 7 > 0`	`False`

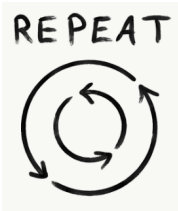
POLITECNICO
MILANO 1863

Loops

Need of iterative constructs

How many persons or for how much time?

```
[…]
walk()
person = meet_someone()
say("Hello!")
walk()
person = meet_someone()
say("Hello!")
walk()
person = meet_someone()
say("Hello!")
walk()
[…]
```



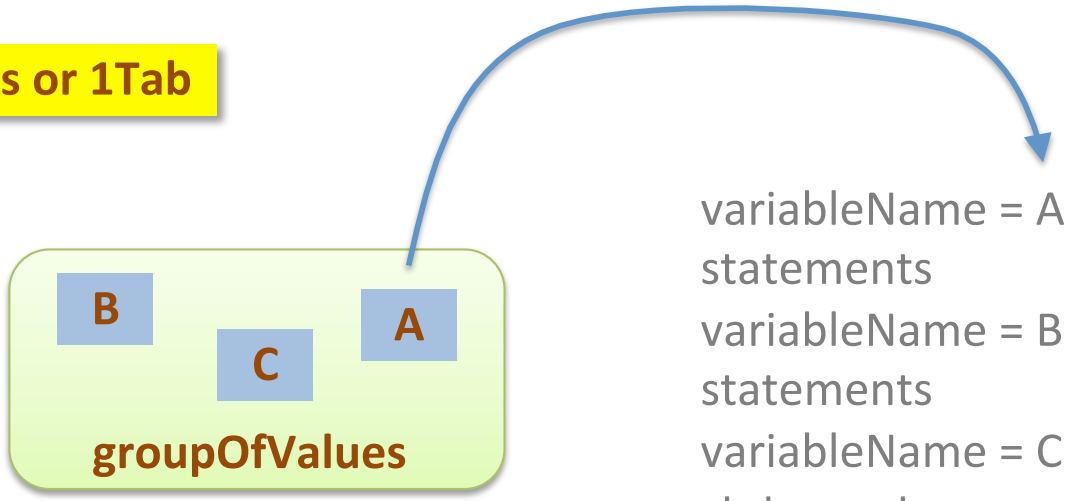

```
while not arrived:
        walk()
        person = meet_someone()
        say("Hello!")

for persons on the way:
        walk()
        person = meet_someone()
        say("Hello!")
```

For loop

Repeats a set of statements over a group of values

```
for variableName in groupOfValues:
    statements
```

4 spaces or 1Tab



variableName = A
statements
variableName = B
statements
variableName = C
statements

Range( )

- The `range` function specifies a range of integers:
  - `range(`***stop***`)`
    - the integers between ***0*** (inclusive) and ***stop*** (exclusive)
  - `range(`***start***`, `***stop***`)`
    - the integers between ***start*** (inclusive) and ***stop*** (exclusive)
  - `range(`***start***`, `***stop***`, `***step***`)`
    - the integers between ***start*** (inclusive) and ***stop*** (exclusive) by ***step***

```
for i in range(3):
    print(i)
```

```
0
1
2
```

```
for i in range(5, 8):
    print(i, i ** 2)
print('The end!')
```

```
5 25
6 36
7 49
The end!
```

```
for i in range(10, 0, -2):
    print(i)
```

```
10
8
6
4
2
```

Exercise:

Write a program that computes the factorial (n!) of an integer

While loop

Executes a group of statements as long as a condition is True.

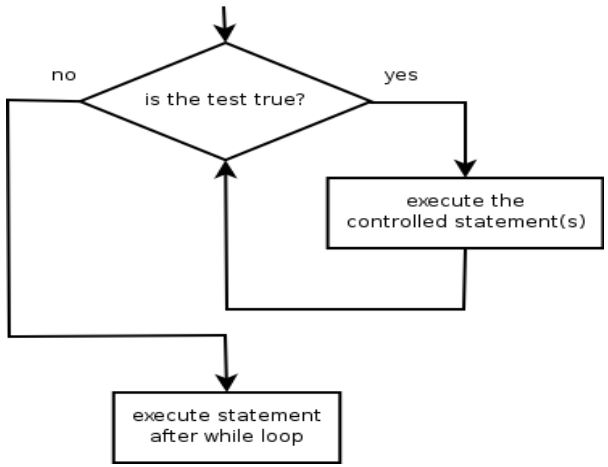
– good for indefinite loops (repeat for an unknown number of times)

Same as for IF

```
while (condition):
    Loop-Body
```

4 spaces or 1Tab

If condition == True:
    Loop-Body
    If condition == True:
        Loop-Body
        If condition == True:
            Loop-Body
            [...]



Example and Exercise:

```
i = 0

while (i<3):
    print("[" + str(i)+"] I'm still inside the loop")
    i = i+1

print("I'm out of the loop, i=" + str(i))
```

```
[0] I'm still inside the loop
[1] I'm still inside the loop
[2] I'm still inside the loop
I'm out of the loop, i=3
```

Exercise: Write a program that reads numbers from the input until the input is an even number.

POLITECNICO
MILANO 1863

Lists & Maps

Lists

Useful to store multiple data (typically of the same type).

Months

January
February
March
April
May
June
July
August
September
October
November
December

Print all the months

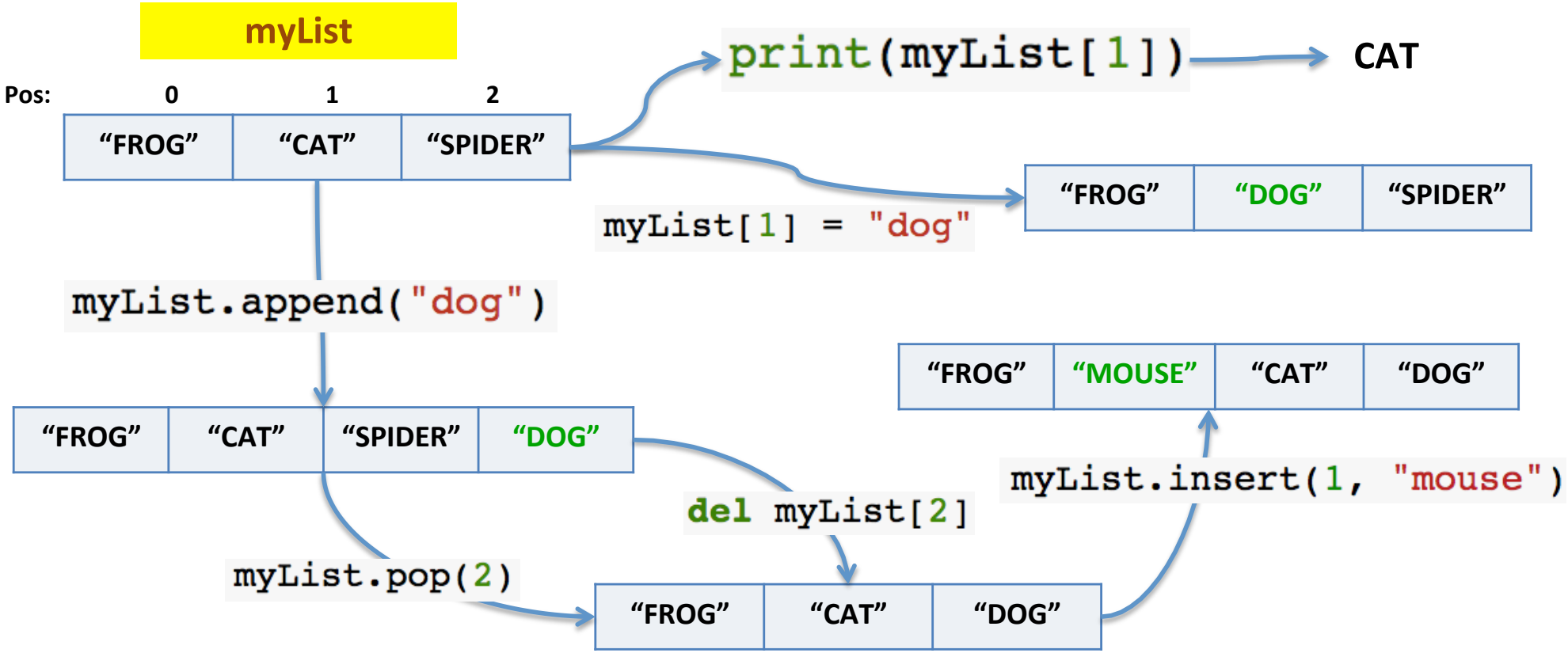
```
month1= "January"
month2 = "February"
month3 = "March"
[...]
print(month1)
print(month2)
print(month3)
[...]
```

```
months= ["January",
"February" ,  "March", ...]

print(months)
```

or

```
for x in months:
    print(x)
```

Lists manipulation



Other useful stuff on lists

```
myList1 = ['a', 'b', 'c']
myList2 = ['e', 'f', 'g']
myList3 = myList1 + myList2
print(myList3)
myList4 = myList1*3
print(myList4)
```

```
['a', 'b', 'c', 'e', 'f', 'g']
['a', 'b', 'c', 'a', 'b', 'c', 'a', 'b', 'c']
```

len(myList) returns the number of elements of the list

- **in** and **not in** operators

```
myList = [56, 76, 33, 56]
print(56 in myList)
print(14 in myList)
print(14 not in myList)
```

```
True
False
True
```

- **Others…**

```
myList = [56, 76, 33, 56]
#Duplicate the list
newList=myList
#Sort of all the elements in the list
myList.sort()
#Reverse the elements in the list
myList.reverse()
#Count how many time there is the element 56
count56 = myList.count(56)
#Remove the first instance of 76 from the list
myList.remove(76)
```

Something “more” on Strings

- Strings can be seen as a *sequence of characters*
- Characters in a string are *numbered as in lists*
- To access an individual character of a string:
  - variableName [index]

name

index	0	1	2	3	4	5	6	7
character	G	I	A	N	L	U	C	A

```
name = "Gianluca"
print(name[0])
print(name[5])
```

```
G
u
```

```
for myCharacter in 'MILANO':
    print(myCharacter)
print('The end!')
```

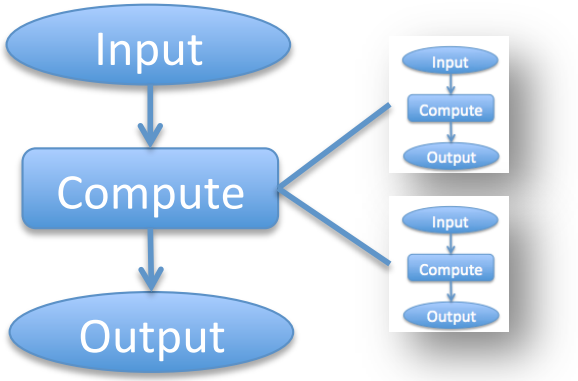
```
M
I
L
A
N
O
The end!
```

POLITECNICO
MILANO 1863

Modularity

Functions

- Functions help breaking our program into smaller and modular chunks
  - more organized and manageable
- Functions avoid repetition and makes code reusable




```
def function_name(parameters):
    body-of-the-function
    return value
```

Functions

Let's suppose that I would like to say "hello" to every person I meet in my school…

```
myName = input("What is your name? ")
print("Ciao " + myName + ". Good morning!")
print("This is my Python program!")
cmName = input("What is the name of one of your class mate? ")
print("Ciao " + cmName + ". Good morning!")
print("This is my Python program!")
mpName = input("What is the name of your Math professor? ")
print("Ciao " + mpName + ". Good morning!")
print("This is my Python program!")
```

```
def greetings(name):
    print("Ciao " + name + ". Good morning!")
    print("This is my Python program!")
    return

myName = input("What is your name? ")
greetings(myName)
cmName = input("What is the name of one of your class mate? ")
greetings(cmName)
mpName = input("What is the name of your Math professor? ")
greetings(mpName)
```

POLITECNICO
MILANO 1863

PyGame!!!

The graphic window

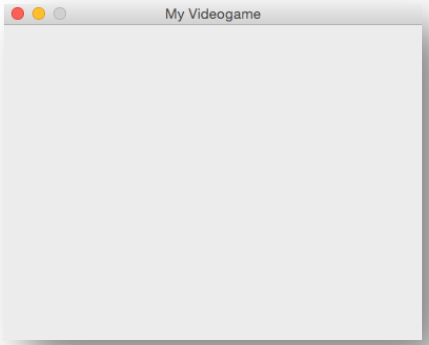
The basic step for a videogame is to create its graphic window

```
import pygame

pygame.init()

sizeX = 400
sizeY = 300

myScreen = pygame.display.set_mode((sizeX, sizeY))
pygame.display.set_caption("My Videogame")

pygame.quit()
```

Module inclusion and initialization

Display settings declaration and customization

Closure of the graphic module at the very end of the program



Main game loop

```python
import pygame

pygame.init()

sizeX = 400
sizeY = 300

myScreen = pygame.display.set_mode((sizeX, sizeY))
pygame.display.set_caption("My Videogame")

pygame.quit()
```

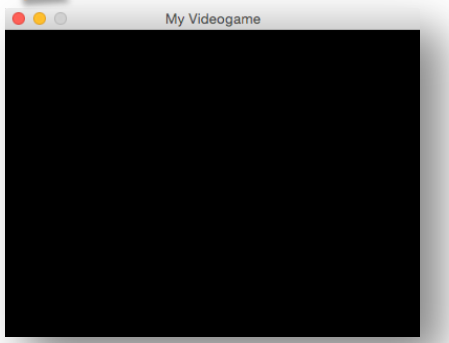
```python
gameActive = True
while (gameActive==True):
    #Clear screen
    myScreen.fill(pygame.Color("black"))

    #event detection
    for event in pygame.event.get():
        #print(event)
        if event.type==pygame.QUIT:
            gameActive = False

    #Display Update
    pygame.display.update()
```

Fill the background of the screen

Detect the event that are happening during the “game” life

Update the screen with the new graphic elements (if any) ☺

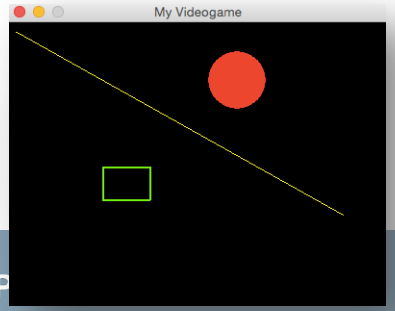
Quit



Graphical elements

Command	Description
`pygame.draw.line(`**`surface,color,(x1,y1),(x2,y2),thickness`**`)`	Draws a line between (***x1***, ***y1***) and (***x2***, ***y2***) with a certain **thickness**
`pygame.draw.circle(`**`surface,color,(x,y),radius,thickness`**`)`	Draws an circle centered in **(x,y)**, with a certain **radius** and **thickness**. If **thickness** is missing, the circle will be filled.
`pygame.draw.rect(`**`surface,color,[x,y,width,height],thickness`**`)`	Draws a rectangle with a top-left corner at (**x**,**y**), and a certain **width**, **height** and **thickness**. If **thickness** is missing, the rectangle will be filled.
[…] more at https://www.pygame.org/docs/ref/draw.html	

```
pygame.draw.line(myScreen, pygame.Color("yellow"), (10,10), (350, 200), 1)
pygame.draw.circle(myScreen, pygame.Color("red"), (240,60), 30)
pygame.draw.rect(myScreen, pygame.Color("green"), [100, 150, 50, 35], 2)
```



Animation:

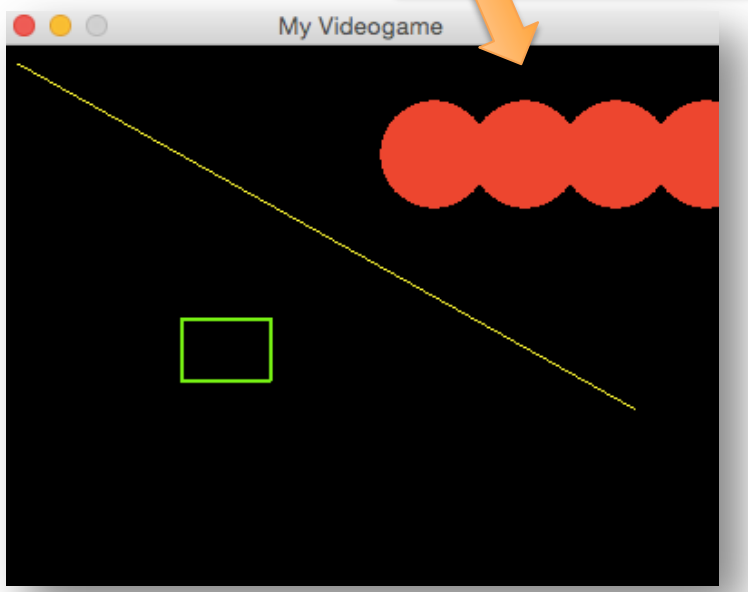
It is enough to draw the same item at different x/y positions within the game loop

```
xCircle = 240
gameActive = True
clock = pygame.time.Clock()

while (gameActive==True):
    #Clear screen
#   myScreen.fill(pygame.Color("black"))

    #Geometric shapes
    pygame.draw.circle(myScreen, pygame.Color("red"), (xCircle,60), 30)
    pygame.draw.line(myScreen, pygame.Color("yellow"), (10,10), (350, 200), 1)
    pygame.draw.rect(myScreen, pygame.Color("green"), [100, 150, 50, 35], 2)
    #Position and dimension update
    xCircle = xCircle + 20

    #Event detection
    for event in pygame.event.get():
        if event.type==pygame.QUIT:
            gameActive = False

    #Display Update
    pygame.display.update()

    #Maximum frame rate
    clock.tick(50)

pygame.quit()
```

We can use a variable and update its value

It is used to limit the speed of the animation (FrameRate)

I removed the "cleaning" of the screen



More on animations:

You can easily make the circle bouncing from left to right, or in any direction

```
# Hint
# Using updateX as the value of the position update
# and xCircle as the x coordinate of the circle...
#
# xCircle= xCircle + updateX
# if (xCircle >= SizeX) or (xCircle <= 0):
#     updateX = -1*updateX
#
```

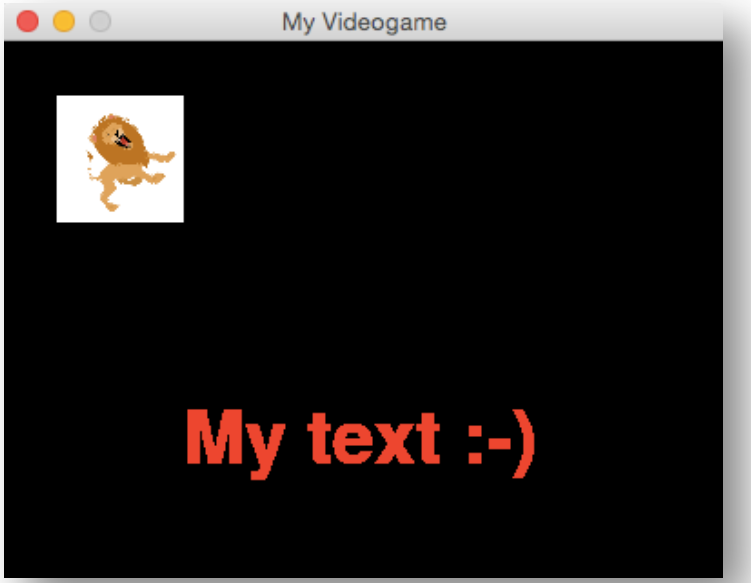
Or you can make disappearing the rectangle when hit by the ball

```
# Hint: define a variable that is used to decide
# if the rectangle should be drawn or not
#
# If hit == False:
#       pygame.draw.rect(...)
```

Text and Images

```
fontSize = 60
font = pygame.font.Font(None, fontSize)
textColor = pygame.Color("red")
text = font.render("My text", True, textColor)
#to get widht and height of a text
#text.get_width() text.get_height()
myScreen.blit(text, (textXpos, textYpos))
```

```
image = pygame.image.load("lion.bmp")
#to get X and Y dimensions
# imageXdim = image.get_size()[0]
# imageYdim = image.get_size()[1]

# to scale the dimensions to 50x50
image = pygame.transform.scale(image,(50,50))
# to rotate the image by 45°
image = pygame.transform.rotate(image, 45)

# include the image in the game window
myScreen.blit(image, (imgXpos, imgYpos))
```



Other possible interactions

```
for event in pygame.event.get():
    # ...
    if (event.type == pygame.MOUSEBUTTONDOWN):
            mouse_pos = pygame.mouse.get_pos()
            mouse_xPos=mouse_pos[0]
            mouse_yPos=mouse_pos[1]

    if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                xPosition = -5
            if event.key == pygame.K_RIGHT:
                xPosition = 5
```

Mouse Pressed

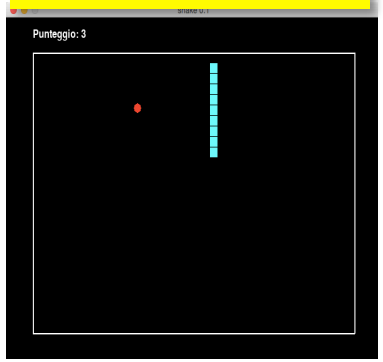
Detects the position of the cursor in the game window. It can be used also outside the event detection loop

Detects when a button of the keyboard has been pushed and which one

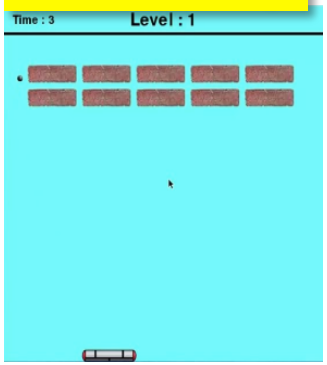
More event.key can be found at
https://www.pygame.org/docs/ref/key.html

Its your turn!!!

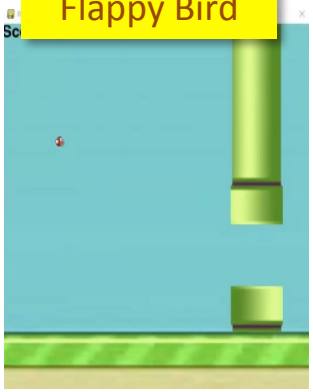
Snake



Arkanoid/Pong

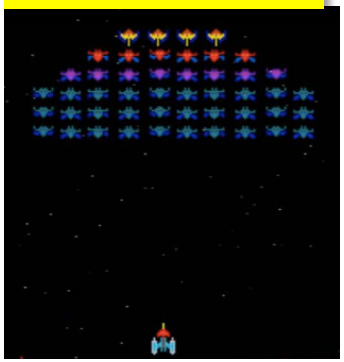


Flappy Bird

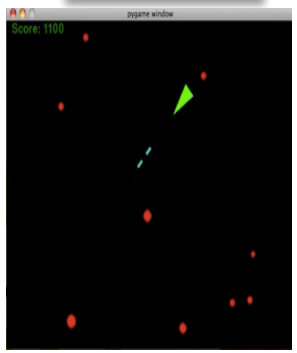


Shot the balloon

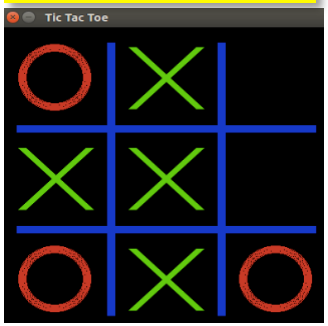
Space Invaders



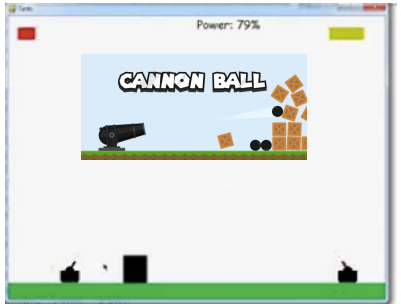
Asteroids



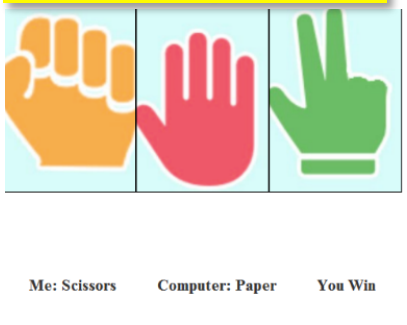
TicTacToe



Cannon Ball

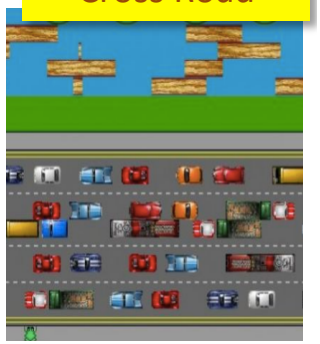


Scissor-Paper-Rock

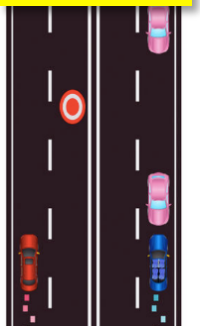


Hangman

Cross Road



Car Race



Other possibilities: 4in a Row, simplified version of Candy Crush, temple run ...