

Trade 4	Sheet metal working and Brazing: 6. Use of sheet metal, working hand tools, cutting , bending , spot welding	06
Trade-5	Forging (Smithy): 7. At least one forging job to be demonstrated and a simple job to be made for Term Work in a group of 4 students.	06

Text Books:

1. Workshop Technology, Volume-I, P.N.Rao, McGrawHill Publication
2. Elements of Workshop Technology, Vol-I, S.K. Hajra Choudhury, A K Hajra Choudhury, Nirjar Roy, Media Promoters & Publishers Pvt Ltd

References:

1. Workshop Technology, Part-II, W A J Chapman, VIVA Books Pvt Ltd
2. A Course in Workshop Technology, B.S. Raghuvanshi, Dhanpat Rai and Co Ltd.

Assessment:

Term Work: Term Work shall consist of at least 3 practicals' based on the above list

Term Work Marks: 25 Marks (Total marks) = 20 Marks (Experiment) + 5 Marks (Attendance)

Course Code	Course Name	Teaching Scheme (Contact Hours)			Credits Assigned			
		Theory	Pract.	Tut.	Theory	Pract.	Tut.	Total
VSEC202	Python Programming	-	2*+2	-	-	2	-	2

Course Code	Course Name	Examination Scheme						
		Theory Marks				Term Work	Practical/ Oral	Total
		Internal assessment (IAT)			End Sem. Exam			
		IAT-I	IAT-II	IAT-I + IAT-II (Total)				
VSEC202	Python Programming	--	--	--	--	25	25	50

Lab Objectives:

1. To familiarize learners with Python's basic syntax, variables, data types, operators, and input/output functions.
2. To reinforce the understanding and application of conditional statements, loops, and functions in Python programming.
3. To instill learners on file handling, exception management, and Python packaging.
4. To Introduce object-oriented programming principles and their application in Python.
5. To explore advanced topics such as regular expressions, pattern matching, and GUI development.
6. To introduce and demonstrate the use of popular Python libraries for data handling.

Lab Outcomes: Learner will be able to

1. Demonstrate the proficiency in basic python programming or Create and perform various operations on data structures like list, tuple dictionaries and strings.
2. Apply Control Flow and Functions for efficient coding to solve problems.
3. Demonstrate proficiency in handling file operations, managing exceptions, and developing Python packages and executable files for modular programming.
4. Illustrate the concept of Object-Oriented Programming used in python.
5. Design Graphical User Interface (GUI) applications, utilizing appropriate Python libraries to create user-friendly interfaces.
6. Investigate and apply popular python libraries to conduct efficient data handling tasks.

Prerequisite: VSEC 102 C Programming

DETAILED SYLLABUS:

Sr. No.	Module	Detailed Content	Hrs	LO Mapping
0	Prerequisite	Introduction to Programming: Understanding basic concepts like algorithms, flowcharts, and pseudocode. Problem-Solving Skills: Ability to approach problems methodically and apply logical thinking to develop solutions.	1	--
1	Introduction to Python	1. Basic Syntax and Data Types - Variables and data types, Operators, Input and output, 2. Data Structures- list, tuple, set and dictionary 3. Understanding the Syntax Transition: From C to Python	4	L1
2	Control Flow and Functions	2.1 Conditional Statements: if, else, elif 2.2 Loops: for and while loop 2.3 Functions- Defining functions, Parameters and return values, Scope and lifetime of variables	4	L2
3	File Handling, Packaging, and Debugging	3.1 File Handling- Reading and writing files, Exception handling	4	L3

		3.2 Creating Python Packages, Modules and executable files 3.3 Dealing with Syntax Errors, Runtime Errors and Scientific Debugging		
4	Object-Oriented Programming (OOP) in Python	4.1 Introduction to OOP: Classes and objects, Encapsulation, inheritance, and polymorphism 4.2 Creating Classes and Objects: Class attributes and methods Constructor and destructor. 4.3 Type of Inheritance: Single, multiple and multilevel inheritance	4	L4
5	Advanced Python Concepts	5.1 Regular Expressions, Pattern matching, Regex functions in Python 5.2 GUI Development using any Python GUI framework	5	L5
6	Python Libraries	6.1 Introduction to Popular Libraries 6.2 NumPy for numerical computing, 6.3 Pandas for data manipulation 6.4 Matplotlib for data visualization	4	L6

Text Books:

1. Core Python Programming, Dr. R. Nageswara Rao, Second Edition, Dreamtech Press.
2. Beginning Python: Using Python 2.6 and Python 3.1. James Payne, Wrox Publication.
3. Python Programming, Anurag Gupta and G. P. Biswas, First Edition, McGraw-Hill Education.

References:

1. Learn Python the Hard Way, Zed Shaw, Third Edition, Addison-Wesley.
2. Python Projects, Laura Cassell, Alan Gauld, First Edition, Wrox Publication.
3. Introduction to computing and problem-solving using python, Balagurusamy, First Edition, McGraw Hill Education.

Online Resources:

Sr. No.	Website Name
1.	Python Tutorial: http://docs.python.org/release/3.0.1/tutorial/
2.	Python for everybody specialization: https://www.coursera.org/specializations/python .

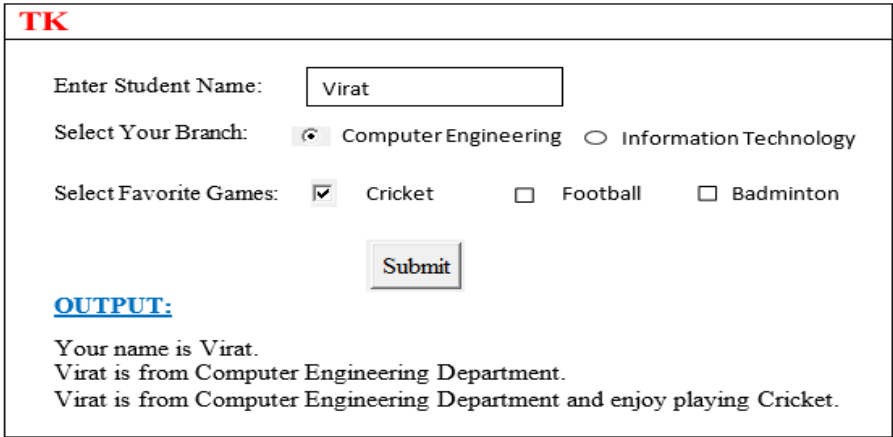
List of Experiments.

The following experiments serve as samples to illustrate the application of concepts covered in each unit. Instructors are encouraged to modify and adapt these experiments to meet the specific needs of the course and the learning objectives. It is essential to ensure that the fundamental concepts and skills outlined in each unit are adequately covered, even with modifications.

Week No	List of Experiments	Hrs
---------	---------------------	-----

01	<p>Objective: To enable learners to transition their understanding of basic programming constructs from C to Python by focusing on Python's syntax, variables, data types, operators, and input/output functions, and comparing these elements with their equivalents in C</p> <ol style="list-style-type: none"> 1. Personalized Greeting Generator* - Write a python code to generate Personalized Greeting. 2. Calculating Areas of Geometric Figures* - Write a python program to calculate areas of any geometric figures like circle, rectangle and triangle. 3. Developing Conversion Utilities: Develop any converter such as Rupees to dollar, temperature convertor, inch to feet etc. 4. Calculating Gross Salary of an Employee*: Write a Python program to calculate the gross salary of an employee. The program should prompt the user for the basic salary (BS) and then compute the dearness allowance (DA) as 70% of BS, the travel allowance (TA) as 30% of BS, and the house rent allowance (HRA) as 10% of BS. Finally, it should calculate the gross salary as the sum of BS, DA, TA, and HRA and display the result. 5. Calculating Simple Interest: Write a Python program to calculate the simple interest based on user input. The program should prompt the user to enter the principal amount, the rate of interest, and the time period in years. It should then compute the simple interest using the formula Simple Interest=(Principal×Rate×Time) /100 and display the result. 6. Exploring Basic Arithmetic Operations in Python*: Write a Python program to explore basic arithmetic operations. The program should prompt the user to enter two numbers and then perform addition, subtraction, multiplication, division, and modulus operations on those numbers. The results of each operation should be displayed to the user. 	02
02	<p>Objective: Mastering Python New Data Structures for Practical Applications</p> <p>Task List Manager*: Develop a Python program to manage a task list using lists and tuples, including adding, removing, updating, and sorting tasks.</p> <p>Student Enrollment Manager *: Create a Python code to demonstrate the use of sets and perform set operations (union, intersection, difference) to manage student enrollments in multiple courses / appearing for multiple entrance exams like CET, JEE, NEET etc.</p> <p>Student Record Keeper *: Write a Python program to create, update, and manipulate a dictionary of student records, including their grades and attendance.</p>	02
03	<p>Objective: To enable students to transition their understanding of control statements and loops from C to Python, emphasizing the adoption of Python syntax while reinforcing logical structures already learned.</p> <ol style="list-style-type: none"> 1. Triangle Pattern Generator Using Loops: Write a Python program to print a triangle pattern (give any), emphasizing the transition from C to Python syntax. 2. Number Type Identifier*: Develop a Python program that takes a numerical input and identifies whether it is even or odd, utilizing conditional statements and loops. 3. Character Type Identifier: Create a Python program to check whether the given input is a digit, lowercase character, uppercase character, or a special character using an 'if-else-if' ladder. 4. Multiplication Table Generator: Write a Python program to take a numerical input from the user and generate its multiplication table using loops. 5. Fibonacci Sequence Generator: Develop a Python program to print the Fibonacci sequence using a while loop. 6. Factorial Generator*: Design a Python program to compute the factorial of a given integer N. 	02

	<p>7. Prime Number Analyzer*: Using function, write a Python program to analyze the input number is prime or not.</p> <p>8. Simple Calculator Using Functions*: Implement a simple Python calculator that takes user input and performs basic arithmetic operations (addition, subtraction, multiplication, division) using functions.</p> <p>9. Interactive Guessing Game: Develop a number guessing game where the program generates a random number, and the user has to guess it. Implement loops and conditional statements for user interaction.</p>	
04	<p>Objective: To enable learners to proficiently handle file operations, manage exceptions, and create Python packages and executable files.</p> <ol style="list-style-type: none"> 1. Extracting Words from Text File *: Develop a Python program that reads a text file and prints words of specified lengths (e.g., three, four, five, etc.) found within the file. 2. Finding Closest Points in 3D Coordinates from CSV: Write a python code to take a csv file as input with coordinates of points in three dimensions. Find out the two closest points. 3. Sorting City Names from File: Write a python code to take a file which contains city names on each line. Alphabetically sort the city names and write it in another file. 4. Building an Executable File*: Create a executable file for any program developed in earlier practical. 	02
05	<p>Objective: To enable learners to proficiently handle errors and exceptions in Python programs, ensuring robust and fault-tolerant code. Learners will also develop debugging skills to identify, diagnose, and fix issues efficiently using scientific debugging methods.</p> <ol style="list-style-type: none"> 1. Basic Exception Handling*: Write a Python program that takes two numbers as input and performs division. Implement exception handling to manage division by zero and invalid input errors gracefully. 2. Custom Exceptions: Develop a Python program that simulates a banking system with a function to withdraw money. Raise custom exceptions for scenarios such as insufficient funds and invalid account numbers 3. Logging for Debugging: Enhance a Python program by adding logging statements to record the flow of execution and error messages. Use the logging module to configure different logging levels (INFO, DEBUG, ERROR). 4. Using a Debugger*: Demonstrate the use of a Python debugger (e.g., pdb or an IDE with debugging capabilities) on a sample program with intentional errors. Guide students on setting breakpoints, stepping through code, and examining variable values. 5. Scientific Debugging Techniques: Provide a Python program with multiple logic and runtime errors. Instruct students to apply scientific debugging techniques, such as binary search debugging, to identify and resolve the issues methodically 	02
06	<p>Objective: To apply object-oriented programming (OOP) principles in Python to model real-world scenarios and systems, fostering the development of modular, reusable, and efficient solutions. Fostering the ability to design and implement solutions for real-world problems.</p> <p>Choose any one real world scenario. Ask student to apply OOP principles such as encapsulation, inheritance, and polymorphism in practical scenarios. The sample real world scenarios are as follows.</p> <ol style="list-style-type: none"> 1. Event Management System: Implement an event management system using OOP concepts to organize and manage various aspects of college festivals or events. Design classes for events, organizers, participants, and activities. Include methods for event registration, scheduling, participant management, and activity coordination. 2. Online Shopping System: Develop classes for products, customers, and shopping carts. Include methods for adding items to the cart, calculating total costs, processing orders, and managing inventory. 	02

	<p>3. Vehicle Rental System: Design a system using classes for vehicles, rental agencies, and rental transactions. Implement methods to handle vehicle availability, rental periods, pricing, and customer bookings.</p>	
07	<p>Objective: To develop a graphical user interface (GUI) application for any use case. Choose any use case from below.</p> <ol style="list-style-type: none"> 1. GUI for Developing Conversion Utilities: Develop a Python GUI application that performs various unit conversions such as currency (Rupees to Dollars), temperature (Celsius to Fahrenheit), and length (Inches to Feet). The application should include input fields for the values, dropdown menus or buttons to select the type of conversion, and labels to display the results. 2. GUI for Calculating Areas of Geometric Figures: Develop a Python GUI application that calculates the areas of different geometric figures such as circles, rectangles, and triangles. Allows users to input the necessary dimensions for various geometric figures and calculate their respective areas. The application should include input fields for the dimensions, buttons to perform the calculations, and labels to display the results. 3. College Admission Registration Form: The college admission registration form collects essential personal, educational, and contact information from prospective students. Create a GUI as shown in Figure-1 that allows the user to input his/her name, branch and favorite game. When the user clicks the Submit button, it should display the output as illustrated.  <p style="text-align: center;">Figure-1: A basic GUI featuring text field and various buttons.</p>	02
08	<p>Objective: To enable learners to effectively utilize regular expressions in Python for pattern matching, validation, and data extraction tasks, enhancing their ability to process textual data efficiently and accurately.</p> <ol style="list-style-type: none"> 1. Script to Validate Phone Number and Email ID *: Write a Python script that prompts the user to enter their phone number and email ID. It then employs Regular Expressions to verify if these inputs adhere to standard phone number and email address formats 2. Password Strength Checker: Write a Python script that prompts the user to enter a password. Use regular expressions to validate the password based on these criteria: At least 8 characters long, Contains at least one uppercase letter, one lowercase letter, one digit, and one special character. 3. URL Validator: Develop a script that verifies if a given string is a valid URL. Use regular expressions to check for standard URL formats, including protocols (http, https), domain names, and optional path segments. Test with various URLs and ensure the validation covers common cases. 4. Extracting Data from Text *: Create a program that reads a text file containing various data (e.g., names, emails, phone numbers). Use regular expressions to extract specific 	02

	types of data, such as email addresses, phone numbers, dates (e.g., MM/DD/YYYY format).	
09	<p>Objective: To equip learners with the skills to utilize the NumPy libraries for efficient numerical computing.</p> <ol style="list-style-type: none"> Creating and Manipulating Arrays*: Write a Python program to create a 1D, 2D, and 3D NumPy array. Perform basic operations like reshaping, slicing, and indexing. Array Mathematics*: Develop a Python script to create two arrays of the same shape and perform element-wise addition, subtraction, multiplication, and division. Calculate the dot product and cross product of two vectors. Statistical Operations*: Write a Python program to calculate mean, median, standard deviation, variance, and correlation coefficients of a given array. 	02
10	<p>Objective: To provide learners with the knowledge and skills necessary to effectively use the Pandas library for data manipulation and the Matplotlib library for data visualization. Learners will engage in tasks that involve analyzing real-world datasets, creating meaningful visualizations, and drawing insights from data.</p> <p>Following task should be performing on a real-world dataset:</p> <p>Task1- Loading and Inspecting Data: Load a CSV file containing information on global COVID-19 cases into a DataFrame. Display the first few rows, check the data types, and summarize basic statistics.</p> <p>Task 2- Data Cleaning: Identify and handle missing values in the dataset. Remove any duplicate rows and ensure data consistency.</p> <p>Task 3-Data Aggregation: Perform aggregation operations to summarize data.</p> <p><i>Task 4- Plotting graphs: Generate a line plot showing the trend / bar plot to compare data/ histogram to show distribution/ scatter plot to examine relationships between variables.</i></p> <p>Instructors can choose other datasets relevant to the course objectives. Sample datasets and task list are as follows.</p> <p>1. Using the Iris Data (https://www.kaggle.com/datasets/saurabh00007/iris.csv), perform the following tasks:</p> <ol style="list-style-type: none"> Read the first 8 rows of the dataset. Display the column names of the Iris dataset. Fill any missing data with the mean value of the respective column. Remove rows that contain any missing values. Group the data by the species of the flower. Calculate and display the mean, minimum, and maximum values of the Sepal length column. <p>2.Using the Cars Data (https://www.kaggle.com/datasets/nameeerafatima/toyotacsv) perform the following tasks:</p> <ol style="list-style-type: none"> Create a scatter plot between the Age and Price of the cars to illustrate how the price decreases as the age of the car increases. Generate a histogram to show the frequency distribution of kilometers driven by the cars. Produce a bar plot to display the distribution of cars by fuel type. Create a pie chart to represent the percentage distribution of cars based on fuel types. Draw a box plot to visualize the distribution of car prices across different fuel types. 	02

Note: * Marks indicate the minimum required programs to be taken. Additional programs should be covered based on the student's learning pace.

The goal of these experiments is to provide a structured approach to learning Python programming concepts. Instructors are encouraged to use these samples as a foundation and customize them to create engaging and effective learning experiences for the students.

Assessment:

Term Work: Term Work shall consist of at least 15 to 18 practicals based on the above list. Since the initial Python programs are small and straightforward, this allows for more practicals to be conducted, providing essential practice needed for mastering any programming language.

Internal Practical Exam: Conduct an internal practical exam after completing the first three modules of the Python course to assess and ensure the learner's understanding.

Term Work Marks: 25 Marks (Total marks) = 10 Marks (Experiment) + 10 Marks (Internal Practical Exam) + 5 Marks (Attendance)

Practical& Oral Exam: An Oral & Practical exam will be held based on the above syllabus.