19EID131: PROBLEM SOLVING AND PROGRAMMING WITH PYTHON
(Common to all)

L	T	P	C
3	1	3	5.5

This course focuses on problem solving using visual programming and flowchart tools. Python being simple and easy to learn syntax, it is used as an introductory coding platform to translate flow charts into programs. The course introduces fundamental programming concepts. Python language is used to present concepts including control structures, functions, data structures followed by important Python packages that will be useful in data analysis.

Course Objectives:

- To introduce programming through Visual programming tool - Scratch
- To teach problem solving through Flow charting tool - Raptor
- To elucidate problem solving through python programming language
- To introduce function-oriented programming paradigm through python
- To train in development of solutions using modular concepts
- To teach practical Pythonic solution patterns

UNIT I: Computational Thinking and Visual Programming Concepts   10 L+6P

Introduction to computational thinking. Visual programming concepts. Scratch environment: sprites -- appearance and motion, angles and directions, repetition and variation, changing costumes, adding background. Input/Output, variables and operators.

Learning Outcomes
After completion of this unit the student will be able to
- develop a program, controlled by a loop. (L3)
- experiment with "costumes" to change the appearance of sprites. (L3)
- perform Input, Output Operations using scratch. (L3)
- perform computation using common mathematical formulas. (L3)
- develop programs by passing messages between sprites. (L3)

UNIT II: Algorithms and Flowchart design through Raptor   10L+6P
Introduction to the idea of an algorithm. Pseudo code and Flow charts. Flow chart symbols, Input/Output, Assignment, operators, conditional if, repetition, procedure and sub charts.

Example problems – Finding maximum of 3 numbers, Unit converters, Interest calculators, multiplication tables, GCD of 2 numbers

Example problems -- Fibonacci number generation, prime number generation. Minimum, Maximum and average of n numbers, Linear search, Binary Search.

Learning outcomes:
After completion of this unit the student will be able to
- select flowchart symbols for solving problems. (L1)

- develop basic flowcharts for performing Input, Output and Computations (L3)
- solve numerical problems using Raptor (L3)
- analyze problems by modular approach using Raptor (L4)

UNIT III: Introduction to Python   **10L+6P**
Python – Numbers, Strings, Variables, operators, expressions, statements, String operations, Math function calls, Input/Output statements, Conditional If, while and for loops, User defined Functions, parameters to functions, recursive functions, Turtle Graphics.

Learning outcomes:
  After completion of this unit the student will be able to
- interpret numbers, strings, variables, operators, expressions and math functions using Python Interactive Mode. (L2)
- solve simple problems using control structures, input and output statements. (L3)
- develop user defined functions (recursive and non-recursive). (L3)
- build Python programs for section 1 raptor flowcharts. (L3)
- develop Python programs for creating various graphical shapes using turtle graphics. (L3)

UNIT IV: Data Structures and Idiomatic Programming in Python   **10L+6P**
Lists, Tuples, Dictionaries, Strings, Files and their libraries. Beautiful Idiomatic approach to solve programming problems.

Learning outcomes:
  After completion of this unit the student will be able to
- summarize the features of lists, tuples, dictionaries, strings and files. (L2)
- demonstrate best practices of "Beautiful Idiomatic Python". (L2)
- build Python programs for section 2 raptor flowcharts. (L3).

UNIT V: Packages   **10L+6P**
Numpy -- Create, reshape, slicing, operations such as min, max, sum , search, sort, math functions etc.
Pandas -- Read/write from csv, excel, json files, add/ drop columns/rows, aggregations, applying functions
Matplotlib -- Visualizing data with different plots, use of subplots.
User defined packages, define test cases and perform unit testing

Learning outcomes:
  After completion of this unit the student will be able to
- read data from files of different formats and perform operations like slicing, insert, delete, update (L3)
- visualize the data (L4)
- ability to define packages (L2)
- define test cases (L1)

Problem Solving and Programming with Python Laboratory

Laboratory Experiments

1. Design a script in Scratch to make a sprite to draw geometrical shapes such as Circle, Triangle, Square, Pentagon.
2. Design a script in Scratch to make a sprite to ask the user to enter two different numbers and an arithmetic operator and then calculate and display the result.
3. Design a Memory Game in Scratch which allows the user to identify positions of similar objects in a 3 x 3 matrix.
4. Construct flowcharts to
    a. calculate the maximum, minimum and average of N numbers
    b. develop a calculator to convert time, distance, area, volume and temperature from one unit to another.
5. Construct flowcharts with separate procedures to
    a. calculate simple and compound interest for various parameters specified by the user
    b. calculate the greatest common divisor using iteration and recursion for two numbers as specified by the user
6. Construct flowcharts with procedures to
    a. generate first N numbers in the Fibonacci series
    b. generate N Prime numbers
7. Design a flowchart to perform Linear search on list of N unsorted numbers (Iterative and recursive)
8. Design a flowchart to perform Binary search on list of N sorted numbers (Iterative and recursive)
9. Design a flowchart to determine the number of characters and lines in a text file specified by the user
10. Design a Python script to convert a Binary number to Decimal number and verify if it is a Perfect number.
11. Design a Python script to determine if a given string is a Palindrome using recursion
12. Design a Python script to sort numbers specified in a text file using lists.
13. Design a Python script to determine the difference in date for given two dates in YYYY:MM:DD format ($0 <= YYYY <= 9999, 1 <= MM <= 12, 1 <= DD <= 31$) following the leap year rules.
14. Design a Python Script to determine the Square Root of a given number without using inbuilt functions in Python.
15. Design a Python Script to determine the time difference between two given times in HH:MM: SS format. ($0 <= HH <= 23, 0 <= MM <= 59, 0 <= SS <= 59$)
16. Design a Python Script to find the value of (Sine, Cosine, Log, PI, *e)* of a given number using infinite series of the function.
17. Design a Python Script to convert a given number to words.
18. Design a Python Script to convert a given number to roman number.
19. Design a Python Script to generate the frequency count of words in a text file.
20. Design a Python Script to print a spiral pattern for a 2 dimensional matrix.
21. Design a Python Script to implement Gaussian Elimination method.
22. Design a Python script to generate statistical reports (Minimum, Maximum, Count, Average, Sum etc) on public datasets.
23. Design a Python script using the Turtle graphics library to construct a turtle bar chart representing the grades obtained by N students read from a file categorizing them into distinction, first class, second class, third class and failed.

Text Book(s):
1. Weingart, Dr. Troy, Brown, Dr. Wayne, An introduction to programming and algorithmic reasoning using raptor.
2. T R Padmanabhan, Programming with python, Springer.
3. Reema Thareja, Python Programming: Using Problem Solving Approach, Oxford University Press.
4. Wes McKinney , Python for Data Analysis, O.Reilly.

Course outcomes:
 After the completion of the course, the student will be able to
- create interactive visual programs using Scratch. (L6)
- develop flowcharts using raptor to solve the given problems. (L3)
- build Python programs for numerical and text based problems(L3)
- develop graphics and event based programming using Python(L3)
- build Python programs using beautiful Pythonic idiomatic practices (L3)