Semester I

Algorithmic Thinking with Python

(Stream A, B & C)

Course Code	Teaching hours per week				Credits	Exam hours	Course Type	CIE Marks	ESE Marks
	L	T	P	R					
26ABCES103	2	0	2	0	4	2 Hrs. 30 Mins	Theory	40	60

1. **Preamble :** This syllabus introduces foundational concepts and strategies essential for effective problem-solving using computational thinking. It covers algorithm design, pseudocode, flowcharts and core python programming constructs. Students will learn decomposition, modularization and recursion for building efficient solutions. Various computational approaches are explored to enhance analytical skills and solution development.

2. **Prerequisites :** Nil

Syllabus

Module Number	Syllabus Description	Contact Hours
1	**Problem Solving Strategies** - Problem-solving strategies defined, Importance of understanding multiple problem-solving strategies, Trial and Error, Heuristics, Means-Ends Analysis and Backtracking (Working backward). **The Problem Solving Process** - Computer as a model of computation, Understanding the problem, Formulating a model, Developing an algorithm, Writing the program, Testing the program and Evaluating the solution. **Algorithm and Pseudocode Representation -** Meaning and Definition of Pseudocode, Reasons for using pseudocode, The main constructs of pseudocode - Sequencing, selection (if-else structure, case structure) and repetition (for, while, repeat-until loops), Sample problems. **Flowcharts** - Symbols used in creating a Flowchart - start and end, arithmetic calculations, input/output operation, decision (selection), module name (call), for loop (Hexagon), flow-lines, on-page connector, off-page connector.	**10**
2	**Essentials of Python Programming** - Creating and using variables in Python, Numeric and String data types in Python, Using the math module, Using the Python Standard Library for handling basic I/O - print, input, Python operators and their precedence.	**15**

Module Number	Syllabus Description	Contact Hours
	Selection and Iteration Using Python - if-else, elif, for loop, range, while loop. **Sequence data types in Python** - list, tuple, set, strings, dictionary, creating and using arrays in Python (using Numpy library).	
3	**Decomposition and Modularization -** Problem decomposition as a strategy for solving complex problems, Modularization, Motivation for modularization, Defining and using functions in Python, Functions with multiple return values **Recursion** - Recursion Defined, Reasons for using Recursion, The Call Stack, Recursion and the Stack, Avoiding Circularity in Recursion, Sample problems - Finding the nth Fibonacci number, greatest common divisor of two positive integers, the factorial of a positive integer, adding two positive integers, the sum of digits of a positive number. **Introduction to files**.	**15**
4	**Computational Approaches to Problem-Solving -** (Introductory diagrammatic/algorithmic explanations only. Analysis not required) :- Brute-force Approach - - Example: Padlock, Password guessing Divide-and-conquer Approach - - Example: The Merge Sort Algorithm - Advantages of Divide and Conquer Approach - Disadvantages of Divide and Conquer Approach Dynamic Programming Approach - Example: Fibonacci series - Recursion vs Dynamic Programming Greedy Algorithm Approach: - Example: Given an array of positive integers each indicating the completion time for a task, find the maximum number of tasks that can be completed in the limited amount of time that you have. - Motivations for the Greedy Approach - Characteristics of the Greedy Algorithm - Greedy Algorithms vs Dynamic Programming Randomized Approach - Example 1: A company selling jeans gives a coupon for each pair of jeans. There are ‘n’ different coupons. Collecting ‘n’ different coupons would give you free jeans. How many jeans do you expect to buy before getting a free one?	**8**
	Total hours	**48**

Sample Questions for Practical Sessions :

- Simple desktop calculator using Python. *Only the five basic arithmetic operators*.
- Create, concatenate and print a string and access a sub-string from a given string.
- Familiarize time and date in various formats (Eg. “Thu Jul 11 10:26:23 IST 2024”).
- Write a program to create, append and remove lists in Python using NumPy.
- Program to find the largest of three numbers.
- Convert temperature values back and forth between Celsius (c) and Fahrenheit (f). [Formula: c/5 = f-32/9]
- Program to construct patterns of stars (*), using a nested for loop.
- A program that prints prime numbers less than ***N***.
- Program to find the factorial of a number using Recursion.
- Recursive function to add two positive numbers.
- Recursive function to multiply two positive numbers.
- Recursive function to find the greatest common divisor of two positive numbers.
- A program that accepts the lengths of three sides of a triangle as inputs. The program should output whether or not the triangle is a right triangle (Recall from the Pythagorean Theorem that in a right triangle, the square of one side equals the sum of the squares of the other two sides). Implement using functions.
- Program to define a module to find Fibonacci Numbers and import the module to another program.
- Program to check whether the given number is a valid mobile number or not using functions.
  - Every number should contain exactly 10 digits.
  - The first digit should be 7, 8 or 9
- Input two lists from the user. Merge these lists into a third list such that in the merged list, all even numbers occur first followed by odd numbers. Both the even numbers and odd numbers should be in sorted order.
- Write a program to play a sticks game in which there are 16 sticks. Two players take turns to play the game. Each player picks one set of sticks (needn’t be adjacent) during his turn. A set contains 1, 2, or 3 sticks. The player who takes the last stick is the loser. The number of sticks in the set is to be input.
- Suppose you're on a game show and you are given the choice of three doors: Behind one door is a car; behind the others, goats. You pick a door, say No. 1, and the host, who knows what is behind the doors, opens another door, say No. 3, which has a goat. He then asks, "Do you want to pick door No. 2?" Is it to your advantage to switch your choice? *(Reference source: https://en.wikipedia.org/wiki/Monty_Hall_problem#:~:text=The%20Monty%20Hall%20pr oblem%20is,the%20American%20Statistician%20in%201975.)*

3. Course Assessment Method

(CIE: 40 marks, ESE: 60 marks)

Continuous Internal Evaluation Marks (CIE) :

Attendance	Continuous Assessment (Accurate Execution of Programming Tasks)	Internal Examination-1 (Written Examination)	Internal Examination-2 (Written Examination)	Internal Examination-3 (Lab Examination)	Total
5	5	10	10	10	40

End Semester Examination Marks (ESE) :

In Part A, all questions need to be answered and in Part B, each student can choose any one full question out of two questions.

Part A	Part B	Total
● 2 Questions from each module. ● Total of 8 Questions, each carrying 3 marks **(8x3 =24 marks)**	● Each question carries 9 marks. ● Two questions will be given from each module, out of which 1 question should be answered. ● Each question can have a maximum of 3 subdivisions. **(4x9 = 36 marks)**	**60**

4. Course Outcomes

At the end of the course students should be able to:

Course Outcomes		Blooms Taxonomy Level
CO1	Apply appropriate problem-solving strategies and formulate algorithms, pseudocode and flowcharts for computational problems.	K3
CO2	Implement Python-based solutions using programming constructs and sequence data types.	K3
CO3	Apply decomposition, modularization, recursion and file handling to construct structured Python programs.	K3
CO4	Explain common algorithmic approaches for problem solving.	K2

Note: K1- Remember, K2- Understand, K3- Apply, K4- Analyze, K5- Evaluate, K6- Create

5. Mapping of Course Outcomes with Program Outcomes

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	3	3	2	-	-	-	-	-	-	2
CO2	3	3	3	2	2	-	-	-	-	-	2
CO3	3	3	2	-	2	-	-	-	-	-	2
CO4	3	3	2	-	-	-	-	-	-	-	2

Note: 1: Slight (Low), 2: Moderate (Medium), 3: Substantial (High), - : No Correlation

6. Text Books

Sl No	Title of Book	Name of Author/s	Publisher	Edition and Year
1	Problem solving & programming concepts	Maureen Sprankle, Jim Hubbard	Pearson	9/e, 2011
2	How to Solve It: A New Aspect of Mathematical Method	George Pólya	Princeton University Press	2/e, 2015
3	Creative Problem Solving: An Introduction	Donald Treffinger., Scott Isaksen, Brian SteadDoval	Prufrock Press	4/e, 2005
4	Python for Everyone	Cay S. Horstmann, Rance D. Necaise	Wiley	3/e, 2024

7. Reference Books

Sl No	Title of Book	Name of Author/s	Publisher	Edition and Year
1	Computational Thinking: A Primer for Programmers and Data Scientists	G Venkatesh Madhavan Mukund	Mylspot Education Services Pvt Ltd	1/e, 2020
2	Computer Arithmetic Algorithms	Koren, Israel	AK Peters /CRC Press	2/e, 2001
3	Introduction to Computation and Programming using Python	Guttag John V	PHI	2/e, 2016
4	Psychology (Sec. Problem Solving.)	Spielman, R. M, Dumper K Jenkins, W., Lacombe, A., Lovett, M., & Perlmutter	H5P Edition	1/e, 2021

8. Video Links or any other Reference Materials

Sl No	Details Like Weblink, Publication Details	Module
1	https://opentextbc.ca/h5ppsychology/chapter/problem-solving	1
2	https://onlinecourses.nptel.ac.in/noc21_cs32	2, 3
3	https://www.w3schools.com/python	2, 3
4	https://ocw.mit.edu/courses/6-046j-introduction-to-algorithms-sma-5503-fall-2005	4

9. Continuous Assessment (5 Marks)

Accurate Execution of Programming Tasks

- Correctness and completeness of the program
- Efficient use of programming constructs
- Handling of errors
- Proper testing and debugging 2. Model Question Paper

10. Evaluation Pattern for Lab Examination (10 Marks)

- Algorithm (2 Marks)
  Algorithm Development: Correctness and efficiency of the algorithm related to the question.
- Programming (3 Marks)
  Execution: Accurate execution of the programming task.
- Result (3 Marks)
  Accuracy of Results: Precision and correctness of the obtained results.
- Viva Voce (2 Marks)
  Proficiency in answering questions related to theoretical and practical aspects of the subject.

11. Model Question Paper

MODEL QUESTION PAPER			
FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY(FISAT) (Autonomous) FIRST SEMESTER B. TECH DEGREE EXAMINATION (2026 Scheme)			
Course Code : 26ABCES103 Course Name : Algorithmic Thinking with Python			
Programme : CE/CSE/CSD/ECE/EEE/EIE/ME			
Max. Marks: 60		**Duration: 2 hours 30 minutes**	
PART A			
(Answer all questions. Each question carries 3 marks)		**CO**	**Marks**
1	You are asked to solve a jigsaw puzzle without a reference picture. How will you solve it using trial and error method?	CO1	(3)

2		Draw the flowchart to generate the first 'n' numbers in the Fibonacci sequence.	CO1	(3)
3		Given the input "12345", write a Python code snippet to check if it is numeric.	CO2	(3)
4		Write the output produced by the following code. (i) for count in range(-10,-20,-2) : print(count, end=" ") (ii) for i in range(10,-1,-2): print(i) (iii) 2*3**2**2	CO2	(3)
5		What is circularity in recursion? How can you prevent circularity in recursive functions?	CO3	(3)
6		Write a Python program to find the sum of even numbers from N given numbers using a function.	CO3	(3)
7		What is the motivation for using a randomized approach in problem-solving?	CO4	(3)
8		You are a software engineer working on a security application for a company that handles sensitive user data. The application has a 4-digit numeric password system for authentication. However, due to a system error, a user has forgotten their password. Write an algorithm for recovering the 4-digit numeric password.	CO4	(3)
PART B				
(Answer any one full question from each module, each question carries 9 marks)				
Module - I			**CO**	**Marks**
9	a)	Explain how backtracking strategy can be applied to solve the Sudoku problem?	CO1	(5)
	b)	Write a pseudocode to implement a grading system based on the following conditions. **1.** Marks ≥ 90: Grade A **2.** Marks 80–89: Grade B **3.** Marks 70–79: Grade C **4.** Marks 60-70: Grade D **5.** Marks 50-69: Grade E **6.** Marks < 50 : Grade F **7.** If the difference between the marks and the next multiple of 10 is less than 3, then convert it to the next grade.	CO1	(4)

		8. If the marks are below 50, no need to convert even if the difference is less than 3.		
10	a)	Your college is located in a metropolitan city and you are new to that city. You would like to go out for dinner with your friends. Explain how heuristics approach can be used to find the best restaurant for dinner.	CO1	(5)
	b)	You are a software developer working on a data analysis tool for a logistics company. The company has a list of delivery times, where each entry represents the time taken (in minutes) for a particular delivery. The team needs to identify delivery times that are divisible by 6 (as they are considered optimal delivery durations) but not divisible by 4 (as they tend to involve more complexity in scheduling). Write an algorithm to print the delivery times that are divisible by 6 but not divisible by 4 from a given set of delivery times.	CO1	(4)
Module - II			**CO**	**Marks**
11	a)	Given the area of a circle. Write a Python program to calculate the radius of the circle using the *math* module.	CO2	(5)
	b)	Write a program that accepts the length of three sides of a triangle as input and determine whether or not the triangle is a right triangle.	CO2	(4)
12	a)	You are working on a text-processing application for a content moderation team. The team needs a feature to analyze user-generated sentences and reorganize their characters for better readability and analysis. Specifically, they want all capital letters to appear first, followed by small letters, then digits, and finally any special characters. Write a Python program to implement this feature, ensuring the reorganized output is displayed as a single word.	CO2	(5)
	b)	Write a Python function isValid_mobile_no to check whether a given number is a valid mobile number or not. The program should validate the number based on the following rules: 1. The number must contain exactly 10 digits. 2. The first digit of the number must be 7, 8, or 9.	CO2	(4)
Module - III			**CO**	**Marks**
13	a)	You are developing a feature for a business analytics tool that processes a list of sales figures for a company. As part of the analytics, the tool must identify the highest and lowest sales figures from the provided data. Write a python program to determine the largest and smallest numbers in a given list of N numbers, ensuring the solution is efficient for real-world applications. Use function.	CO3	(5)
	b)	You are working as a software engineer for a financial analysis company. The company uses Fibonacci sequences to model certain financial	CO3	(4)

		growth patterns, such as project milestones or revenue projections. You are tasked with developing a tool that generates the first 'n' numbers in the Fibonacci sequence to assist in creating projections for clients. Use recursion to solve it.		
14	a)	You are given two positive integers, a and b. Write a Python program using recursion to find the Least Common Multiple (LCM) of these integers. You are not allowed to directly calculate the LCM formula, but must instead use the relationship between GCD and LCM. Note: For any two positive integers, the product of their LCM and GCD is equal to the product of the two numbers. In other words, $lcm(a,b) \times gcd(a,b) = ab$	CO3	(5)
	b)	Write a Python program to Input two lists from the user. Merge these lists into a third list such that in the merged list, all even numbers occur first followed by odd numbers. Both the even numbers and odd numbers should be in sorted order. Use function.	CO3	(4)
		Module - IV	**CO**	**Marks**
15	a)	Illustrate the process of sorting the array [15, 8, 3, 12, 6, 10, 4, 1] using the merge sort algorithm. Draw a diagram showing how the array is split and merged at each stage.	CO4	(5)
	b)	Compare and contrast greedy approach and dynamic programming approach.	CO4	(4)
16	a)	Explain memoization and tabulation techniques in dynamic programming for calculating the n-th Fibonacci number.	CO4	(5)
	b)	Solve the following: You are given an array of positive integers where each integer represents the time taken to complete a task. Develop a greedy algorithm to determine the maximum number of tasks you can complete in a given total time limit. Explain your reasoning behind each step of the solution.	CO4	(4)
		******		

Focus on Excellence

FISAT
Focus on Excellence