TEMPLATE FOR COURSE SYLLABUS FOR NEP IMPLEMENTATION

Discipline: Science [√] Arts, Humanities & Social Science ☐

Commerce ☐ BBA ☐ BCA ☐

Subject Name: COMPUTER SCIENCE

Subject Code: UCMSMIN40004 (Will be provided by the University)

Semester: Semester I ☐ Semester II ☐ Semester III ☐ Semester IV ☐

Semester V ☐ Semester VI ☐ Semester VII [√] Semester VIII [√]

Course Name: PYTHON PROGRAMMING

Course Code: (Will be provided by the University)

Course Credit:	Theoretical	3	Practical/Tutorial	1
Marks Allotted:	Theoretical	40	Practical/Tutorial	20
	Continuing Evaluation	10	Attendance	

Course Type (tick the correct alternatives):

Major Core	☐	AEC	☐
Interdisciplinary/ DSE	☐	SEC	☐
Minor / Generic Elective	[√]	VAC	☐
Research Project/Dissertation	☐	Vocational	☐

Is the course focused on employability / entrepreneurship? YES [√] NO ☐

Is the course focused on imparting life skill? YES [√] NO ☐

Is the course based on Activity? YES ☐ NO [√]

Remarks by Chairman, UG BOS, if any

UG BOS Meeting Reference Number: Date:

Course Code: UCMSMIN40004

Course Name: PYTHON PROGRAMMING

Prerequisite(s) and/or Note(s):
(1) High school mathematics.
(2) Note(s): Syllabus changes yearly and may be modified during the term itself, depending on the circumstances. However, students will be evaluated only on the basis of topics covered in the course.

Course Objectives

Knowledge Acquired:
(1) Fundamental Concepts: Students acquire knowledge of fundamental programming concepts such as variables, data types, loops, conditionals, and functions in Python.
(2) Data Structures: They learn about essential data structures like lists, tuples, dictionaries, and sets, understanding their usage and implementation.

Skills Gained:
(1) Coding Proficiency: Through hands-on practice and assignments, students develop coding proficiency in Python, enabling them to write clear, concise, and functional code.
(2) Problem-Solving: They enhance their problem-solving skills by applying Python programming concepts to solve various computational problems and algorithms.
(3) Debugging and Troubleshooting: Students acquire skills in debugging code and troubleshooting errors, learning how to identify and fix common programming mistakes effectively.

Competency Developed:
(1) Logical Thinking: Python programming exercises require logical thinking and algorithmic problem-solving skills, helping students develop a logical mindset.
(2) Attention to Detail: Writing code necessitates attention to detail to ensure accuracy and functionality. Students develop this competency through debugging and code review processes.
(3) Collaboration and Documentation: Students learn to collaborate on coding projects using version control systems like Git and to document their code effectively, enhancing their ability to work in teams and communicate technical concepts clearly.

Detailed Syllabus
4th Year: Semester 7 or 8

UCMSMIN40004: PYTHON PROGRAMMING **[Credits: 3, Lecture :45]**

Unit 1: Introduction to Programming (6 Lectures)
Concept of problem solving, Problem definition, Program design, Debugging, Types of errors in programming, Documentation. Flowcharts, algorithms, Types of programming paradigms – unstructured, procedure oriented, object oriented. Programming methodologies - top-down and bottom-up programming.

Unit 2: Introduction to Python (12 Lectures)
Structure of a Python Program, Elements of Python, Entering Expressions into the Interactive Shell, The Integer, Floating-Point, and String Data Types, String Concatenation and Replication, Storing Values in Variables**.**

Unit 3: Flow control and Functions (12 Lectures)
Boolean Values, Comparison Operators, Boolean Operators, Mixing Boolean and Comparison Operators, Elements of Flow Control, Program Execution, Flow Control Statements, Importing Modules, Ending a Program Early with sys.exit(), def Statements with Parameters, Return Values and return Statements, The None Value, Keyword Arguments and print(), Local and Global Scope, The global Statement, Exception Handling.

Unit 4: List, Dictionary, String and Tuples (15 Lectures)
String, String functions, Manipulating Strings, Lists: Creating Lists; Operations on Lists; Built-in Functions on Lists; Implementation of Stacks and Queues using Lists; Nested Lists. Dictionaries: Creating Dictionaries; Operations on Dictionaries; Built-in Functions on Dictionaries; Dictionary Methods; Populating and Traversing Dictionaries. Tuples and Sets: Creating Tuples; Operations on Tuples; Built-in Functions on Tuples; Tuple Methods; Creating Sets; Operations on Sets; Built-in Functions on Sets; Set Methods.

Suggested Readings:

1. Yashavant Kanetkar and Aditya Kanetkar , Let Us Python,BPB,3rd Edition.
2. Sheetal Taneja and Naveen Kumar, Python Programming- A modular approach with Graphics, Database, Mobile and Web applications, Pearson, Sixteenth Impression-2023,
3. T. Budd, Exploring Python, TMH, 1st Ed, 2011
4. Python Tutorial/Documentation www.python.or 2015
5. Allen Downey, Jeffrey Elkner, Chris Meyers, How to think like a computer scientist: learning with Python , Freely available online.2012

UCMSMIN40004: PYTHON PROGRAMMING LAB **[Credits:1, Tutorial Hours: 30]**

Students are advised to do laboratory/practical practice not limited to, but including the following types of problems:

1. Write a menu driven program to convert the given temperature from Fahrenheit to Celsius and vice versa depending upon users' choice.

2. WAP to calculate total marks, percentage and grade of a student. Marks obtained in each of the three subjects are to be input by the user. Assign grades according to the following criteria:

Grade A: Percentage >=80
Grade B: Percentage>=70 and <80
Grade C: Percentage>=60 and <70
Grade D: Percentage>=40 and <60
Grade E: Percentage<40

3. Write a menu-driven program, using user-defined functions to find the area of rectangle, square, circle and triangle by accepting suitable input parameters from user.

4. WAP to display the first n terms of Fibonacci series.

5. WAP to find factorial of the given number.

6. WAP to implement the use of arrays in Python.

7. WAP to implement String Manipulation in python in Python.

8. WAP to find sum of the following series for n terms: 1 – 2/2! + 3/3! - - - - - n/n!

9. WAP to calculate the sum and product of two compatible matrices.

10. WAP to create Class and Objects in Python.

12. WAP to implement Data Hiding in Python.

13. WAP to implement constructor and destructor for a class in Python.

14. WAP to implement constructor and destructor in Python.

15. WAP to implement different types of inheritance in Python.

16. WAP to implement concept of Overriding in Python.

17. Write programs to create mathematical 3D objects using class.
    a. curve b. sphere c. cone d. arrow e. ring f. cylinder

18. WAP to Check if a Number is Positive, Negative or 0

19. WAP to Check leap year or not.

20. WAP to display calendar of the given month and year.