b) Sorting (1) Insertion sort (2) Selection sort

9. Write a program to copy one file to another, use command line arguments.
10. Write a program to mask some bit of a number (using bit operations)
11. An array of records contains information of managers and workers of a company. Print all the data of managers and workers in separate files.

Semester	**: II**
Course Type	**: SEC**
Course Code	**: CSCSEC151**
Name of the Course	**: Python Programming**
Learning level	**: Foundation or Introductory Course**
Credits	**: 3**
Contact Hours	**: 30**
Total Marks	**: 100**
End Semester Marks	**: 80 [50 (Theory) + 30 (Lab)]**
Internal Marks	**: 20**

Course Objectives:

1. *Familiarize students with the basics of Python programming language, including syntax, variables, data types, and control structures syntax, variables, data types, and control structures.*
2. *Introduce students to fundamental programming concepts such as variables, data types, operators, conditional statements, loops, and functions, within the context of Python.*
3. *Explore various data structures in Python, such as lists, tuples, dictionaries, and sets, and develop the skills to manipulate, access, and organize data using these structures.*
4. *Teach students how to read from and write to files using Python, including text files and CSV files, enabling them to handle data stored in external files.*
5. *Develop skills in handling errors and exceptions in Python programs, allowing for graceful error handling and robustness.*

UNIT-I

Introduction: Basic Elements of Python, Python character set, Python tokens (keyword, identifier, literal, operator, punctuator), variables, concept of l-value and r-value, use of comments, Knowledge of data types: Number(integer, floating point, complex), boolean, sequence(string, list, tuple), None, Mapping(dictionary), mutable and immutable data types.Operators: arithmetic operators, relational operators, logical operators, assignment operators, augmented assignment operators, identity operators (is, is not), membership operators (in not in)Expressions, statement, type conversion, and input/output: precedence of operators, expression, and evaluation of an expression, type-conversion (explicit and implicit conversion), accepting data as input from the console and displaying output.

UNIT-II

Flow of Control: introduction, use of indentation, sequential flow, conditional and iterative flow; Conditional statements: if, if-else, if-elif-else, flowcharts, simple programs: e.g.:

absolute value, sort 3 numbers and divisibility of a number. Iterative Statement: for loop,range(), while loop, flowcharts, break and continue statements, nested loops, suggested programs: generating pattern, summation of series, finding the factorial of a positive number, etc.Strings: introduction, string operations (concatenation, repetition, membership and slicing), traversing a string using loops, built-in functions/methods.

UNIT-III

Lists: introduction, indexing, list operations (concatenation, repetition, membership and slicing), traversing a list using loops, built-in functions/methods, nested lists. **Tuples:** introduction, indexing, tuple operations (concatenation, repetition, membership and slicing); built-in functions. Set: creating set, changing/ adding elements to a set, removing elements to a set, python set operations, python set methods. **Dictionary:** introduction, accessing items in a dictionary using keys, mutability of a dictionary (adding a new term, modifying an existing item), traversing a dictionary, built-in functions/methods. Introduction to Python modules: Importing module using 'import <module>' and using from statement, importing math Module**. String Manipulation:** Basic Operations, Slicing

UNIT-IV

Python functions, types of functions, function definition, function call, types of function arguments, pas by value and pass by reference/object reference, recursion, advantages and disadvantages of recursion, scope and lifetime variables. Python Modules, Python Package.

UNIT-V

Python Files: Python File Operation, Python Directory , Python Exception Handling, python built-in exception, Try, Except and Finally, Python user defined exception, Python graph plotting using Matplotlib.

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

1. *Demonstrate a solid understanding of Python syntax, including variables, data types, control structures, functions, classes, and modules.*
2. *Develop problem-solving skills and the ability to design, implement, and debug Python programs to solve a variety of computational problems.*
3. *Demonstrate the ability to read from and write to files, process data from external sources, and handle input/output operations using Python.*

Text Books

1. John V. Guttag, Introduction to Computation and Programming Using Python, 2nd Edition, PHI 2 Core, 2016
2. R. Nageswara Rao, Python Programming, dreamtech Press, 2nd Edition, 2018.

Reference Books

3. Ramsey Hamilton, Python: Programming: A Beginner's Guide, Create space Independent Pub, 2nd Edition, 2016
4. YashavantKanetkar, Let Us Python, BPB Publications, 5th Edition, 2022
5. R.S. Salaria, Programming in Python, Khanna Publishing House, 2nd Edition, 2019.
6. P. Sharma, Programming in Python, BPB, 2nd Edition, 2014
7. T. Budd, Exploring Python, TMH, 1st Edition, 2011

LAB: PART B: Python Programming (Practical): 30 Hours. (Practical/Project/Field work): Total marks: 30 (ESE+CCA)

Pass marks: 12 (ESE+CCA)

This part provides the practical knowledge of Programming with Python.

***Course Objectives*:**

1. *Provide students with hands-on experience in writing Python code.*
2. *Allows practicing programming concepts, syntax, and techniques in a real-world coding environment.*
3. *Help students understand and apply fundamental programming concepts such as variables, data types, control structures (loops and conditionals), functions, and input/output operations in Python.*
4. *Introduce students to various Python libraries and modules that extend the functionality of Python. Students learn how to leverage these libraries to solve complex problems and perform tasks such as data manipulation, visualization, and scientific computing.*

*This paper provides practical knowledge of python programming. List of laboratory programming assignments (not limited to these)***:**

1. Using a loop, print a table of Celsius/Fahrenheit equivalences. Let c be the Celsius temperatures ranging from 0 to 100, for each value of c, print the corresponding Fahrenheit temperature.
2. Using a while loop, produce a table of sins, cosines and tangents. Make a variable x in range from 0 to 10 in steps of 0.2. For each value of x, print the value of sin(x), cos(x) and tan(x).
3. Write a program that reads an integer value and prints “leap year” or “not a leap year”.
4. Write a program that takes a positive integer n and then produces n lines of output shown as follows.
5. For example enter a size: 5
   *
   **
   ***
   ****
   *****
6. Write a function that takes an integer ‘n’ as input and calculates the value of
   1 + 1/1! + 1/2! + 1/3! + … + 1/n
7. Write a function that takes an integer input and calculates the factorial of that number.
8. Write a function that takes a string input and checks if it’s a palindrome or not.
9. Write a list function to convert a string into a list, as in list (‘abc’) gives [a, b, c].
10. Write a program to generate Fibonacci series.
11. Write a program to check whether the input number is even or odd.

12. Write a program to compare three numbers and print the largest one.
13. Write a program to print factors of a given number.
14. Write a method to calculate GCD of two numbers.
15. Write a program to sort a list using insertion sort and bubble sort and selection sort.

16. Problems related to File handling, exception handling in python, usage of user defined exceptions and assertions.
17. Problems related to string manipulations, basic operations , slicing.
18. Write a program to draw a line, bar, histograms, pie chart etc.

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

1. *Students should develop a strong foundation in Python programming language, including syntax, data types, control structures, functions, and file handling.*
2. *Students should be able to analyze problems, design algorithms, and implement efficient solutions using Python.*
3. *Students should be capable of developing small-scale applications using Python.*
4. *Students should develop skills in identifying and resolving errors in Python code. They should be proficient in using debugging tools and techniques to identify and fix issues, ensuring the correctness and reliability of their programs.*