Gyanmanjari
Innovative University

Syllabus
Gyanmanjari College of Computer Application
Semester-I (MCA)

Subject: Python Programming - MCA2XX11501

Type of course: Major Core

Prerequisite: Fundamentals of Computer Science: basic knowledge of any programming language.

Rationale:

This course provides a comprehensive and practical foundation in Python programming for MCA Semester-I students. It covers Python syntax, data types, control structures, functions, object-oriented programming, file handling, exception handling, modules, and standard libraries. Students gain hands-on experience through coding exercises and lab work, developing logical thinking and real-world problem-solving skills. The course bridges the gap between theoretical concepts and application development, preparing students for advanced computing, data science, and software development subjects. Practical exercises and simulations reinforce learning and enable students to write efficient, well-structured Python programs.

Teaching and Examination Scheme:

Teaching Scheme			Credits	Examination Marks		Total Marks
CI	T	P	C	SEE	CCE	
2	0	4	4	100	50	150

Legends: CI-Classroom Instructions; T-Tutorial; P -Practical; C –Credit; SEE -Semester End Examination; MSE-Mid Semester Examination; V–Viva; CA-Continuous Assessment; ALA-Active Learning Activities.

Course Content:

Sr. No	Course Content	Hrs.	% Weightage
1	**Introduction to Python Programming** **Theory Topics:** Introduction to Python: History, Features, and Applications of Python. Python Installation and Environment Setup (Anaconda, IDLE, VS Code, Google Colab). Python Interpreter – Interactive Mode and Script Mode. Python Keywords, Identifiers, and Naming Conventions. Variables and Data Types: int, float, complex, bool, str, NoneType. Type Conversion and Type Casting (implicit and explicit). Input and Output Functions: input(), print(), format(), f-strings. Operators: Arithmetic, Relational, Logical, Bitwise, Assignment, Identity (is, is not), Membership (in, not in). Operator Precedence and Associativity. Comments and Docstrings.	18	20%

Practical:

1. Install Python and write your first "Hello, World!" program using IDLE and VS Code.
2. Write a program to demonstrate all Python data types using the type() function.
3. Write a program to perform type conversion: int to float, float to str, str to int.
4. Write a program to accept user input and display name, age, and city using print() with f-strings.
5. Write a program to demonstrate all arithmetic operators with sample expressions.
6. Write a program to demonstrate relational and logical operators with boolean results.
7. Write a program to demonstrate bitwise operators (AND, OR, XOR, NOT, left shift, right shift).
8. Write a program to calculate simple interest and compound interest using variables.
9. Write a program to swap two numbers using a temporary variable and without a temporary variable.
10. Write a program to find the area and perimeter of a rectangle and a circle.
11. Write a program to check if a number is even or odd using the modulo operator.
12. Write a program using membership (in, not in) and identity (is, is not) operators on lists and variables.
13. Write a program to demonstrate operator precedence using complex arithmetic expressions.
14. Write a program to read temperature in Celsius and convert it to Fahrenheit and Kelvin.
15. Solve the unsolved questions on data types, operators, and expressions given in the reference book.

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Python Basics Proficiency Check :** Students will write a Python program demonstrating correct use of variables,data types, type conversion, all categories of operators (arithmetic, relational, logical, bitwise, identity, membership), and formatted output. The program must accept user input interactively, perform computations, and display results using f-strings or str.format(). Submit the script with sample output for evaluation.	20	-

2	**Python Environment Explorer :** Students install Python and set up an IDE (IDLE or VS Code). They write a program covering data types, operators, and input/output, explore built-in functions such as type(), id(), and abs(), and submit a PDF report with screenshots and observations on the GMIU Portal.	-	5
3	**LinkedIn Profile Enhancement :** Students update their LinkedIn profile by completing essential sections such as profile photo, headline, summary, education, and skills. They create and publish a LinkedIn post on any Python-related topic with a relevant image. Students compile their LinkedIn profile link, profile screenshots, and post screenshot into a single PDF report and upload it on the GMIU Portal	-	5
	Total	**20**	**10**

Examination Style:

Python Basics Proficiency Check [20 Marks] :
Students will receive a question from faculty to write a comprehensive Python program demonstrating fundamental concepts. The program should accept user input, demonstrate correct use of data types, perform type conversions, apply all operator categories (arithmetic, relational, logical, bitwise, identity, membership), and produce formatted output using f-strings or str.format(). The solution must reflect understanding of Python syntax, operator precedence, and proper variable usage. Submit the script with sample outputs for evaluation.

Python Environment Explorer [5 Marks] :
Students will design interactive Python programs using input(), print(), and formatted strings to solve basic real-world problems such as temperature conversion, interest calculation, and area computation. The assignment should include source code, outputs, and a short explanation of the logic used in each program.

LinkedIn Profile Enhancement [5 Marks] :
Students will create or update their LinkedIn profile by completing key sections such as profile photo, headline, summary, education, skills, and achievements. They will also design and publish a LinkedIn post on any Python-related topic, accompanied by a relevant self-created image, infographic, or visual representation. The assignment should include the LinkedIn profile link, screenshots of the updated profile, a screenshot of the published post, and a brief reflection on the importance of professional networking and knowledge

	sharing. Students must compile all evidence into a single PDF report and upload it on the GMIU Portal.		
2	**Control Structures and Functions** **Theory Topics:** Conditional Statements: if, if-else, if-elif-else, Nested if. Looping Constructs: for loop, while loop, Nested loops. Loop Control Statements: break, continue, pass, range() function. Comprehensions: List Comprehension, Dictionary Comprehension, Set Comprehension. Functions: Defining and Calling Functions, Function Arguments – Positional, Keyword, Default, Variable-length (*args, **kwargs). Return Statement and Multiple Return Values. Scope of Variables: Local, Global, Nonlocal. Lambda Functions. Higher-order Functions: map(), filter(), reduce(). Recursion and Recursive Functions. Built-in Functions: len(), max(), min(), abs(), round(), sorted(), enumerate(), zip(). Docstrings and Annotations. **Practical:** 16. Write a program using if-elif-else to assign a grade based on student marks. 17. Write a program to find the largest among three numbers using nested if. 18. Write a program to print the multiplication table of a number using a for loop. 19. Write a program to find the sum of natural numbers from 1 to N using a while loop. 20. Write a program to print the Fibonacci series up to N terms using a loop. 21. Write a program to print a pyramid star pattern using nested loops. 22. Write a program to demonstrate break, continue, and pass inside loops. 23. Write a program using list comprehension to generate even numbers from 1 to 50. 24. Write a function to calculate the factorial of a number using recursion. 25. Write a function with default arguments to compute the power of a number. 26. Write a program using *args and **kwargs to demonstrate variable-length arguments. 27. Write a program to use lambda functions with map() and filter() on a list. 28. Write a program to implement binary search using a recursive function. 29. Write a program to check whether a string is a palindrome using recursion. 30. Write a program using zip() and enumerate() to iterate over two lists simultaneously. 31. Solve unsolved questions on control structures and functions given in the reference book. **Evaluation Methods :** **Sr. No.** | **Evaluation Methods** | **SEE** | **CCE** 1 | **Shell Scripting Proficiency Check :** Students will write an advanced Python program using | 20 | -	18	20%

	conditional statements (if-elif-else), looping constructs (for/while with break/continue), and functions. The script must include at least one recursive function, use of lambda with map() or filter(), and demonstrate loop control statements with meaningful sample output.		
2	**Algorithm Visualization & Complexity Analysis Lab** : Students implement and trace sorting or searching algorithms (e.g., bubble sort, linear search, binary search) using Python functions and loops. They compare their behaviour using different inputs, provide screenshots of output, and upload an individual PDF report on the GMIU Portal.	-	10
	Total	**20**	**10**

Examination Style:

Control Flow & Function Writing Check [20 Marks] :
Students will write a Python program using arithmetic, relational, and logical operations through conditional statements and looping constructs. The script must use at least one recursive function, demonstrate lambda expressions with map() or filter(), implement loop control statements (break, continue), and provide sample output. Ensure the script clearly shows control flow, function design, and accurate program behaviour.

Algorithm Visualization & Complexity Analysis Lab [10 Marks] :
Students will implement and trace searching or sorting algorithms (e.g., linear search, binary search, bubble sort) using Python functions and looping constructs. They will compare the behaviour and performance of the algorithms using different input sizes and analyse their basic time complexity (Best Case, Average Case, and Worst Case). Students must include program code, sample outputs, execution observations, complexity comparison tables, and screenshots in an individual PDF report uploaded on the GMIU Portal.

3	**Data Structures in Python** **Theory Topics:** Lists: Creation, Indexing, Slicing, List Methods (append, extend, insert, remove, pop, sort, reverse, index, count, copy, clear). Lists as Stack and Queue. Tuples: Creation, Indexing, Tuple Methods, Immutability, Packing and Unpacking. Sets: Creation, Set Operations (union, intersection, difference, symmetric difference). Set Methods (add, discard, remove, pop, clear, copy, update). Dictionaries: Creation, Accessing and Updating, Dictionary Methods (keys, values, items, get, pop, update, copy, setdefault). Nested Dictionaries. Strings: String Methods (upper, lower, strip, split, join, replace, find, count, startswith, endswith). String Slicing. String Formatting (%-formatting, str.format(), f-strings). Mutable vs Immutable Types.	18	20%

Introduction to the collections module: Counter, defaultdict, OrderedDict, deque, namedtuple.

Practical:

32. Write a program to perform all list operations: append, insert, remove, pop, sort, and reverse.
33. Write a program to implement a stack using a list with push, pop, and display operations.
34. Write a program to implement a queue using a list with enqueue, dequeue, and display operations.
35. Write a program to demonstrate tuple packing, unpacking, and indexing.
36. Write a program to perform all set operations: union, intersection, difference, and symmetric difference.
37. Write a program to create and manipulate a dictionary of student records (name, roll, marks).
38. Write a program to demonstrate nested dictionary for storing department-wise employee data.
39. Write a program to count word frequency in a paragraph using a dictionary.
40. Write a program to demonstrate all string methods with examples on a sample string.
41. Write a program to reverse a string and check whether it is a palindrome.
42. Write a program to sort a list of strings alphabetically and also by length.
43. Write a program to remove duplicate elements from a list using sets.
44. Write a program using Counter from the collections module to count character frequency in a string.
45. Write a program to merge two dictionaries and find common keys between them.
46. Write a program using deque from the collections module to simulate a circular queue.
47. Solve unsolved questions on lists, tuples, sets, and dictionaries given in the reference book.

Evaluation Methods :

Sr. No.	Evaluation Methods	SEE	CCE
1	**Data Structure Operations Lab :** Students will write a Python program demonstrating key operations on lists, tuples, sets, and dictionaries. The script must include string manipulation, use of at least one comprehension (list or dict), and demonstrate at least two methods from the collections module (Counter, deque, defaultdict, etc.) with well-formatted output clearly showing each data structure's behaviour.	20	-

	2	**Data Structure Exploration Task :** Students solve a real-world problem (e.g., contact book, inventory management, word frequency analyser) using the most appropriate Python data structures. They justify their choice, provide code with output, and submit an individual PDF report on the GMIU Portal.	-	5	
	3	**DSA Problem Solving Challenge :** Students will solve at least five Data Structures and Algorithms (DSA) problems on either HackerRank or LeetCode and submit proof of successful completion through a PDF report on the GMIU Portal.	-	5	
		Total	**20**	**10**	

Examination Style:

Data Structure Operations Lab [20 Marks] :
Students will write a Python script demonstrating operations on lists, tuples, sets, and dictionaries. The script must apply string methods, implement at least one comprehension (list or dict), use at least two methods from the collections module (Counter, deque, defaultdict, etc.), and display well-formatted output clearly showing each data structure's behaviour.

Data Structure Exploration Task [5 Marks] :
Students solve a real-world problem (e.g., contact book, inventory management, word frequency analyser) using the most appropriate Python data structures. They justify their choice, provide code with output, and submit an individual PDF report on the GMIU Portal.

DSA Problem Solving Challenge [5 Marks] :
Students will solve a minimum of five DSA problems from either HackerRank or LeetCode. The problems may cover topics such as arrays, strings, searching, sorting, recursion, linked lists, stacks, queues, or other fundamental DSA concepts. Students should take screenshots of each successfully solved problem, including the problem title and submission status. All screenshots must be compiled into a single PDF report and uploaded on the GMIU Portal. The report should also include a brief summary of the problems solved and the key concepts learned during the activity.

4	**Object-Oriented Programming in Python** **Theory Topics:** Object-Oriented Programming Concepts: Classes and Objects, Constructors (__init__), Destructor (__del__), Instance Variables and Class Variables, Instance Methods, Class Methods (@classmethod), Static Methods (@staticmethod) Access Modifiers: Public, Protected, Private Encapsulation Inheritance: Single, Multiple, Multilevel, Hierarchical, Hybrid Method	18	20%	

Overriding and super() Function. Polymorphism: Method Overriding, Operator Overloading (__add__, __str__, __len__, __eq__, __lt__). Abstract Classes and Interfaces using the abc module. Iterators and Generators (yield, __iter__, __next__). Decorators: Function Decorators, @property Decorator. Magic/Dunder Methods. Using OnlineGDB for writing, executing, and debugging OOP programs.

Practical:

48. Write a program to create a Student class with attributes and a method to display student details.
49. Write a program to demonstrate the constructor (__init__) and destructor (__del__).
50. Write a program to demonstrate class methods (@classmethod) and static methods (@staticmethod).
51. Write a program to implement encapsulation using private variables with getter and setter methods.
52. Write a program to implement single inheritance with method overriding.
53. Write a program to implement multiple inheritance and demonstrate the MRO (Method Resolution Order).
54. Write a program to implement multilevel inheritance (Animal → Mammal → Dog).
55. Write a program to demonstrate polymorphism through method overriding in a shape hierarchy.
56. Write a program to overload the +, *, and __str__ operators for a custom Vector class.
57. Write a program to implement an abstract class using the abc module.
58. Write a generator function that yields square numbers from 1 to N.
59. Write a program using the @property decorator to implement controlled attribute access.
60. Write a program to implement a custom iterator class using __iter__ and __next__.
61. Write a program to demonstrate a function decorator that logs the execution time of a function.
62. Write a program to implement a simple Bank Account system using OOP with encapsulation and inheritance.
63. Solve unsolved questions on OOP concepts from the reference book.

Evaluation Methods :

Sr. No.	Evaluation Methods	SEE	CCE
1	**Real-World OOP Modelling Task :** Students will design and implement a Python program using OOP principles. The program must include class and object creation, constructor, destructor,	20	-

		encapsulation with access modifiers, at least two levels of inheritance, method overriding, operator overloading using magic dunder methods, and use of @property or abstract class. Submit the complete program with sample output.			
	2	**OOP Design Case Study :** Students choose a real-world system (Library, Hospital, or Banking) and design its class hierarchy using OOP principles. They implement the system in Python, document the design with a simple UML class diagram, and submit an individual PDF report on the GMIU Portal.	-	10	
		Total	**20**	**10**	

Examination Style:

OOP Implementation Project [20 Marks] :
Students will design and implement a Python program using OOP principles. The program must include class and object creation, constructor/destructor, encapsulation with access modifiers, at least two levels of inheritance, method overriding, operator overloading using magic/dunder methods, and use of @property or abstract class. Submit the complete program with sample output.

OOP Design Case Study [10 Marks] :
Students choose a real-world system (Library, Hospital, or Banking) and design its class hierarchy using OOP principles. They implement the system in Python, document the design with a simple UML class diagram, and submit an individual PDF report on the GMIU Portal.

5	**File Handling, Exception Handling, Modules and Libraries** **Theory Topics:** File Handling: Opening and Closing Files (open(), close(), with statement). File Modes (r, w, a, rb, wb, r+). Reading Files: read(), readline(), readlines(). Writing Files: write(), writelines(). File Pointer: seek() and tell(). Working with CSV Files (csv module). Working with JSON Files (json module). Exception Handling: Types of Errors – Syntax, Runtime, Logical. try-except Block, Multiple except Clauses, else and finally Clause. Raising Exceptions (raise). Custom/User-defined Exceptions, assert Statement. Modules and Packages: Importing Modules (import, from-import, as). Built-in Modules: os, sys, math, random, datetime, re, collections. Creating User-defined Modules and Packages. pip and Virtual Environments (venv). Introduction to Libraries: NumPy – arrays and array operations. Pandas – Series, DataFrame, read_csv(), basic statistics. Matplotlib – line plot, bar chart, histogram.	18	20%

Practical:

64. Write a program to create a file, write text into it, and read it back line by line.
65. Write a program to append content to an existing file and display all lines.
66. Write a program to count the number of lines, words, and characters in a text file.
67. Write a program to copy the contents of one file to another.
68. Write a program to read and write student records using the csv module.
69. Write a program to store and retrieve employee data in JSON format using the json module.
70. Write a program to demonstrate try-except for handling ZeroDivisionError and Value Error.
71. Write a program to handle multiple exception types using separate except blocks.
72. Write a program to demonstrate the else and finally clauses in exception handling.
73. Write a program to create and raise a custom exception for invalid age input.
74. Write a program to demonstrate the os and sys modules (file path, directory listing).
75. Write a program using the math and random modules for computations and simulations.
76. Write a program using the date time module to display the current date and time and calculate age.
77. Write a program to demonstrate regular expressions using the re module (search, match, findall, sub).
78. Write a program using NumPy to create arrays, perform arithmetic operations, and use array slicing.
79. Write a program using Pandas to read a CSV file and display basic statistics (describe, head, groupby).
80. Write a program using Matplotlib to plot a bar chart and a line graph from sample data.
81. Create a user-defined module with utility functions and import it in another Python script.
82. Solve unsolved questions on file handling, exception handling, and modules from the reference book.

Evaluation Methods :

Sr. No.	Evaluation Methods	SEE	CCE
1	**Python File Handling & Data Processing :** Students will write a Python program demonstrating file I/O (text, CSV, JSON), comprehensive exception handling (multiple except, custom exception, finally).	20	-

	use of at least two built-in modules (os, math, random, datetime, re), and basic data manipulation with NumPy or Pandas and a Matplotlib chart. Include sample data files and outputs.		
2	**Python Library Showcase :** Students build a mini application (e.g., student grade analyser, CSV-based report generator, or expense tracker) using NumPy, Pandas, and Matplotlib. They document the project with code, generated charts, and a brief analysis, and upload an individual PDF report on the GMIU Portal.	-	10
	Total	**20**	**10**

Examination Style:

File & Exception Handling Mini Project [20 Marks] :
Students will write a Python program demonstrating file I/O (text, CSV, JSON), comprehensive exception handling (multiple except, custom exception, finally), use of at least two built-in modules (os, math, random, datetime, re), and basic data manipulation with NumPy or Pandas and a Matplotlib chart. Include sample data files and outputs.

Python Library Showcase [10 Marks] :
Students build a mini application (e.g., student grade analyser, CSV-based report generator, or expense tracker) using NumPy, Pandas, and Matplotlib. They document the project with code, generated charts, and a brief analysis, and upload an individual PDF report on the GMIU Portal.

Suggested Specification table with Marks (Theory): 100

Distribution of Theory Marks (Revised Bloom's Taxonomy)						
Level	Remembrance (R)	Understanding (U)	Application (A)	Analyze (N)	Evaluate (E)	Create (C)
Weightage %	10%	15%	20%	20%	15%	20%

Note: This specification table shall be treated as a general guideline for students and teachers. The actual distribution of marks in the question paper may vary slightly from the above table.

Course Outcome:

After learning the course, the students should be able to	
CO1	Understand the fundamentals of Python programming including syntax, data types, operators, and control structures.
CO2	Develop Python programs using functions, recursion, and built-in data structures such as lists, tuples, sets, and dictionaries.
CO3	Apply object-oriented programming concepts including classes, objects, inheritance, polymorphism, and encapsulation using Python.
CO4	Implement file handling, exception handling, and module/package management in Python applications.
CO5	Build real-world applications using Python libraries for data processing, visualization, and automation.

Instructional Method:

The course delivery method will depend upon the requirement of content and the needs of students.

A minimum of 20–30% of the course content shall be suggested to deliver through Flipped Learning, where students study digital learning resources before the classroom session. Classroom time shall be utilized for discussion, problem-solving, practical applications, and higher-order learning activities.

Students shall be encouraged to use AI-powered learning platforms, simulation software, virtual laboratories, digital libraries, research databases, online certification courses, NPTEL, SWAYAM, MOOCs, and other technology-enabled learning resources to strengthen conceptual understanding and practical competence.

Continuous assessment shall be conducted through Active Learning Assignments (ALAs), quizzes, skill demonstrations, presentations, practical exercises, projects, case studies, innovation activities, peer assessment, reflective learning, and industry-oriented tasks instead of relying primarily on conventional written assignments.

Reference Books:

[1] **Python Programming using Problem Solving Approach** – Reema Thareja, Oxford University Press.
[2] **Core Python Programming** – R. Nageswara Rao, Dreamtech Press.
[3] **Python Crash Course** – Eric Matthes, No Starch Press.
[4] **Learning Python** – Mark Lutz, O'Reilly Media.
[5] **Automate the Boring Stuff with Python** – Al Sweigart, No Starch Press.
[6] **Programming in Python 3** – Mark Summerfield, Addison-Wesley.
[7] **Head First Python** – Paul Barry, O'Reilly Media.

Suggested Assessment Guidelines:

Module 1: Introduction to Python Programming

Program Implementation Task (20 Marks) :		
Criteria	**Description**	**Marks**
Python Environment Setup	Successfully installs Python, configures IDE (IDLE/VS Code), and executes basic programs	5
Understanding of Data Types & Variables	Correct use of variables, data types, type casting, and input output operations	5
Operators Implementation	Demonstrates arithmetic, relational, logical, bitwise, identity, and membership operators correctly	5
Program Execution & Documentation	Proper execution, output verification, explanation, and report submission	5
Total		**20**

Module 2: Control Structures and Functions

Program Implementation Task (20 Marks) :		
Criteria	**Description**	**Marks**
Conditional & Loop Implementation	Correct implementation of decision-making statements and loops	5
Function Design & Usage	Proper use of functions, arguments, return values, and recursion	5
Lambda & Higher-order Functions	Effective use of lambda, map(), filter(), and reduce()	5
Logic Analysis & Report Preparation	Proper explanation, output tracing, complexity understanding, and documentation	5
Total		**20**

Module 3: Data Structures in Python

Program Implementation Task (20 Marks) :		
Criteria	**Description**	**Marks**
Data Structure Operations	Correct implementation of lists, tuples, sets, and dictionaries with built-in methods	5
String Manipulation Skills	Effective use of string operations, slicing, and formatting	5
Collections Module Usage	Demonstrates use of Counter, deque, defaultdict, or other collections effectively	5
Problem Solving & Documentation	Real-world application, logic explanation, and proper report presentation	5
Total		**20**