|| Jai Sri Gurudev ||
Sri Adichunchanagiri Shikshana Trust (R)
SJB Institute of Technology
BGS Health and Education City, Dr. Vishnuvardhana Road, Kengeri, Bengaluru-560060
Approved by AICTE, New Delhi.
Autonomous Institute affiliated to Visvesvaraya Technological University, Belagavi
Accredited by NAAC with 'A+'grade, Certified by ISO 9001 - 2015
Recognized by UGC, New Delhi with 2(f) & 12 (B)

BE (CSE/ISE/AI&ML/CSE(DS)

Semester:	I/II	**Course Type:**	IESC		
Course Title: Python Programming					
Course Code:	25PLT15A/25PLT25A			**Credits:**	04
Teaching Hours/Week (L:T:P:S)			3:0:2:1	**Total Hours:**	40+(10-12 lab Slots)
CIE Marks:	50	**SEE Marks:**	50	**Total Marks:**	100
SEE Type:	Theory			**Exam Hours:**	03

Pre-requisites (Self Learning):

I. Course Objectives:

Students are expected to have very basic knowledge of school-level mathematics (simple algebra and probability), logical thinking and problem-solving skills, and familiarity with everyday computer use (files, folders, internet). A minimal awareness of simple programming concepts like variables, loops, and conditions is **helpful but not compulsory**

II. Teaching-Learning Process (General Instructions):

The following are some of the strategies that teachers can employ to facilitate the achievement of various course outcomes:

- Begin with **simple real-life examples** before introducing programming concepts.
- Use **live coding demonstrations** in class and encourage students to practice alongside.
- Employ **visual tools** like (https://pythontutor.com/) to help students trace and understand program execution step by step.
- Blend **chalk & talk**, multimedia resources, and interactive quizzes to reinforce concepts.
- Encourage **pair programming and group problem-solving** so students learn collaboratively.
- Provide **graded exercises**:
  - ✓ Easy tasks for building confidence (slow learners).
  - ✓ Extension problems and mini-projects for advanced learners.
- Conduct **weekly recap sessions** with short coding challenges and doubt clearing.
- Relate concepts to **real-world applications** (e.g., games, text processing, file handling) to keep learning engaging.

COURSE CONTENT

Theory

Module-1: Getting Started with Python	8 Hours

Introduction to Python and Interactive Shell
Entering Expressions into the Interactive Shell, The Integer, Floating-Point, and String Data Types, String Concatenation and Replication, Storing Values in Variables, Your First Program, Dissecting Your Program - The print() Function, The input() Function, The len() Function, The str(), int(), and float() Functions
Statements and Operators:
Statements, Evaluating expressions, Operators and operands, Type converter functions, Order of operations, Operations on strings, Input, Composition, The modulus operator.
Iteration: Assignment, Updating variables, the for loop, the while statement, The Collatz 3n + 1 sequence, tables, two-dimensional tables, break statement, continue statement, paired data, Nested Loops for Nested Data.
Functions: Functions with arguments and return values.

Textbook 1: 2.4 to 2.12, 4.4, 4.5
Textbook 2: Chapter 1

RBT Levels: L1, L2, L3

Module-2: Strings and Tuples	8 Hours
Strings: Working with strings as single things, working with the parts of a string, Length, Traversal and the for loop, Slices, String comparison, Strings are immutable, the in and not in operators, A find function, Looping and counting, Optional parameters, The built-in find method, The split method, Cleaning up your strings, The string format method. **Tuples**: Tuples are used for grouping data, Tuple assignment, Tuples as return values, Composability of Data Structures Textbook 1: 5.1, 5.2	
RBT Levels: L1, L2, L3	
Module-3: Lists and Dictionaries	8 Hours
Lists: List values, accessing elements, List length, List membership, List operations, List slices, Lists are mutable, List deletion, Objects and references, Aliasing, cloning lists, Lists and for loops, List parameters, List methods, Pure functions and modifiers, Functions that produce lists, Strings and lists, list and range, Nested lists, Matrices. **Dictionaries**: The Dictionary Data Type - Dictionaries vs. Lists, The keys(), values(), and items() Methods, Checking Whether a Key or Value Exists in a Dictionary, The get() Method, The setdefault() Method Textbook 1: 5.3 Textbook 2: Specified topics from chapter 5	
RBT Levels: L1, L2, L3	
Module-4: Files and Modules	8 Hours
Files: About files, file operations-read, write and display, turning a file into a list of lines, Reading the whole file at once, working with binary files, Directories, fetching something from the Web. **Modules**: Random numbers, the time module, the math module, creating modules, Namespaces, Scope and lookup rules, Attributes and the dot Operator, Three import statement variants. Textbook 1: 7.1 - 7.8, 8.1 to 8.8	
RBT Levels: L1, L2, L3	
Module-5: Exception Handling and Basics of Object-Oriented Programming	8 Hours
Debugging: Raising Exceptions, Getting the Traceback as a String, Assertions **Object Oriented Programming:** Classes and Objects — The Basics, Attributes, Adding methods to class, Instances as arguments and parameters, Converting an instance to a string, Instances as return values. Mutable Objects, Sameness, Copying. Textbook 2: 10 Textbook 1: 11.1, 11.2.2 - 11.2.4	
RBT Levels: L1, L2, L3	
III(b). PRACTICAL PART	
Programs	
1. Write a Python program to implement a simple calculator (addition, subtraction, multiplication, and division) 2. Write a Python program to generate and display a formatted multiplication table for numbers from 1 to N with suitable row and column headers. 3. Write a Python program to check whether a given string is a palindrome. 4. Write a Python program to process a line of text and perform the following operations a. Convert it to lowercase b. Split the input to words c. Display total number of unique words 5. Write a Python function that accepts a list of numbers and returns a tuple containing their sum and average. Call the function and display the results.	

6. Write a Python program to accept N numbers into a list and perform the following:
   a. Find the maximum and minimum elements
   b. Sort the list in ascending order
   c. Count the frequency of a given element
7. Write a Python program to demonstrate the use of keys(), values() and items() on employee details.
8. Write a Python program to create a text file named student.txt, write student names into it, and then read and display its contents line by line.
9. Write a Python program to accept a number and display its square root, factorial, and sine value using the math module.
10. Write a Python function that accepts a number and raises a Value Error if the number is negative. Demonstrate exception handling in the main program.
11. Write a Python program that defines a class **Student** with the following:
   a. Attributes: name, roll_no, and marks.
   b. A method **get_result()** that returns "Pass" if marks ≥ 40, otherwise "Fail".

COURSE OUTCOMES: At the end of this course, students will be able to

CO1	Apply fundamental Python programming concepts to solve simple real world problems.
CO2	Apply control structures such as loops, conditional statements, and functions to develop modular Python programs.
CO3	Demonstrate string manipulation techniques and the use of tuples for grouping and returning data.
CO4	Implement programs using lists, dictionaries, file handling, and standard Python modules for data processing.
CO5	Develop Python applications incorporating exception handling and basic object-oriented programming concepts.

III. CO-PO-PSO MAPPING (mark H=3; M=2; L=1)

PO/PSO	1	2	3	4	5	6	7	8	9	10	11	12	S1	S2	S3
CO1	2	1			1								1		
CO2	2	1			1								1		
CO3	2	1	1		1								1		
CO4	2	2	1		2								1		
CO5	2	2	1		2								1	`	

IV. Assessment Details (CIE & SEE)

General Rules: Refer CIE and SEE guidelines based on course type for autonomous scheme 2023 Dated on 10-02-2025.

Continuous Internal Evaluation (CIE): Refer Annexure section 2

Semester End Examination (SEE): Refer Annexure section 2

V. Learning Resources

VII(a): Textbooks:

Sl. No.	Title of the Book	Name of the author	Edition and Year	Name of the publisher
1	***How to Think Like a Computer Scientist: Learning with Python 3 Documentation***	***Peter Wentworth, Jeffrey Elkner, Allen B. Downey & Chris Meyers***	***3rd Edition, 2020***	***Available under GNU Free Document License https://app.readthedocs.org/projects/howtothink/downloads/pdf/latest/***
2	Automate the Boring Stuff with Python	Al Sweigart,	No Starch Press	2015 (Available under CC-BY-NC-SA license at https://automatetheboringstuff.com/)

VII(b): Reference Books:

1	Python Programming for Beginners: Learn Python in One Day and Master It	Jason Cannon	Independently Published	2nd Edition, 2021
2	Python Programming: An Introduction to Computer Science	John Zelle	Franklin, Beedle & Associates	4th Edition, 2020

VII(c): Web links and Video Lectures (e-Resources):

1. https://www.learnbyexample.org/python/
2. https://www.learnpython.org/
3. https://pythontutor.com/visualize.html#mode=edit
4. https://automatetheboringstuff.com/
5. http://greenteapress.com/thinkpython2/thinkpython2.pdf
6. https://www.learnpython.org/
7. https://www.youtube.com/playlist?list=PLhQjrBD2T382hIW-IsOVuXP1uMzEvmcE5
8. https://www.youtube.com/watch?v=rfscVS0vtbw

VIII: Activity Based Learning / Practical Based Learning/Experiential learning:

✓ **Pair Programming:** Students work in pairs to solve coding exercises → helps slow learners learn from peers.

✓ **Mini- Projects:**

- Beginner track: Word counter, calculator, palindrome checker.
- Advanced track: Quiz generator, automated file backup, library system using OOP.

✓ **Debugging Challenges:** Provide buggy code snippets for students to fix.

✓ **Peer Presentations:** Small 5-minute demos of their mini-projects in class.