CURRICULA-2020

MCA
(2 Years - CBCS Scheme)

DEPARTMENT OF COMPUTER SCIENCE

FACULTY OFNATURAL SCIENCES

Jamia Millia Islamia: New Delhi, India

Recommended by the Board of studies DCS-FNS-JMI on 26/08/2020
Detailed Syllabi approved by the BOS DCS-FNS-JMI on 20/01/2021

A.	**Reference**	1. Notification No.AICTE/AB/MCA/2020-21/03-07020) 2. Adaptation by BOS dated 06-07-20 for MCA(2-Years) program.
B.	**Motivation**	1. Rapidly changing academic and technological scenarios around the world. 2. Conformance to prescriptions from the regulatory bodies, and JMI ordinances. 3. Keeping abreast with the current and future industrial skill-set requirements, and placement related tests.
C.	**Bases and Constraints**	UGC Guideline on Choice Based Credit Based System (CBCS) – with greater autonomy to students on the selection of courses – by reinforcing the following: 1. Retaining prescribed types as Core, Electives, CBCS, Ability/Skill enhancement courses. 2. Retaining the 4thsemeater as full project-semester, because of the mandate of MCA as a professional program and placements. 3. CBCS courses are to be open for any PG students including DCS students, subject to availability of seats. 4. One CBCS-Elective and CBCS-Ability/Skill-Enhancement course each, to be compulsory in each of the semesters (1-3).
D.	**Abbreviations**	Computer Science: **CS**, Core: C, Elective: E, Ability/Skill: S
E.	**Course Codes**	Unique code to be assigned to each of the typical courses at (UG/PG level separately), offered by the Department using the following coding scheme: • Computer Science Core/elective: CSC/E • Computer Science (CBCS Elective): CBCSE • Computer Science (CBCS Ability/Skill-Enhancement):CBCSS
F.	**Course L-T-P**	1. Core Theory Courses: 3-0-0 2. Lab-Oriented Theory Courses (Electives): 3-0-2 3. Lab-Courses: 0-0-4 4. CBCS Elective Courses: 4-0-0/3-0-2 5. Ability/Skill Enhancement Courses: 2-0-2 6. Major Project: 0-4-32
G.	**Eligibility for Admission**	1. Bachelor’s Degree with at least 50% Marks in (a) Computer Science/Engineering/Applications/Equivalent Allied, OR (b) any other discipline with Mathematics at 10+2 Level. 2. In case of 1(b), a candidate must either produce ‘a valid certificate of passing at least 6 credits of Computer Science courses (with at least C grade/50% marks) from any Govt-approved mode at 10+2/graduation level’ **OR** ‘complete the bridge courses from the Department in the first semester’.
H.	**Bridge Courses**	Applicable to those qualifying under G1(b), under which a student must produce or complete at least 6 credits of the courses as advised by the department, necessarily including a course on programming.
I.	**Special Considerations**	• Accommodation of non-Computer Science graduates and the bridge-courses. • Applied discipline and professional nature of programs. • Balancing academic, technological, and industrial imperatives. • National and global connect. • ACM, UGC, AICTE, and other central universities' curriculum.
J.	**Track Specialization**	MCA degree may be awarded in a specialization (Advanced Computing OR Informatics), provided a student, besides fulfilling all the requirements: (i) opts all the electives from that track, and (ii) requests formally in writing to the Department.
K.	**Remarks**	1. Students are encouraged to enroll in courses about Communication Skills and Management from other Departments, as CBCS electives. 2. CBCS courses of minimum 3 credits each that may be chosen from other departments subject to students’ requirements and convenience. 3. Lab and project courses shall have independent practical and viva-voce examinations. 4. At least two courses shall be offered for each of the typical electives, provided at least 15 students opt for a course. 5. Faculty members may be allocated 2 to 4 lab-periods/week subject to feasibility. 6. Department may float any other elective, beyond the listed ones, subject to feasibility and endorsement of BOS.

MCA(2 Years) Programme Structure: 2020

SEM	CODE	COURSE TITLE	CREDITS	REMARKS
Bridge Courses	CSC01	Computer Fundamentals	3	Any two courses as advised by the Department, as per clause H.
	CSC02	Programming and Problem Solving using C	4	
	CSC03	Applied Operating Systems	3	
	CSC04	Information Systems	3	
I	CSC11	Digital Logic and Computer Architecture	3	Periods/Week: ~24 Credits: ~20
	CSC12	Discrete Mathematics	3	
	CSC13	Algorithmics and Program Design	3	
	CSC14	Database Management Systems	3	
	CSC15	Lab-I (APD)	2	
	CSC16	Lab-II (DBMS)	2	
	CBCSE17	CBCSE-I	4	
II	CSC21	Software Engineering	3	Periods/Week: ~31 Credits:~27
	CSC22	Object Oriented Programming	3	
	CSC23	Advanced Data Structures	3	
	CSC24	Operating Systems and Shell Programming	3	
	CSE25	Elective-I	4	
	CSC26	Lab-III (OOP)	2	
	CSC27	Lab-IV(ADS+SP)	2	
	CBCSE28	CBCSE-II	4	
	CBCSS29	CbcsS-I	3	
III	CSC31	Data Communication and Networks	3	Periods/Week: ~31 Credits: ~27
	CSC32	Artificial Intelligence	3	
	CSC33	Information Security	3	
	CSC34	Analysis and Design of Algorithms	3	
	CSE35	Elective-II	4	
	CSC36	Lab-V (AI)	2	
	CSC37	Lab-VI (ADA)	2	
	CBCSE38	CBCSE-III	4	
	CBCSS39	CbcsS-II	3	
IV	CSC41	Major Project	20	Periods/Week: 36 and Credits: 20
Summary: Core-Courses(12)+Elective(2)+Lab-Courses(6)+CBCS-Elective(3)+CBCS-Skill(2)+Major-Project (1)=26 Courses				

II Semester Syllabus

CSC21: Software Engineering	

LEARNING OUTCOMES

Analyse and specify software requirements, and model its software design.
Understand the software life cycle models, and suitable model for the problem.
Illustrate design concepts and models and use suitable methods for software design

1. **Software Process:** Software Engineering and Development, Software and its Components; Software characteristics; problem of Size and Complexity; Evolving Role of software; Changing Nature; Legacy Software and Software Myths; Software Engineering – A Layered approach, Process Framework, CMMI; Technology, Product and Process.
2. **Software Process Models:** Prescriptive Models: Waterfall Model; Incremental Process Models , RAD; Evolutionary Models – Prototyping, Spiral, and concurrent Models; The Unified Process – Phases and Work Products; Agile Process Models – Extreme Programming and Adaptive; and Dynamic Software Development – Scrum, Crystal, Feature Driven and Agile Modeling.
3. **SE Principles and Practices:** Software Engineering Practices – Essence and Principles; Communication Practices; Planning Process; Modelling Principles; Construction Practices – Coding principles and Concepts; Testing Principles and Deployment; Computer based Systems; System Engineering Hierarchy – System Modelling and Simulation; Business Process Reengineering; Product Engineering; system modeling.
4. **Requirements Engineering and Modelling:** Requirements Engineering Tasks; Requirements Engineering Process; Eliciting Requirements; Developing Use-Cases; Analysis Modelling; Negotiating Requirements; and Validations. Requirements Analysis; Analysis Modelling Approaches; Object Oriented Analysis; Scenario Based and Flow Oriented Modelling.
5. **Design Concepts and Models:** Design concepts and principles**,** Software Design and Software Engineering, Design Context, Process and Quality; Design Concept – Abstraction, architecture, Pattern, modularity, information hiding, functional independence, refinement, design classes; design models – data elements, interface elements, architecture elements; User Interface Design- Process and Models, User Interface Design-The Golden Rules, Component-Level Design.

REFERENCES

Roger S. Pressman: Software Engineering – A practitioners’ Approach. McGraw Hill

K.K. Aggarwal & Yogesh Singh: Software Engineering. New Age International Publishers
P. Jalote: Software Engineering. Narosa

CSC22: Object Oriented Programming

LEARNING OUTCOMES

Understand OOP concepts and features.
Make use of objects and classes for developing programs.
Learn to develop software using OO approach.

1. **OO Concepts:** Programming Paradigms: Unstructured Programming, Structured Programming, Object Oriented Programming; ADT; Class; Object; Message; Encapsulation; Polymorphism; Inheritance; Pros and Cons of Object-oriented Methodology; cin and cout Objects.
2. **Classes and Objects:** Classes; Friend Functions: Benefits and Restrictions, Friends Classes; Inline Functions; Constructor, Parameterized Constructor; Destructor and its usages; Static Data Member and Static Member Functions; Creating Object; Passing and Returning Object(s) to/from a Function; Object Assignment; Nested and Local Classes; Arrays of Objects; Pointer to Objects; this Pointer, Pointer to Derived Type; References; Reference vs Pointer; Reference Parameters; Dynamic Memory Allocation.
3. **Function and Operator overloading:** Function overloading: Rules, Overloading Constructors, Copy Constructors; Default Function Arguments vs. Function Overloading. Operator Overloading: Operators that cannot be Overloaded, Overloading Operators using Member Function and Friends Functions, Overloading different operators including prefix and postfix form of ++ and – operators.
4. **Inheritance & Virtual function**: Inheritance: Types of Inheritances, Base-Class Access Control, Protected Members, Protected Base-class Inheritance, Multiple Inheritance and problems, Solution to Multiple Inheritance Problem, Passing Parameters to Base Class Constructors; Virtual functions: Introduction, Calling a Virtual Function using Base Class Reference, Pure Virtual Function, Abstract Class.
5. **Generic Function, Exception and File Handling:** Generic Functions: Benefits, Functions with Two Generic Types, Explicitly Overloading a Generic Function, Overloading a generic function, Restriction, Generic Classes. Exception Handling, user defined Exception. C++ Streams; C++ File Handling: Opening/Closing a File, Reading /Writing a Text File, Random Access, Reading /Writing Object to a File.

REFERENCES

Herbert Schildt: The Complete Reference C++. McGraw Hill
H.M. Deitel & P.J. Deitel: C++ How to Program. PHI
A. N. Kamthane: Object Oriented Programming with ANSI and TURBO C++. Pearson

CSC23: Advanced Data Structures

LEARNING OUTCOMES

Illustrate terminology and concepts of data structures.
Derive the mapping functions to maps the indices of multi dimensional arrays to index of 1D array.
Design the efficient algorithms for different matrix operations for various special matrices.
Design algorithm for various operations of different data structures.
Applications of various data structures to solve different problems.
Device applications based on Graph data structures.

1. **List and Matrices:** Data Structure, Linear Data Structure, Array Data Structure, Multi Dimensional Array, Mapping of Indices of 2D and 3D Arrays to the Index of 1D Array, Matrix, Mapping of Indices of Matrix Elements to One Dimensional (1D) Array Index, Special Matrices, Triangular, Diagonal, Tri-Diagonal, Representation in Row Major and Column Major Order, Mapping of non-null Elements in 1D Array, Sparse Matrix, Single Linked List, Circular Linked List, Doubly Linked List, Circular Doubly Linked List, Applications of Linked Lists: Bin Sort, Radix Sort, Convex Hull.
2. **Stacks and Queues:** Stack Data Structure, Various Stack Operations, Representation and Implementation of Stack using Array and Linked List, Applications of Stack: Conversion of Infix to Postfix Expressions, Parenthesis Matching, Towers of Hanoi, Rat in a Maze, Implementation of Recursive Functions, Queue Data Structure, Various Queue Operations, Circular Queue, Representation and implementation of queues using Array and Linked List, Applications of Queue Railroad Car Rearrangement, Machine Shop Simulation, Image-Component Labeling, Priority Queues: Priority Queue Using Heap; Max and Min Heap; Insertion into Heap; Deletion from a Heap; Applications of Priority Queue: Heap Sort.
3. **Trees:** Binary Trees and their Properties; Representation of Binary Trees: Array-Based and Linked Representations; Binary Tree Traversals; Binary Search Trees (BST); Operations on BST: Search, Insertion and Deletion; BST with Duplicates; Applications of BST: Histogramming, Best- Fit Bin Packing, AVL Trees; AVL Tree Representation; Introduction to Red-Black and Splay Trees, B-Trees and their Representation; Operations on B-Tree: Search, Insertion and Deletion; B+-Trees.
4. **Sorting, Searching, and Hashing:** Insertion Sort, Bubble Sorting, Quick Sort, Merge Sort, Shell sort, Sequential search, binary search, Introduction to Hashing, Hash Table Representation, Hash Functions, Collision and Overflows, Linear Probing, Random Probing, Double Hashing, and Open Hashing.
5. **Graphs and Disjoint Sets:** Graph Terminology & Representations, Graphs & Multi- graphs, Directed Graphs, Representations of Graphs, Weighted Graph Representations; Graph Traversal Methods: Breadth-First Search and Depth-First Search; Spanning Tree and Shortest Path Finding Problems, Disjoint Sets, Various Operations of Disjoint Sets, Disjoint Sets Implementation.

REFERENCES

Sartaj Sahni: Data Structures, Algorithms and Applications in C++. Universities Press

D. Samanta: Classic Data Structures, PHI
Narasimha Karumanchi: Data Structures and Algorithms Made Easy. CareerMonk

CSC24: Operating System and Shell Programming	

LEARNING OUTCOMES
To understand design of an operating system and services provided by the OS.
To understand what a process is and how processes are synchronized and scheduled.
To acquire knowledge on different approaches to memory management.
To understand the structure and organization of the file system and disk.
Be familiar with various types of operating systems including UNIX, Linux and windows.

1. **Introduction:** Operating Systems functions, Computer-System Organization, Computer-System Architecture, Operating-System Structure, Operating-System Operations, Process Management, Memory Management, Storage Management, Protection and Security, Distributed Systems, Special-Purpose Systems, Computing Environments, Open Source Operating Systems, Operating-System Services, User Operating-System Interface, System Calls, Types of System Calls, System Programs, Operating-System Design and Implementation, Virtual Machines, Operating-System Generation, System Boot.
2. **Process, Threads & Scheduling:** Process Concept, Process Scheduling, Operations on Processes, Inter-process Communication, IPC Systems, Communication in Client-Server Systems, Threads: Overview, Multithreading Models, Thread Libraries, Threading Issues, Process Scheduling: Basic Concepts, Scheduling Criteria, Scheduling Algorithms, Thread Scheduling, Multiple-Processor Scheduling, Operating System Examples, Algorithm Evaluation.
3. **Synchronization & Deadlocks:** Background, The Critical-Section Problem, Peterson's Solution, Synchronization Hardware, Semaphores, Classic Problems of Synchronization, Monitors, Synchronization Examples, Atomic Transactions, Deadlocks: System Model, Deadlock Characterization, Methods for Handling Deadlocks, Deadlock Prevention, Deadlock Avoidance, Deadlock Detection, Recovery from Deadlock.
4. **Memory Management Strategies:** Background, Swapping, Contiguous Memory Allocation, Paging, Structure of the Page Table, Segmentation, Virtual Memory: Background, Demand Paging, Copy-on-Write, Page Replacement, Allocation of Frames, Thrashing, Memory-Mapped Files, Allocating Kernel Memory, Operating-System Examples.
5. **File-System & Shell Programming:** File Concept, Access Methods, Directory and Disk Structure, File-System Mounting, File Sharing, Protection, File-System Structure, File-System Implementation, Directory Implementation, Allocation Methods, Free-Space Management, Efficiency and Performance, Recovery, NFS, The WAFL File System, Shell Programming: Types of shells, Shell functionality, Environment, writing & executing shell scripts, Debugging script, Making interactive scripts, Variables, default variables, Functions & file manipulations, Regular Expression & Filters.

REFERENCES
A. Silberschatz, P.B., Galvin, and G. Gagne: Operating System Concept. JW
W. Stallings: Operating Systems - Internals and Design Principles. Pearson

CSC26: Lab-III (OOP)

Implementation of at least ONE specific assignment concerning each of the following:

1. Structure and nested structure with methods.
2. Creation of class along with constructors, destructors.
3. Friend function, static data members and member functions
4. Passing objects to a function, returning object from a function and nested classes.
5. Creating array of objects, pointer to objects and pointer to class members.
6. Dynamic memory allocation and de-allocation using new and delete, this pointer & reference variable.
7. Function and constructor overloading, operator overloading, copy constructor, default function arguments.
8. Different types of inheritance, virtual functions, abstract class and exception handling.
9. Creation, use and deployment of generic functions
10. File creation and handling using FileStream classes, command line arguments

CSC27: Lab-IV(ADS+SP)

Implementation of at least ONE specific assignment concerning each of the following:

1. Creation of Matrix and Special Matrices classes OR Implementation of the different functions of the Matrix and Special Matrices.
2. Creation Array class OR implementation of the different functions of the Array D.S.
3. Creation of the Different types of Linked list classes OR Implementation of the various functions of the Different types of

Linked list D.S. 4. Implementation of the bucket and radixSort algorithms using single linked list D.S.and convex hull using doubly circular linked list D.S. 5. Creation of the Generic Stack class and implementation of the infixTopostfix, EvaluatePostfix, tower of Hanoi problem, Rat on a maz problems, and recusive functions such as factorial(n), Fib(n), gcd(m, n), DecimalToBinary(n) using Stack D.S. 6. Creation of the Generic CircularQueue class and implementation of the Railroad Car Rearrangement, Image-Component Labeling using Queue. 7. Creation of Array and linked representation of the tree data structure class. 8. Creation of the linked representation of the Expression tree data structure. 9. Creation of the Heap tree data structure and implementation of the heapsort and priority Queue D.S. 10. Creation of the binary search tree data structure. 11. Implementation of various functions of the graph. 12. Implement the depth first search and breadth first search of a graph. 13. Implementation of the topological sort. 14. Use of Basic UNIX Shell Commands: ls, mkdir, cd, cat, touch, file, wc, sort, cut, grep etc. 15. Shell Programming based on control statements and operators. 16. Shell Programming for file handling. 17. Shell Programming for managing the access of the files and users.

(Remark: Detailed syllabus of some Electives and CBCSE/CBCSS courses shall be designed and shared whenever floated for teaching – after BOS approval.)

Section-2.1: Elective Courses (Elective-I)

CSE25.1: Elective-I: Advanced DBMS	
LEARNING OUTCOMES Recap to use DBMS features and be familiar with advanced SQL usage Understanding of Query Processing and Query Optimization Be proficient with Transactions, Concurrency Control and Recovery Systems Be exposed to parallel, distributed and deductive databases and object database systems	
1. Coping with System Failures: Introduction to ADBMS, ACID properties, Issues and Models for Resilient Operation, Undo Logging, Redo Logging, Undo/Redo Logging, Logging Rules, Recovery using different Logging methods, Quiescent and NonquiescentCheck pointing a Log, Recovery with a checkpointed Log, Protecting against Media Failures, Nonquiescent Archiving, Recovery using an Archive and Log, Transactions in SQL,Serializability, Atomicity, Read-only Transactions, Dirty Reads, other Isolations Levels, Review of PL/SQL. **2. Concurrency Control:** Serial and Serializable Schedules, Conflict-Serializability, Precedence Graphs and a Test for Conflict-Serializability, Enforcing Serializability by Locks, The Locking Scheduler, Two-Phase Locking (2PL), Locking Systems with several Lock Modes: shared and Exclusive Locks, Compatibility Matrices, Upgrading Locks, Update Locks, Increment Locks, An architecture for a Locking Scheduler, The Lock Table, Managing Hierarchies of Database Elements: Locks with Multiple Granularity, The Tree Protocol, Concurrency Control by Timestamps, Concurrency Control by Validation, Constraints and Triggers. **3. Advanced Transaction Management:** Serializability and Recoverability, Recoverable Schedules, ACR, Logical Logging, Recovery from Logical Logs, View Serializability, Polygraphs and the Test for View-Serializability, Resolving Deadlocks, Deadlock Prevention by Ordering Elements and Timestamps, Distributed Databases: Distributed Commit, Two-phase Commit (2PC), Distributed Locking, Long-duration Transactions, Sagas and Compensating Transactions **4. The Query Compiler:** Parsing, Estimating the cost of operations, Query optimization, Completing the Physical-Query-Plan and Query Execution; Storage management**.** **5. Database System Architectures:** Object Definition Language (ODL), Object-relational Model,XML and its Data Model, Object-orientation in Query Languages,Logical Query Languages, Centralized and Client-Server Architectures,Parallel Databases,Spatial and Geographic Databases, Multimedia Databases, Mobility and Personal Databases.	
REFERENCES H. Garcia-Molina, J. D. Ullman, and J. Widom: Database Systems: The Complete Book. Pearson A. Silberschatz, H. F. Korth, and S. Sudarshan: Database System Concepts. Mc Graw Hill R. Ramakrishnan & J. Gehrke, Database Management Systems. Mc Graw Hill	
CSE25.8: Elective-I: Data Mining and Warehousing	
LEARNING OUTCOMES Explain the context and concepts; and apply techniques related Data Mining and warehousing. Learn various clustering and classification algorithms. Able to identify the appropriate classification/ clustering techniques for a given dataset.	

Becomes familiar with various data mining tools.
1. **Data Mining:** Introduction, Data warehouses, Transactional databases, Advanced Data Information Systems and Applications, Data Mining Functionalities, Classification of data mining systems, data mining task primitives, Integration of data mining systems with a data warehouse systems, Data Preprocessing: Descriptive data summarization, Data cleaning, Data Integration and Transformation, Data Reduction, Data discretization and Concept hierarchy generation. 2. **Data Warehouse and OLAP technology:** Multidimensional data model, Data Warehouse architecture and Implementation: OLAP, ROLAP, MOLAP, HOLAP etc., Data Cubes, Indexing OLAP data, OLAP queries, Discovery-driven exploration of data cubes. 3. **Frequent Patterns, Associations:** Association Rules, Frequent Itemsets, Closed Itemsets, Apriori algorithm, Generating association rules from frequent itemsets, Mining Closed Frequent Itemsets, Correlation Analysis, Metarule guided mining of Association Rules, Constraint Pushing, Classification v/s Prediction methods, Classification by Decision Tree Induction, Bagging and Boosting. 4. **Classification:** Bayesian classification, K- Nearest-Neighbor Classifier, Case-based Reasoning, Decision Tree, Random Forest, Backpropagation learning, Prediction: Linear v/s Non-linear Regression, Accuracy and Error measures: Hold-out method, Cross-validation, Bootstrap, estimating confidence intervals, Confusion matrix, ROC curves. 5. **Clustering:** Types of data in Cluster Analysis, Categorization of Clustering methods, Partitioning Methods: k-means, k-Medoids, CLARANS, Hierarchical Methods: BIRCH; Mining Time-series data, Introduction to Text Mining, Graph Mining, Social Network Analysis, and Web or Link Mining
REFERENCES J. Han, M. Kamber, and J. Pei: Data Mining: Concepts and Techniques. Elsevier I. Witten, E. Frank, and M. Hall: Data Mining: Practical Machine Learning Tools and Techniques. Elsevier

Section-3.2: CBCS Courses (Sem-II)

CBCSE28.1: Modeling and Simulation	
LEARNING OUTCOMES Define the basics of simulation modeling and replicating the practical situations in organizations Generate random numbers and random variates using different techniques. Develop simulation model using heuristic methods. Analysis of Simulation models using input analyzer, and output analyzer Explain Verification and Validation of simulation model.	
1. **Basics:** Concepts of Systems, Models, and Simulation. Distributed Lag Model, Cobweb Models; The process of a simulation Study, Exponential Growth Models, Exponential Decay Models, Type of simulation, Discrete-Event Simulation: Time-Advance Mechanisms, Components and Organization of a Discrete Event Simulation Model. 2. **Simulation Problems**: Monte Carlo Method, **Discrete** Simulation and Continuous Simulation and their Examples; **Discrete Simulation**: Simulation of Inventory problem, Simulation of Single-Server Queuing System, **Continuous Simulation:** Pure-pursuit Problem. 3. **Random Number Generators:** Linear Congruential Generators, Other kinds of Generators, Testing Random-Number Generators; Generating Random Variates: Various Approach Approaches. 4. **Output Data Analysis for a Single System & Simulation Languages:** Transient and Steady-State Behavior of a Stochastic Process, Type of Simulations with regard to output Analysis and Statistical Analysis for Testing Simulation; Comparisons of Simulation Packages with programming languages Introduction to different types of Simulation Languages. Factors in Selection of discrete system simulation; Object-Oriented Simulation. 5. **Verification and Validation:** Model Building, Verification of Simulation Models: Validating first-time model, Subsystem validity, internal validity, sensitive analysis, face validity; Calibration and Validation of Models, Validation of Model Assumptions, Validating Input, Output Transformations.	
REFERENCES Geoffrey Gordon: System Simulation. PHI M. Law & W. D. Kelton: Simulation Modeling and Analysis. Mc Graw Hill Fred Maryanski: Digital Computer Simulation. Hayden Book Co Jerry Banks: Handbook of Simulation: Principles, Methodology, Advances, Applications and Practice. Wiley P. B. Zeigler: Theory of Modelling and Simulation. Krieger J. Banks et al: Discrete Event System Simulation. Pearson	

Section-4.1: Ability/ Skill Enhancement courses (Sem-II)

CBCSS29.1: Programming with Python	
LEARNING OUTCOMES Understand the basic construct of Python programming language Apply various constructs and control structures in problem solving Understand the object-oriented program design and development in Python Write clear and effective python code Access database using python programming	
1. **Introduction:** Getting Started: Setting up Programming Environment, Python on Different Operating Systems, Running	

Python Programs from a Terminal. Variables & Simple Data Types: Variables, Strings, Numbers, Comments, The Zen of Python. Working with Lists: What is a List, Changing, Adding, Removing Elements, Organizing a List, Avoiding Index Errors, Looping through an Entire List, Avoiding Indentation Errors, Making Numerical Lists, Slicing a List, Working with Tuples and Dictionaries.

2. **Basic Constructs:** User Inputs: input() and int() Functions, Accepting Input in Python. Conditional Tests: if Statements, Using if Statement with Lists. While Loop: Introducing while Loops, using a flat, break, continue, Using a while Loop with Lists and Dictionaries.
3. **Functions, Classes, & Modules:** Functions: Defining a Function, Passing Arguments, Return Values, Passing a List, Passing an Arbitrary Number, Storing Your Functions in Modules. Classes: Creating and Using a Class, Working with Classes and Instances, Inheritance, _init_() Method for a Child Class, Overriding Methods, Instances as Attributes, Importing Classes, Modules, Storing Multiple Classes in a Module, Importing Classes from a Module, Importing a Module into a Module.
4. **Files & Exceptions:** Reading from a File, Reading an Entire File, File Paths, Reading Line-by-line, Making a List of Lines from a File, Working with a File's Contents, Large Files, Writing to a File, Writing to an Empty File, Writing Multiple Lines, Appending to a File. Exceptions: Handling the ZeroDivisionError Exception, Using try-except Blocks, Using Exceptions to Prevent Crashes, The else Block, Handling the FileNotFoundError Exception, Analyzing Text, Working with Multiple Files, Failing Silently, Deciding with Errors to Report, Storing Data, Using json.dump() and json.load(), Saving the Reading User-Generated Data, Refactoring.
5. **Advanced Python Features:**Data Visualization: Generating Data, Installing matplotlib, Plotting a Simple Line Graph, Rolling Dice with Pygal, Making a Histogram. CSV File Format: Parsing CSV File Headers, Extracting and Reading Data, Plotting Data in a Temperature Chart, the datetime Module, Plotting Dates. Working with APIs: Using a Web API, Git and GitHub, Requesting Data using an API Call, Installing Requests, Using GIT for Version Control. Overview of Django.

REFERENCES

Eric Matthes: **Python Crash Course: A Hands-On, Project-Based Introduction to Programming**. No Starch Press
Mark Lutz: Learning Python. O’Reilly
Zed A. Shaw: Learn Python the Hard Way. Addison-Wesley