GANPAT UNIVERSITY

FACULTY OF DIPLOMA ENGINEERING

Programme	Diploma in Computer Engineering/ Information Technology/Electronics and Communication		
Semester	III	Version	1.0.0.0
Effective from Academic Year	2026-27	Effective for the batch Admitted in	JULY 2025
Course Code	1CEIT3103	Course Name	Python Programming

I. TEACHING-LEARNING AND ASSESSMENT SCHEME

Course Type	Course code	Course Title	Teaching & Learning Scheme									Examination Scheme							
			Credit				Actual Contact Hrs/week			SLH	Total Learning Hrs/Week	TH			PR			SLA	Total
			CL	TL	LL	Total	CL	TL	LL			CE	SEE	Total	CE	SEE	Total		
DSC	1CEIT3103	PYTHON PROGRAMMING	3	-	2	5	3	-	4	2	9	40	60	100	40	60	100	20	220

Abbreviation:	**CL-** Classroom Learning	**TL -** Tutorial Learning	**LL -** Laboratory Learning
	SLH - Self Learning Hours	**SLA -** Self Learning Assessment	**CE -** Continuous Evaluation
	SEE – Semester End Examination		

II. PRE-REQUISITES

Basic familiarity with mathematical logic and problem-solving skills is desirable.

III. INDUSTRY /EMPLOYER EXPECTED OUTCOMES

- Develop efficient, modular, and readable Python programs.
- Write programs for automation, data processing, and basic system-level tasks.
- Implement logic-based solutions using control structures, functions, and data structures in Python.
- Manipulate and transform data effectively using Python's built-in collections and comprehensions.
- Build applications by applying object-oriented programming principles, file handling, and exception management.

IV. COURSE LEARNING OUTCOMES

At the end of the course, students will be able to achieve the following course learning outcomes:

CO1. Understand the fundamentals of Python programming, including its history, features, syntax rules, variables, data types, operators, and basic input/output operations.
CO2. Implement control structures such as conditional statements, looping constructs, and control flow mechanisms to develop logic-based Python programs for real-world problem solving.
CO3. Develop modular and reusable Python programs using functions, recursion, modules, and effective string handling techniques for efficient program design.
CO4. Utilize Python's built-in data structures such as lists, tuples, sets, and dictionaries, along with comprehensions, for efficient data manipulation and transformation.
CO5. Apply object-oriented programming principles, file handling techniques, and exception handling mechanisms in Python to design robust, maintainable, and real-world applications.

V. THEORY LEARNING OUTCOMES AND ALIGNED COURSE CONTENT:

Name of Unit	Theory Learning outcomes (TLO's) aligned to CO's	Learning Content mapped with Theory Learning outcomes (TLO's)&CO's	Marks	Hours
Unit-1 **Introduction to Python, Data Types and Operators**	**TLO 1.1** Describe history and key features of Python. **TLO 1.2** Explain advantages and major applications of Python. **TLO 1.3** Install Python and set up the environment. **TLO 1.4** Use basic IDEs and	**1.1** Origin and evolution of Python **1.2** Key features and application areas **1.3** Python installation and environment setup	**9**	**8**

	development tools. **TLO 1.5** Apply Python syntax rules and dynamic typing. **TLO 1.6** Work with variables and built-in data types. **TLO 1.7** Perform type conversion operations. **TLO 1.8** Use different types of operators in expressions. **TLO 1.9** Understand operator precedence and associativity.	**1.4** IDEs: IDLE, VS Code, Jupyter Notebook **1.5** Syntax rules, indentation, keywords **1.6** Variables and built-in data types **1.7** Type conversion functions **1.8** Arithmetic, relational, logical and other operators **1.9** Operator precedence and associativity		
Unit-2 **Control Structures and Looping Techniques**	**TLO 2.1** Explain the need for conditional execution. **TLO 2.2** Implement simple and compound conditional statements. **TLO 2.3** Use multi-way decision-making constructs. **TLO 2.4** Apply while loop for indefinite iteration. **TLO 2.5** Use for loop with range and sequences. **TLO 2.6** Develop programs using nested loops. **TLO 2.7** Control loop execution using jump statements. **TLO 2.8** Solve real-world problems using control structures. **TLO 2.9** Identify and debug common control flow errors.	**2.1** Decision making concept **2.2** if and if-else statements **2.3** if-elif-else ladder **2.4** while loop **2.5** for loop and range() **2.6** Nested loops **2.7** break, continue, pass statements **2.8** Real-world problem solving **2.9** Common errors and debugging	**11**	**9**
Unit-3 **Functions, Modules and String Handling**	**TLO 3.1** Explain the concept and benefits of functions. **TLO 3.2** Define functions with parameters and return values. **TLO 3.3** Use different types of function arguments. **TLO 3.4** Understand variable scope and recursion. **TLO 3.5** Create and import user-defined modules. **TLO 3.6** Use important built-in modules. **TLO 3.7** Manipulate strings using operations and methods. **TLO 3.8** Apply different string formatting techniques. **TLO 3.9** Solve common string processing tasks.	**3.1** Functions – concept and advantages **3.2** Function definition and return **3.3** Positional, keyword, default arguments **3.4** Variable scope and recursion **3.5** User-defined modules **3.6** Built-in modules (math, random, datetime) **3.7** String operations and methods **3.8** String formatting techniques **3.9** Common string processing tasks	**13**	**10**
Unit-4	**TLO 4.1** Create and modify lists and tuples.	**4.1** List creation and modification	**15**	**10**

Lists, Tuples, Sets, Dictionaries and Comprehensions	**TLO 4.2** Apply list methods and operations. **TLO 4.3** Use tuples and their special properties. **TLO 4.4** Perform set operations. **TLO 4.5** Create and manipulate dictionaries. **TLO 4.6** Use list and dictionary comprehensions. **TLO 4.7** Work with nested data structures. **TLO 4.8** Compare mutability and performance of data structures. **TLO 4.9** Select appropriate data structure for given problem.	**4.2** List methods and operations **4.3** Tuple creation and properties **4.4** Set operations **4.5** Dictionary operations and methods **4.6** List and dictionary comprehensions **4.7** Nested data structures **4.8** Mutability and performance comparison **4.9** Data structure selection guidelines		
Unit-5 **Object-Oriented Programming, File Handling and Exception Handling**	**TLO 5.1** Explain core principles of object-oriented programming. **TLO 5.2** Define classes and create objects. **TLO 5.3** Implement constructors and instance methods. **TLO 5.4** Apply inheritance and polymorphism. **TLO 5.5** Implement encapsulation in Python. **TLO 5.6** Perform basic file input/output operations. **TLO 5.7** Use advanced file handling techniques. **TLO 5.8** Implement exception handling mechanisms. **TLO 5.9** Raise and handle custom exceptions.	**5.1** OOP principles **5.2** Class and object **5.3** Constructor and instance methods **5.4** Inheritance and polymorphism **5.5** Encapsulation and access control **5.6** File modes and basic operations **5.7** with statement and file methods **5.8** try-except-else-finally **5.9** Raising exceptions and custom exceptions	**12**	**8**

VI. LABORATORY LEARNING OUTCOME AND ALIGNED PRACTICAL			
SR. NO	**PRACTICAL/LABORATORY LEARNING OUTCOME(LLO)**	**PRACTICAL TITLES**	**RELEVANT COs**
1	**LLO 1.1** Install and set up Python environment **LLO 1.2** Use variables, data types and basic I/O	To install Python, configure IDE and develop programs using variables, data types, input() and print() functions	CO1
2	**LLO 1.3** Apply arithmetic, relational and logical operators **LLO 1.4** Use assignment, bitwise, membership and identity operators	To develop programs demonstrating all categories of operators with precedence rules	CO1
3	**LLO 1.5** Perform type conversion operations **LLO 1.6** Solve simple expression-based problems	To develop programs using type conversion functions and expressions involving multiple operators	CO1
4	**LLO 2.1** Implement conditional statements (if, if-else, elif)	To develop programs using if, if-else, if-elif-else and nested if (grading system, menu-driven selection, eligibility check)	CO2

	LLO 2.2 Use nested conditional constructs		
5	**LLO 2.3** Implement while and for loops **LLO 2.4** Apply loop control statements (break, continue, pass)	To develop programs using while, for, nested loops and control statements (series generation, pattern printing, search)	CO2
6	**LLO 2.5** Solve real-world problems using control structures **LLO 2.6** Debug common control flow errors	To develop and debug logic-based programs (number guessing game, simple calculator, ATM-like menu simulation)	CO2
7	**LLO 3.1** Define functions with parameters and return values **LLO 3.2** Use different types of function arguments	To develop modular programs using positional, keyword, default and variable-length arguments (*args, **kwargs)	CO3
8	**LLO 3.3** Implement recursive functions **LLO 3.4** Create and import modules (user-defined & built-in)	To develop recursive solutions (factorial, Fibonacci, GCD) and programs using math, random, datetime modules	CO3
9	**LLO 3.5** Perform string operations, slicing and methods **LLO 3.6** Apply string formatting techniques	To develop programs using string slicing, common methods and formatting (f-strings, .format(), %)	CO3
10	**LLO 4.1** Create and manipulate lists with comprehensions **LLO 4.2** Work with tuples and sets	To develop programs using list operations/comprehensions, tuples (immutability) and set operations	CO4
11	**LLO 4.3** Create and manage dictionaries **LLO 4.4** Use dictionary comprehensions	To develop programs using dictionary operations, methods and dictionary comprehensions	CO4
12	**LLO 4.5** Apply advanced comprehensions and lambda expressions **LLO 4.6** Solve data transformation problems	To develop efficient programs using list/dict/set comprehensions and lambda with map/filter/sorted	CO4
13	**LLO 5.1** Design classes, objects and constructors **LLO 5.2** Implement instance methods and self-parameter	To develop OOP programs demonstrating classes, objects, 'init' constructor and instance methods	CO5
14	**LLO 5.3** Apply inheritance and method overriding **LLO 5.4** Demonstrate polymorphism and encapsulation	To develop programs using single/multiple inheritance, method overriding, polymorphism and encapsulation concepts	CO5
15	**LLO 5.5** Perform file handling operations (text & CSV) **LLO 5.6** Implement exception handling and integrate with OOP	To develop programs for file read/write/append (text/CSV) using with statement + exception handling + OOP integration (e.g., Student/Inventory record system with file storage)	CO5

VII. SUGGESTED MICRO PROJECT/ASSIGNMENTS/ACTIVITIES FOR SELF LEARNING/SKILL DEVELOPMENT (SELF LEARNING)

Micro Projects (Mini Applications / Use Cases)

- Develop a Simple Calculator (Menu-driven using functions and control structures).
  Skills: Operators, conditional statements, loops, functions
- Create a To-Do List Manager.
  Skills: Lists/dictionaries, file handling, user input validation
- Build a Number Guessing Game.
  Skills: Random module, loops, conditional logic, exception handling
- Design a Student Marks Management System.
  Skills: OOP (classes, objects, inheritance), file I/O, data structures
- Develop a Basic Inventory Tracking System.
  Skills: Dictionaries, lists, file handling, comprehensions
- Create a Unit Converter (e.g., Length, Weight, Temperature).
  Skills: Functions, conditional statements, string formatting
- Implement a Password Strength Checker.
  Skills: Strings, loops, conditional checking, regular expressions (optional)

Self-Learning / Skill Building Activities

1. Create a free account on online platforms like LeetCode / HackerRank / CodeChef and solve at least 30 Python easy-level problems.
   Goal: Improve problem-solving and coding speed
2. Implement recursive solutions for problems like Factorial, Fibonacci, GCD/LCM, and Tower of Hanoi.
   Goal: Understand recursion and base cases
3. Practice data structure operations by building Employee Database or Book Inventory List applications.
   Goal: Master lists, tuples, sets, and dictionaries
4. Use online Python compilers/IDEs (Replit, Programiz, PythonAnywhere) to experiment with code anytime.
   Goal: Gain confidence in writing and testing code independently
5. Prepare a poster/flowchart explaining Python memory management (variables, objects, references).
   Goal: Visualize concepts like immutability and object references
6. Record a short screen video tutorial on "Creating Your First Python Program and Running It".
   Goal: Improve communication and teaching skills
7. Research and present (written/oral) how Python is used in real-world domains (Web, Data Science, Automation, AI).
   Goal: Understand industry relevance and career opportunities
8. Develop a menu-driven ATM Simulation Program.
   Goal: Integrate loops, conditionals, functions, and basic OOP
9. Explore and use command-line arguments with Python scripts (using sys.argv).
   Goal: Learn script automation and real-world execution
10. Document 10 common Python errors (e.g., NameError, TypeError, IndexError) and their fixes.
    Goal: Develop debugging skills and error-handling mindset
11. Build a Personal Portfolio Console App (display skills, projects, contact info using menus).
    Goal: Apply strings, functions, file storage, and structured programming

VIII. LIST OF INSTRUMENTS / EQUIPMENT / TRAINER BOARD

1	**Code Editors / IDEs:**

	Visual Studio Code / PyCharm Community Edition / IDLE / Jupyter Notebook |
2	**Python Interpreter:** Latest stable version of Python 3.x (from python.org)
3	**Online Compilers & Simulators** • Replit: https://replit.com • Programiz: https://www.programiz.com/python-programming/online-compiler/ • Google Colab: https://colab.research.google.com
4	**Collaboration / Documentation Tools:** • GitHub / Git: Version control and collaboration • Google Docs / Word: Report and assignment writing • diagrams.net / Excalidraw: Flowcharts, memory maps, logic design

IX. LIST OF REFERENCE BOOKS			
Sr.No	**Title**	**Author**	**Publication**
1	Let Us Python	Yashavant Kanetkar & Aditya Kanetkar	BPB Publications
2	Head First Python	Paul Barry	O'Reilly Media
3	Core Python Programming	Wesley J. Chun	Pearson Education
4	Python Programming	Reema Thareja	Oxford University Press

X. LINK OF LEARNING WEB RESOURCE	
1	Official Python Tutorial: https://docs.python.org/3/tutorial/index.html
2	W3Schools Python Tutorial: https://www.w3schools.com/python/
3	GeeksforGeeks Python Tutorial: https://www.geeksforgeeks.org/python-programming-language/
4	Real Python Basics Tutorials: https://realpython.com/tutorials/basics/
5	Programiz Learn Python Programming: https://www.programiz.com/python-programming

XI. SUGGESTED WEIGHTAGE TO LEARNING EFFORTS & ASSESSMENT PURPOSE							
Unit	**Unit Title**	**Aligned COs**	**Learning Hours**	**R-Level**	**U-Level**	**A-Level**	**Total Marks**
1	Unit-1: Introduction to Python, Data Types and Operators	CO1	8	5	3	1	9
2	Unit-2: Control Structures and Looping Techniques	CO2	9	3	4	2	11
3	Unit-3: Functions, Modules and String Handling	CO2, CO3	10	2	4	4	13
4	Unit-4: Lists, Tuples, Sets, Dictionaries and Comprehensions	CO3	10	2	3	5	15
5	Unit-5: Object-Oriented Programming, File Handling and Exception Handling	CO4, CO5	8	1	2	5	12
	Grand Total		**45**	**13**	**16**	**17**	**60**

XIII. COs AND POs AND PSOs MAPPING		
Course	**Programme Outcomes (POs)**	**Programme Specific Outcomes**

Outcome (Cos)								**(PSOs)**		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PSO1	PSO2	PSO3
CO1	3	2	1	2	0	1	0	3	1	1
CO2	3	3	2	2	0	1	0	3	1	2
CO3	3	3	3	3	1	2	1	3	1	2
CO4	3	3	2	3	1	2	1	3	2	2
CO5	3	3	3	3	2	2	1	3	2	3

Legends: - *3-High; 2-Moderate/Medium; 1-Slight/Low; 0-None*