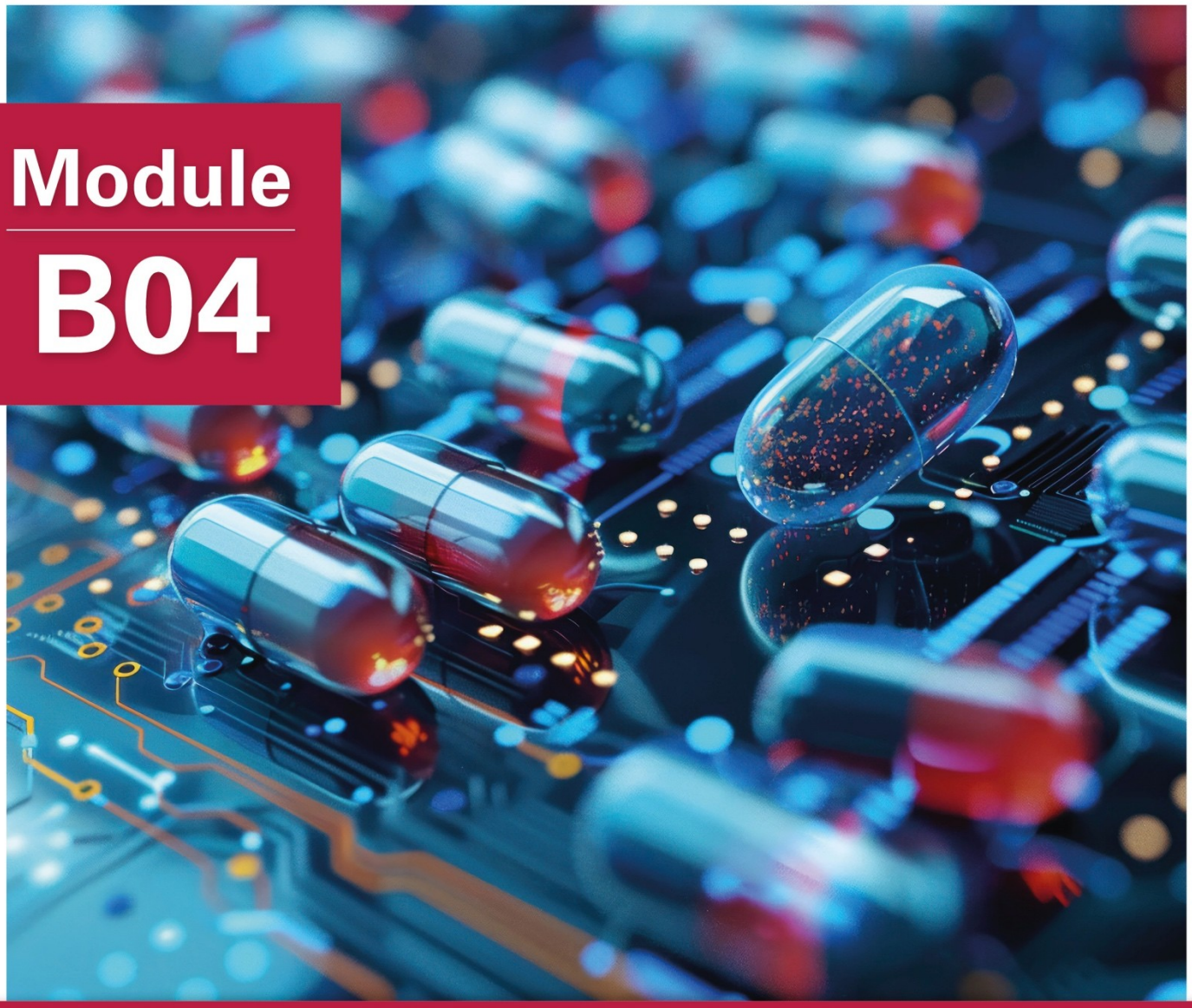
AfricaCDC
Centres for Disease Control and Prevention

Safeguarding Africa's Health



Module B04

Bioinformatics Foundational Course

Introduction to Programming

NGS Academy for the Africa CDC

ASLM
AFRICAN SOCIETY FOR LABORATORY MEDICINE

H3ABioNet
Pan African Bioinformatics Network for H3Africa

ASLM ACADEMY

Module B04

Introduction to Programming

back to the table of modules

Module last updated:
December 2024

Suggested or approximate number of sessions	5-6
Suggested or approximate total learning time	20-24 hours (practice required)
Target audience	Bioinformaticians
Delivery format	Lectures, videos, practicals with data examples, code review sessions
Level of the module	Introductory to Intermediate

Contributors

Hocine Bendou, George Githinji, John Juma, Shahiid Kiyaga, Perceval Maturure, Kennedy Mwai, Nicola Mulder and Verena Ras.

Suggested prerequisite module(s)

- Module B01. Introduction to Unix/Linux, Command Line, and Shell Scripting
- Module B02. Introduction to Version Control

Module description

Programming tasks include developing new algorithms, managing and analyzing large volumes of data generated in research projects, integrating diverse datasets to tackle complex problems, and performing other computational tasks essential to scientific endeavors. This module aims to assist novice programmers in writing modular code for data analysis. The module recommends teaching language-agnostic programming principles, but examples can be introduced using different programming languages, such as Bash, Python, or R. These principles include automation with loops and encapsulation with functions, as well as best practices for scientific software development. These practices are rooted in extensive research and practical experience and aim to enhance scientists' productivity and the reliability of their software. In this module, participants are also introduced to the following topics and/or concepts:

- Basic data types such as integers, strings, and floating-point numbers
- Using variables -value to assign a value to a variable in order to record it in memory. Variables are created on demand whenever a value is assigned to them.
- Data structures & algorithms
- Control flow/conditionals and loops e.g if, elif, else statements and loop or while loops
- Introduction to a programming interfaces, e.g an IDE or a Jupyter Notebooks
- Debugging and troubleshooting
- Using print(something) to display the value of something.
- Using # to provide some kind of explanation or to add comments to programs.
- Built-in functions
- Reading and writing I/O text files
- Working with modules for data science and visualization, including Numpy, Pandas, and Matplotlib.
- Writing scripts and introducing the concept of object-oriented programming.
- Introducing Biopython using real genomics data.
- How to find, load, and run Biopython modules.
- Parallelizing code
- Work with AI-generated code

Summary of best practices to teach trainees

- Write programs for people, not computers.
- A program should not require its readers to hold more than a handful of facts in memory at once.
- Make names consistent, distinctive, and meaningful.
- Make code style and formatting consistent.
- Let the computer do the work.
- Make the computer repeat tasks.
- Save recent commands in a file for re-use.
- Use a build tool to automate workflows.
- Make incremental changes.
- Work in small steps with frequent feedback and course correction.
- Use a version control system.
- Put everything that has been created manually in version control.
- Don't repeat yourself (or others).
- Every piece of data must have a single authoritative representation in the system.
- Modularize code rather than copying and pasting.
- Re-use code instead of rewriting it. Plan for mistakes.
- Add assertions to programs to check their operation.
- Use an off-the-shelf unit testing library.
- Turn bugs into test cases. Use a symbolic debugger.
- Optimize software only after it works correctly.
- Use a profiler to identify bottlenecks.

- Write code in the highest-level language possible.
- Document design and purpose, not mechanics.
- Document interfaces and reasons, not implementations.
- Refactor code in preference to explaining how it works.
- Embed the documentation for a piece of software in that software.
- Collaborate. Use pre-merge code reviews.
- Use pair programming when bringing someone new up to speed and when tackling particularly tricky problems. Use an issue-tracking tool.

Module learning outcomes

On completion of this module, participants will have a basic knowledge of, or will be able to:

- Write effective and efficient programmes
- Work within a programming interface, e.g. Jupyter Notebooks
- Understand data structures & algorithms
- Use the different functions and libraries
- Troubleshoot and debug code
- Follow software best practices

Module assessments

Module practical: Practical available on the ASLM platform
Module quiz: Assessment questions available on the ASLM platform

Module resources

- PLOS| Article - Best Practices for Scientific Computing
- PLOS | Article - Good enough practices in scientific computing
- SIB-SWISS | GitHub - Training Collection
- The Carpentry | GitHub - Programming with Python
- The Carpentry | GitHub - Analyzing Patient Data
- Sanfoundry | Webpage - Python MCQ (Multiple Choice Questions)
- Javatpoint | Webpage - Python MCQ (Multi Choice Questions)
- Biopython | Webpage - Biopython Tutorial & Cookbook
- GeeksforGeeks | Webpage - Python MCQ (Multiple Choice Questions) with Answers
- Interviewbit | Webpage - Python MCQs With Answers

Acknowledgements

We would like to thank the following individuals, in alphabetical order of last name, for their valuable time and effort spent in designing (i.e., drafting, reviewing, and refining) this module: **Hocine Bendou, George Githinji, John Juma, Shahiid Kiyaga, Perceval Maturure, Kennedy Mwai, Nicola Mulder and Verena Ras**.

Furthermore, we would like to thank the following institutions, societies, journals and individuals from whom we sourced open-access resources, used in this module:
Biopython, Geeks for Geeks, Interviewbit, Javatpoint, Public Library of Science, Sanfoundry, The Carpentry; Silvano Aldà, Dhavide Aruliah, Jennifer Bryan, Charles Brown, Neil Chue Hong, Karen Cranston, Matt Davis, Iulian Dragan, Vincent Emonet, Robin Engler, Geert van Geest, Richard Guy, Steven Haddock, Kathryn Huff, Justin Kitzes, Ian Mitchell, Lex Nederbragt, Mark Plumbley, Smoretti, Jarosław Surkont, Tracy Teal, Ben Waugh, Ethan White, Greg Wilson, Paul Wilson, Vioannid.