SEMESTER S1

ALGORITHMIC THINKING WITH PYTHON

(Common for all branches)

Course Code	**ALP105**	**CIE Marks**	40
Teaching Hours/Week (L: T:P: R)	3:0:2:0	**ESE Marks**	60
Credits	4	**Exam Hours**	2 Hrs 30 mins
Prerequisites (if any)	None	**Course Type**	Theory

Course Objectives:

1. To provide students with a thorough understanding of algorithmic thinking and its practical applications in solving real-world problems.
2. To explore various algorithmic paradigms, including brute force, divide-and-conquer, dynamic programming, and heuristics, in addressing and solving complex problems.

SYLLABUS

Module No.	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. 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.	7
2	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*	9

	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. * - Evaluate an expression, d=a+b*c, find simple interest, determine the larger of two numbers, determine the smallest of three numbers, determine the grade earned by a student based on KTU grade scale (using if-else and case structures), print the numbers from 1 to 50 in descending order, find the sum of n numbers input by the user (using all the three loop variants), factorial of a number, largest of n numbers (Not to be limited to these exercises. More can be worked out if time permits). ** Only for visualizing the control flow of Algorithms. The use of tools like RAPTOR (https://raptor.martincarlisle.com/) is suggested. Flowcharts for the sample problems listed earlier may be discussed	
3	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). 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 **. * The idea should be introduced and demonstrated using Merge sort, the problem of returning the top three integers from a list of n>=3 integers as examples. (Not to be limited to these two exercises. More can be worked out if time permits). ** Not to be limited to these exercises. More can be worked out if time permits.	10
4	COMPUTATIONAL APPROACHES TO PROBLEM-SOLVING (Introductory diagrammatic/algorithmic explanations only. Analysis not required) :-	10

	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? - Example 2: n people go to a party and drop off their hats to a hat-check person. When the party is over, a different hat-check person is on duty and returns the n hats randomly back to each person. What is the expected number of people who get back their hats? - Motivations for the Randomized Approach	

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 =24marks)**	• 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**

Course Outcomes (COs)

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

Course Outcome		Bloom's Knowledge Level (KL)
CO1	Utilize computing as a model for solving real-world problems.	K2
CO2	Articulate a problem before attempting to solve it and prepare a clear and accurate model to represent the problem.	K3
CO3	Utilize effective algorithms to solve the formulated models and translate algorithms into executable programs.	K3
CO4	Interpret the problem-solving strategies, a systematic approach to solving computational problems, and essential Python programming skills	K3

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

CO-PO Mapping Table:

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

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

Text Books				
Sl No.	**Title of the book**	**Name of the Author/s**	**Name of the Publisher**	**Edition and Year**
1	Fundamentals of Python: First Programs	Kenneth A Lambert	Cengage Publishing	2/e, 2016
2	Python for Data Analysis	Wes McKinney	Shroff / O'Reilly publishers	2/e, 2017
3	The Complete Reference Python	Martin C Brown	Mc Graw Hill	1/e, 2018
4	Introduction to Python for Science & Engineering	David J. Pine	CRC Press	1/e. 2021

Reference Books				
Sl No.	**Title of the book**	**Name of the Author/s**	**Name of the 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 Stead-Doval	Prufrock Press	4/e,2005
4	Psychology (Sec. Problem Solving.)	Spielman, R. M., Dumper, K., Jenkins, W., Lacombe, A., Lovett, M., & Perlmutter, M	H5P Edition	1/e, 2021
5	Computational Thinking: A Primer for Programmers and Data Scientists	G Venkatesh Madhavan Mukund	Mylspot Education Services Pvt Ltd	1/e, 2020
6	Computer Arithmetic Algorithms	Koren, Israel	AK Peters/CRC Press	2/e, 2001
7	Python for Everyone	Cay S. Horstmann, Rance D. Necaise	Wiley	3/e, 2024
8	Introduction to Computation and Programming using Python	Guttag John V	PHI	2/e., 2016

Video Links (NPTEL, SWAYAM…)	
Module No.	**Link id**
1.	https://opentextbc.ca/h5ppsychology/chapter/problem-solving/
2.	https://onlinecourses.nptel.ac.in/noc21_cs32/preview

1. **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. **Evaluation Pattern for Lab Examination (10 Marks)**

   1. **Algorithm (2 Marks)**

      Algorithm Development: Correctness and efficiency of the algorithm related to the question.

   2. **Programming (3 Marks)**

      Execution: Accurate execution of the programming task.

   3. **Result (3 Marks)**

      Accuracy of Results: Precision and correctness of the obtained results.

   4. **Viva Voce (2 Marks)**

      Proficiency in answering questions related to theoretical and practical aspects of the subject.

Sample Classroom Exercises:

1. Identify three ill-defined problems and well-defined problems
2. Identify five use cases for Trial and error, Heuristics, backtracking, and Means-ends analysis.
3. Use a diagram to solve the Tower of Hanoi for three pegs with the minimum number of moves.
4. Evaluate different algorithms discussed earlier based on their efficiency by counting the number of steps.
5. A recursive function that takes a number and returns the sum of all the numbers from zero to that number.
6. A recursive function that takes a number as an input and returns the factorial of that number.
7. A recursive function that takes a number 'n' and returns the nth Fibonacci number.
8. A recursive function that takes an array of numbers as input and returns the product of all the numbers in the array.
9. A program to reverse the contents of an 1D array without using a second array.
10. To register for the end-semester examination, you need to log into the University portal with your credentials. Write a program to validate the credentials. Assume that the usernames are stored in an array of strings called USERNAME and the corresponding passwords are stored in another array of strings called PASSWORD such that password[i] is the password for the user username[i].
11. You are given a list and your task is to divide it to make two smaller lists. The sublists should be made from alternate elements in the original list. So if the original list is {5,1,4,12,6}, then one sublist should be {5,4,6} and the other should be {1,12}.
12. A program that takes three points in a 2D plane and determines whether they are collinear. Two pairs of points are collinear if they have the same slope.

LAB Experiments:

1. Simple desktop calculator using Python. Only the five basic arithmetic operators.
2. Create, concatenate, and print a string and access a sub-string from a given string.
3. Familiarize time and date in various formats (Eg. “Thu Jul 11 10:26:23 IST 2024”).
4. Write a program to create, append, and remove lists in Python using NumPy.
5. Program to find the largest of three numbers.
6. Convert temperature values back and forth between Celsius (c), and Fahrenheit (f). [Formula: c/5 = f-32/9]
7. Program to construct patterns of stars (*), using a nested for loop.
8. A program that prints prime numbers less than N.
9. Program to find the factorial of a number using Recursion.
10. Recursive function to add two positive numbers.
11. Recursive function to multiply two positive numbers.
12. Recursive function to find the greatest common divisor of two positive numbers.
13. 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.
14. Program to define a module to find Fibonacci Numbers and import the module to another program.
15. Program to check whether the given number is a valid mobile number or not using functions. Rules: 1. Every number should contain exactly 10 digits. 2. The first digit should be 7 or 8 or 9
16. 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.
17. 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.
18. 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? (source:https://en.wikipedia.org/wiki/Monty_Hall_problem#:~:text=The%20Monty%20Hall%20problem%20is,the%20American%20Statistician%20in%201975.)