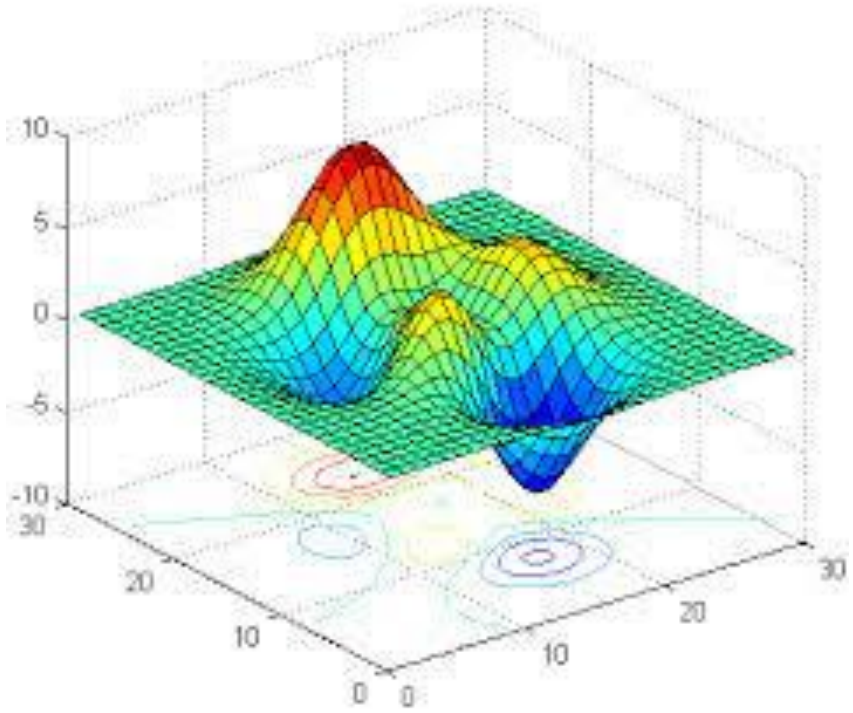
**Khemis Miliana University**
**Faculty of Sciences of Matter and Computer Science**
**Department of Physics**

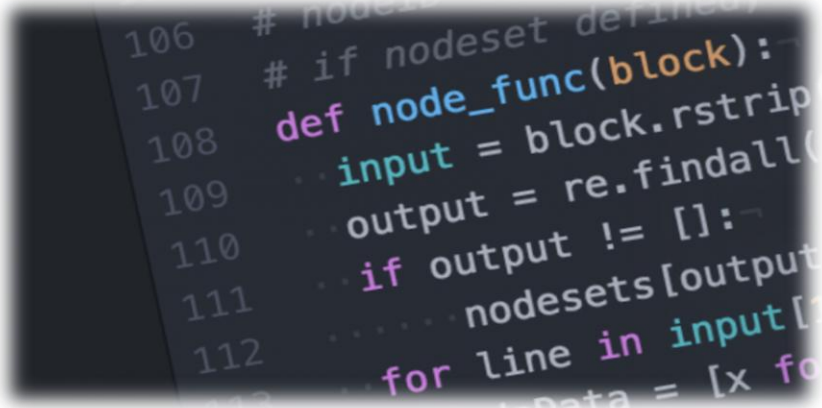


# Numerical Methods & Scientific Programming

***Dr. Salah-Eddine BENTRIDI***
***s.bentridi@univ-dbkm.dz***
***Univ. Khemis-Miliana***

# Content of the program

- Chapter 01: ***Initiation to a programming language (Python)***
  - ***Hands-on-Pythn; Basics of Python***
- Chapter 02: ***Numerical Integration***
  - ***Trapezoidal rule;  Simpson's method***
- Chapter 03: ***Numerical Solution of equations***
  - ***Bisection method; Newton's Method***
- Chapter 04: ***Numerical resolution of differential equations***
  - ***Euler's method;  Runge-Kutta method***
- Chapter 05: ***Numerical resolution of linear systems***
  - ***Gauss method, Gauss-Seidel method***





# Python Basics

***These slides are prepared thanks to available sources***
***"Python Crash Course", by Ying Liu***
***https://pythonbook.org***


***Dr. Salah-Eddine BENTRIDI***
***s.bentridi@univ-dbkm.dz***
***Univ. Khemis-Miliana***

## *Chapter 01: Initiation to a programming language*

# Say “Hello World!”


It is a tradition (since C programming language book) to display **"Hello World!"** in a computer console (terminal).

print is a Python built-in function name. A function performs some tasks.

Here it prints the message (a string) **"Hello World!"** to the console.

```
print("Hello World")
```

# Function Call


A function, like a computer, processes its input and performs some tasks.

A function name followed by a pair of parentheses is a function call – the computer runs/executes the function.

For example, `print("Hello World")`

The string `"Hello World!"` is the input to the print function.

A string is written inside a pair of double quotes or a pair of single quotes. A pair of single quotes also works:

```
print('Hello World!').
```

# Python is Simple and Easy


In Python:

```
print("Hello World")
```

In Java:

```
public class Main
{
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

# A Computer Can Compute


It is a good idea to put a space before and after the + operator to make it easy to read.

Python interpreter will ignore those spaces:

```
2+3
```

Or

```
2 + 3
```

Output:

5

# Two Execution Modes


You run Python code in one of two modes:

**Interactive**

- Type and run code directly inside the Python Interpreter.
- It is convenient to try small code snippets.

**Script**

- Type code into a file (script file), then run the script file as:

```
> python3 filename.py
```

- It is used for most Python applications. However, you need I/O functions such as built-in function `print()` to see the output.

# More Computations


Python supports common mathematic operations.

Use # to mark a start of comments, either at the start of a line or middle of a line.

Comments work as help text for developers. Python Ignores comments.

```
# this is a comment, ignored by Python Script
1 + 2

5 - 3

3 * 5 - 2

7 / 5

2 ** 10 # exponential

7 // 5 # quotient

7 % 5 # remainder
```

# Operation Orders


It is a best practice to use parenthesis for operation order.

```
# they are optional here but better
(3 * 5) - 2

# subtraction before multiplication
3 * (5 - 2)

# the order of operation is not clear
2 ** 3 ** 2

# no ambiguity, pay attention to code format
(2 ** 3) ** 2

# no ambiguity
2 ** (3 ** 2)
```

# Complex Math


Python put more math functions in the **`math`** module. A module is a Python file that provides multiple functions.

You need to import it first before use.

A function may return a value that can be used as an input to other functions.

```
import math

print(math.sqrt(2))

print(math.log(2))
```

# Python Functions


There are three types of sources for Python functions:

- Built-in: they are essential functions that are part of the Python programming language. You can call these functions directly.
- Standard library: these are common functions that come with Python installation but you must `import` them before use them.
- Installation packages: you need to download and install them online first, then `import` them before use. For example, all AI packages.

# Built-in Functions


- Python has about 80 built-in functions such as “print”, “input”, “abs”, “sum” etc., for the very basic programming tasks required by almost all programs.
- You can use these functions directly.
- The operators such as “+”, “-”, “*” etc. are another type of built-in functions that are presented as natural math operations.
- List of built-in functions: https://docs.python.org/3/library/functions.html

# Standard Library


- Python provides more functions in so-called standard library.
- Functions are organized into service modules such as `math`, `statistics`, `os`, `time` etc., that provide additional common functions.
- You need to `import` a module first to use its functions.
- You call the function using its module prefix such as `math.sqrt(2)`.
- Documentation: https://docs.python.org/3/library/

# Comparison


The comparison result is a Boolean value that is either **`True`** or **`False`**

```
import math

3 > 2
3 == 2
3 != 2
(7 / 5) >= 1
math.sqrt(3) > 2

# can be chained
2 > math.sqrt(3) > math.sqrt(2)
```

# Logical Operations


There are three basic logical operations: **`not`**, **`and`**, and **`or`**.

These operations work on boolean values and the results are either **`True`** or **`False`**.

```
not (3 == 2)

not (3 != 2)

(3 > 2) and ((7 / 5) > 3)

(3 == 2) or (3 >= 2)
```

# Literals Are Not Enough


- So far, the operation data are *literals* such as numbers: `5`, `3` or text `"Hello World!"`
- But you need something when you want to
  - Name a data !!!
  - Store the result of an operation
  - Use a short reference in multiple places
  - Refer a non-existing data from input

# Variables


A Python variable is a symbolic name that is a reference to a data.

Examples:

- Name a data: `pi = 3.141592653589793`
- Store the result of an operation: `diameter = 2 * radius`
- Use a short reference in multiple places: `circumference = pi * diameter`
- Refer a non-existing data from input:

```
radius_input = input("Please input radius: ")
```

# A simple programing exercise


```
pi = 3.14159

# display a message and take user input
radius_input = input("Please input radius: ")

# convert input string into a float number
radius = float(radius_input)

diameter = 2 * radius
circumference = pi * diameter

print(circumference)
```

# Create a Python Program


You create a program by writing a code/script file.

Python code file has a `.py` postfix.

A script file usually has multiple lines.

Python runs the script file line by line, from top to the bottom.

IDE (Integrated Development Environment) such as VS Code provides syntax highlight and checks code errors.

# Better Output


You can use an *f-string* (formatted string literals) to create a text string from variables.

An f-string is created by prefixing the string with **`f`** or **`F`** and writing variable as **`{variable}.`**

For example:

- **`f"The circumference of radius {radius} is {circumference}."`**

You can also add format modifiers:

- **`f"The circumference of radius {radius:.2f} is {circumference:.4f}."`**
- The **`:.2f`** is a format modifier that shows 2 places after the decimal.

# Better Output


```
import math
# display a message and take user input
radius_input = input("Please input radius: ")
# convert input text into a float number
radius = float(radius_input)
diameter = 2 * radius


# Python has a predefined pi variable in the math module
circumference = math.pi * diameter
print(f"The circumference of radius {radius:.2f} is
  {circumference:.4f}.")
```

# Life is Not Linear


You want take different actions based on different conditions/assumptions.

_____Python uses an *indentation* (4 spaces) to mark a *code block*.

Python runs a code block if its above condition is True.

Go back to same level as the if statement.

# Life is Not Linear


```
# display a message and take user input
score_input = input("Please input your score: ")
# convert input text into an integer
score = int(score_input)
if score >= 900:
    # nice
    grade = "A"
elif score >= 800:
    # ok
    grade = "B"
else:
    # too bad
    grade = "C"
# back to normal
result = f"Grade is {grade}."
print(result)
```

# Life is Full of Repetitions


You keep studying and practicing until ...

Python code is easy to understand even if it is the first time you see the following code.

A code block may have more than one lines of code at the same indentation level. They are executed top-down.

# Life is Full of Repetitions


```
score_input = input("Please input your score: ")
score = int(score_input)

while score < 900:
    print("Please keep studying and practicing....")
    score_input = input("Please input new score: ")
    score = int(score_input)

# back to normal
print("Great, you made it!")
```

# It is Turing Complete


You have learned all the fundamental programming concepts:

- Variables
- Sequential execution
- Branch (if else)
- Loop (while …)

# Real World is Complex


You need to deal with many many types of compound data:

- A class has 50 students.
- Each student has name, age, major, and GPA.
- An AI model may use billions of record.
- ...

# List


A sequence of values can be written as a list of comma separated items: in a pair of square brackets.

You can access an item by its index number in a pair of square brackets, starting from 0.

If the index is out of range, your code crashes with an IndexError exception

```
scores = [930, 790, 367, 827]
grades = ["A", "B", "C", "D", "F"]

print(scores[0])
print(grades[4])
print(grades[6])
```

# For Loop


Python has a **for** loop to iterate over a list of items.

```
scores = [930, 790, 367, 827]
# in each iteration of this for loop
# the score is assigned the next value in the list
# starting from index 0 to the last one.
for score in scores:
    if score >= 900:
        print(f"Great, you got an A.")
    else:
        difference = 900 - score
        print(f"You need {difference} points to get an A.")

print("Done.")
```

# Dictionary


A dictionary has a set of items.

Each item is a pair of **`key: value`**.

```
# a dictionary has a comma-separated list of key:value pairs within the
braces
student = {"Name": "Alice", "Age": 20, "Major": "IS"}

# use a key to read or change a dictionary value
print(student["Name"])
student["Age"] = 21
print(student["Age"])

# access every key and its value
for key in student.keys():
    message = f"Key {key} has a value of {student[key]}"
    print(message)
```

# Define a Function


**You want define a function that can be reused for different reasons:**

- **No redundant code.**
- **Function name shows its purpose.**
- **It can  be used in other places.**

```
import math
# define a function with a parameter radius
# the body is a code block to do the real work
def get_circumference(radius):
    diameter = 2 * radius
    circumference = math.pi * diameter
    return circumference

radiuses = [1, 5, 10]
for radius in radiuses:
    # here the radius is the argument of the function call
    circumference = get_circumference(radius)
    print(f"The circumference is {circumference:.4f}.")
```

# Function Concepts


- Functions name: describe the purpose of the function.
- Function call: execute the function by appending a pair of parentheses after the function name.
- Parameters: zero, one or more inputs used by the function body.
- Arguments: the actual input data used in function call.
- Function body: the code block in function definition. It is executed in function call.
- Return value: the output of a function call.

# Objects and Types


In Python, all data are *objects*.

Every object has a ***type*** that determines the valid operations of the object.

You can use built-in function ***type()*** to get an object's type.

For example:

```
type("hi") # <class 'str'>
type(3) # <class 'int'>
```

# Method and Attribute


Function defined inside a type is called a *method*. You call a method using the *dot notation*.

An object may contain data that is called an *attribute*. You use a dot notation to access an attribute.

For example, the **`math`** module is an object that has an object called **`pi`**

```
print("hi".title())

import math

print(math.pi)
```

```
Hi
3.141592653589793
```

# Computational/Algorithmic Thinking


- Arithmetic and logic operations
- Variables
- Control flow
  - Sequential execution
  - Branch: if ... else ...
  - Loop: for, while
- Composite data
- Function
- Type (method and attribute)

# Get deeper


The following both education websites, dedicated for python learning, are highly recommended:

- https://pythonbook.org
- https://marko-knoebl.github.io/slides/python-all-en.html