TEMPLATE FOR COURSE SYLLABUS FOR NEP IMPLEMENTATION

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

Commerce ☐ BBA ☐ BCA ☐

Subject Name: COMPUTER SCIENCE

Subject Code: UCMSMAJ35009 (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: DESIGN AND ANALYSIS OF ALGORITHMS

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	5

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: UCMSMAJ35009

Course Name: DESIGN AND ANALYSIS OF ALGORITHMS

Brief Course Description:

Design and Analysis of Algorithms is a core subject in computer science that focuses on developing efficient procedures for solving computational problems and assessing their performance. It involves crafting algorithms using techniques like divide and conquer, dynamic programming, and greedy methods, and evaluating their efficiency through time and space complexity using Big O notation. The subject also includes optimizing algorithms for better performance and exploring complexity classes, such as P and NP, to understand computational limits. Mastery of these concepts is essential for creating effective and efficient software and systems.

Prerequisite(s) and/or Note(s):

Students interested in this course should have had an exposure to introductory courses on programming and data structures. A basic of mathematical logic is essential for success in this course**.**

Course Objectives:

The objective of learning Design and Analysis of Algorithms is to develop efficient methods for solving computational problems and to evaluate their performance using time and space complexity. It aims to equip students with skills to optimize algorithms for better speed and resource usage while understanding problem complexity and classification. This foundational knowledge is crucial for advancing in computer science and creating effective software solutions.

Knowledge acquired:

1. Understanding how to measure and improve the efficiency of algorithms.
2. Proficiency in analyzing and comparing time and space complexity.
3. Skills to enhance algorithms for better performance and lower resource consumption.
4. Mastery of various algorithmic techniques such as divide and conquer, dynamic programming, and greedy methods.

Skills gained:

1. Ability to conceptualize and develop step-by-step procedures for solving complex problems.
2. Proficiency in analyzing the time and space complexity of algorithms to gauge their efficiency.
3. Skills in refining algorithms to enhance their performance and minimize resource usage.
4. Mastery of various Problem-Solving Techniques, such as divide and conquer, dynamic programming, and greedy strategies.
5. Insight into the classification of problems and algorithms based on complexity classes like P, NP,etc.

Competency Developed:

1. Ability to evaluate and compare the efficiency of different algorithms using time and space complexity metrics.
2. Proficiency in refining and optimizing algorithms to improve performance and resource utilization.
3. Expertise in applying various algorithmic strategies and techniques to effectively solve diverse computational problems.
4. Enhanced capability to analyze and reason about the computational limits and trade-offs of different algorithmic approaches.

Detailed Syllabus

3rd Year: Semester 5

UCMSMAJ35009: DESIGN AND ANALYSIS OF ALGORITHMS [Credits:3, Lecture :45]

Unit 1: Introduction (6 Lectures)
Definitions, Algorithms vs Programs, Basic Algorithm Design Techniques, Correctness of Algorithm, Algorithm classification: Iterative techniques, Divide and Conquer, Dynamic Programming, Greedy Algorithms, Algorithm analysis (formal and informal), calculation of running time of an algorithm, loop invariants.

Unit 2: Growth of Functions (7 Lectures)
Complexity of Algorithms, Best, Worst and Average case complexity, growth rate of functions, Asymptotic Analysis, Analyzing algorithm control structures,

Unit 3: Recurrences (8 Lectures)
Introduction, Substitution method, Iteration method, Master method (Master theorem), Simple problems using Master theorem.

Unit 4: Analysis of simple Sorting and Searching algorithms (16 Lectures)
Elementary sorting techniques: Bubble Sort, Selection sort, Insertion Sort, Shell sort, Merge Sort, Advanced Sorting techniques - Heap Sort, Quick Sort, Sorting in Linear Time - Bucket Sort, Radix Sort and Count Sort, Searching Techniques.

Unit 5: Graphs (8 Lectures)
Graph Algorithms–Breadth First Search, Depth First Search and its Applications, Directed acyclic graphs, Single source shortest paths: Dijkstra's algorithm, Minimum Spanning Trees algorithms: Prim's algorithm, Kruskal's Algorithm.

Suggested Readings:

1. T.H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein Introduction to Algorithms, PHI, 3rdEdition 2009
2. Sarabasse & A.V. Gelder Computer Algorithm – Introduction to Design and Analysis, Publisher Pearson 3rdEdition 1999
3. Rajesh K. Shukla, Analysis and Design of Algorithms: A Begineers Approach, Wiley
4. Udit Agarwal, Algorithms design and analysis, Dhanpat rai & Co.

UCMSMAJ35009:DESIGN AND ANALYSIS OF ALGORITHMS LAB [Credits:1, Lecture Hours:15]

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

1. Implement Bubble Sort (The program should report the number of comparisons)
2. Implement Selection Sort (The program should report the number of comparisons)
3. Implement Insertion Sort (The program should report the number of comparisons)
4. Implement Merge Sort(The program should report the number of comparisons)
5. Implement Randomized Quick sort (The program should report the number of comparisons)
6. Implement Radix Sort(The program should report the number of comparisons)
7. Implement Heap Sort(The program should report the number of comparisons)
8. Implement Breadth-First Search in a graph
9. Implement Depth-First Search in a graph
10. Write a program to determine the minimum spanning tree of a graph

TEMPLATE FOR COURSE SYLLABUS FOR NEP IMPLEMENTATION

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

Commerce ☐ BBA ☐ BCA ☐

Subject Name: COMPUTER SCIENCE

Subject Code: UCMSMAJ35010 (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: NUMERICAL ANALYSIS

Course Code: (Will be provided by the University)

Course Credit: Theoretical 3 Practical/Tutorial 1

Marks Allotted: Theoretical 60 Practical/Tutorial 0

Continuing Evaluation 10 Attendance 5

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:

Prepared by CIRM

Course Code: UCMSMAJ35010

Course Name: NUMERICAL ANALYSIS

Brief Course Description:

The subject Numerical Analysis focuses on developing and implementing algorithms for solving mathematical problems involving continuous data, such as equations, integrals, and differential equations. It aims to ensure accuracy and efficiency in numerical computations and error management.

Prerequisite(s) and/or Note(s):

To enroll in this course, students should have a basic understanding of several key areas of mathematics and computing This course is suitable for beginners and those looking to expand their knowledge in numerical computing.

Course Objectives:

The objective of a Numerical Analysis course for computer science students is to equip them with the skills to develop, analyze, and implement numerical methods for solving mathematical problems computationally. Students gain a deep understanding of numerical methods, error analysis, and algorithm development, ensuring accuracy and efficiency in computations. The course emphasizes practical applications, enabling students to tackle real-world problems in science and engineering. Additionally, students become proficient in using programming languages and numerical libraries to implement these methods. By understanding the theoretical foundations and enhancing problem-solving skills, students are prepared for complex computational challenges in their careers and research.

Knowledge acquired:

1. Understanding and applying various numerical techniques to solve mathematical problems.
2. Identifying, analyzing, and minimizing errors in numerical computations.
3. Designing and implementing efficient and accurate numerical algorithms.
4. Applying numerical methods to real-world scientific and engineering problems.
5. Using programming languages and numerical libraries to implement and test numerical solutions.

Skills gained:

1. Ability to select and apply appropriate numerical methods to solve complex mathematical problems.
2. Proficiency in programming languages and numerical libraries for implementing numerical algorithms.
3. Skills in analyzing the accuracy, stability, and efficiency of numerical methods.
4. Competence in identifying, quantifying, and minimizing various types of computational errors.
5. Expertise in developing and optimizing algorithms for numerical computation tasks.

Competency Developed:

1. The ability to rigorously analyze and understand the behavior and performance of numerical algorithms.
2. Proficiency in implementing and executing numerical methods using appropriate software and programming languages.
3. The skill to apply numerical techniques to solve complex real-world problems in science, engineering, and other fields.
4. Enhanced ability to evaluate the accuracy, efficiency, and reliability of different numerical methods and to make informed decisions about their use.
5. A strong foundation in mathematical concepts and theories underlying numerical methods, allowing for deeper insight and innovation in computational approaches.

Detailed Syllabus

3rd Year: Semester 5

UCMSMAJ35010: NUMERICAL ANALYSIS [Credits: 3, Lectures: 45]

Unit 1: Introduction (15 Lectures)
Floating point representation and computer arithmetic, Significant digits, Errors: Round-off error, Local truncation error, Global truncation error, Absolute error, Relative error, Percentage Error, Rules of rounding off, Convergence &terminal conditions, Efficient computations: Bisection method, Regula−Falsi method, Newton−Raphson method, Gauss elimination method and Gauss−Jordan method, Matrix inversion.

Unit 2: Iterative methods (7 Lectures)
Jacobi and Gauss-Seidel iterative methods

Unit 3: Interpolation (7 Lectures)
Lagrange's Interpolation, Newton forward and backward Interpolation

Unit 4: Numerical integration (6 Lectures)
Trapezoid rule, Simpson's rule (only method)

Unit 5: Extrapolation methods (7 Lectures)
Gaussian quadrature, Ordinary differential equation: Euler's method, Taylor Series Expansion, Runge-Kutta second method , Classical 4th order Runge-Kutta method

Unit 6: Eigen-values & Eigen-vectors (3 Lectures)
Definitions and properties

Suggested Readings

1. Laurence V. Fausett, Applied Numerical Analysis, Using MATLAB, Pearson, 2/e (2012)
2. M.K. Jain, S.R.K. Iyengar and R.K. Jain, Numerical Methods for Scientific and Engineering Computation,New Age International Publisher, 6/e (2012)
3. Steven C Chapra, Applied Numerical Methods with MATLAB for Engineers and Scientists, Tata McGrawHill, 2/e (2010)

UCMSMAJ35010: NUMERICAL ANALYSIS TUTORIAL [Credit:1, Tutorial Hours: 30]

Numerical Analysis tutorial as assigned and advised by teacher(s)

TEMPLATE FOR COURSE SYLLABUS FOR NEP IMPLEMENTATION

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

Commerce ☐ BBA ☐ BCA ☐

Subject Name: COMPUTER SCIENCE

Subject Code: UCMSMAJ35011

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

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

Course Name: FUNDAMENTALS OF MICROPROCESSORS

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 5

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: UCMSMAJ35011

Course Name: FUNDAMENTALS OF MICROPROCESSORS

Brief Course Description:

The Microprocessor Fundamentals and Programming course provides a thorough introduction to the architecture, functioning, and programming of microprocessors. Students will explore the internal workings of microprocessors, learning how they interact with memory and peripherals to perform various tasks. This course emphasizes both theoretical concepts and practical programming skills, enabling students to develop a solid understanding of microprocessor-based systems.

Prerequisite(s) and/or Note(s):

Students should have a background in basic electronics and digital logic before enrolling in this course. Familiarity with assembly language or a high-level programming language is advantageous. This course is ideal for those pursuing studies in computer engineering, electrical engineering, or related fields.

Course Objectives:

The primary objectives of the Microprocessor Fundamentals and Programming course are to provide students with a comprehensive understanding of microprocessor architecture and operation. The course aims to teach students how to program microprocessors in assembly language, interface with various hardware components, and design simple microprocessor-based systems. By the end of the course, students will be equipped with the knowledge and skills to develop and troubleshoot microprocessor applications.

Knowledge acquired:

1. Students will gain detailed knowledge of the CPU, memory hierarchy, and input/output mechanisms.
2. They will learn about instruction sets, addressing modes, and the execution of instructions.
3. The course covers the principles of assembly language programming, enabling students to write and understand low-level code.
4. Students will learn about interfacing microprocessors with external devices, such as sensors and actuators.

Skills gained:

1. Students will develop practical skills in programming microprocessors using assembly language.
2. They will learn how to write efficient code to control hardware, manage data, and perform arithmetic and logic operations.
3. Students will gain experience in debugging and optimizing assembly programs.
4. They will acquire skills in designing and implementing interfaces between the microprocessor and peripheral devices.

Competency Developed:

1. Students will have developed the competency to design, program, and troubleshoot microprocessor-based systems.
2. They will be capable of developing assembly language programs to solve complex problems and interface with various hardware components.
3. This competency will enable students to work in fields such as embedded systems, hardware design, and computer engineering.
4. The course provides a strong foundation for advanced studies in microprocessor technology and related areas.
5. Students will be prepared for careers in the rapidly evolving field of computing & electronics

Detailed Syllabus

3rd Year: Semester 5

UCMSMAJ35011: FUNDAMENTALS OF MICROPROCESSORS [Credits: 3, Lectures:45]

Unit 1: Microprocessor 8085 Architecture (15 Lectures)
Introduction to Microprocessor – Introduction to Microprocessors, Components of Microprocessor: Registers, ALU, timing and control, CPU, I/O devices, clock, memory, bussed architecture, tri-state logic, Bus Structure, address bus, data bus and control bus. Block diagram of 8085, pin configuration of 8085, Architecture of Microprocessor 8085, Internal registers (8-bit & 16-bit), CPU, ALU, multiplexing and demultiplexing address/data bus, Instruction Register and Decoder, Timing and Control Unit, Interrupts and Serial I/O.

Unit 2: Instruction Set-I (10 Lectures)
Machine Language and Assembly Language, Addressing modes, types of instruction format, Data Transfer type instructions, Arithmetic and logical instructions, Branching instructions -looping, Timing diagram for opcode fetch, memory read, memory write, I/O read, I/O write.

Unit 3: Instruction Set-II and Programming (10 Lectures)
Special Instructions: Rotate instructions - stack and subroutine related instructions. Assembly Language Programs – Addition, Subtraction, Multiplication (8-bit), Division (8-bit), sorting-Ascending / Descending Order, Largest/Smallest (single byte), Addition of N numbers (single byte).

Unit 4: Memory/IO Interface and Interrupts (10 Lectures)
Memory Interface (Basics) – memory mapped I/O & I/O mapped I/O. Interrupts in 8085- Types, Generation of RST codes-Hardware, software interrupts and their function, Edge triggered and level triggered interrupts, Interrupt priority, Vectored and non-vectored interrupt -SIM and RIM instructions, Comparison of Microprocessor and Microcontroller.

Suggested Readings:

1. Microprocessor Architecture, Programming and Application with the8085, Ramesh S. Gaonakar, Penram International Publishing, Mumbai, (2011).
2. Fundamental of Microprocessor 8085: Architecture Programming, and Interfacing, V. Vijayendran, Viswanathan, S., Printers & Publishers Pvt. Ltd (2009).
3. The 8051 Microcontroller, Architecture, Program and application, Kenneth J Ayala, Pen ram
4. Microprocessor Organisation and Architecture, Leventhal L.A , Prentice Hall India.
5. B. Ram, Fundamentals of microprocessors and microcomputers – Dhanpat Rai Publications, New Delhi
6. The 8080/85 Family: Design, Programming & Interfacing, John Uffenbeck, , PHI India.
7. A. K. Ray & K. M.Bhurchandani, Advance Microprocessor and Peripherals, 2ndEdition, Tata McGraw Hill, 2006
8. Mathur A.P., Introduction to Microprocessors. 3rd edn, Tata McGraw, New Delhi,
9. Muhammed Ali Mazidi, Janice Gillispie Mazidi – The 8051 Microcontroller and Embedded systems
10. Microprocessors & Microcontrollers by B. P. Singh, Galgotia publications Pvt. Ltd.

UCMSMAJ35011: FUNDAMENTALS OF MICROPROCESSORS LAB [Credit:1, Lab Hours: 30]

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

ASSEMBLY LANGUAGE PROGRAMMING in 8085

1. Write an assembly language program to check whether a number is even or odd.
2. Write an assembly language program to find the number of ones and the number of zeros in an 8-bit number.
3. Write an assembly language program to find the smaller of two numbers.
4. Write an assembly language program to multiply two 8-bit numbers.
5. Write an assembly language program to find the smallest among 10 integers stored in memory locationsstarting from 2050H
6. Write an assembly language program to find the largest among 10 integers stored in memory locations starting from 2050H
7. Write an assembly language program to sort 10 numbers using bubble sort.
8. Write an assembly language program to generate the first 10 fibonacci series and store the result at memorylocation starting from FC50H
9. Write an assembly language program to find the sum of the series 12 + 22 + 32 + 42 + ….. + 102
10. Write an assembly language program to find the perfect square of any number and if the number is not aperfect square, display FFH
11. Write an assembly language program for linear search.
12. Write an assembly language program to calculate the following expression using a single register: Y = X2 +2X + 3XZ
13. Write an assembly language program to find the sum of the first 10 even natural numbers.
14. Write an assembly language program to find the sum of the first n odd natural numbers.
15. Write an assembly language program to create an odd parity generator.
16. Write an assembly language program to create an even parity generator.
17. Write an assembly language program to find the sum of five 8-bit numbers.
18. Write an assembly language program to convert decimal to binary.
19. Write an assembly language program to convert octal to binary.
20. Write an assembly language program to convert hexadecimal to binary.
21. Write an assembly language program to convert hexadecimal to decimal.
22. Write an assembly language program to check whether an 8-bit number is palindrome or not.
23. Write an assembly language program to display the truth table for and AND gate.
24. Write an assembly language program to display the truth table for and OR gate.
25. Write an assembly language program to display the truth table for and XOR gate.
26. Write an assembly language program to perform n byte addition of two numbers.
27. Write an assembly language program to implement a simple sub routine call.
28. Write an assembly language program to check whether a number is prime or not

TEMPLATE FOR COURSE SYLLABUS FOR NEP IMPLEMENTATION

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

Commerce ☐ BBA ☐ BCA ☐

Subject Name: COMPUTER SCIENCE

Subject Code: UCMSMAJ35012 (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 5

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:

TEMPLATE FOR COURSE SYLLABUS FOR NEP IMPLEMENTATION

Course Code: UCMSMAJ35012

Course Name: PYTHON PROGRAMMING

Brief Course Description

The Python Programming course offers a comprehensive introduction to one of the most versatile and widely-used programming languages in the industry. Python's simplicity and readability make it an ideal language for beginners, while its powerful libraries and frameworks make it invaluable for experienced developers. This course covers fundamental programming concepts, Python syntax, and advanced topics, preparing students to use Python for various applications, from web development to data analysis.

Prerequisite(s) and/or Note(s):

No prior programming experience is required to enroll in this course, making it suitable for beginners. However, a basic understanding of computer operations and familiarity with high school-level mathematics will be beneficial. This course is ideal for students from diverse backgrounds, including those in computer science, engineering, data science, and other fields that benefit from programming skills.

Course Objectives:

The primary objectives of the Python Programming course are to provide students with a solid understanding of Python's syntax and core programming concepts. The course aims to teach students how to write clean, efficient, and well-documented Python code. Additionally, students will learn to use Python libraries and frameworks to solve real-world problems, develop web applications, and perform data analysis. By the end of the course, students will be equipped to tackle a wide range of programming challenges using Python.

Knowledge Acquired:

1. Students will acquire comprehensive knowledge of Python programming, starting with basic concepts like variables, data types, and control structures, and progressing to advanced topics such as object-oriented programming and exception handling.
2. They will learn about functions, modules, file handling, and working with essential libraries like NumPy, pandas, and Matplotlib.

Skills Gained:

1. Students will develop practical skills in writing Python code for algorithm implementation, data structure manipulation, and creating functions and classes.
2. They will gain experience in using Python for data analysis and visualization tasks.
3. Students will learn to build web applications using Python.
4. The course emphasizes developing debugging and testing skills to ensure robust, reliable code.

Competency Developed:

1. Students will have the competency to design, develop, and maintain Python-based applications.
2. They will be capable of using Python to solve complex problems and analyze data.
3. Students will be prepared for careers in software development, data science, web development, and other fields requiring strong programming skills.
4. The course provides a strong foundation for advanced study and specialization in Python and related technologies, preparing students for success in the tech industry.

Detailed Syllabus

3rd Year: Semester 5

UCMSMAJ35012: PYTHON PROGRAMMING [Credits: 3, Lectures:45]

Unit 1: Planning the Computer Program (8 Lectures)
Concept of problem solving, Problem definition, Program design, Debugging, Types of errors in programming, Documentation.

Unit 2: Techniques of Problem Solving (8 Lectures)
Flowcharting, decision table, algorithms, Structured programming concepts, Programming methodologies viz. top-down and bottom-up programming.

Unit 3: Overview of Programming (5 Lectures)
Structure of a Python Program, Elements of Python.

Unit 4: Introduction to Python (12 Lectures)
Python Interpreter, Using Python as calculator, Python shell, Indentation. Atoms, Identifiers and keywords, Literals, Strings, Operators (Arithmetic operator, Relational operator, Logical or Boolean operator, Assignment, Operator, Ternary operator, Bit wise operator, Increment or Decrement operator)

Unit 5: Creating Python Programs (12 Lectures)
Input and Output Statements, Control statements (Branching, Looping, Conditional Statement, Exit function, Difference between break, continue and pass.), Defining Functions, default arguments.

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

UCMSMAJ35012: PYTHON PROGRAMMING LAB **[Credit:1, Lab 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:
   a. Grade A: Percentage >=80
   b. Grade B: Percentage>=70 and <80
   c. Grade C: Percentage>=60 and <70
   d. Grade D: Percentage>=40 and <60
   e. 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.
11. WAP to implement Data Hiding in Python.
12. WAP to implement constructor and destructor for a class in Python.
13. WAP to implement constructor and destructor in Python.
14. WAP to implement different types of inheritance in Python.
15. WAP to implement concept of Overriding in Python.
16. Write programs to create mathematical 3D objects using class.
    a. curve b. sphere c. cone d. arrow e. ring f. cylinder
17. WAP to Check if a Number is Positive, Negative or 0
18. WAP to Check leap year or not.
19. WAP to display calendar of the given month and year.
20. WAP to find whether a number is prime or not.