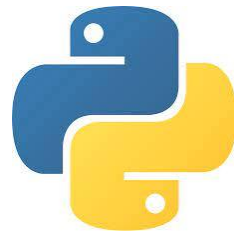




DIGISPORT
GRADUATE SCHOOL

ÉCOLE UNIVERSITAIRE DE RECHERCHE

ALGORITHM AND PROGRAMMING



ALGORITHM AND PYTHON FOR BEGINNERS
APPLICATION TO MOTION ANALYSIS

GOAL OF THIS LESSON

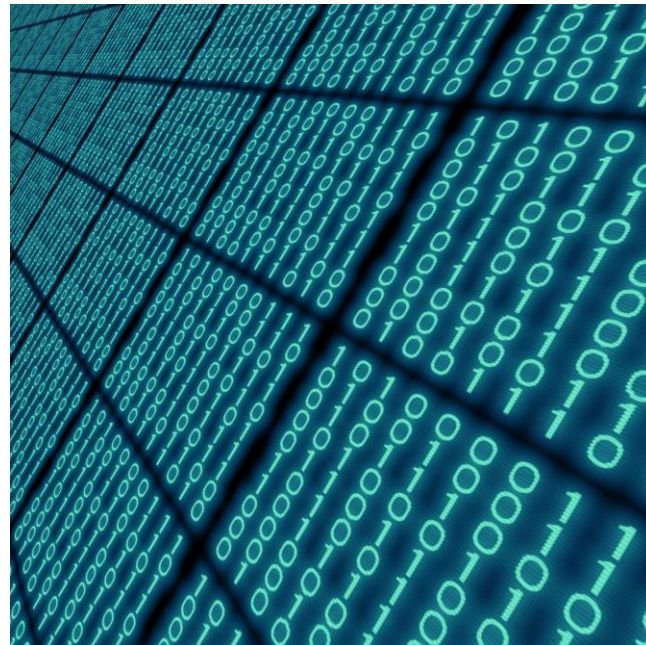
- Design programs to
 - Automate data processing
 - Communicate with devices
 - Make statistical analyses
- You thus need to
 - Implement code
 - Give commands to computer
 - Format and store results
 - Design algorithm
 - Structure your program
 - Create generic and reusable codes
- Implemented in Python programming language

1

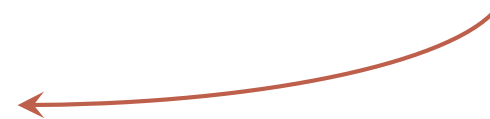
INTRODUCTION

WHAT IS PYTHON? WHY PYTHON?

- Python is a **programming language**
 - Computers work on bytes, groups of 8 bits, that are binary data (0/1)
 - Programming language allows to give instructions to computers



```
16 # Load mocap file
17 filename = 'data/trial02.csv'
18 #filename = 'data/back05.csv'
19 trajs, freq, markers = file.load_c
20 print('- file', filename, 'loaded v
21 print(' - size:', np.shape(trajs)
22 print(' - frequency:', freq)
23 print(' - markers', markers)
24
25 # Create time column
26 t = np.arange(0, np.shape(trajs)[0
27
28 # Compute the center of the right
29 RWRA = traj.get_marker_traj_from_r
30 RWRB = traj.get_marker_traj_from_r
31 right_wrist = (RWRA+RWRB)/2
32
33 # Compute the center of the right
34 RRAD = traj.get_marker_traj_from_r
35 RHUM = traj.get_marker_traj_from_r
36 right_elbow = (RRAD+RHUM)/2
37
38 # Compute the length of the forear
39 right_forearm = right_wrist - right
40 right_forearm_lg = bm.norm_3d_traj
41 plt.figure(0)
42 plt.plot(t, right_forearm_lg, 'b-
43 plt.xlabel('Time (s)')
44 plt.ylabel('Values (mm)')
45
```



WHAT IS PYTHON? WHY PYTHON?

- Python is a **general-purpose** programming language
 - Computers work on bytes, groups of 8 bits that are binary data (0/1)
 - Programming language that allows to give instructions to computers
- Widely used to create applications by engineers, mathematicians, data analysts, accountants and by many companies



WHAT IS PYTHON? WHY PYTHON?

- Python is a **high-level** general-purpose programming language
 - Computers work on bytes, groups of 8 bits that are binary data (0/1)
 - Programming language that allows to give instructions to computers
- Widely used to create applications by engineers, mathematicians, data analysts, accountants and by many companies
- **Abstraction from machine language**
 - Variables, objects, arrays... instead of memory addresses, call stacks...

```
# Definition of the function
def get_marker_traj_from_index(data, marker_index):
    col_X = (marker_index+1)*3-2
    col_Z = (marker_index+1)*3
    return data[:, col_X:col_Z+1]

# Use of the function
traj_marker_0 = get_marker_traj_from_index(back05, 1)
print(traj_marker_0)
```



	Encoding								Grammar								Parse File							
	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15		
0x000000	CF	FA	ED	FE	07	00	00	01	03	00	00	00	06	00	00	00	0F	00	00	00	30	07		
0x000016	00	00	85	00	10	00	00	00	00	00	00	19	00	00	00	78	02	00	00	5F	5F	54	45	
0x00002C	58	54	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	90	
0x000042	1D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	90	1D	00	00	00	00	00	
0x000058	07	00	00	00	05	00	00	00	07	00	00	00	00	00	00	00	5F	5F	74	65	78	74		
0x00006E	00	00	00	00	00	00	00	00	00	00	00	00	5F	5F	54	45	58	54	00	00	00	00	00	
0x000084	00	00	00	00	00	16	00	00	00	00	00	00	41	02	18	00	00	00	00	00	00	00	16	
0x00009A	00	00	04	00	00	00	00	00	00	00	00	00	00	00	00	00	04	00	80	00	00	00	00	
0x0000B0	00	00	00	00	00	00	00	00	5F	5F	73	74	75	62	73	00	00	00	00	00	00	00	00	
0x0000C6	00	00	5F	5F	54	45	58	54	00	00	00	00	00	00	00	00	00	00	00	00	42	18	18	00
0x0000DC	00	00	00	00	6C	06	00	00	00	00	00	00	42	18	18	00	01	00	00	00	00	00	00	

WHAT IS PYTHON? WHY PYTHON?

- Python is a **interpreted** high-level general-purpose programming language
 - Computers work on bytes, groups of 8 bits that are binary data (0/1)
 - Programming language that allows to give instructions to computers
- Widely used to create applications by engineers, mathematicians, data analysts, accountants and by many companies
- Abstraction from machine language
 - Variables, objects, arrays... instead of memory addresses, call stacks...
- **Code can be directly executed without compilation on Windows, Mac, mobiles**

WHAT IS PYTHON? WHY PYTHON?

- Python is a interpreted high-level general-purpose programming language
 - Computers work on bytes, groups of 8 bits that are binary data (0/1)
 - Programming language that allows to give instructions to computers
- Widely used to create applications by engineers, mathematicians, data analysts, accountants and by many companies
- Abstraction from machine language
 - Variables, objects, arrays... instead of memory addresses, call stacks...
- Code can be directly executed without compilation on Windows, Mac, mobiles
- Free, open-source language
- Large community, many resources, libraries and tutorials

WHY LEARN PYTHON?

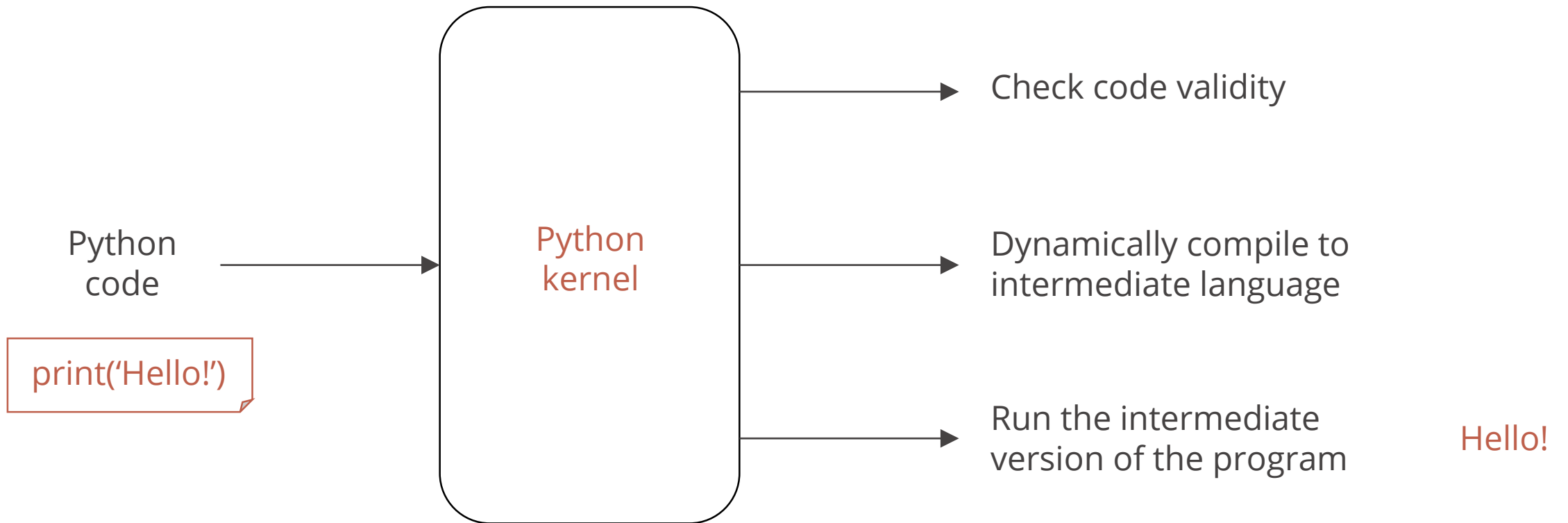
- To automate motion analysis
 - High frequency, motion duration, repetitions, sample size
 - Huge number of data
 - Divide and conquer
 - Create libraries of your functions
 - Create complex computation from the combination of simpler ones
- Collaborate with other fields
 - Used for statistics, machine learning, multimedia programs, signal analysis...
- Make yourself adaptable to other languages and multidisciplinary projects

2

HOW IT WORKS

FROM WRITING TO EXECUTION

- Python is an interpreted programming language



FROM WRITING TO EXECUTION

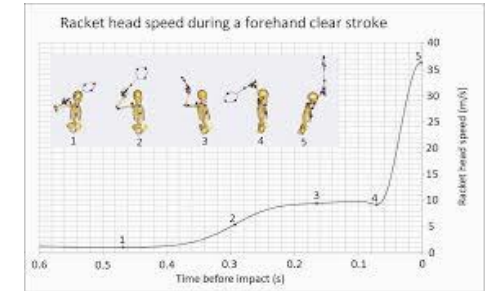


RAM



Hard drive

abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	



Code lines

Functions

Modules Libraries

Program

```
a = np.arange(12).reshape(3, 4)
print('a =', a)
b1 = np.array([False, True, True])
b2 = np.array([True, False, True, False])

print('a[b1, :] =', a[b1, :])
print('a[b1] =', a[b1])
print('a[:, b2] =', a[:, b2])
print('a[b1, b2] =', a[b1, b2])
```

```
# Definition of the function
def get_marker_traj_from_index(data, marker_index):
    col_X = (marker_index+1)*3-2
    col_Z = (marker_index+1)*3
    return data[:, col_X:col_Z+1]

# Use of the function
traj_marker_0 = get_marker_traj_from_index(back05, 1)
print(traj_marker_0)
```

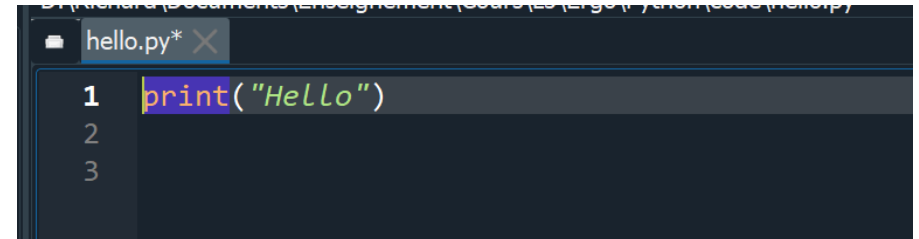
SEVERAL WAYS TO EXECUTE CODE

- Interactively with Python commands
 - Python REPL (Read Evaluate Print Loop)
 - Also in Anaconda Powershell Prompt
- To start Python: simply write *python* in the shell
 - Ready to execute Python commands

```
C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe
(base) PS C:\Users\rkulpa> python
Python 3.8.5 (default, Sep 3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print ('Hello')
Hello
>>>
```

SEVERAL WAYS TO EXECUTE CODE

- Interactively with Python commands
 - Python REPL (Read Evaluate Print Loop)
 - Also in Anaconda Powershell Prompt
- To start Python: simply write *python* in the shell
 - Ready to execute Python commands
 - From a Python source file

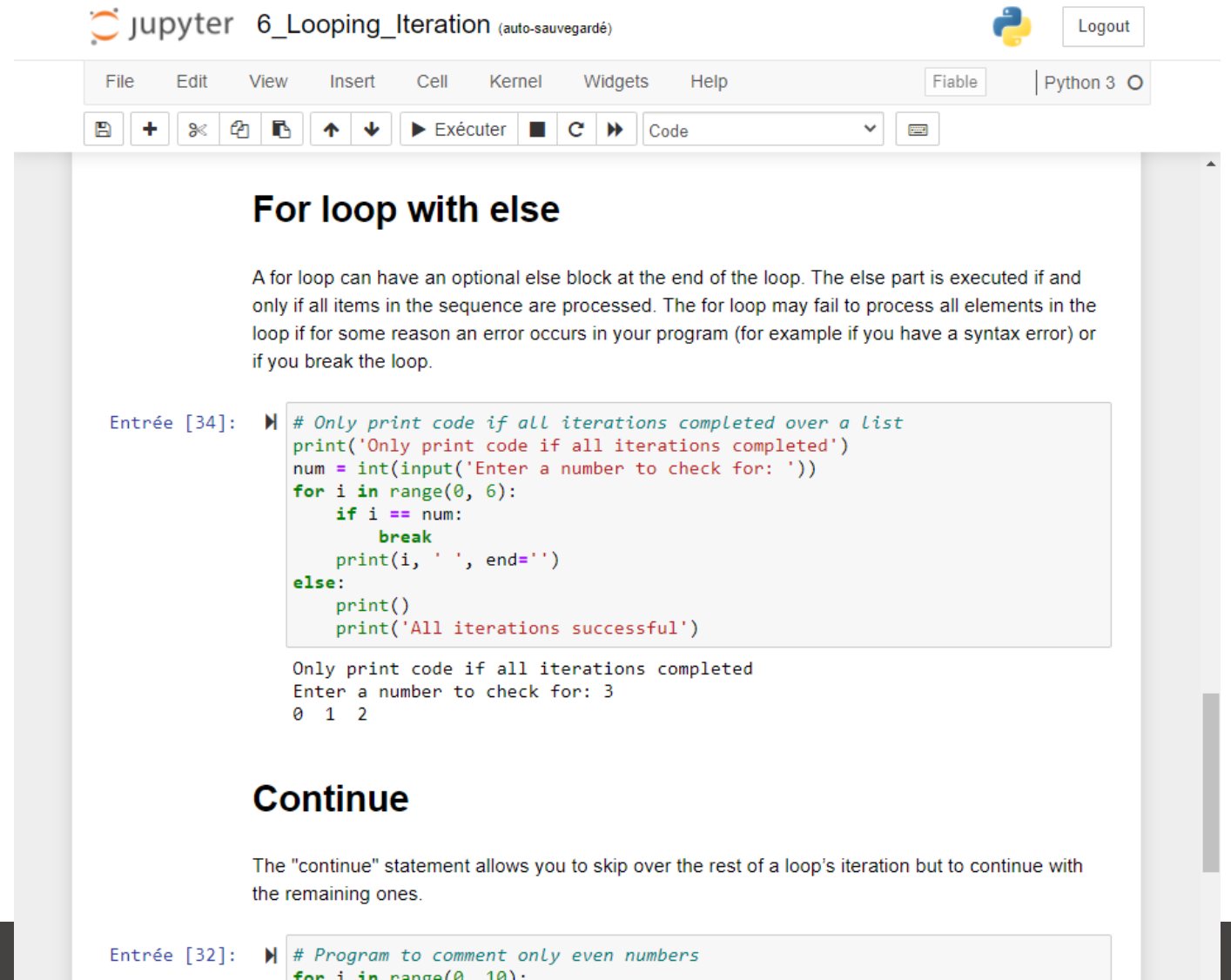


```
hello.py*  
1 print("Hello")  
2  
3
```

```
C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe  
(base) PS C:\Users\rkulpa> python  
Python 3.8.5 (default, Sep 3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>> print('Hello')  
Hello  
>>>
```

SEVERAL WAYS TO EXECUTE CODE

- With a notebook (in the navigator)
- First part of the course on this medium



The screenshot shows a Jupyter Notebook interface. At the top, the title is "jupyter 6_Looping_Iteration (auto-sauvegardé)". The menu bar includes "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". The status bar shows "Fiable" and "Python 3". The toolbar contains icons for file operations and execution. The main content area has a heading "For loop with else" followed by a paragraph explaining that a for loop can have an optional else block. Below this is a code cell labeled "Entrée [34]:" containing Python code that uses a for loop with a break statement and an else block. The output of the code cell shows the execution of the loop and the final message "All iterations successful".

For loop with else

A for loop can have an optional else block at the end of the loop. The else part is executed if and only if all items in the sequence are processed. The for loop may fail to process all elements in the loop if for some reason an error occurs in your program (for example if you have a syntax error) or if you break the loop.

```
Entrée [34]: ▶ # Only print code if all iterations completed over a List
print('Only print code if all iterations completed')
num = int(input('Enter a number to check for: '))
for i in range(0, 6):
    if i == num:
        break
    print(i, ' ', end='')
else:
    print()
    print('All iterations successful')
```

Only print code if all iterations completed
Enter a number to check for: 3
0 1 2

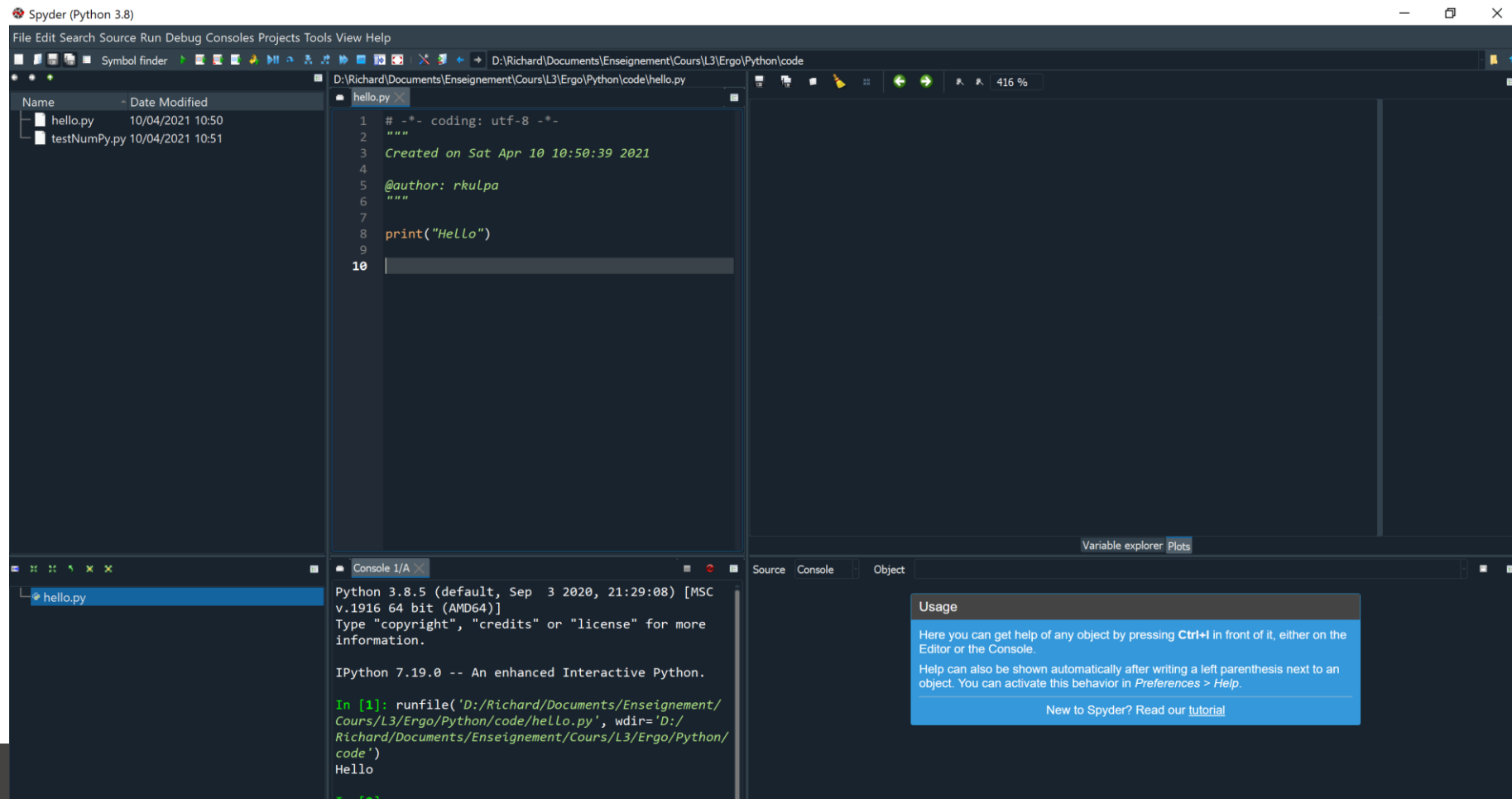
Continue

The "continue" statement allows you to skip over the rest of a loop's iteration but to continue with the remaining ones.

```
Entrée [32]: ▶ # Program to comment only even numbers
for i in range(0, 10):
```


SEVERAL WAYS TO EXECUTE CODE

- With an IDE (Integrated Development Environment): Spyder



3

INSTALLATION

INSTALLATION

- Large community and development tools
 - Many Python developing environments
- This lesson based on a reference tool: Anaconda
 - Gathers all necessary tools
 - Jupyter Notebook
 - Allows to keep track of each code and to see the result directly, possibility to make web pages
 - First part of the lesson is created on it
 - Spyder
 - More complete development environment
 - And libraries
 - NumPy: numerical computation
 - Matplotlib: visualization



ANACONDA®

anaconda.com

4

ALGORITHM

ALGORITHM

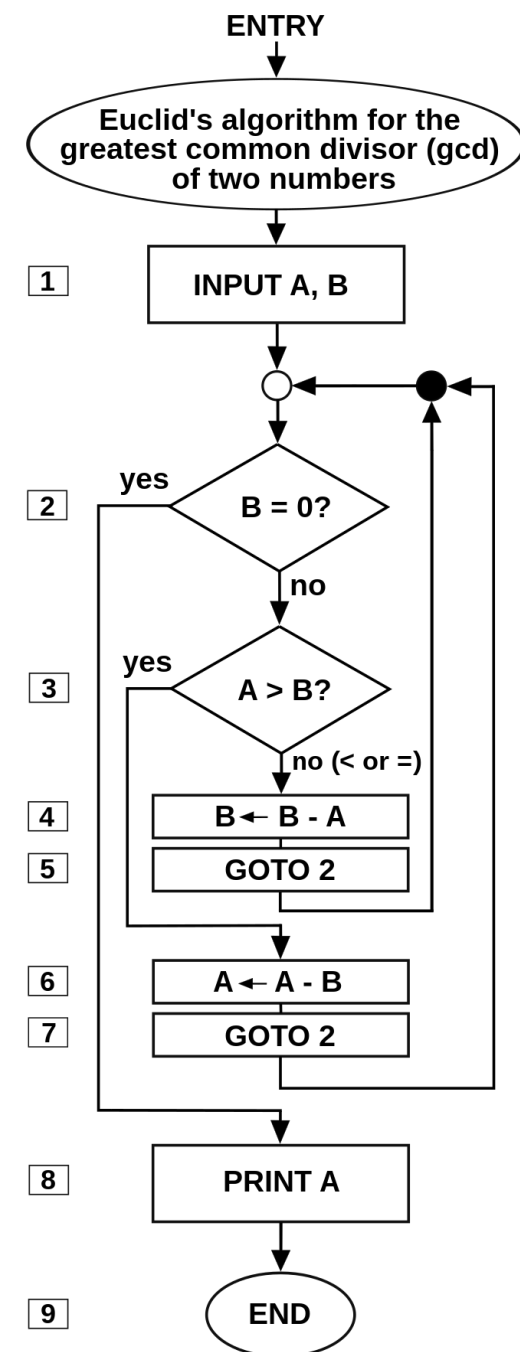
- Overall objective: get the computer to do what you want it to do
- Problem: how to translate a concept, a problem, an analysis into a series of instructions understandable by the computer?
 - Going from ideas in natural language to a programming language
 - Sometimes problem / calculation is complex: divide and conquer approach
 - Focus on the steps of the calculation first in a sequential order
 - Divide the calculation into sub-steps
 - Arrange all these steps in order to handle all cases (using conditional test for example)
 - Translate these steps one after the other into the programming language

ALGORITHM

- A distinction is made between:
 - The algorithm, which takes data as input, describes the processing steps to be performed and provides data as output
 - Program which is the series of instructions that carries out (implements) an algorithm. It is written with a programming language (Python for this course)
- The algorithm is therefore independent of the programming language

ALGORITHM

- Two main approaches for writing an algorithm
 - Flowchart
 - Pseudo-language
- Example of the Greatest Common Divisor (GCD)
 - *Plus Grand Commun Diviseur* (PGCD) in French
- Flowchart
 - Starts from ENTRY and finished at END
 - Ellipse: comment
 - Rectangle: command/action/computation
 - Diamond: conditional test



ALGORITHM

- Pseudo-language

ALGORITHM GCD

{Greatest Common Divisor (PGCD in French)}

VARIABLES

A : integer, B : integer

ENTRY

1 INPUT A, B

2 IF B \neq 0

3 IF A \leq B

4 B \leftarrow B - A

5 GOTO 2

... ...

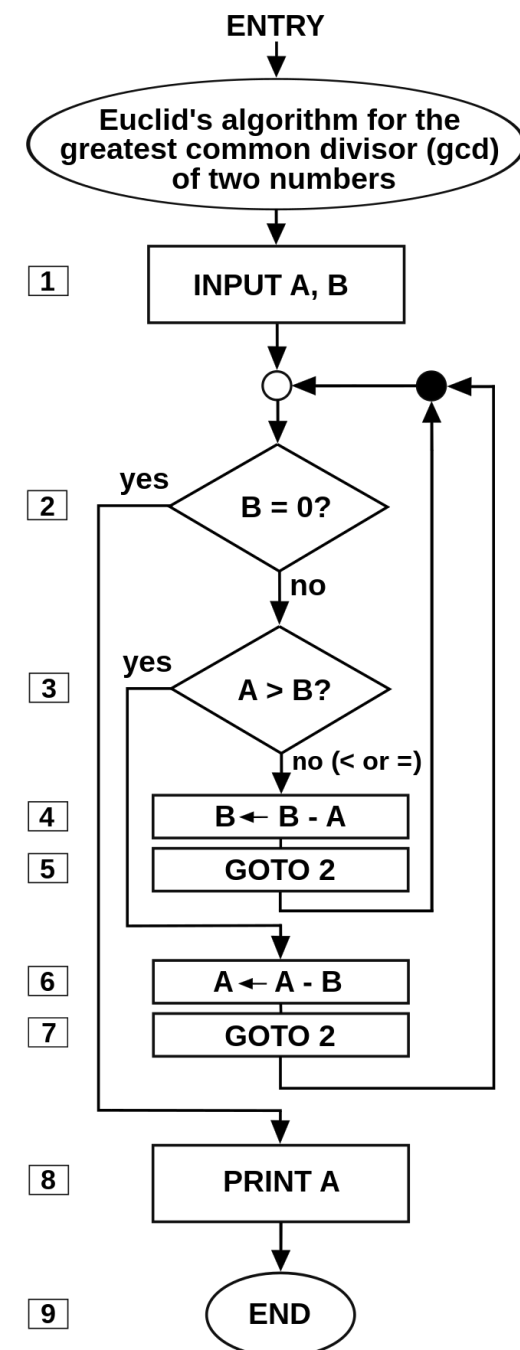
 ENDIF

ENDIF

PRINT A

END

- Importance of indentation !
 - Mandatory in Python
 - There is no end, only based on indent



ALGORITHM

- The better the design, the easier the implementation

ALGORITHM GCD

{Greatest Common Divisor (PGCD in French)}

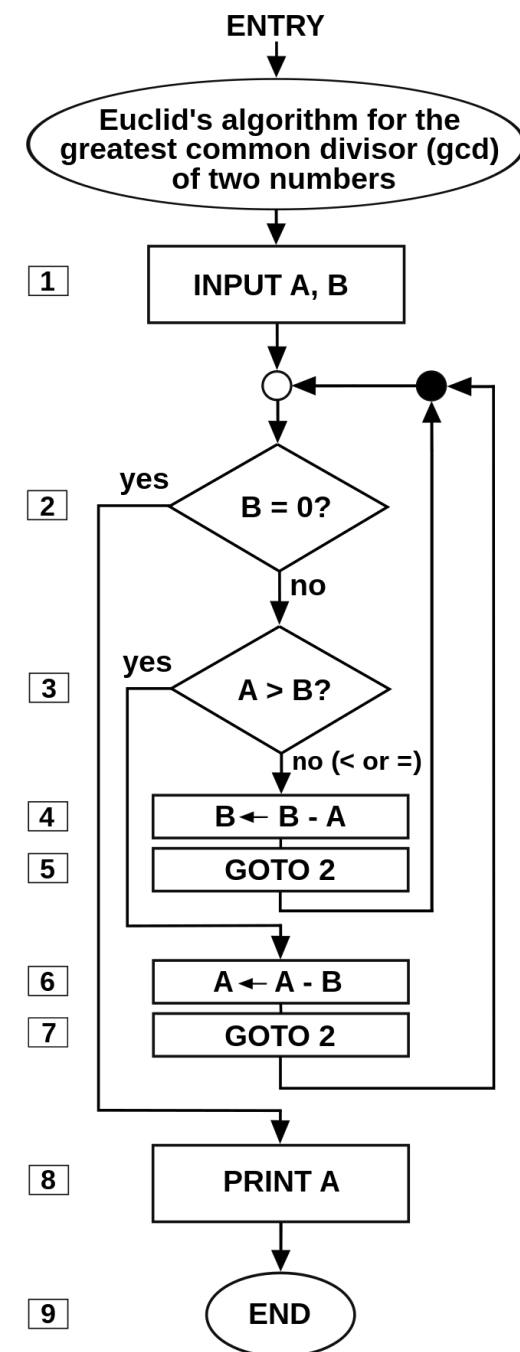
VARIABLES

A : integer, B : integer

ENTRY

```
1 INPUT A, B
2 WHILE B <> 0
3     IF A <= B
4         B ← B - A
5     ELSE
6         A ← A - B
7     ENDIF
8 ENDIF
9 PRINT A
END
```

- Avoid the GOTO!
 - Use loops instead
 - Or order differently



ALGORITHM

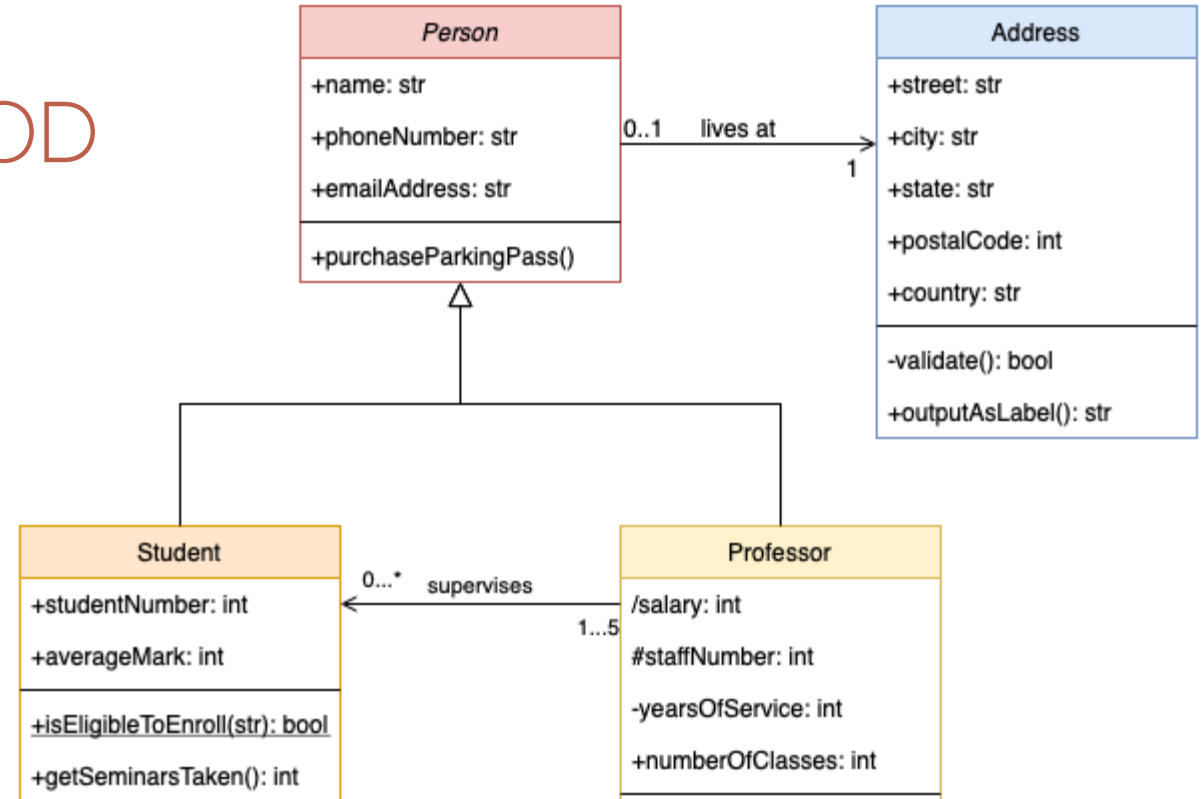
- In Python
 - You'll do it by yourself soon ^^

ALGORITHM

- Algorithms are also intended to address:
- Complexity
 - How long does it take for an algorithm to achieve the desired result?
 - How much memory space does it need?
- Computability
 - Are there tasks for which no algorithm exists?
 - Given a task, can we say whether there is an algorithm that solves it?
- Correction
 - Can we be sure that an algorithm answers the problem for which it was designed?
 - And that it works for all possible cases?

OBJECT-ORIENTED METHOD

- New types of data
 - Data structure
 - Set of variables
 - Class
 - Variables
 - Methods (functions)
 - Access levels (internal access only...)
 - Inheritance
 - ...
- Modeling language to design programs
 - Unified Modeling Language (UML)



GOOD PROGRAMMING

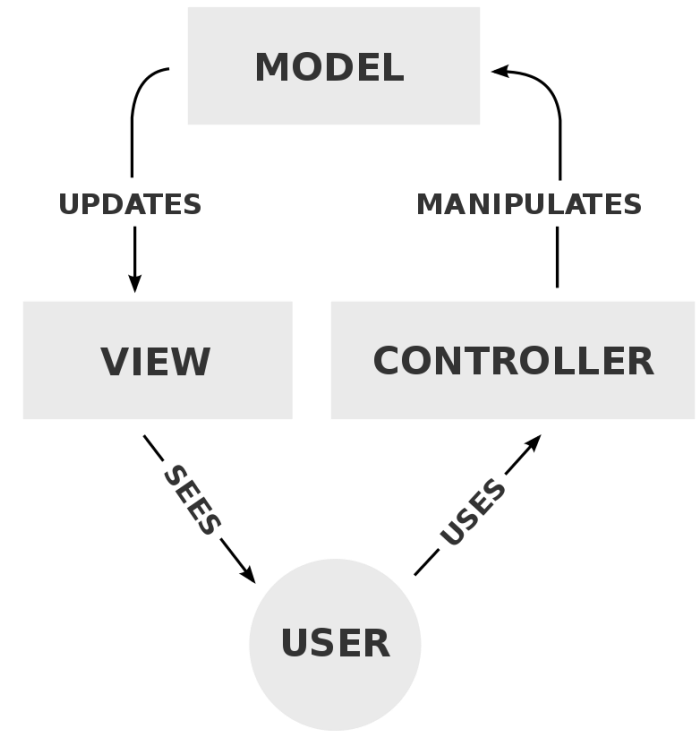
- Good programming
 - Easy to read (even by other programmers)
 - Easy to debug
 - Easy to improve
 - Easy to apply to a new device, a new language...
- Use smart object names, avoid a,b,c...
- Respect programming rules
- Comment your code!
 - For the other developers
 - For you later
 - Some parsers used to ask for a minimum ratio of comments

GOOD PROGRAMMING

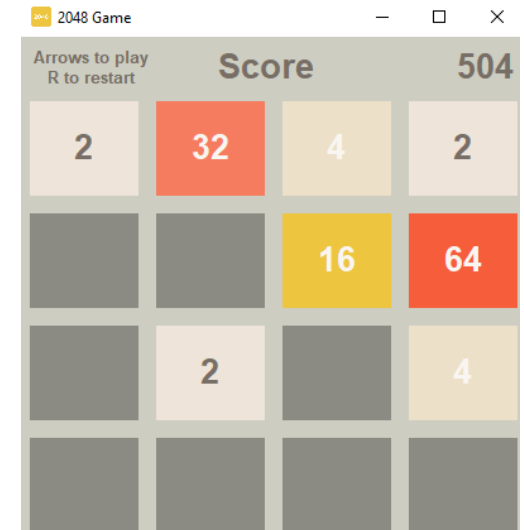
- Organize your program to be as generic as possible
 - Small elementary functions used by other functions
 - Code independant of input and output devices for example
- Example of functions
 - local_com computes the local COM of a segment
 - global_com uses local_com to calculate the COM of the posture
 - derivate allows to compute the derivative to determine the acceleration of the COM (used twice)

GOOD PROGRAMMING

- Organize your program to be as generic as possible
 - Small elementary functions used by other functions
 - Code independant of input and output devices for example
 - Example of Model-view-controller (MVC) design



```
2048_Game.py x Controller.py x View.py x Model.py x
1  #-*- coding: utf-8 -*-
2  """
3  Created on Sun Jun  5 14:53:23 2022
4
5  @author: rkulpa
6  """
7
8  # In Spyder, when there are multiples files, use once Ctrl+Enter to start this
9  # file, then use Alt+Enter whatever the file opened in the editor to still
10 # run this one
11
12 from View import View
13 from Model import Model
14 from Controller import Controller
15
16 if __name__ == "__main__":
17     model = Model()
18     view = View(model)
19     controller = Controller(model, view)
20     controller.start()
21
22
```



5

PROGRAMMING LANGUAGE

LET'S CODE NOW

- Basic rules
 - Take advantage of the large community: someone has already encountered your problem
 - Take advantage of your interdisciplinary promotion too
 - Read and understand the error messages!
- Course divided into steps
- Each part of the course made of theoretical knowledge + exercises
- You should challenge yourself and others with additional exercises and answers
- Make a compilation of the exercises that would grow each year
 - For each part of the course
 - Possibly by level of difficulty
- Some parts are more advanced and not useful at first, left to your personal work

LET'S SWITCH TO JUPYTER NOTEBOOK

Jupyter Notebook

The Python notebook includes a kernel that runs your code. So to stop working with your notebook, click on File/Close and Halt. This will stop the kernel too.

Cells

Notebook are composed of cells. The default cell is for code but you can change it to Markdown tu put text (such as this one). Here are how to format text to help you make your cells more readable:

- Use # to create Title1, ## to create Title2...
- Use -, * or 1. to create bullets
- Use ** to bold texte
- Use ` (backtick) to highlight text
- Use *** to create horizontal line
- Use two spaces at the end of a line to return to next line without creating a new paragraph

And read the help section in the menu ! ;-)

EUR DIGISPORT

DIGITAL SPORT SCIENCES



L'Ecole Universitaire de Recherche **DIGISPORT** bénéficie d'une aide de l'Etat gérée par l'**Agence Nationale de la Recherche** au titre du Programme d'Investissements d'Avenir portant la référence **ANR-18-EURE-0022**

Avec le soutien de la **Région Bretagne** et **Rennes Métropole**





DIGISPORT

GRADUATE SCHOOL