Competency 9: Develops algorithms to solve problems and uses python programming language to encode algorithms

Competency Level 9.1: Uses problem-solving process
Competency Level 9.2: Explores the top down and stepwise refinement methodologies in solving problems

Time: 06 periods

Learning Outcomes:

- Describes the steps of problem solving process
- Implements the solution Briefly describes Candidate key, primary key, alternate key and foreign key
- Uses stepwise refinement methodology to solve problems
- Draws structures charts to illustrate a solution for a system

Contents:

- Understanding the problem
- Defining the problem and boundaries
- Planning solution
- Implementation
- Modularization
- Top down design and stepwise refinement
- Structure charts

Concepts and terms to be highlighted:

- Appreciate the process of problem solving, using critical thinking and logical process
- Design a system by breaking the system into small subsystem and connecting them logically
- Structure charts
- Illustrate how structure charts can be used to show the breakdown of a system to its lowest manageable levels

Guidance for lesson plans:

- Divide the students into groups
- Provide simple problems to each group and ask them to discuss the problem solving process and present it
- Discuss the systematic process of problem solving
- Ask each group to identify a problem to solve
- Ask the group to break the problem into small sub parts (subsystems)
- Discuss the top down and stepwise refinement methodology in problem solving
- Introduce Structure Charts

Guidance for assessments and evaluations:

- Provide a problem and ask the students to write down the steps to solve it

- Ask the student to draw the Structure Chart for the problem

Quality inputs:

- Printout with problems, presentations

Competency Level 9.3: Uses algorithmic approach to solve problems

Time: 06 periods

Learning Outcomes:

- Briefly describes algorithms
- Identifies the standard symbols used to draw flow charts
- Draws flow charts to illustrate solutions to a given problem
- Writes pseudo codes to illustrate solutions to a given problem
- Uses hand traces to verify the solutions

Contents:

- Algorithms
  - Flow charts
  - Pseudo codes
  - Hand traces

Concepts and terms to be highlighted:

- Algorithm
- Algorithm is the step by step process to be followed in solving a problem
- Sequence
- Selection
- Repetition
- Flow chart
- Flow chart is a diagram used to represent an algorithm
- Pseudo code
- Pseudo code is a high level description used to represent an algorithm

Guidance for lesson plans:

- Provide a problem and ask the students to write down the solution in stepwise
- Discuss their stepwise solution and introduce the term algorithm and its representations (flow chart and pseudo code)
- Provide some situations of sequence, selection, repetition and discuss each of them
- Provide some sample problems and discuss how to show their solutions in flow charts and pseudo code
- Provide some problems and ask the students to create algorithms in both flowcharts and pseudo code and present it

Guidance for assessments and evaluations:

- Provide problems involving sequence only, selection, iteration selection and iteration and ask the students to create algorithms in both flow chart and pseudo code
- Provide flow charts and ask the students to find the output of the problem

Quality inputs:

- Printouts with flowcharts and pseudo code, presentations

Competency Level 9.4: Compares and Contrasts different programming paradigms

Time: 02 periods

Learning Outcomes:

- Describes the evolution of programming language in terms of generations
- Compares and contrasts imperative, declarative, object oriented languages

Contents:

- Evolution of programming languages
- Programming paradigms
  - Imperative languages
  - Declarative languages
  - Object oriented languages

Concepts and terms to be highlighted:

- Programming languages
- Evaluation of programming languages
- Programming paradigms
- The specific style of programming is called Programming Paradigms
- According to the paradigms of programming, the programming languages can be classified as follows
- Imperative languages
- Declarative languages
- Object oriented languages

Guidance for lesson plans:

- Divide the student into groups
- Ask each group to search about the evolution of programming language in internet and present their findings.
- Discuss programming paradigms. (imperative, declarative and object oriented)

Guidance for assessments and evaluations:

- Divide the student into groups
- Ask each group to prepare a presentation on evolution of programing languages and present it.
- Ask student to find at least one language under each programing paradigm.

Quality inputs:

- Internet access facilities
- Presentations
- Computers with presentation software

Competency Level 9.5: Explores the need of program translation and the type of program translators

Time: 02 periods

Learning Outcomes:

- Describes the need of translation of a program
- Compares the source and object program
- Lists and briefly describes the types of program translators
- Briefly describes the function of linkers

Contents:

- Need of program translation
- Source program
- Object program
- Program translators
  - Interpreters
  - Compilers
  - Hybrid approach

  Linkers

Concepts and terms to be highlighted:

- Appreciating that a computer programs are executable in binary format only (binary files)
- Identifying the difference between source program and object (executable) program
- Appreciating the need for translating (interpreting/ compiling) the computer programs written in high level languages (source program) to binary format (executable program)
- Appreciating the difference between compiling and interpreting
- Compiling Process – Translating the complete source program as a whole to equivalent binary format and make it as one executable program
- Interpreting Process – Translating the source code line by line to equivalent binary format at the time of execution
- Hybrid Approach – Combining of interpreting and compiling process
- Appreciating how linkers do dynamic translation/ compiling
- Source Program, Object Program, Interpreter, Compiler, Linker, Loader

Guidance for lesson plans:

- Show a source file and the corresponding object file of a computer program, and ask students to identify the differences in them, and discuss.
- Discuss the process of translating a natural language
- Discuss the process of translating a computer programming language to make students appreciate the fact that some kind of interface (translator) is necessary in order for the computer to understand what is written (commands) by humans
- Show an example source program of an interpreting language and execute it and demonstrate that interpreting and executing are happening line by line
- Show an example source program of a compiling language and execute it and demonstrate that compiling and executing are happening at separate times
- Show the system overview of an example process where both an interpreting and compiling are done (Hybrid Approach)

- Discuss the function of Linkers/ Loaders and how they function using a program flow diagram

Guidance for assessments and evaluations:

- Provide questions like what is a compiler/ interpreter? Why is it necessary to use a Compiler/ Interpreter? etc.

Quality inputs:

- Computers with python programming language, Presentations

Competency Level 9.6: Explores integrated development environment (IDE) to identify their basic features

Time: 04 periods

Learning Outcomes:

- Identifies the basic features of IDE
- Practices the instructions to
  - Open and save files
  - Compile, execute programs
- Uses the debugging facilities in IDE

Contents:

- Basic features of IDE
- Instructions to use
  - Opening and saving files
  - Compiling, executing programs
- Debugging facilities

Concepts and terms to be highlighted:

- Appreciate the fact that to do programing, it is necessary to have software like Editors, Compilers, Debuggers and/ or Frameworks (IDEs)
- Appreciate the fact that IDEs are comprehensive software containing Editors, Compilers, and Debuggers etc.
- Editors, Compilers, Debuggers, IDEs.

Guidance for lesson plans:

- Show an IDE of a particular language, and explain & discuss the features (Editing, Compiling, Debugging etc. functions)

Guidance for assessments and evaluations:

- Provide a IDE of a particular language and ask the students to identify the its' features

Quality inputs:

- Computers with python software
- Presentations

Competency Level 9.7: Uses an imperative programming language to encode algorithms

Time: 10 periods

Learning Outcomes:

- Identifies the structure of a program
- Uses comments to identify the usage of code for future reference
- Uses constants and variables in a program appropriately
- Identifies the primitive data types of a given program language
- Identifies and uses operators in a program
- Identifies precedence of operators
- Writes programs with the facilities of input from keyboard and output to standard devices

Contents:

- Structure of a program
- Comments
- Constants and Variables
- Primitive data types
- Operator categories
  - Arithmetical, relational, Logical, bitwise
- Operator precedence
- Input / output
  - Input from keyboard
  - Output to standard devices

Concepts and terms to be highlighted:

- Appreciate the fact that a computer programs should contain all the commands explicitly and they should be constructed according to a standard structure
- Appreciate the fact that a structure should necessarily have an identifier (a header), definitions (declaration of variables, constants etc.), a body (where the processing is done), and an output section (output commands) in that order
- Appreciate the need for adding non executable statements (comments) in order to understand the functions of the statement at a later time or by another person
- Appreciate the difference between constants and variables, and that they are necessary in order to do calculations in the body of a program (Example: In y= mx + c, m and c are constants and x is a independent variable, x, c, and m determines the value of y, the dependent variable)
- Appreciate that each language has its' own data types
- Identify the types of data of the language being used
- Appreciate that data may be operated by an operation to get an output that combines or compares two or more data items
- Identify the categories of operators: Arithmetic, Relational, Logical, Bitwise
- Appreciate the fact that any equation has operators of more than one category and they need to be given an order of evaluation (precedence) according to a standard method
- Appreciate the fact that in any system, there should be an input device to input data in to a processing unit to process the data, and an output device to display the processed data, and there should be statements to input data and output data according to a specific language

- Appreciate the data input and the need for type conversion of input data
- Data, Data types, Variables, Constants, Comments, Operators, Operator Precedence, Input/ Output

Guidance for lesson plans:

- Discuss the need for a structure of any Document (Letter, Essay etc.) – Header, Body, Conclusion
- Discuss the need for a structure for a computer program
- Show a sample program (in the programming language being taught) discuss the structure of the program: Program identifier, Declaration of variables & constants, Body (where the processing is done), and Output section (output commands) in that order
- Divide students into groups
- Provide the groups with a simple sample program in the programming language being taught (without comments) and make them realize that it is difficult to know what the program statements do without a comment in natural language at the end of the statement
- Discuss the need for comments in programs, as well as the syntax of the programming language being taught
- Give the equation y= mx + c, where c and m are constants and x is an independent variable, x, c and m determine the value of y, the dependent variable, and the groups to identify their (c, x and y) nature and behavior
- Discuss and introduce the nature and characteristic constants and variables and how they are declare and use (their syntax) in the programming language being taught
- Discuss the concept of data types and different data types of the language being used, and their syntax in the programming language being taught
- Show some sample programs that contain different data types and ask students to identify them
- Ask student groups to write a simple program of few lines, compile it and run. If there are compiling errors, ask students to identify them. If there are no compiling errors, ask them to introduce an error and study the error report after compiling
- Discuss commonly used operators
- Discuss and explain the operators and their syntax of the language being taught
- Ask student groups to write simple programs which contain as many operators (arithmetic, relational, logical, and bitwise) as possible
- Give an example equation containing operators of different precedence and ask the groups to solve the equation and get the answer
- Discuss with students the solving procedure (according to precedence) and how there can be different answers in a different precedence
- Discuss the importance of a standard precedence of operators, and introduce the precedence of operators of the language being taught
- Ask student groups to write simple programs to input data, do some processing and display

Guidance for assessments and evaluations:

- Provide simple equations and ask students to write a program with comments to read data from keyboard/ mouse and display the results
- Check whether students have followed the correct structure when writing the program
- Check whether the comments written by the students are appropriate
- Ask students to develop programs to some sample real world situations (Example: Converting Farenheight to Centigrade using C=(F-32)*5/9) and ask students to identify the variable(s) and constant(s) in it ask the groups to write a simple program to input different input values in Farenheight and display the corresponding Centigrade value on a display
- Write an equation with many data types (integer, floating point, string, date etc.) and ask students to categorize them into types
- Ask students to identify different operators in the equation and the correct precedence

- Give a complex equation and ask students to use brackets in cases where the compiler is not able to identify the precedence by default

Quality inputs:

- Computers with python programming language
- Sample programs
- Presentations

Competency Level 9.8: Uses control structures in developing programs

Time: 12 periods

Learning Outcomes:

- Briefly describes control structures
- Lists and briefly describes the types of control structures
- Uses control structures appropriately in programming
- Applies nested control structures in programs

Contents:

- Control Structures
  - Sequence
  - Selection
  - Repetition
    - ❖ Iteration
    - ❖ Looping

Concepts and terms to be highlighted:

- Appreciate the need for control structures to execute a program appropriately
- Appreciate the fact that when executing a program, the commands are executed sequentially
- Appreciate the fact that program could execute in separate paths (two or more) according to a condition (If, If and Else)
- Appreciate the fact that some sequences need to be repeated (pre-determined or post- determined number of times – Iteration and Looping)
- Appreciate the fact that there can be sub repetitions inside a repetition (Nested Loops)
- Control structures – sequence, selection. repetitions – iteration, looping

Guidance for lesson plans:

- Show a complex program and explain the structure highlighting Conditional Statements.
- Divide the students into groups
- Ask student groups to get the output of a program and compare it with the source code and identify that the output is in sequence with the corresponding input commands in case of sequential (non-conditional and non-repetitive) execution
- Ask student groups to write a program to solve a simple problem using conditional statements (Example: To apply a 30 per cent discount for all purchases of more than 400,000 Rs., else no discount)
- Ask student groups to extend their solution to different levels (20 per cent for more than 300,000 Rs., and 10 per for more than 100,000 Rs., less than 100,000 Rs., no discount)
- Ask student groups to develop a grading systems (A. B, C, and F for marks ranges 75-100, 50-74, 40-49, Less than 40 respectively)
- Ask student groups to write a program add numbers from 1 to 100 using “For” statements
- Ask student groups to write a program to display the first ten triangular numbers
- Ask students to write a program to draw a given triangular number graphically using a “While” loop
- Ask the student groups to write a program to add numbers 1, 2, 3, 4 an so on until the total is less than 50

Guidance for assessments and evaluations:

- Show a complex program and ask students to identify the structure highlighting Conditional Statements.
- Provide the output of a program and ask student groups to compare it with the source code and identify that the output is in sequence with the corresponding input commands in case of sequential (non-conditional and non-repetitive) execution
- Ask student groups to write a program step by step (while executing each structure and observing the output) incorporating all the control structures (Conditions and Repetitions)
- Given a practical problem, ask the groups to develop a solution using appropriate control structures and write a program

Quality inputs:

- Computers with python programming language
- Sample programs
- Presentations

Competency Level 9.9: Uses sub-programs in programming

Time: 10 periods

Learning Outcomes:

- Briefly describes the functions
- Lists and briefly describes the types of functions
- Identifies the structure of a function
- Compares local and global variables
- Identifies the behavior of a variable in terms of life time
- Identifies the need of return values and writes functions to obtain the appropriate return value
- Writes functions using relevant parameters and arguments
- Uses user defined functions

Contents:

- Types of subprograms
  - Built in
  - User defined
    - Structure
    - Parameter passing
    - Return values
    - Default values
    - Scope of variables

Concepts and terms to be highlighted:

- Appreciate the fact that a program is composed of logical sub components (programs) connected together
- Appreciate the fact that such components could be pre-programmed and stored in a library for later use by many programs (Built in sub programs)
- Appreciate the fact that in case where there are no built in sub programs to be used as per the programmer's specific need, the programmer can write sub programs (user defined sub programs)
- Appreciate the fact that such user defined sub programs could be stored in a library for later use
- Identify the essential structure and operation of sub programs – inputting data from main program to sub program (parameter passing), and returning the output values from sub program to main program (return values)
- Identify the characteristics of variables that are passed as input and as output (local and global variables, parameter passing by reference (address) and by values)
- Built in and User defined sub programs, Parameter passing, Return values, Default values

Guidance for lesson plans:

- Divide the students into groups
- Provide the groups with sample programs composed few sub programs
- Discuss and introduce a sub program and its' function and behavior
- Ask student groups to identify sub programs in the sample programs
- Describe some built in sub programs in the language being taught

- Question students what they would do if there are no built in sub programs and make students appreciate that such programs can be written by themselves (User defined sub programs)
- Taking a built in sub program as an example, discuss the structure of sub program and how values are being exchanged between the main program and the sub program
- Discuss the nature of global variables and local variables
- Discuss parameter passing by value and by address

Guidance for assessments and evaluations:

- Ask the students to Identify sub programs in the given sample programs
- Ask students to identify which built in sub programs can be used for a given scenario and use them to solve the problem
- Ask students to write a sub program to draw a specific triangular number
- Ask students to write a program to calculate a simple physical quantity (Example: Perimeter of a circle, Area of a square etc.)
- Ask students to make it as a sub program that can be later used in main program (considering parameter passing by reference/ by value, global/ local variable etc.)

Quality inputs:

- Computers with python programming language
- Sample programs
- Presentations

Competency Level 9.10: Uses data structures in programs

Time: 08 periods

Learning Outcomes:

- Briefly explains the use of data structures
- Uses relevant data structures in programming

Contents:

- Data structures
  - Strings
  - Lists
  - Tuples
  - Dictionaries

Concepts and terms to be highlighted:

- Appreciate the fact that data need to be organized in a certain pattern and order and saved before inputting the data to a program
- Appreciate that different problems need different data structures suitable for the purpose
- Appreciate the characteristics and the method of implementation (array based or otherwise) of the data structures of the language being taught
- Data structures – Strings, Lists, Tuples, Dictionaries

Guidance for lesson plans:

- Explain and demonstrate the data structures Strings, Lists, Tuples, Dictionaries in detail
- Divide the students into groups
- Give a set of data and ask student groups to organize them in a suitable way for inputting to serve a particular purpose (Example: Taking data items one after the other, Taking a pair of data (Example: item name and price) one after the other)
- Ask student groups to write programs using the data defined according to the above mentioned structures

Guidance for assessments and evaluations:

- Ask students to identify the data structures Strings, Lists, Tuples, Dictionaries used in a sample program
- Ask student to write programs to create data structures Strings, Lists, Tuples, Dictionaries to be used in a program
- Ask student groups to write programs using the above mentioned data structures

Quality inputs:

- Computers with python programming language
- Sample programs
- Presentations

Competency Level 9.11: Handles files and databases in programs

Time: 06 periods

Learning Outcomes:

- Uses basic file operations (open, close, read write and append)

Contents:

- File handling
    - Basic file operations

Concepts and terms to be highlighted:

- Appreciate the fact that programs need to get data and information from saved files
- Appreciate the fact that such data can be in different as text files
- Appreciate how to refer to a file in a program to get connected to a file
- Appreciate how to open a file, how to read & append data items in it and how to close the file
- Appreciate the situations where file handling is useful in problem solving process
- File handling, basic file operations (open, read/ append and close)

Guidance for lesson plans:

- Discuss how a file can be opened and closed in applications software (Example: Notepad, Word etc.)
- Create a suitable text file and a program for demonstrating file handling operations
- Demonstrate how to create a text file by using a program
- Demonstrate how a file can be accessed via a program in execution
- Divide the students into groups
- Provide some text files and ask the groups to write programs to access them and read and append data
- Ask the groups to write a program for a practical situation

Guidance for assessments and evaluations:

- Ask students to create a sample text file using a Text Editor (Example: Notepad)
- Ask students to open/ read/ close a text file manually and using a program
- Ask students to create a text file and do file handling operations by using a program

Quality inputs:

- Computers with python programming language
- Sample programs
- Presentations

Competency Level 9.12: Manages data in databases

Time: 04 periods

Learning Outcomes:

- Embeds SQL statements in programming languages to retrieve, add, modify and delete data

Contents:

- Connecting to a database
- Retrieve data
- Add, modify and delete data

Concepts and terms to be highlighted:

- Appreciate the fact that programs need to get data and information from saved database file
- Appreciate how to create a simple database to do database handling operations
- Appreciate how to refer to a database in a program to get connected to a database
- Appreciate how to open a database and how to create a suitable interface program containing forms with validations to retrieve, update, delete and search a database record conveniently
- Appreciate the application of SQL statements to do the above mentioned operations
- Appreciate the simple multiple tables applications of databases to solve practical problems
- Database connectivity, Database handling operations

Guidance for lesson plans:

- Divide the students in to groups
- Ask the groups to create a suitable simple database
- Ask the groups to write a program to get connected to a database
- Ask students to create a program for a given scenario (database) with validations to retrieve, update, delete and search a database record conveniently

Guidance for assessments and evaluations:

- Ask the groups to create a suitable simple database for a given scenario
- Ask the groups to write a program to get connected to the developed database
- Ask students to create a program for a given scenario (database) with validations to retrieve, update, delete and search a database record conveniently

Quality inputs:

- Computers with python programming language and database software
- Sample programs
- Presentations

Competency Level 9.13: Searches and sorts data

Time: 04 periods

Learning Outcomes:

- Uses sequential searching technique appropriately
- Implements bubble sort technique appropriately

Contents:

- Searching techniques
    - Sequential search
- Sorting techniques
    - Bubble sort

Concepts and terms to be highlighted:

- Appreciate the fact that in real life situations, we have to sort things in order to search and find them fast and easily
- Appreciate the need for sorting data in order to search and find them fast and easily
- Appreciate the concept of searching using Sequential Search techniques in a data set stored in an array
- Appreciate the concept of sorting using Bubble Sort technique
- Appreciate the concept and technique (the need for a dummy variable) of swapping two data items
- Sorting and Searching, Sequential Search, Bubble Sort

Guidance for lesson plans:

- Give numbers randomly to a set of students in the class
- Ask a student to find the student holding a specific number and let the students feel the difficulty in finding the number due to the fact that the numbers are not sorted
- Ask the groups to write a program according to the physical method they were using for searching (Sequential Search)
- Ask students to stand in to a row in an order of the number (ascending or descending) using the Bubble Sort technique starting from the original unsorted set
- Ask the groups to write a program according to the physical method they were using for sorting (Bubble Sort)

Guidance for assessments and evaluations:

- Ask the students to write a program to search a data according to a given situation using Sequential Search
- Ask the students to write a program to sort data according to a given situation using Bubble Sort

Quality inputs:

- Computers with python programming language
- Sample programs
- Presentations