Government of Karnataka

DEPARTMENT OFTECHNICAL EDUCATION

Curriculum Structure

II Semester Scheme of Studies - Diploma in Computer Science and Engineering

Sl. No.	Teaching Department	Course Code	Course Name	Hours per week			Total Contact Hours/week	Credits	CIE Marks		Theory SEE Marks		Practice SEE Marks		Total Marks
				L	T	P			Max	Min	Max	Min	Max	Min	
Integrated Courses															
1	SC	25SC21I	Engineering Mathematics-II	4	0	4	8	6	50	20	50	20	-	-	100
2	ENG	25EG01I	Essential English Communication	4	0	4	8	6	50	20	-	-	50	20	100
3	ME	25ME02I	Computer Aided Engineering Graphics	3	0	4	7	5	50	20	-	-	50	20	100
4	CS	25CS21I	Thinking Programming with Python	4	0	4	8	6	50	20	50	20	-	-	100
Audit Course															
5	CS	25CS22T	Indian Constitution	2	0	0	2	2	50	20	-	-	-	-	50
6	Personality Development		NCC/NSS/YOGA/SPORTS...	Students are expected to engage in any one of these activities from 1st semester to 6th semester (No Credits)											
Total				**17**	**0**	**16**	**33**	**25**	**250**	**-**	**100**	**-**	**100**	**-**	**450**

SEMESTER 2

Government of Karnataka
DEPARTMENT OF TECHNICAL EDUCATION

Program	**Computer Science and Engineering**	**Semester**	2
Course Name	**Thinking Programming with Python**	**Type of Course**	Integrated
Course Code	**25CS21I**	**Contact Hours**	8 Hours per week
Teaching Scheme	4:0:4	**Credits**	6
CIE Marks	**50**	**SEE Marks**	**50 (Theory)**

1. Rationale:

The course aims to combine foundational problem-solving skills with practical programming experience in Python. This integrated approach enables learners to develop a logical mindset while gaining hands-on coding experience in one of the world's most versatile and beginner-friendly programming languages. The course goes beyond teaching programming syntax. It's an essential step toward developing logical, computational, and technical proficiency for success in the digital age.

2. Course Outcomes: At the end of the Course, the student will be able to:

CO-01	Apply computational thinking to solve the given problem and illustrate the solution as an algorithm
CO-02	Develop programs using fundamental programming concepts.
CO-03	Design, code, and debug a Python program to solve a problem
CO-04	Identify and resolve both syntactical and semantic errors.
CO- 05	Apply modular programming approach to optimize the program.

3. Course Content

Week	CO	PO	Lecture(4HRS) (Knowledge Criteria)	Practice(4HRS) (Performance Criteria)
1	**1**	**1,2**	**Computational Thinking** Introduction, Components, Importance and Applications of CT in various fields. Decomposition: Consider a problem statement to learn decomposition (refer table 1 for examples) – Pattern recognition: Analogy; Classification; Sequencing; Ranking; Series	Organize Games and Activities to instill Computational Thinking
2	**1**	**1,2**	**Algorithmic thinking** Problem Types for Algorithmic Solutions [definition and examples] ▪ Searching Problems ▪ Sorting Problems ▪ Optimization Problems	For a given problem ▪ Identify the key components (input, output and logic) of an algorithm ▪ Develop a step-by-step algorithm to solve it.

			▪ Graph Problems ▪ String Processing Problems ▪ Numerical Problems ▪ Combinatorial Problems ▪ Cryptographic Problems **Algorithm** Representation: Natural Language, Pseudocode, Flowchart *[Note: Sorting Algorithms: designed to arrange data in a specific order* *Examples: Bubble Sort, Merge Sort, Quick Sort]*	
3	**2**	**1**	**Introduction to Programming** What is programming? ; Why we need Programming; How to Think like a programmer; Programming Paradigms; Programming Languages and their Paradigms	Organize Games and Activities to instill Programmer Thinking
4	**2**	**1,4**	**Basic Programming Concepts:** Syntax ; Tokens and types; Variable -Rules for creating variables; constants; datatypes; errors ; comments ; Best programming practices;	Setup and get familiar with Your Development Environment such as VsCode
5	**2**	**1,4**	**Introduction to python programming:** Features and applications ; Data types (primitive); Assignment statement; Type conversion;	Practice Programs to understand datatype, and their conversion.
6	**3,4**	**1,2, 4**	**Input and output statements:** input (); print (); Formatting output - string concatenation, format() and f-strings; Debugging techniques.	Practice Programs to understand reading input and formatting output
7	**3,4**	**1,2, 4**	Operators and their Precedence Expressions	Practice Programs to understand Operators and Expressions
8	**3,4**	**1,2, 3,4**	**Flow control** Conditional statements: if, if-else, if-elif, match case	Practice programs to understand the concept of conditional statements
9	**3,4**	**1,2, 3,4**	**Iterative statement – for loop:** structure of the for loop using range();concept of break and continue with the for loop;	Practice programs to understand the concept of for loop
10	**3,4**	**1,2, 3,4,**	**Iterative statement - While loop :** structure; concept of **Break, continue, pass** statements with while loop;	Practice programs to understand the concept of while loop

11	**3,4**	**1,2, 3,4**	**Nested loops :** Use cases of nested loops; control flow in nested loops using **break, continue, and else;**	Practice programs to understand the concept of nested loops
12	**3,4, 5**	**1,2, 3,4, 5,7**	**Functions :** Need for functions ; create function ; function call: with and without arguments; return statements; Variable scope	Practice programs to understand the concept of functions
13	**3,4, 5**	**1,2, 3,4, 5,7**	**Modules and Packages :** Significance of modules and packages; Import and use built-in modules; Create new module; Create a package with multiple modules; import statement (import, import…as .., from….import, import *)	Practice programs to understand the concept of modules and packages

4. References:

Sl No	Description
1	*Think Like a Programmer* (V. Anton Spraul)
2	*Automate the Boring Stuff with Python* – Al Sweigart
3	*Python Crash Course* – Eric Matthes
4	Introduction to Computation and Programming Using Python – John V. Guttag
5	How to Think Like a Computer Scientist: Learning with Python – Allen B. Downey
6	Programming for the Puzzled – Srini Devadas
7	Introduction to Algorithmic Thinking – Daniel Zingaro
8	CS50 Python Course - Harvard's CS50P: Introduction to Python
9	Google Python Course -
10	Codecademy Python Course
11	Coursera: Python for Everybody – University of Michigan

5. Suggestive Online courses

Sl no	Topic Name	Reference Courses	Self Assessment Link	Source
1	**Computational Thinking**	TOC - Problem Solving Using Computational Thinking | Infosys Springboard		Coursera
2	**Algorithmic thinking**	TOC - Programming Fundamentals using Python - Science Graduates - Foundation Program | Infosys Springboard		Infosys Wingspan
3	**Introduction to Programming**	TOC - Programming Fundamentals using Python - Science Graduates - Foundation Program | Infosys Springboard		Infosys Wingspan
4	**Basic Programming Concepts:**	TOC - Programming Fundamentals using Python - Science Graduates - Foundation Program | Infosys Springboard		Infosys Wingspan
5	**Introduction to python programming**:	TOC - Basics of Python | Infosys Springboard	Basics of Python - Self Assessment - Viewer Page | Infosys Springboard	Infosys Wingspan

6	**Input and output statements**:	TOC - Programming Fundamentals using Python - Science Graduates - Foundation Program | Infosys Springboard	Assessment - Programming Fundamentals using Python - Science Graduates - Viewer Page | Infosys Springboard	Infosys Wingspan
7	**Operators and their Precedence Expressions**			
8	**Flow control Practice programs to understand the Conditional statements**			
9	**Iterative statement**			
10	**Nested loops**			
11	**Functions**			
12	**Modules and Packages**			

6. Suggestive Program List

week	Suggested program/ activity list
1	Case based learning: ▪ Improving the Transport System in the Countryside ▪ Analyzing the Reach of Government Schemes ▪ Smart Irrigation System for Optimal Water Usage ▪ Impact of Social Media on Students
2	Problem / use case based learning Devise an Algorithm and draw flowchart for problems such as ▪ Swapping two values ▪ Finding largest / smallest among two/three number ▪ Computing area/ perimeter of given shape (circle, triangle, rectangle, square) ▪ Metric conversion (meter – KM, pound – kilo gram, Celsius – Fahrenheit) ▪ Determine given number is even or odd, positive or negative.
3	**Think Like a Programmer - Google Books** Think Like a Programmer: Introduction (youtube.com) General problem solving techniques- refer the book or youtube videos **https://www.codecademy.com/resources/blog/how-to-think-like-a-programmer/**
4	**Prepare Your Development Environment:** **1.** Download and install the necessary compiler or interpreter ▪ Python: Download from python.org. ▪ Java: Install the Java Development Kit (JDK) from Oracle or OpenJDK. 2. Verify the installation by checking the version using the terminal/command prompt 3. Install an IDE (for python VSCode or Pycharm)

	▪ Open your IDE and explore its features: ▪ Create a new project. ▪ Write a simple "Hello, World!" program. ▪ Learn to run and debug your code. ▪ Explore useful features like syntax highlighting, auto-completion, and integrated terminal.
5	Create variables, assign values, and display their data types: a. Create variables of different data types and assign values b. Display the values and their respective data types using the type() function. Assignment Statements: a. Single variable assignment: Assign a value to a single variable. b. Multiple variable assignment: Assign values to multiple variables in one statement. c. Assign the same value to multiple variables. Type Conversion a. Demonstrate Python's ability to convert types automatically during operations. b. Use functions like int(), float(), str(), or bool() to explicitly convert data types.
6	1. Write a program that: a. Accepts user input for name and age. b. Prints a formatted message using different string formatting methods. 2. Read message from the user and Format it with different methods and display. 3. Demonstrate swapping the values(numerical) of two variables using a temporary variable.
7	1. Write python equivalent expressions for math expressions such as a. f = ax +b b. $f = a^2 + b^2 + 2ab$. c. $f = a^3 + b^3 + 3ab(a + b)$ d. $area = \pi r^2$ e. $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. f. $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ 2. Translate textual problem statements into Python expressions 3. Write a program that: a. Accepts two numbers from the user. b. Converts them to integers (if they are not already). c. Performs arithmetic operations (add, subtract, multiply, divide) and displays the results with the data types and appropriate message. 4. Calculate the total cost of an item after applying a 15% discount to its original price of Rs150. 5. Calculate the compound interest on a principal of Rs10000 at an annual interest rate of 5% for 3 years, compounded annually. 6. Compute area of triangle, rectangle. 7. Swapping values of two variables without using a temporary variable

8	1. Create a program to manage airline ticket bookings based on the passenger's selected class: *"economy"*, *"business"*, or *"first"*. a. Prompt the user to input details such as name, age, destination, and class preference (*economy, business, or first*). b. Validate inputs (e.g., ensure the name contains only letters, age is a positive integer, and class selection is valid). c. Display appropriate error messages for invalid entries and allow the user to re-enter the data. 2. Build a system to suggest clothing based on the temperature. a. If temperature ≥30°C, suggest "Wear light clothes." b. If 20°C ≤ temperature < 30°C, suggest "Wear moderate clothing." c. If temperature < 20°C, suggest "Wear warm clothes." 3. Implement a discount system for an e-commerce website. a. If the total purchase is ≥Rs5000, apply a 20% discount. b. If Rs2000 ≤ total < Rs5000, apply a 10% discount. c. If total < Rs2000, no discount is applied. 4. Create a program that simulates an ATM machine to check: a. If the entered PIN is correct, allow the user to proceed. b. Validate that the withdrawal amount doesn't exceed the account balance. c. Ensure that the withdrawal amount is in multiples of a specific denomination (e.g., Rs100).
9	1. Generate numbers from 1 to 10 using range(). 2. Generate numbers for a given range. 3. Write a program to display square numbers from 1 to 10. 4. Write a Python program that accepts an integer input from the user and determines whether the number is a prime number or not. If the number is prime, display an appropriate message; otherwise, indicate that it is not prime. 5. Write a Python program that accepts a message and a number from the user. The program should then print the specified message the given number of times. 6. Write a Python program to calculate the factorial of a given number. The program should: a. Accept a non-negative integer as input. b. Use an iterative approach to compute the factorial. c. Handle invalid inputs (e.g., negative numbers or non-numeric inputs) by displaying appropriate error messages. 7. Evaluation of mathematical series like **$S=1+2+3+\ldots\ldots+n$** **$S=1^2+2^2+3^2+\ldots\ldots+n^2$** **$H_n=1+1/2+1/3+\ldots\ldots+1/n$**
10	1. Implement the Euclidean algorithm to find the GCD of two integers using a while loop. 2. Collect daily weather data (e.g., temperature, humidity) from the user. Allow data entry to continue until the user decides to stop or inputs a sentinel value. 3. Simulate a loan repayment system where a borrower pays a fixed amount each month. Continue reducing the loan balance until it is fully paid off, and display the balance after each payment.

	4. Write a program that simulates a countdown timer. Start with a given number of seconds and decrement until it reaches zero, displaying the countdown at each step. 5. Write a program that takes an integer and calculates the sum of its digits using a while loop. 6. Write a program that simulates a simple user authentication system. The program should: a. Repeatedly ask the user to input their username and password. b. Verify the entered credentials against a pre-defined username and password. c. Allow the user up to three attempts to enter the correct credentials. d. Provide appropriate feedback for each attempt (e.g., "Incorrect username or password"). e. If the correct credentials are provided within three attempts, display a success message and terminate the program. f. If all three attempts are exhausted, display a failure message and terminate the program.
11	1. Write a program to identify all prime numbers within a user-defined range. The program should: a. List all the prime numbers within the range. b. Calculate and display the total count of prime numbers found. c. Compute and display the sum of these prime numbers. 2. Write a Python program to perform the following tasks for a user-defined range of integers: a. Identify and list all non-prime numbers within the given range. b. Categorize the non-prime numbers into even and odd. c. Count the total number of even and odd non-prime numbers. d. Calculate and display the sum of even and odd non-prime numbers separately. 3. Write a Python program to perform the following tasks for a user-defined range of integers: a) Identify and list all palindrome numbers within the specified range. b) Count the total number of palindrome numbers found. c) Display the results in a user-friendly format. 4. Write a program to generate and display a multiplication table for numbers 1 to 10. 5. Generate star or number patterns like a pyramid or diamond shape.
12	1. Write simple functions for arithmetic calculations (e.g., addition, factorial). 2. Define a function assign_priority() for ticketing system which assigns priorities to support tickets based on the issue type (low, medium, high and returns a priority level. 3. Define function calculate_interest() for banking system, to calculate the interest on savings accounts based on the principal amount, rate, and time period. 4. An online store offers a discount based on the total purchase amount. If the customer's total purchase exceeds a certain threshold, they get a percentage discount. Define a function apply_discount() that accepts the total amount and returns the final price after discount.

13	1. Create a module that includes a function to calculate the area of a circle and use it in another script. 2. Create a package for a small project, such as a calculator application with modules for arithmetic, trigonometric, and logarithmic operations.

7. CIE Assessment Methodologies

Sl.No	CIE Assessment	Test Week	Duration (minutes)	Max marks	
1.	CIE-1TheoryTest	4	90	50	Average of all CIE=50 Marks
2.	CIE-2Practice Test	7	180	50	
3	CIE-3TheoryTest	10	90	50	
4.	CIE-4Practice Test	13	180	50	
5	CIE-5 ▪ Portfolio evaluation (20) ▪ Online Course/s of minimum 10 Hrs. in Infosys Spring Board/ Swayam/NPTEL/AWS /any other (30)	1-13		50	
				Total	**50 arks**

Note:

Portfolio evaluation

Each laboratory exercise will be evaluated for a total of 20 marks. The evaluation will include the following components:

- Written description of the experiment in the observation book.
- The results obtained from the experiment.
- Corrections and evaluations of the experiment completed in the previous class, documented in the record book.

The average of all exercises shall be considered for the final assessment at the end of course.

Rubrics for the Mini Project (if included) should be defined by the course coordinator.

8. SEE - Theory Assessment Methodologies

Sl. No	SEE – Theory Assessment	Duration	Exam Paper Max marks	Exam Paper Max Marks scale down to (Conversion)	Min marks to pass
1.	Semester End Examination-Theory	3 Hours	100	50	20

9. CIE Theory Test model question paper

Program	**Computer Science and Engineering**			**Semester – 2**	
Course Name	**Thinking Programming with Python**			**Test**	**I/III**
Course Code	25CS21I	**Duration**	**90 min**	**Marks**	**50**
Name of the Course Coordinator:					

Note: Answer any one full question from each section. Each full question carries equal marks.

Q.No	**Questions**	**Cognitive Level**	**Course Outcome**	**Marks**
	Section – 1			
1	a. You are tasked with organizing a class party. The goal is to ensure everything is well-prepared, including food, decorations, activities, and invitations. Use decomposition to break down this problem into smaller, manageable tasks. – (10 M) b. Identify the pattern and provide an explanation to complete the series. (10M) 1. 3, 6, 11, 18, 27,____,____,_____,_____ 2. 1, 1, 2, 3, 5, 8, ___,____,_____,_____ 3. 2, 4, 12, 48, ___,____,_____,____ 4. A2, B4, C8, D16, ___,____,_____,____ c. Explain the four pillars of computational thinking. – (5M)	L2,L3	1	25
2	**a.** You are tasked with organizing a birthday party for your best friend. The goal is to ensure the party runs smoothly. To accomplish this, break the task into smaller, manageable steps and write down the plan. – (10M) b. A magical machine only accepts even numbers and rejects odd ones. If you input an even number, the machine lights up green; if you input an odd number, it lights up red. Write an algorithm and draw flowchart to help a wizard decide whether a number can be accepted by the machine or not.- (7M) **c.** Identify different types of problems and provide an example algorithm to solve each. - (8M)	L2,L3	1	
	Section – 2			
3	a. Explain the concept of a variable. Why are variables essential in programming? b. Define tokens. List and explain the different types of tokens with examples. c. List the rules for naming variables in Python. Provide examples of valid and invalid variable names. d. Why are constants important in programming? Illustrate with an example where a constant would be beneficial.	L2	2	25

	e. What are syntax errors, logical errors, and runtime errors?			
4	a. How can learning programming enhance one's logical thinking, creativity, and problem-solving abilities? b. Explain the concept of "thinking like a programmer" and its importance in solving real-world problems. c. Explain how identifying patterns in problems can lead to efficient solutions. Use an example to demonstrate this process. d. Trace the evolution of programming paradigms through the generations of programming languages. e. A team is building a web application with a real-time chat feature. Discuss which programming paradigms and languages might be best suited for this task and why.	L2	2	
Note for the Course coordinator: Each question may have one, two or three subdivisions. Optional questions in each section carry the same weightage of marks, cognitive level and course outcomes.				

Signature of the Course Coordinator **Signature of the HOD** **Signature of the IQAC Chairman**

10. CIE Practice Test model question paper

Program	**Computer Science and Engineering**			**Semester**	**2**
Course Name	**Thinking Programming with Python**			**Test**	**II/IV**
Course Code	**25CS21I**	**Duration**	**180 min**	**Marks**	**50**
Name of the Course Coordinator:					
Questions				**CO**	**Marks**
Write a program that performs the following tasks: 1. Generate Random Numbers: Generate a random number within a given range [a, b], where **a** and **b** are user inputs. 2. Determine non-prime Number: Determine that generated random number is not prime. 3. Classify non-prime Number: Further classify numbers into: Even / Odd Instructions: ▪ Clearly identify and describe the key concepts needed to develop the program. ▪ Structure the program into separate, reusable modules or functions.				**1,2,3,4**	**50**
Scheme of assessment a) Program Design and Conceptual Clarity (Clear identification of the key concepts, Explanation of the logic and methodology used and organization of the program.) - 10 b) Implementation and Execution - 30 c) Best Practices (Code Readability and Error Handling)- 10					
				Total Marks	**50**

Signature of the Course Coordinator **Signature of the HOD**

11. Equipment/software list with Specification for a batch of 30 students

Sl.No.	Particulars	Specification	Quantity
01	Desktop/Laptop PC with Windows/Linux	Intel i3, 500GB Hard Disk/SSD, 8GB RAM, Monitor, Mouse, Keyboard or higher configuration	30
02	Internet Connection	100 Mbps speed or higher subscription	1
03	LAN connectivity/ High speed Wireless AP	32 Port Switch with LAN cabling/ Wifi Adapters (32 No.)	1
04	Online UPS	5KV with 3 -6 hours backup	1
05	Projector	Multimedia Projector	1
06	White Board	Plane white board / Smart Board/Smart TV	1
07	Audio Speakers	Multimedia, Two-way hybrid speaker system	2