FORMAN CHRISTIAN COLLEGE (A CHARTERED UNIVERSITY), LAHORE

DEPARTMENT OF COMPUTER SCIENCE

COURSE OUTLINE

COMP 102 — Programming Fundamentals

BS Computer Science | NQF Level 6 | Fall Semester 2026

Aligned with HEC Undergraduate Education Policy V 1.1 (2023)

1. Course Information

Course Code	COMP 102	**Course Title**	Programming Fundamentals
Credit Hours	4 (3–3)	**Contact Hours**	3 theory + 3 lab / week
Course Type	Computing Core (Mandatory Major)	**Semester**	Fall 2026 (Semester 1)
Pre-requisites	None	**Co-requisites**	None
Programme	BS Computer Science	**Knowledge Area**	Software Development Fundamentals
Language / Tool	Python 3.x, Thonny IDE	**Semester Weeks**	15 weeks (27 lectures)
Instructor	Fakhir Shaheen	**Email**	`fakhirshaheen@fccollege.edu.pk`
Office Hours	TR 9:15–11:00, W 9:00–9:30	**Classroom / Lab**	*As per timetable*

Note: Per HEC (2025), 1 credit hour of practical work requires three contact hours per week throughout the semester; hence the 4(3–3) weighting for this course.

2. Course Introduction / Description

This course introduces computational problem solving to first-semester students, progressing from expressions, variables and control flow to functions, structured data and functional abstraction. Python is the vehicle language: it exercises the underlying concepts; objects, binding, scope, abstraction, with minimal syntactic overhead while remaining current and market oriented.

Emphasis throughout is on *reasoning about* programs: tracing execution, arguing correctness, analysing cost — and not merely on producing code that runs. As a pre-requisite to much of the later programme (Object Oriented Programming, Data Structures, Algorithms, Database Systems), students are advised to achieve the Course Learning Outcomes to the maximum possible level.

3. Programme Learning Outcomes (PLOs)

The PLOs below are adopted from the HEC Revised Curriculum for Computer Science (2025) and the Seoul Accord graduate attributes.

PLO	Title	Statement
PLO-1	Academic Education	Prepare graduates as computing professionals.
PLO-2	Knowledge for Solving Computing Problems	Apply computing, mathematics and domain knowledge to abstract and conceptualise computing models from defined problems.
PLO-3	Problem Analysis	Identify, formulate and solve complex computing problems, reaching substantiated conclusions from first principles.
PLO-4	Design / Development of Solutions	Design and evaluate solutions, systems, components or processes meeting specified needs, with due regard for health, safety, cultural, societal and environmental factors.
PLO-5	Modern Tool Usage	Create, select and apply appropriate techniques, resources and modern computing tools, understanding their limitations.
PLO-6	Individual and Team Work	Function effectively as an individual and as a member or leader of diverse, multi-disciplinary teams.
PLO-7	Communication	Communicate effectively with the computing community and society at large through reports, documentation, presentations and clear instructions.
PLO-8	Computing Professionalism and Society	Assess societal, health, safety, legal and cultural issues, and the responsibilities these place on professional computing practice.
PLO-9	Ethics	Commit to professional ethics, responsibilities and norms of computing practice.
PLO-10	Life-long Learning	Recognise the need, and have the ability, to engage in independent, continual professional learning.

4. Course Learning Outcomes (CLOs)

By the end of this course the student will be able to:

CLO No.	Course Learning Outcome	Bloom's Taxonomy	PLO
CLO-1	**Explain** core programming concepts: objects, variables, expressions, data types, control structures, functions and Python's built-in collections.	BT2 (Understand)	PLO-2
CLO-2	**Write, execute and debug** Python programs using sequential, conditional, iterative and modular constructs, following coding best practices.	BT3 (Apply)	PLO-5
CLO-3	**Develop** algorithms — search, approximation, recursion, sorting — to solve computational problems, and reason about correctness and order of growth.	BT3 (Apply)	PLO-3
CLO-4	**Design and implement** modular, reusable solutions, selecting and combining algorithmic strategies — exhaustive search, bisection, recursion, simulation — appropriate to the problem.	BT6 (Create)	PLO-4
CLO-5	**Analyse** program behaviour and troubleshoot errors using environment diagrams, assertions, exceptions and systematic testing.	BT4 (Analyse)	PLO-3

CLO–PLO Articulation (Mapping) Matrix

Strength of contribution: **H** = High (directly taught and directly assessed), M = Medium (taught and partially assessed), L = Low (introduced or reinforced), – = not addressed at this level.

CLO	PLO1	PLO2	PLO3	PLO4	PLO5	PLO6	PLO7	PLO8	PLO9	PLO10
CLO-1	L	**H**	M	–	–	–	–	–	–	L
CLO-2	L	M	–	L	**H**	–	–	–	–	M
CLO-3	L	M	**H**	M	L	–	–	–	–	L
CLO-4	L	M	M	**H**	L	–	L	–	–	L
CLO-5	L	L	**H**	L	M	–	–	–	L	M

Mapping rationale. **PLO-2** ← CLO-1 (core concepts); **PLO-3** ← CLO-3, CLO-5 (formulation, tracing, cost); **PLO-4** ← CLO-4 (designing solutions); **PLO-5** ← CLO-2 (IDE, debugger, testing). **PLO-1** is programme-level, hence Low throughout. **PLO-6** and **PLO-8** are not assessed here: all work is individual, and societal/legal impact belongs to later courses. **PLO-9** and **PLO-10** appear at Low via academic integrity and self-directed work.

5. Course Outline (Contents)

Problem solving and the von Neumann architecture; the role of the interpreter and linker; objects, types and expressions; variables, binding and assignment; strings and string operations; standard input/output; comparison and logical operators, Boolean expressions; conditional statements and execution flow; repetitive statements (`while`, `for`); functions, definition, calling, parameters, return

values, scope, the call stack, environment diagrams; tuples and lists, memory organisation, aliasing, cloning and mutation; list comprehensions and 2-D lists; searching and approximation *(linear search, bisection, Newton-Raphson)*; assertions, exceptions and defensive programming; dictionaries and hashing; measuring performance, order of growth (Big-O) and elementary sorting; classical ciphers and frequency analysis; recursion, base cases and the call stack; randomness and simulation *(random walks, Monte Carlo methods)*; plotting and data visualisation; testing, debugging and validation; type annotations as checkable specifications; profiling and performance tuning.

Coverage note: Extends the HEC-prescribed content for Programming Fundamentals. Pointers and memory allocation are treated through Python's object/reference model.

6. Weekly Lecture Plan

Wk	Lec	Date	Topic	Lab	Deliverable	CLO
1	1	Tue, 1 Sep	Introduction to Programming; Objects	—	—	CLO-1
1	2	Thu, 3 Sep	Expressions, Variables	OS navigation, command line, Thonny	—	CLO-1, 2
2	3	Tue, 8 Sep	Strings, Input/Output	—	A0 released	CLO-1, 2
2	4	Thu, 10 Sep	Boolean Expressions	Expressions, Variables	—	CLO-1
3	5	Tue, 15 Sep	Conditions	—	**Quiz 1**	CLO-2
3	6	Thu, 17 Sep	Problem Set 1	I/O, Boolean expressions	A0 due; A1 released	CLO-2, 3
4	7	Tue, 22 Sep	Iteration	—	—	CLO-2
4	8	Thu, 24 Sep	Problem Set 2	Conditions	—	CLO-2, 3
5	9	Tue, 29 Sep	Code Abstraction, Functions	—	**Quiz 2**	CLO-2, 4
5	10	Thu, 1 Oct	Environment Diagrams	Iteration	—	CLO-5
6	11	Tue, 6 Oct	Tuples, Lists	—	A1 due; A2 released	CLO-1, 2
6	12	Thu, 8 Oct	Aliasing, Cloning	Functions	**Quiz 3**	CLO-5
7	13	Tue, 13 Oct	Comprehensions, 2-D Lists	—	—	CLO-2, 3
7	14	Thu, 15 Oct	Problem Set 3	Lists	—	CLO-3
8	—	Tue, 20 Oct	**MIDTERM EXAMINATION**	—	A2 due; A3 released	CLO-1–3, 5
8	—	Thu, 22 Oct	*Fall Break — no class*	—	—	—
9	15	Tue, 27 Oct	Search, Approximation	—	—	CLO-3
9	16	Thu, 29 Oct	Assertions, Exceptions	Advanced Lists	—	CLO-5
10	17	Tue, 3 Nov	Dictionaries	—	**Quiz 4**	CLO-2, 3
10	18	Thu, 5 Nov	Counting Steps, Big-O	Search, Approximation, Exceptions	A3 due; A4 released	CLO-3
11	19	Tue, 10 Nov	Sorting	—	—	CLO-3
11	20	Thu, 12 Nov	Codes & Codebreakers	Dictionaries, Big-O	—	CLO-2, 3
12	21	Tue, 17 Nov	Recursion and the Call Stack	—	**Quiz 5**	CLO-3, 5
12	22	Thu, 19 Nov	Thinking Recursively	Sorting, Ciphers	—	CLO-3
13	23	Tue, 24 Nov	Randomness and Simulation	—	—	CLO-3, 4
13	24	Thu, 26 Nov	Plotting	Recursion	A4 due	CLO-2, 4
14	25	Tue, 1 Dec	Testing, Debugging; Type Annotations	—	**Quiz 6**	CLO-2, 5

Continued on next page. . .

Wk	Lec	Date	Topic	Lab	Deliverable	CLO
14	26	Thu, 3 Dec	Making Python Faster	Simulation, Plotting	—	CLO-3, 5
15	27	Tue, 8 Dec	Course Review & Exam Preparation	—	—	CLO-1–5
15	—	Thu, 10 Dec	**FINAL EXAMINATION**	—	—	CLO-1–5

7. Teaching and Learning Methodology

- **Interactive lectures** with live coding: every concept is demonstrated by writing, running and deliberately breaking a program in front of the class.
- **Finger exercises** punctuating each lecture: a short task with a 1–2 minute pause for students to attempt it before it is worked through.
- **Environment-diagram tracing** so that students can predict program state before executing it (the principal instrument for CLO-5).
- **Weekly closed labs** in which students implement, from a specification, the constructs covered in the preceding two lectures.
- **In-class problem sets (CCC)** solved collaboratively, modelling the process of decomposing an unfamiliar problem.
- **Graded programming assignments (A0–A4)** of increasing scope, each accompanied by a student-facing test file so that students learn to validate their own work.
- **Real-world hooks** anchoring each unit to a familiar system (recommendation feeds, playlists, game boards, spell-checkers, banking abstractions), to motivate the construct before formalising it.

8. Assessment Scheme and CLO Mapping

Assessment Instrument	Count	Weight (%)	CLOs Assessed
Programming Assignments (A0–A4)	5	15	CLO-2, CLO-3, CLO-4, CLO-5
Labs	13	10	CLO-2, CLO-5
Quizzes (Q1–Q6)	6 (best 5)	15	CLO-1, CLO-2, CLO-3
Midterm Examination	1	25	CLO-1, CLO-2, CLO-3, CLO-5
Final Examination *(comprehensive)*	1	35	CLO-1–CLO-5
Total		**100**	

CLO–Assessment Coverage

CLO	Quizzes	Assignments	Labs	Midterm	Final
CLO-1	✓	–	–	✓	✓
CLO-2	✓	✓	✓	✓	✓
CLO-3	✓	✓	–	✓	✓
CLO-4	–	✓	✓	–	✓
CLO-5	–	✓	✓	✓	✓

Every CLO is assessed by at least three independent instruments, and every CLO is represented in the final exam, so that attainment can be computed per CLO for OBE reporting.

9. Grading Policy

Grading is **criterion-referenced**: marks reflect demonstrated attainment of the Course Learning Outcomes and are not curved to a predetermined distribution. Letter grades are awarded on the university's published letter-grade scale, under which the minimum passing aggregate is **60%**. Assessment instruments are blueprinted against the CLOs and moderated before use, so that the standard represented by each grade is consistent across sections and semesters.

For Outcome-Based Education reporting, a CLO is deemed *attained* when at least 60% of the enrolled students score 50% or above on the items mapped to that CLO; CLOs falling below this threshold are recorded in the course file with a corrective action for the next offering.

Per HEC policy, a student acquiring less than 2.00/4.00 GPA in a semester but passing all papers is promoted on probation, subject to achieving above 2.00 GPA in the following semester.

10. Reference Materials

Text Book

1. John V. Guttag, *Introduction to Computation and Programming Using Python: With Application to Computational Modeling and Understanding Data*, 3rd ed., MIT Press, 2021.

Reference Books (or any other standard and latest books)

2. Tony Gaddis, *Starting Out with Python*, 5th ed., Pearson, 2021.
3. Allen B. Downey, *Think Python: How to Think Like a Computer Scientist*, 3rd ed., O'Reilly, 2024.

Online Resources

6. The Python Tutorial — `https://docs.python.org/3/tutorial/`
7. PEP 8 — Style Guide for Python Code — `https://peps.python.org/pep-0008/`
8. Python Tutor (online environment-diagram visualiser) — `https://pythontutor.com/`

11. Course Policies

Attendance Requirement — Mandatory

A student may miss at most 5 lectures. A sixth absence bars the student from the final examination **regardless of all other performance**.

Academic Integrity.
All graded work is individual unless explicitly stated otherwise. Discussion of ideas is encouraged; submission of another person's code, from a classmate, a repository, or a generative AI tool — as one's own is plagiarism and is reported under the university's disciplinary policy. Students must be able to explain any line of code they submit.

Use of AI Assistants.
Permitted for explanation and for debugging code the student has already written; not permitted for generating solutions to assessed work. Any AI use must be disclosed in the submission header.

Late Submission.
Assignments **lose 10% per day** late, **up to three days**; **no credit afterwards**. Quizzes and labs cannot be made up except on documented medical or university grounds.

Communication.
Course announcements, slides, labs and assignment specifications are distributed through the course website.