Scientific Programming

Course Preface & Learning Guide

2nd Year Licence Physics — Semester 3

Prepared by: Dr OUAAR Fatima
MCA in Applied Mathematics — University of Biskra

1. Welcome to Scientific Programming

Welcome to **Scientific Programming**. This course is designed for second-year Physics students who are beginning to use programming as a practical tool for scientific work.

You are not expected to be an experienced programmer at the beginning of the semester. We will therefore progress **step by step**: first understanding the basic idea, then observing a short example, and finally trying a similar problem ourselves.

The main objective is not to turn you into professional software developers. The objective is to help you become comfortable with Python as a **scientific working tool** that can support calculations, data manipulation, visualization, and the exploration of physical problems.

A simple philosophy:
Understand → **Observe** → **Practice** → **Apply**

2. Course Overview and Teaching Philosophy

Scientific Programming introduces Physics students to Python as a practical tool for scientific computing. The course begins with simple data structures and gradually moves toward basic numerical and data-handling tools. Its philosophy is to keep explanations short, concrete, and closely connected to practice. Each concept is first introduced through a brief explanation, then illustrated with a small example, and finally reinforced through guided and independent exercises. The emphasis is on understanding the logic of a program rather than memorizing commands. Throughout the course, Python is presented as a useful instrument for calculations, data organization, visualization, and the first steps toward computational work in Physics.

Main idea:
The course develops a **progressive scientific programming mindset**: start from simple structures, learn how to manipulate numerical data, visualize results, and finish with elementary search and sorting algorithms.

3. How will we work?

This is a **practical work (TP) course**. According to the official Semester 3 table, Scientific Programming has **15 hours** of volume, with **1 hour of TP per week**. The course has **coefficient 1** and **1 credit**. The official evaluation mode is **50% continuous assessment and 50% examination**.

Each practical session follows a predictable structure so that you always know what to expect.

1. **Brief theory:** a small amount of explanation before using the computer.
2. **Guided examples:** short Python examples are studied and explained.
3. **Hands-on practice:** you run, modify, and test code.
4. **Independent exercises:** a few exercises are left for you to solve yourself.
5. **Quick visual review:** a short visual summary helps you remember the essential ideas before assessment.

Important: You do not need to understand everything at once. In programming, making small mistakes is part of learning. Read the error message, identify the line concerned, correct it, and run the program again.

4. Course Content: Main Chapters

The course content follows the official scientific programming syllabus and is organized into three main chapters.

Chapter 1 — Dictionaries and Sets in Python

- Dictionaries: creation, access, manipulation, adding, deleting, and modifying key–value pairs.
- Sets: creation, basic operations, and common uses of sets and dictionaries.

Chapter 2 — Introduction to Scientific Tools in Python

- **NumPy:** arrays, creation, indexing, and basic operations including arithmetic and comparisons.
- **Matplotlib:** introduction to figures and simple plots such as points and lines.
- **CSV files:** reading simple CSV files using Python's `csv` module.
- **Pandas and DataFrames:** introduction to DataFrames, reading data from lists or CSV files, and basic display/manipulation.
- **Simple statistics:** calculation of the mean, median, and standard deviation using Python.
- **Mathematical functions:** use of the `math` module for common functions such as sin, cos, log, exp, and $\sqrt{}$.

Chapter 3 — Basic Search and Sorting Algorithms

- Search algorithms: linear search and an introduction to binary search.
- Sorting algorithms: bubble sort, insertion sort, and selection sort.

From data structures → scientific tools → basic algorithms

The practical sessions are organized progressively from these chapters while keeping the level appropriate for second-year Physics students.

5. Why Python for Physics?

Physics is increasingly connected with computation. A short Python program can help us perform repetitive calculations, organize measurements, visualize a relationship between physical quantities, and inspect numerical data.

For this reason, we will use examples that are close to the way a Physics student thinks: **numbers, arrays, measurements, functions, tables, and graphs**. The programming syntax will always serve the scientific idea rather than replace it.

Think like a physicist, program like a scientist.
Python is not the final goal. It is a tool that can help you calculate, investigate, visualize, and communicate scientific results.

6. The Role of the Moodle Resources

After each session, the Moodle space will contain the material for that session separately. This is intentional: you should not have to search through one large document to find what you need.

- **TP PDF:** the main practical document for the session.
- **Visual Review:** a short, colourful file combining a **Mind Map**, a **Quick Review**, and a small **Quick Practice**.
- **Optional Python Companion:** additional material for students who want to go a little further. It is a support resource, not a replacement for the official TP content.

Before a test: Start with the Visual Review, then revisit the relevant TP examples and exercises. The Visual Review is designed for quick recall, not for replacing practice.

7. Evaluation and Grading

According to the official Semester 3 table, Scientific Programming is evaluated by:

Component	Weight
Continuous assessment	50%
Final examination	50%

During the practical work, there will be **five continuous-assessment tests**. Each test is marked on **4 points**, giving a total of **20 points** for the continuous-assessment component.

Thus, if the five test marks are

$$T_1, T_2, T_3, T_4, T_5 \qquad (0 \leq T_i \leq 4),$$

the continuous-assessment mark is

$$C = T_1 + T_2 + T_3 + T_4 + T_5, \qquad 0 \leq C \leq 20.$$

The final course mark combines the continuous-assessment component and the examination component according to the official **50%–50%** weighting.

Five tests, five opportunities to practice.
The purpose of the short assessments is to encourage regular work throughout the semester rather than waiting until the end of the course.

8. What Is Expected from You?

To progress comfortably, you are encouraged to:

- attend and participate in the practical sessions;
- run the examples yourself instead of only reading them;
- type and modify short programs;
- keep your own notes about commands and errors;
- attempt the unsolved exercises before looking for help;
- review the Visual Review before each assessment;
- ask questions whenever a Python instruction or scientific idea is unclear.

No previous mastery of Python is assumed. What matters most is **regular practice**.

9. A Small Learning Rule

Do not just copy the code.

Read it → Predict the result → Run it → Change something → Run it again.

This simple habit will gradually turn Python from something unfamiliar into a useful scientific instrument.

10. A Final Message

You do not need to become an expert programmer in ten sessions. The goal is to build a **solid first foundation** and, more importantly, to develop the confidence to continue learning.

Some students may discover that they enjoy programming and wish to explore more advanced Python later. For that reason, optional resources will be available on Moodle for students who want to extend their skills beyond the core practical work.

Learn a little. Practice a little. Improve every week.

Python is a tool; your scientific curiosity is what gives it a purpose.

Prepared by: Dr OUAAR Fatima
MCA in Applied Mathematics — University of Biskra
Scientific Programming — 2nd Year Licence Physics