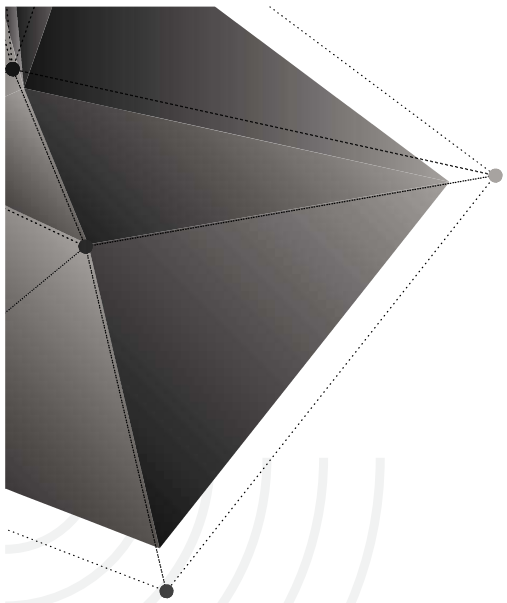
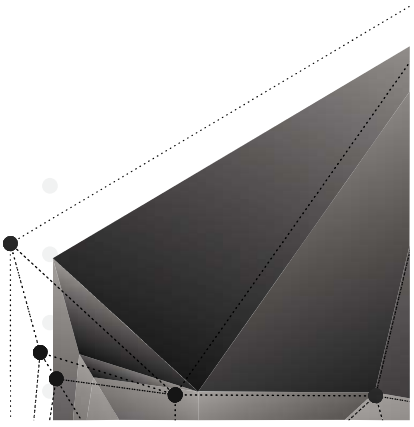


FOURTH SEMESTER

'CYBER FORENSICS
&
INFORMATION SECURITY'



FUNDAMENTAL OF PYTHON PROGRAMMING

Course Code:	494001
Course Title	Fundamental of Python Programming
No. of Credits	5 (TH:4,T:0,P:2)

COURSE OUTCOMES: At the end of the course, the student will be able to:

1. Understand the fundamental concepts of algorithmic problem-solving, including building blocks of algorithms, notation, and simple strategies for developing algorithms (iteration and recursion).
2. Familiarize with data types and expressions in Python, including int, float, Boolean, string, and list. Gain proficiency in variable usage, statements, tuple assignment, and operator precedence.
3. Acquire skills in control flow constructs, such as conditionals (if, if-else, if-elif-else) and iteration (while, for), along with their practical application in problem-solving.
4. Develop knowledge of functions in Python, including function definition, function composition, recursion, and the concept of local and global scope.
5. Learn about data structures like lists, tuples, and dictionaries, and their operations, methods, and parameter usage. Acquire proficiency in list comprehension and advanced list processing.
6. Gain practical experience in file handling, including reading and writing text files, handling exceptions, working with modules and packages, and handling command-line arguments.

COURSE CONTENTS

Unit- I : Algorithmic Problems

Algorithms, building blocks of algorithms (statements, state, control flow, functions), notation (pseudo code, flow chart, programming language), algorithmic problem solving, simple strategies for developing algorithms (iteration, recursion). Illustrative problems: find minimum in a list, insert a card in a list of sorted cards, guess an integer number in a range, Towers of Hanoi.

Unit- II : Data, Expressions, Statements

Python interpreter and interactive mode; values and types: int, float, Boolean, string, and list; variables, expressions, statements, tuple assignment, precedence of operators, comments; modules and functions, function definition and use, flow of execution, parameters and arguments; Illustrative programs: exchange the values of two variables, circulate the values of n variables, distance between two points.

Unit - III : Control Flow, Functions

Conditionals: Boolean values and operators, conditional (if), alternative (if-else), chained conditional (if-elif-else); Iteration: state, while, for, break, continue, pass; Fruitful functions: return values, parameters, local and global scope, function composition, recursion; Strings: string slices, immutability, string functions and methods, string module; Lists as arrays. Illustrative programs: square root, gcd, exponentiation, sum an array of numbers, linear search, binary search.

Unit - IV : Lists, Tuples, Dictionaries

Lists: list operations, list slices, list methods, list loop, mutability, aliasing, cloning lists, list parameters; Tuples: tuple assignment, tuple as return value; Dictionaries: operations and methods; advanced list processing - list comprehension; Illustrative programs: selection sort, insertion sort, merge sort, histogram.

Unit - V : Files, Modules, Packages

Files and exception: text files, reading and writing files, format operator; command line arguments, errors and exceptions, handling exceptions, modules, packages; Illustrative programs: word count, copy file.

Practical Outcomes: Upon completion of the course, students will be able to:

1. Write, test, and debug simple Python programs.
2. Implement Python programs with conditionals and loops.
3. Develop Python programs step-wise by defining functions and calling them.
4. Use Python lists, tuples, dictionaries for representing compound data.
5. Read and write data from/to files in Python.

List of Practicals:

1. Programs related to basic input/output.
2. Programs related to variables, strings, numbers
3. Programs related to Lists and Tuples
4. Programs related to Functions
5. Programs related to If Statements
6. Programs related to While Loops and Input
7. Programs related to Basic Terminal Apps
8. Programs related to Dictionaries
9. Programs related to Classes
10. Programs related to Exceptions
11. Case study of application areas of python.

References / Suggested Learning Resources:

1. Allen B. Downey, ``Think Python: How to Think Like a Computer Scientist'', 2nd edition, Updated for Python 3, Shroff/O'Reilly Publishers, 2016
2. Guido van Rossum and Fred L. Drake Jr, "An Introduction to Python – Revised and updated for Python 3.2, Network Theory Ltd., 2011.
3. John V Guttag, "Introduction to Computation and Programming Using Python'', Revised and expanded Edition, MIT Press , 2013
4. Robert Sedgewick, Kevin Wayne, Robert Dondero, "Introduction to Programming in Python: An Inter-disciplinary Approach, Pearson India Education Services Pvt. Ltd., 2016.
5. Timothy A. Budd, "Exploring Python", Mc-Graw Hill Education (India) Private Ltd.,, 2015.
6. Kenneth A. Lambert, "Fundamentals of Python: First Programs", CENGAGE Learning, 2012.
7. Charles Dierbach, "Introduction to Computer Science using Python: A Computational Problem- Solving Focus, Wiley India Edition, 2013.
8. Paul Gries, Jennifer Campbell and Jason Montojo, "Practical Programming: An Introduction to Computer Science using Python 3", Second edition, Pragmatic Programmers, LLC, 2013

*****

INTRODUCTION TO DBMS

Course Code:	434006
Course Title	Introduction to DBMS
No. of Credits	5 (TH:4,T:0,P:2)

COURSE OUTCOMES: At the end of the course, the student will be able to:

1. Understand the fundamental concepts of databases, advantages of DBMS, and the structure of a standard DBMS.
2. Conceptualize, design, and implement data structures using prominent data models, emphasizing the Entity-Relationship Model.
3. Acquire skills to implement and manage relational databases, apply security measures, and proficiently use query languages.
4. Grasp the principles of database normalization, functional dependencies, and apply them to design efficient databases.
5. Develop proficiency in SQL and PL/SQL for effective data manipulation, control structures, error handling, and understanding advanced features like cursors and triggers.
6. Understand and apply foundational concepts in database security, forensics, and auditing, ensuring the integrity and security of database applications.

COURSE CONTENTS

Unit-1: Introduction to Database Systems and Data Modelling:

- Fundamental concepts of databases and advantages over traditional file systems.
- Structure and components of a standard Database Management System (DBMS).
- Introduction to prominent Data Models with a focus on the Entity-Relationship (E-R) Model.
- Overview of the Client-Server Architecture in databases.
- Detailed study of the E-R Model: Entities, Attributes, Relationships, and their degrees.

Unit-2: The Relational Data Model: Concepts and Security Measures

- Introduction to the Relational Model: Attributes, domains, and key concepts.
- Understanding of primary and candidate keys and associated integrity constraints.
- Study of data security, access control, and authorization in databases.
- Introduction to foundational Query Languages: Relational Algebra and Views.

Unit-3: Structured Query Language (SQL) and Procedural Language/SQL (PL/SQL)

- **Basic SQL operations:** Creating, updating, deleting tables, and implementing constraints.
- **Introduction to advanced SQL concepts:** Set operations, aggregate functions, and joins.

- **Basics of PL/SQL:** Structure, variables, control mechanisms, and error handling.
- **Overview of specialized PL/SQL features:** Cursors, triggers, and packages.

Unit-4: Principles of Relational Database Design and Storage Systems

- Importance of database normalization and associated challenges.
- Basics of Functional Dependencies and steps to normalization.
- Introduction to file organization in databases.
- Fundamental concepts of indexing, hashing, and query optimization.

Unit-5: Fundamentals of Query and Transaction Processing

- Strategies for efficient query processing.
- Basics of transactions in databases and their properties.
- Understanding concurrency in transactions and related concepts.

Unit-6: Case Study on Database Security and Forensics

- Study of security models applicable to databases.
- Introduction to database auditing, activity monitoring, and forensic measures

Practical Outcomes: At the end of the course:

1. Student will be proficient in defining, altering, and managing database structures using Data Definition Language (DDL) commands.
2. Student will grasp the importance of data integrity through constraints like PRIMARY KEY, FOREIGN KEY, etc., and will leverage group functions to derive key insights from datasets.
3. Student will master data manipulation using DML commands, set operators, and essential string functions to optimize and analyse table data.
4. Student will become competent in managing date and time data types, utilizing SQL's specialized functions for time-sensitive queries and analyses.
5. Student will understand and apply conditional constructs like if-then-else, ensuring dynamic and responsive database operations.
6. Student will gain foundational skills in PL/SQL, especially in variable and constant declarations, facilitating efficient temporary data storage and manipulation in their programs.

List of Practicals:

1. Creating and Executing DDL in SQL.
2. Creating and Executing Integrity constraints in SQL.
3. Creating and Executing DML in SQL.
4. Executing relational, logical and mathematical set operators using SQL.
5. Executing group functions.
6. Executing string operators & string functions.
7. Executing Date & Time functions.

8. Executing Data Conversion functions.
9. Program using if-then-else in PL/SQL.
10. Program for declaring and using variables and constant using PL/SQL.

Text and Reference Books:

1. Fundamentals of Database Systems, Elmasri & Navathe, Pearson Education
2. Database Management Systems, Raghurama Krishnan, Johannes Gehrke, Tata McGraw Hill.
3. Database System Concepts, Abraham Silberschatz, Henry F. Korth, S. Sudarshan, McGraw- Hill, New Delhi, India.
4. Introduction to Database Systems, C .J. Date, Pearson Education.
5. Introduction to SQL, Rick F. VanderLans, Pearson Education

*****

CYBER SECURITY LAWS

Course Code:	494002
Course Title	Cyber Security Laws
No. of Credits	4 (TH:4,T:0,P:0)

Course outcomes: After completion of the course, students will be able to:

1. Acquire good understanding of IT ACT and Programs.
2. Acquire good understanding of Privacy acts and policy.
3. Perform search and seizure of computers to obtain electronic evidence.
4. Acquire good understanding of IPR.

COURSE CONTENTS

Unit - 1 : IT ACT and Programs:

Aims and Objectives; Overview of the Act; Jurisdiction; Electronic Governance; Electronic Evidence; Digital Signature Certificates; Digital signatures; Duties of Subscribers; Role of Certifying Authorities; The Cyber Regulations Appellate Tribunal; Internet Service Providers and their Liability; Impact of the Act on other Laws.

Unit - 2 : International Aspects of Cyber Crime:

International Perspectives; Main features of Budapest Convention on Cybercrime; Net neutrality; Features of Web Content Accessibility Guidelines (WCAG).

Unit - 3 : Privacy Acts and Policy:

Introduction to Privacy, Personally Identifiable Information (PII); Overview of international legal standards on privacy, Data Protection Bill.

Unit - 4 : Cyber Crime & Role of Electronic Evidence:

Definition of Cyber Crimes, Investigation Agencies in India, Digital Evidence definition, Evidence Collection, Fundamentals of Digital Forensics (Brief idea only).

Text books:

1. Cyber Law & Cyber Crimes By Advocate Prashant Mali; Snow White publications, Mumbai
2. CYBER LAW & CYBER CRIMES SIMPLIFIED by Cyber Infomedia by Adv. Prashant Mali (Author)
3. Cyber Law in India by Farooq Ahmad; Pioneer Books
4. Information Technology Law and Practice by Vakul Sharma; Universal Law Publishing Co. Pvt. Ltd.
5. The Indian Cyber Law by Suresh T. Vishwanathan; Bharat Law House New Delhi
6. Guide to Cyber and E – Commerce Laws by P.M. Bukshi and R.K. Suri; Bharat Law House, New Delhi
7. Guide to Cyber Laws by Rodney D. Ryder; Wadhwa and Company, Nagpur
8. Scene of the Cybercrime: Computer Forensics Handbook by Syngress.
9. Security and Incident Response by Keith J. Jones, Richard Bejtloich and Curtis W. Rose
10. Introduction to Forensic Science in Crime Investigation by Dr. (Smt) Rukmani Krishnamurthy.

Reference books:

1. Cyber Forensics: A Field Manual for Collecting, Examining, and Preserving Evidence of Computer Crimes

*****

DESIGN & ANALYSIS OF ALGORITHMS

Course Code:	494003
Course Title	Design & Analysis of Algorithms
No. of Credits	5 (TH:4,T:0,P:2)

COURSE OUTCOMES: At the end of the course, the student will be able to:

1. Understand and design basic algorithms for sorting, searching, tree and graph operations.
2. Analyze the time and space complexity of algorithms.
3. Implement various algorithms in a modern programming language.
4. Compare the performance of different sorting and searching techniques.
5. Apply different programming paradigms in a high-level language.
6. Demonstrate familiarity with major algorithms and data structures.

COURSE CONTENTS

Unit-1: Introduction

- Algorithm: Definition and Characteristics
- Performance Measurement: Time and Space Trade-offs

Unit-2: Elementary Data Structures

- Linear Data Structures: Stack, Queue, Lists
- Non-linear Data Structures: Graphs, Trees (Basic Terminologies, Binary Trees)
- Dictionaries: Binary Search Trees
- Hashing: Open Hashing (Separate Chaining), Closed Hashing (Open Addressing)
- Priority Queues, Heaps, Heap Sort
- Sets and Disjoint Set Union

Unit-3: Sorting and Searching

- Binary Search
- Selection Sort, Bubble Sort
- Quick Sort, Merge Sort
- Performance Analysis

Unit-4: Divide-and-Conquer

- General Method
- Finding the Maximum and Minimum

Unit-5: Brute Force and Exhaustive Search

- Sequential Search
- Brute-Force String Matching

- Traveling Salesman Problem
- Assignment Problem
- Depth-First Search and Breadth-First Search

Unit-6: Greedy Method

- Basic Method, Use, Examples
- Prim's Algorithm
- Kruskal's Algorithm
- Dijkstra's Algorithm
- Huffman Trees and Codes

Unit-7: Dynamic Programming

- Basic Method, Use, Examples
- Matrix-chain Multiplication
- Floyd Warshall's All Pair Shortest Path Algorithm
- 0/1 Knapsack Problem

Unit-8: Backtracking

- Basic Method, Use, Examples
- 8-Queen Problem
- Graph Coloring Problem

Unit-9: NP-hard and NP-complete Problems

- P Class, NP-hard Class, NP-complete Class
- Circuit Satisfiability Problem

Practical Outcomes: Upon completion of the course, student will be able to:

- Implement Binary Search
- Implement various sorting algorithms like Bubble Sort, Quick Sort, Merge Sort, etc.
- Implement BFS/DFS
- Implement Travelling Salesman Problem
- Implement Matrix-Chain Multiplication problem
- Implement Minimum Spanning Tree algorithms
- Implement Dijkstra's algorithm
- Find Kth smallest element
- Implement different backtracking algorithms

List of Practical:

1. Implement Sequential Search
2. Implement Binary Search
3. Implement Insertion Sort
4. Implement Bubble Sort
5. Implement Selection Sort
6. Implement Merge Sort
7. Implement Quick Sort
8. Implement Counting Sort
9. Implement BFS/DFS
10. Implement Dijkstra's algorithm
11. Find Kth smallest element
12. Implement Minimum Spanning Tree algorithms
13. Implement Travelling Salesman Problem
14. Implement Matrix-Chain Multiplication problem

Text and Reference Books:

1. Introduction to Algorithms, Cormen, Leiserson, Rivest, and Stein, MIT Press
2. Algorithms, Sedgewick and Wayne, Pearson
3. Fundamentals of Computer Algorithms, Ellis Horowitz, S Sahni, University Press
4. Introduction to The Design and Analysis of Algorithms, Anany Levitin, Pearson
5. Introduction to Theory of Computation, Sipser Michael, Cengage Learning
6. Design & Analysis of Algorithms, Gajendra Sharma, Khanna Publishing House

*****

OBJECT ORIENTED PROGRAMMING USING JAVA

Course Code:	474005
Course Title	Object Oriented Programming Using Java
No. of Credits	5 (TH:4,T:0,P:2)

COURSE OUTCOMES: After completion of this course, student will be able to:

1. Understand the fundamentals of object-oriented programming and distinguish it from procedure-oriented programming.
2. Implement various language constructs such as variables, types, data types, operators, iteration statements, and conditionals.
3. Create and work with classes, objects, constructors, and packages in Java.
4. Understand and implement inheritance, access control, abstract classes, and interfaces.
5. Apply polymorphism through method and constructor overloading and overriding.
6. Handle exceptions effectively and utilize multithreading concepts for concurrent programming.

COURSE CONTENTS

1. Introduction and Features:

Fundamentals of object oriented programming – procedure oriented programming Vs. object oriented programming (OOP), Object oriented programming concepts – Classes, object, object reference, abstraction, encapsulation, inheritance, polymorphism, Introduction of eclipse (IDE) for developing programs in Java

2. Language Constructs:

variables, types and type declarations, data types : Integer, floating point type, character, Boolean, all Operators, iteration and jump statement, if then else clause; conditional expressions, input using scanner class and output statement, loops, switch case, arrays, methods.

3. Classes and Objects:

Class fundamentals, constructors, declaring objects (Object & Object Reference), creating and accessing variables and methods, static and non-static variables/methods defining packages, Creating and accessing a package, Importing packages, Understanding CLASSPATH, auto boxing, String , String Buffer

4. Inheritance:

Definition of inheritance, protected data, private data, public data, constructor chaining, order of invocation, types of inheritance, single inheritance, multilevel inheritance, hierarchical inheritance, hybrid inheritance , access control (Private Vs Public Vs Protected Vs Default)

5. Abstract Class and Interface:

Defining an interface, difference between classes and interface, Key points of Abstract class & interface, difference between an abstract class & interface, implementation of multiple inheritance through interface.

6. Polymorphism:

Method and constructor overloading, method overriding, up-casting and down-casting.

7. Exception Handling:

Definition of exception handling, implementation of keywords like try, catches, finally, throw& throws, built in exceptions, creating own exception sub classes importance of exception handling in practical implementation of live projects

8. Multithreading:

Difference between multi-threading and multi-tasking, thread life cycle, creating threads, thread priorities, synchronizing threads.

PRACTICAL OUTCOMES: Upon completion of the course, student will be able to:

1. Understand the use of the super and this keywords and their significance in class implementation.
2. Gain proficiency in designing classes with inheritance and static methods for modeling real-world scenarios.
3. Demonstrate knowledge of abstract methods and classes and their application in program design.
4. Develop a comprehensive understanding of string manipulation and the implementation of string-related operations.
5. Apply class hierarchies and inheritance concepts to model and manipulate objects in specific domains, such as cars or employees.
6. Learn to effectively handle exceptions and implement error-handling mechanisms in programs.

List of Practical:

1. WAP to create a simple class to find out the area and perimeter of rectangle and box using super and this keyword.
2. WAP to design a class account using the inheritance and static that show all function of bank (withdrawal, deposit).
3. WAP to design a class using abstract methods and classes.
4. WAP to design a string class that perform string method (equal, reverse the string, change case).
5. Consider we have a Class of Cars under which Santro Xing, Alto and Wagon R represents individual Objects. In this context each Car Object will have its own, Model, Year of Manufacture, Colour, Top Speed, etc. which form Properties of the Car class and the associated actions i.e.,

object functions like Create(), Sold(), display() form the Methods of Car Class.

6. In a software company Software Engineers, Sr. Software Engineers, Module Lead, Technical Lead, Project Lead, Project Manager, Program Manager, Directors all are the employees of the company but their work, perks, roles, responsibilities differs. Create the Employee base class would provide the common behaviors of all types of employee and also some behaviors properties that all employee must have for that company.
7. Using the concept of multiple inheritance create classes: Shape, Circle, Square, Cube, Sphere, Cylinder. Your classes may only have the class variable specified in the table below and the methods Area and/or Volume to output their area and/or volume.

Class	Class Variable	Constructor	Base class
Shape	String name	Shape ()	
Circle	double radius	Circle (double r, String n)	Shape
Square	double side	Square (double s, String n)	Shape
Cylinder	double height	Cylinder (double h, double r, String n)	Circle
Sphere	None	Sphere (double r, String n)	Circle
Cube	None	Cube (double s, String n)	Square

8. WAP to handle the exception using try and multiple catch block.
9. WAP that implement the Nested try statements.
10. WAP to create a package that access the member of external class as well as same package.
11. WAP that show the partial implementation of interface.
12. WAP to create a thread that implement the Runnable interface.

Text & Reference Books:

1. Programming with Java: A Primer; E. Balagurusamy
2. Head First Java, O-REILLY, Kathy Sierra & Bert Bates.
3. OCA Java SE Programmer I Certification Guide , Wiley Publisher , Mala Gupta
4. PROGRAMMER'S GUIDE TO JAVA SE 8 , Pearson , Khalid E Mughal
5. e-books/e-tools/relevant software to be used as recommended by AICTE/ UBTER/ NITTTR.

*****

INTRODUCTION TO OPERATING SYSTEMS

Course Code:	434003
Course Title	Introduction to Operating System
No. of Credits	5 (TH:4,T:0,P:2)

COURSE OUTCOMES: After completion of this course, student will be able to:

1. Understand the basic concepts and architecture of UNIX/LINUX operating systems.
2. Gain knowledge of process management, including process concepts, operations, and inter-process communica-tion.
3. Learn about different process scheduling algorithms and their implementations.
4. Develop an understanding of memory management techniques such as allocation, swapping, paging, and segmentation.
5. Explore file management concepts, including file access methods, directory structure, and file system implementation.
6. Acquire knowledge of OS security principles, including authentication, access control, and system logs.

COURSE CONTENTS

UNIT - 1:

1.1 Overview of Operating System

1.2 Basic concepts

1.3 UNIX/LINUX Architecture

1.4 Kernel

1.5 Services and systems calls

1.6 System programs.

UNIT - 2 :

2.1 Overview of Process management Concept

2.2. Process scheduling Methods

2.3 Multi- threaded programming

2.4 Memory management Techniques

2.5 Virtual memory

UNIT - 3 :

3.1 File management

- 3.1.1 Concept of a file
- 3.1.2 Access methods

3.2 Directory structure

3.3 Overview of File system structure and implementation

3.4 Different types of file systems

UNIT - 4 :

4.1 I/O system

4.2 Mass storage structure

4.2.1 Overview

4.2.2 Disk structure

4.2.3 Disk attachment

4.3 Basic Idea of Disk scheduling algorithms

4.4 Swap space management

4.5 Raid.

UNIT - 5:

5.1 OS Security

5.2 Authentication

5.3 Access Control

5.4 Access Rights

5.5 System Logs

Practical Outcomes: Upon completion of the course, student will be able to:

1. Simulate CPU scheduling algorithms (FCFS, SJF, Round Robin, Priority) using a C program.
2. Implement the producer-consumer problem simulation using semaphores in C.
3. Simulate the Dining-philosophers problem to understand synchronization and resource allocation.
4. Implement memory allocation techniques (MVT, MFT) in a simulation.
5. Simulate contiguous memory allocation techniques (Worst Fit, Best Fit, First Fit) using a C program.
6. Implement page replacement algorithms (FIFO, LRU, Optimal) in a simulation.
7. Simulate different file organization techniques (Single Level Directory, Two Level Directory).

List of Practicals:

1. Simulate the following CPU scheduling algorithms.

   (a) FCFS (b) SJF (c) Round Robin (d) Priority.

2. Write a C program to simulate producer-consumer problem using Semaphores
3: Write a C program to simulate the concept of Dining-philosophers problem.
4. Simulate MVT and MFT.
5. Write a C program to simulate the following contiguous memory allocation Techniques

   (a) Worst fit (b) Best fit (c) First fit.

6. Simulate all page replacement algorithms

   (a) FIFO (b) LRU (c) OPTIMAL

7. Simulate all File Organization Techniques
8. Simulate all file allocation strategies
   (a) Sequential (b) Indexed (c) Linked.
11. Write a C program to simulate disk scheduling algorithms.

References

1. Operating System Concepts, Silberschatz and Galvin, Wiley India Limited
2. UNIX Concepts and Applications, Sumitabha Das, McGraw-Hill Education
3. Operating Systems, Internals and Design Principles, Stallings, Pearson Education, India
4. Operating System Concepts, Ekta Walia, Khanna Publishing House
5. Modern Operating Systems, Andrew S. Tanenbaum, Prentice Hall of India
6. Operating systems, Deitel & Deitel, Pearson Education, India

*****

‘Elective 1-1’
ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Course Code:	454007
Course Title	Artificial Intelligence & Machine Learning
No. of Credits	4 (TH:4,T:0,P:0)

COURSE OUTCOMES: At the end of this course, the student will be able to:

1. Understand the history and foundations of artificial intelligence, including its origins and key milestones in its development.
2. Gain knowledge and proficiency in problem-solving techniques in AI.
3. Develop an understanding of adversarial search in decision support systems and technologies.
4. Acquire knowledge of representation, reasoning, expert systems, and the basics of planning in AI.
5. Learn the basics tools and techniques used in machine learning.

COURSE CONTENTS

Unit - I : Introduction

History & foundations of AI, Problem solving: Uninformed and informed Search.

Unit - II : Adversarial Search

Two players games, games with uncertainty; Decision support systems and technologies; Knowledge representation, Reasoning.

Unit - III : Machine Learning Basics

Decision trees, Ensemble learning, Reinforcement learning, Evolutionary computation, Neural networks, Visualization.

Unit - IV :

Basic idea of Linear regression, concept of SSE; gradient descent; closed form; normal equations; features.

Unit - V :

Classification problems; Decision boundaries; Probability and classification, Bayes optimal decisions.

References:

1. Russell, Norvig, Artificial intelligence: A modern approach, 2nd edition. Pearson/Prentice Hall.
2. M.C. Trivedi, A classical approach to Artificial Intelligence, Khanna Publishing House, New Delhi (2018)
3. V.K. Jain, Machine Learning, Khanna Publishing House, New Delhi (2018)
4. Ethem Alpaydin, Introduction to Machine Learning, Second Edition,
5. http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=12012.

*****

‘Elective 1-2’
SOFT COMPUTING

Course Code:	454008
Course Title	Soft Computing
No. of Credits	4 (TH:4,T:0,P:0)

COURSE OUTCOMES : At the end of the course, the student will be able to:

1. Classify and differentiate problem solving methods and tools.
2. Apply A*, AO*, Branch and Bound search techniques for problem solving.
3. Formulate an optimization problem to solve using evolutionary computing methods.
4. Design and implement GA, PSO and ACO algorithms for optimization problems in Mechanical Engineering.
5. Apply soft computing techniques for design, control and optimization of Manufacturing systems.

COURSE CONTENTS

Unit - I : Introduction

Soft Computing, Difference between Hard and Soft computing, Requirement of Soft computing, Major Areas of Soft Computing, Applications of Soft Computing.

Unit - II : Neural Networks

Introduction to Neural Network, Learning rules and various activation functions, Single layer Perceptrons, Back Propagation networks, Architecture of Backpropagation (BP) Networks, Neural Network.

Unit - III : Fuzzy Systems

Fuzzy Control Systems, Fuzzy Classification.

Unit - IV : Genetic Algorithm

History of Genetic Algorithms (GA), Working Principle, Various Encoding methods.

Unit - V : Hybrid Systems

Sequential Hybrid Systems, Auxiliary Hybrid Systems, Embedded Hybrid Systems.

Text & Reference Books:

1. Tettamanzi Andrea, Tomassini and Marco, Soft Computing Integrating Evolutionary, Neural and Fuzzy Systems, Springer, 2001.
2. Elaine Rich, Artificial Intelligence, McGraw Hill, 2/e, 1990.
3. Kalyanmoy Deb, Multi-objective Optimization using Evolutionary Algorithms, John Wiley and Sons, 2001.

*****

'Audit Course'
ESSENCE OF INDIAN TRADITIONAL KNOWLEDGE

Course Code	AS401
Course Title	Essence of Indian Traditional Knowledge
No. of Credits	0 (TH:2,T:0,P:0)

COURSE OUTCOMES: After completion of this course, student will be able to:

1. Develop a comprehensive understanding of the essence of Indian knowledge and tradition.
2. Explore the rich philosophical systems of ancient India and their relevance today.
3. Gain familiarity with the Vedic literature and scriptures, and appreciate their wisdom.
4. Analyze Indian epics and mythology to understand their cultural and spiritual significance.
5. Learn and apply principles of yoga, meditation, and mindfulness for personal well-being.
6. Discover the principles and practices of Ayurveda and natural healing for holistic health.

COURSE CONTENTS

1. Introduction to Indian Knowledge and Tradition
2. Ancient Indian Philosophical Systems
3. Vedic Literature and Scriptures
4. Indian Epics and Mythology
5. Yoga, Meditation, and Mindfulness Practices
6. Ayurveda and Natural Healing Systems
7. Indian Classical Arts and Music
8. Indian Architecture and Sculpture
9. Indian Festivals and Rituals
10. Ethical and Moral Values in Indian Culture

References /Suggested Learning Resources:

1. "Indian Philosophy: A Very Short Introduction" by Sue Hamilton
2. "The Vedas: An Introduction to Hinduism's Sacred Texts" by Roshen Dalal
3. "The Ramayana: A Shortened Modern Prose Version of the Indian Epic" by R.K. Narayan
4. "The Upanishads" translated by Eknath Easwaran
5. "Autobiography of a Yogi" by Paramahansa Yogananda
6. "Ayurveda: The Science of Self-Healing" by Dr. Vasant Lad.

*****

MINOR PROJECT WORK

Course Code:	AS402
Course Title	Minor Project Work
No. of Credits	2 (TH:0,T:0,P:4)

OBJECTIVE:

The Minor Project work is an integral part of the Engineering Diploma program, designed to provide students with an opportunity to apply theoretical knowledge gained throughout their studies to real-world engineering challenges. This module aims to foster creativity, problem-solving abilities, and practical skills essential for successful engineering professionals.

PRACTICAL OUTCOMES: After undergoing the minor project work, the student will be able to:

1. Understand the practical applications of engineering concepts in real-world scenarios.
2. Develop hands-on experience in designing, implementing, and testing engineering projects.
3. Enhance problem-solving and critical thinking skills through project execution.
4. Improve documentation and presentation skills for effective project communication.

GENERAL GUIDELINES:

1. Introduction to Minor Projects

- Overview of the module's purpose and objectives
- Importance of practical application in engineering
- Understanding the project life cycle and its stages

2. **Project Ideation and Proposal Development**
- Identifying engineering problems and project ideas
- Formulating clear project objectives and scope
- Developing a comprehensive project proposal

3. **Project Planning and Management**
- Creating a project plan with defined milestones and timelines
- Resource allocation and budgeting for the project
- Risk assessment and mitigation strategies

4. **Engineering Design and Analysis**
- Principles of engineering design and problem-solving
- Conducting feasibility studies and simulations (if applicable)
- Engineering analysis techniques and tools

5. **Prototyping and Implementation**
- Hands-on development of project prototypes
- Conducting experiments and data collection
- Troubleshooting and problem-solving during implementation

6. **Project Documentation and Reporting**
- Techniques for effective project documentation
- Writing comprehensive project reports and design documentation
- Organizing and presenting project data

7. **Project Presentation and Communication**

- Principles of effective communication in engineering
- Preparing engaging & informative project presentations
- Addressing questions & feedback during the presentation

8. **Project Evaluation and Assessment**

- Criteria for evaluating project success and achievement of objectives
- Conducting fair and unbiased project assessments
- Peer evaluations and constructive feedback.

ACTIVITIES AND EXECUTION GUIDELINES

1. **Project Proposal Submission:**

Students will submit their project proposals to the assigned mentors. The proposals should be well-structured, indicating the project's significance, expected outcomes, resources required, and a preliminary plan of action.

2. **Project Execution:**

During this period, students will work on their projects under the guidance of their mentors. They are encouraged to employ innovative techniques and apply engineering principles to achieve project objectives successfully.

3. **Project Documentation:**

Students will submit their final project reports and related documentation. The documentation should encompass all project phases, methodologies, experimental data, analysis, and outcomes.

4. Project Presentation:

Each student will deliver a comprehensive presentation to a panel of evaluators, showcasing their project's key aspects, results, and conclusions.

ASSESSMENT CRITERION

1. Project Proposal and Objective (10%)

Students are required to submit a comprehensive project proposal outlining the problem statement, objectives, scope, and methodology of the project. This component will account for 10% of the total marks.

2. Project Implementation (60%)

The core of the assessment will be based on the successful implementation of the project. Students will be evaluated on their ability to execute the project plan, adhere to timelines, and demonstrate practical engineering skills. This segment will carry 60% of the total marks.

3. Documentation (15%)

Proper documentation is vital to effective project management and communication. Students will be evaluated on the clarity, completeness, and organization of their project reports, design diagrams, code (if applicable), and any other relevant material. This component will contribute 15% of the total marks.

4. Project Presentation (15%)

Communication and presentation skills are crucial for engineers to articulate their ideas effectively. Students will be assessed based on their ability to present their project's

objectives, methodology, results, and conclusions in a clear and concise manner. This segment will be worth 15% of the total marks.

The Minor Project module is a pivotal component of the Engineering Diploma program that provides students with hands-on experience, encourages critical thinking, and prepares them for real-world engineering challenges. By adhering to the module guidelines and distribution of marks, students can excel in their projects and demonstrate their engineering prowess effectively.

*****