Gyanmanjari Innovative Univer ty

Sy llabus
Gyanmanjari Institute of Technology
Semester-3 (B Tech)

Subject: Python Programming -BET2EE13307

Type of course: Engineering Science Course

Prerequisite: Basic Computer Operations, Basic Mathematics, Basic English

Rationale: The course is designed to introduce concepts from scratch, ensuring a smooth learning curve. Emphasis on hands-on practice, real-life examples, and step-by-step coding exercises will help them grasp programming logic effectively. Additionally, the course provides exposure to practical applications such as data analysis and automation, making it relevant for various engineering and non-technical domains.

Teaching and Examination Scheme:

Teaching Scheme			Credits	Examination Marks		Total Marks
CI	T	P	C	SEE	CCE	
2	0	2	3	70	30	100

Legends: CI-Class Room Instructions, T – Tutorial, P – Practical, C – Credit; SEE - Semester End Evaluation, CCE-Continuous and Comprehensive Evaluation

Course Content:

Sr. No.	Course Content	Hrs.	% Weightage
1	**Python Programming Foundations and Problem-Solving** Introduction to Python, Python installation, Python interpreter, IDE / text editor, Creating Python files, Running Python programs, Basic Python syntax, Comments, Error messages, Troubleshooting, Indentation, Basic output, print(), Variables, Variable naming rules, Strings, Single and double quotes, String methods, Changing case, Removing whitespace, Combining strings, f-strings, Formatting output, Numbers, Integers, Floats, Arithmetic operations, Lists, Indexing, List positions, Modifying elements, Adding elements, Removing elements, Sorting, len(), List errors, Iterating through lists, for loops, Loop variables, Iterating over lists, range(), Numerical lists, Statistics, List comprehensions, Repeated operations, Avoiding repetitive code, List slicing, Selecting ranges, Copying lists, List references, Tuples, Immutable data, Looping through tuples, Code organization, PEP 8 principles, Readable Python code, Conditional tests, Equality, Inequality, Comparison operators, Boolean expressions, and, or, not, in, not in, if, elif, else, Conditional logic with lists, Decision-making programs	12	20%

Practical:

1) Python Environment Setup
2) Personal Introduction Program
3) Debugging Challenge
4) Student Profile Generator
5) Personal Finance Calculator
6) Bill and Invoice Generator
7) Personal Shopping List Manager
8) Student Marks List
9) List Debugging Challenge
10) Student Marks Analyzer
11) Number Analysis Tool
12) List Comprehension Challenge
13) Data Slicing Activity
14) Fixed Configuration Using Tuples
15) Code Quality Improvement
16) Student Eligibility Checker
17) Restaurant Ordering System
18) Mini Python Business Application

Evaluation Method :

Sr. No.	Evaluation methods	SEE (Marks)	CCE (Marks)
1	**Develop a Python-based Restaurant or Small-Shop Order Management and Billing System :** Students independently create a complete console-based billing and order management application.	10	--
2	**Develop a Complete Real-World Python Application :** Students independently design and implement a small real-world console application integrating all Module 1 concepts.	10	--
3	**Develop a Python-based Student Performance and Activity Analyzer :** Students develop a complete interactive Python application that demonstrates their	--	10

	understanding of the fundamental Python programming concepts taught throughout		
	Total	20	10

Examination style:

(1) Develop a Python-based Restaurant or Small-Shop Order Management and Billing System. (10 marks)

The student should demonstrate a successfully working interactive Python Customer Ordering and Billing Application that displays shop details and a menu of at least eight products, allows customers to select multiple items with quantities and modify or remove orders, calculates item totals, subtotal, conditional discounts, tax, and final payable amount, performs required list operations using loops, range(), slicing, sorting, and list comprehension, generates a professionally formatted invoice, produces correct results for small, medium, and large order test cases, and successfully implements and demonstrates the additional modification given by the examiner.

(2) Develop a Complete Real-World Python Application. (10 marks)

The student should independently develop and demonstrate a successfully working Python application based on the problem statement provided by the examiner that accepts and processes user input, uses meaningful variables, strings, numbers, lists and tuples, performs indexing, adding, modifying, removing, sorting and slicing operations, uses loops, range() and list comprehension for meaningful processing, applies if-elif-else, comparison and Boolean operators for decision-making, performs relevant numerical calculations, handles valid, invalid and boundary input conditions, follows good coding practices, produces clear formatted output, successfully incorporates the additional requirement given by the examiner without rewriting the complete program, and demonstrates the final application with at least three different test cases while explaining the use of data structures, loops, conditions and list processing.

(3) Develop a Python-based Student Performance and Activity Analyzer. (10 marks)

The student should demonstrate a successfully working interactive Python Student Academic Performance Analyzer that accepts student details and marks, displays a formatted student profile, calculates total, average, highest and lowest marks, performs list operations using indexing, slicing, loops and list comprehension, generates values using range(), determines the student's performance category using conditional and Boolean logic,

	displays fixed course information using a tuple and produces correct results for high-, average-, and low-performing student test cases		
2	**Intermediate Python Programming, Functions, OOP, Files, Exceptions and Testing:-** Dictionaries, Key-value pairs, Creating dictionaries, Accessing values, Adding key-value pairs, Modifying values, Removing values, Looping through dictionaries, Nested dictionaries, Lists of dictionaries, input(), Numerical input, while loops, Loop conditions, Flags, break, continue, Interactive programs, Lists with while loops, Dictionaries with while loops, Defining functions, Calling functions, Parameters, Arguments, Default arguments, Return values, Passing lists, Arbitrary arguments, Modules, Importing modules, Classes, Objects, Attributes, Methods, Constructors, self, Instance data, Inheritance, Parent classes, Child classes, Method reuse, Importing classes, Reading files, Writing files, Appending files, File paths, File errors, File Not Found Error, try, except, Exception handling, JSON, json.dump(), json.load(), Testing, Why testing matters, unittest, Test cases, Testing functions, Testing classes, Assertions, Test-driven improvement, Expanding applications safely **Practical:** 19) Student Profile Dictionary 20) Nested Student Database 21) Product Inventory System 22) Interactive Menu Program 23) Interactive Inventory Manager 24) User Input Validation Challenge 25) Function-Based Calculator 26) Student Management Functions 27) Modular Utility Library 28) Class-Based Student System 29) Class-Based Inventory System 30) Inheritance-Based Management System 31) Text File Reader 32) Safe File Processing System 33) JSON-Based Data Storage 34) Function Unit Testing 35) Class Testing 36) Freelance-Ready Python Application **Evaluation Method :** <table><tr><th>Sr. No.</th><th>Evaluation Component</th><th>SEE (Marks)</th><th>CCE (Marks)</th></tr></table>	12	20%

	1	**Develop an Object-Oriented Inventory Management System :** Students develop a complete object-oriented inventory Management application	10	-	
	2	**Develop a Complete Client-Ready Python Application :** Students receive a real-world client-style problem and independently develop a complete Python application	10	-	
	3	**Develop a Menu-Driven Student Record Management System :** Students develop an interactive record management application	-	10	
		Total	20	10	

Examination style :

(1)Develop an Object-Oriented Inventory Management System (10 marks)

The student should demonstrate a successfully working modular Python Inventory Management Application that uses Product and Inventory classes, creates and manages multiple product objects, demonstrates inheritance through a specialized product class, and allows users to add, view, search, update, and remove products and calculate total inventory value through an interactive menu. The application must save and restore inventory data using JSON, handle missing files and invalid inputs without crashing using exception handling, organize the program into multiple Python files with appropriate imports, and successfully demonstrate product creation, inheritance, inventory operations, calculations, JSON save/load, and error handling through correct executable output.

(2) Develop a Complete Client-Ready Python Application. (10 marks)

This practical exam evaluates a student's ability to analyze a domain requirement and independently design, develop, and test a fully functional Python application. The student must deliver a modular codebase consisting of custom imported modules, structured nested data, and an interactive menu powered by a while loop that supports full CRUD operations, saving, loading, and graceful exit routines. Evaluation relies on assessing two related classes demonstrating constructors, methods, and active child-class inheritance, alongside a persistent .json storage file generated via json.dump() and json.load(). The examiner must verify application robustness by observing real-time error handling that gracefully recovers from File Not Found Error and invalid inputs without crashing. Furthermore, the student must demonstrate a unittest suite with at least five

	test cases, including a live demonstration of a deliberate test failure, its resolution, and updated tests following an on-the-spot requirement change request. Ultimately, marks are awarded based on an end-to-end practical demonstration showing data persistence across restarts, clean code modularity, passing test suites, and the student's brief architectural explanation. **(3) Develop a Menu-Driven Student Record Management System (10 marks)** This practical assessment evaluates the student's ability to build a robust, interactive student management system in Python using structured data, custom modules, and defensive input validation. Evaluation relies on examining a multi-file architecture (e.g., main.py and student_utils.py) where major operations—such as adding, searching, updating, and deleting records—are separated into modular functions demonstrating parameters, return values, and default arguments. The student must demonstrate a pre-populated collection of at least five student records formatted as a list of dictionary of nested records containing IDs, names, ages, courses, and marks. Marks are awarded based on an interactive menu powered by a while loop that uses break for exit routines and continue to gracefully handle invalid user choices, age/marks validation, and duplicate ID prevention without crashing. The student must conduct an end-to-end workflow demonstration showcasing viewing, adding, searching, updating, and deleting a record, followed by confirming its removal. Additionally, the student must demonstrate resilience against invalid inputs and successfully adapt the application live if asked by the examiner to integrate an additional data field into every record		
3	**Object-Oriented Game Application Development Using Pygame:-** Introduction to Pygame, Installing Pygame, Project structure, Creating a Pygame project, Initializing Pygame, Creating a display surface, Window dimensions, Background colors, Event loop, QUIT event, Main game loop, Updating the screen, Frame refresh, Program termination, Classes in game development, Creating a ship class, Constructor, Image loading, Rectangles, Positioning objects, Screen boundaries, Keyboard events, Key press, Key release, Movement flags, Continuous movement, Horizontal movement, Preventing the ship from leaving the screen, Bullet class, Object creation, Sprite-like game objects, Bullet position, Bullet speed, Bullet movement, Firing bullets, Bullet groups, Managing multiple bullets, Removing off-screen bullets, Limiting bullets, Collision preparation, Alien class, Creating individual aliens, Creating rows, Creating columns, Fleet generation, Spacing, Fleet movement, Horizontal movement, Boundary detection, Changing fleet direction, Moving fleet downward, Rebuilding fleets, Collision detection, Bullet-alien collisions, Removing collided objects, Ship-alien collisions, Game statistics, Remaining ships, Resetting	12	20%

game start. End-of-game conditions. Restarting the game. Score calculation. Scoreboard. High score. Level tracking. Difficulty progression, increasing speed. Start button. Game activation. Game reset. Complete game architecture

Practicals

37) Create the Alien Invasion Project Environment
38) Configurable Game Settings
39) Event and Frame Management
40) Create the Ship Class
41) Keyboard-Based Ship Movement
42) Continuous Ship Movement
43) Create the Bullet Class
44) Implement the Fire Mechanism
45) Bullet Lifecycle Management
46) Create the Alien Class
47) Build an Alien Fleet
48) Implement Fleet Movement
49) Bullet-Alien Collision System
50) Ship Collision and Player Lives
51) Game State Management
52) Scoreboard System
53) Levels and Increasing Difficulty
54) Complete Alien Invasion Application

Evaluation Method :

Sr. No.	Evaluation Component	SEE (Marks)	CCE (Marks)
1	**Develop an Alien Fleet Shooting Application using Pygame.:** Students will extend their understanding of Pygame by independently developing an interactive application containing a player-controlled ship, bullets and a moving alien fleet.	10	-
2	**Develop and Demonstrate a Complete Alien Invasion-Style Pygame Game. :** Students will independently develop or complete a fully functional interactive Pygame application that integrates the complete knowledge acquired throughout the module.	10	-

	Develop a Player-Controlled Spacecraft Application using Pygame : Students will independently create a small interactive Pygame application that demonstrates their understanding of the foundational concepts		10
	Total	20	10

Examination style

(1) Develop an Alien Fleet Shooting Application using Pygame.(10 marks)

The student should demonstrate the successful creation of a Python Pygame application in which a player controls a ship, fires multiple bullets, and interacts with a programmatically generated alien fleet. The application should correctly implement bullet creation, movement, management, off-screen removal, bullet limits, alien creation, multi-row and multi-column fleet generation with appropriate spacing, horizontal fleet movement, boundary detection, direction changes, downward movement, and examiner-requested modifications without affecting the existing functionality.

(2) Develop and Demonstrate a Complete Alien Invasion-Style Pygame Game. (10 marks)

The student should demonstrate the successful creation of a complete, modular Python Pygame game that includes a playable game window, player-controlled ship, continuous keyboard movement, bullets, a programmatically generated alien fleet, collision detection, scoring, high score, levels, increasing difficulty, remaining ships, game-state management, game over, restart functionality, and a Play button for game activation. The application should correctly manage gameplay progression, reset and restart the game without affecting existing functionality, retain the high score where appropriate, and successfully implement examiner-requested modifications while maintaining the overall operation of the game

(3) Develop a Player-Controlled Spacecraft Application using Pygame. (10 marks)

The student should demonstrate the successful creation of a Python Pygame application that opens a game window with a visible background, continuously executes the game loop, correctly handles window close events, displays a ship object, supports smooth keyboard-controlled movement with proper key press and key release handling, restricts the ship within the screen boundaries, and correctly implements examiner-requested modifications without affecting the existing functionality.

4	**Data Generation, Visualization and Api-Based Data Applications** **Theory Topics:** Random data, Random number generation, Data simulation, Dice, Probability experiments, Repeated trials, Data collections, NumPy arrays, Numerical processing, Matplotlib, Line charts, Scatter plots, Axis labels, Titles, Data ranges, Customization, Pygal, Bar charts, Histograms, Interactive charts, Tooltips, Data comparison, CSV files, JSON files, Reading structured data, Data cleaning, Data transformation, Data analysis, Visualizing external data, APIs, HTTP requests, requests, Response objects, Status codes, JSON API responses, Processing API data, API errors, GitHub API, Repository data, API query parameters, API response processing, Rate limits, Data visualization, Automated data retrieval. **Practical:** 55) Random Data Generator 56) Dice Simulation 57) Multiple Dice Experiment 58) NumPy Data Processor 59) Scatter Plot Analyzer 60) Customized Data Visualization 61) Interactive Bar Chart 62) Statistical Visualization 63) Data Story 64) CSV Population Analyzer 65) JSON Data Processor 66) Real Dataset Visualization 67) First API Request 68) API JSON Processor 69) API Data Collector 70) GitHub Repository Analyzer 71) API Error Handling 72) API Visualization Dashboard **Evaluation Method :**	12	20%

Sr. No.	Evaluation Component	SEE (Marks)	CCE (Marks)
1	**Students are given an unfamiliar real-world dataset. Develop a Python Data Analysis Application that converts the raw data into meaningful information and visualizations.:** Students will receive an unfamiliar CSV or JSON dataset.	10	

	The student must independently load, inspect, process, clean, transform, analyze and visualize the dataset		
3	**Develop a complete API-powered Data Analysis and Visualization Application.**: Students must independently develop an application that retrieves live data from an online API, validates the response, processes JSON data, handles errors, analyzes the retrieved data and communicates findings through multiple visualizations.	10	
1	**Develop a Python Data Experiment and Visualization Application.** : Students will independently create a Python data-processing and visualization application. The student must generate or simulate numerical data, process the data, calculate meaningful statistical information and create appropriate visualizations.	–	10
	Total	20	10

Examination style

(1) Students are given an unfamiliar real-world dataset. Develop a Python Data Analysis Application that converts the raw data into meaningful information and visualizations (10 marks)

The student should demonstrate the ability to load and read a CSV or JSON dataset programmatically and correctly identify the available data fields. The application should extract relevant numerical and categorical data and process it appropriately. The student should identify and resolve at least one data-quality issue, such as missing, invalid, or duplicate records, and demonstrate the cleaning process. The application should calculate and display the minimum, maximum, and average values and correctly identify the highest and lowest records. The student should create at least two meaningful visualizations, including one using Matplotlib, to represent different aspects of the dataset. The visualizations should be clear, properly labelled, and accurately reflect the processed data. The student should explain at least three meaningful observations based on the analysis and visualizations. When the examiner modifies an analytical requirement, the application should automatically update the data processing, statistical analysis, and visualizations without manual editing of the final output. The application should execute successfully without errors and produce accurate, meaningful, and well-presented results.

	(2) Develop a complete API-powered Data Analysis and Visualization Application. (10 marks) The student should demonstrate the ability to retrieve live data from a public API using Python and validate the API response before processing the data. The application should extract meaningful information from the returned JSON data and perform appropriate statistical analysis such as minimum, maximum, average, count, or category comparison. The student should correctly display the analytical results in a clear and readable format. The application should create at least two meaningful visualizations, including one interactive visualization where appropriate, using the retrieved API data. The student should implement proper error handling for unsuccessful requests, invalid responses, or missing data and display meaningful error messages without terminating unexpectedly. The student should explain how the data flows from the API request through processing, analysis, and visualization. When the examiner modifies the API query or analysis requirement, the application should successfully update the retrieved data, processing, calculations, and visualizations. The application should execute successfully without errors and produce accurate, meaningful, and well-presented analytical results based on dynamically retrieved data. **(3) Develop a Python Data Experiment and Visualization Application. (10 marks)** The student should demonstrate the ability to generate a dataset of numerical values automatically without manual data entry and store the generated data correctly. The application should calculate and display the minimum, maximum, and average values accurately. The student should use NumPy to perform at least one numerical processing operation on the dataset. The application should create a well-formatted Matplotlib chart with an appropriate title and clearly labelled X-axis and Y-axis. The student should also generate an interactive visualization that allows the examiner to inspect or compare data values. The application should present meaningful results in a clear and readable format. The student should correctly identify and explain at least two observations from the generated dataset based on the displayed results or visualizations. When the examiner changes a requirement, such as increasing the number of generated values, the application should automatically generate the new dataset without manual modification. All calculations, charts, and observations should update correctly according to the modified data. The overall application should execute successfully without errors and produce accurate, meaningful, and well-presented outputs.		
5	**Django Web Application Development, Authentication, Version Control and Deployment:-** Virtual environments, Django installation, Project creation, Application creation, Django project structure, Development server, Settings, URL configuration, Django models, Model	12	20%

fields, Database migrations, Migration commands, Django admin, Model registration, URL routing, Views, Templates, Template inheritance, Dynamic content, Context data, Django forms, Model Forms, Form validation, Create, Read, Update, Delete (CRUD), Redirects, User registration, Authentication, Login, Logout, User-specific data, Authorization, Access restrictions, Ownership, CSS integration, Application styling, Git repository, Git commits, Version control, Deployment preparation, Production settings, Deployment.

Practical:

71) Create Django Environment
74) Create Django Project
75) Create Django Application
76) Create Data Model
77) Database Migration
78) Admin Data Management
79) URL Routing
80) Dynamic Views
81) Dynamic Templates
82) Add Data Form
83) Edit Data Form
84) Complete CRUD Workflow
85) User Registration
86) Login and Logout
87) User-Owned Records
88) Style the Application
89) Git Version Control
90) Deploy the Application

Evaluation Method :

Sr. No.	Evaluation Component	SEE (Marks)	CCE (Marks)
1	**Develop a multi-user Django web application in which registered users can create and manage their own records.:** The practical will assess whether the student can move beyond basic Django pages and create a functional multi-user web application.	10	-
2	**Develop and deploy a complete Django web application for a real-world use case.:** The practical assesses whether students can	10	

	transform their Python and Django knowledge into a realistic, functional and deployable web application		
3	**Develop a Django web application for managing a collection of real-world records.** Students will independently create a functional Django web application that demonstrates the fundamental concepts	–	10
	Total	20	10

Examination style

(1)Develop a multi-user Django web application in which registered users can create and manage their own records. (10 marks)
The teacher should evaluate whether the student can develop a working multi-user Django application where each registered user can independently manage their own records. The student should demonstrate successful user registration, login, and logout, with protected pages inaccessible to unauthenticated users. The application should allow logged-in users to create new records through a validated form and automatically assign ownership of each record to the correct user. The student should demonstrate complete Create, Read, Update, and Delete (CRUD) functionality through the web interface. Record list and detail pages should retrieve and display information dynamically from the database. Using two different user accounts, the student should demonstrate that each user can see and manage only their own records. The teacher should verify that one user cannot view, edit, or delete another user's protected records. Successful operations should provide appropriate confirmation or redirect users to the correct page. When the teacher requests a change, the student should correctly update the model, migration, form, view, and template while preserving authentication and ownership restrictions. The final application should run without errors and demonstrate secure multi-user record management with working CRUD, authentication, authorization, and examiner-requested modifications

(2)Develop and deploy a complete Django web application for a real-world use case (10 marks)
The teacher should evaluate whether the student can independently develop and deploy a complete Django web application for a meaningful real-world use case. The application should demonstrate a properly structured Django project with working models, database migrations, admin interface, URLs, views, templates, and database-driven dynamic content. The student should demonstrate complete Create, Read, Update, and Delete (CRUD) operations using validated forms with appropriate redirects. The

	application should provide working user registration, login, and logout functionality and restrict protected pages to authenticated users. Using at least two user accounts, the student should prove that each can access and modify only their own records and cannot access another user's protected data. The application should have a consistent and usable interface with integrated CSS, styled forms, and clear navigation. The student should demonstrate a Git repository containing multiple meaningful commits that show progressive development of the application. The teacher should access the deployed application and verify that its major features work correctly in the live environment. When a new requirement is given, the student should modify the appropriate model, migration, form, view, URL, and template components without breaking existing functionality. The student should also explain, using their own application, how Models, URLs, Views, Templates, Forms, Database, Authentication, and Authorization work together to form the complete Django application **(3) Develop a Django web application for managing a collection of real-world records. (10 marks)** The teacher should evaluate whether the student can independently set up a Django development environment and successfully run the project. The student should demonstrate a working Django project with a properly created and registered application. The application should contain a meaningful database model with at least four fields and successfully applied migrations. The student should demonstrate the Django admin panel by creating at least five records, modifying one record, and deleting one record. The application should retrieve stored records from the database and display them dynamically through a web page. The student should demonstrate correctly configured URLs and views by navigating to the required pages. The application should use a base template and child templates with template inheritance and provide both record-list and record-detail pages. Navigation between the list and detail pages should work correctly. When the teacher requests a modification, such as adding a new model field, the student should update the model, migration, database, and relevant pages correctly. The final application should run without errors and demonstrate that the student can modify the existing Django application without rebuilding it from scratch.		

Suggested Specification Table with Marks:

Distribution of Marks (Revised Bloom's Taxonomy)						
Level	Remembrance (R)	Understanding (U)	Application (A)	Analyze (N)	Evaluate (E)	Create (C)
Weightage %	10%	20%	35%	15%	05%	15%

Note: This specification table shall be treated as a general guideline for students and teachers. The actual distribution of marks in the evaluation may vary slightly from the above table.

Course Outcome:

After learning the course, the students should be able to:	
CO1	Demonstrate the fundamentals of Python programming by using variables, data types, operators, control structures, functions, lists, dictionaries and object-oriented programming concepts to develop Python programs.
CO2	Develop Python applications by applying modular programming, file handling, exception handling, testing techniques and reusable code to solve real-world computational problems.
CO3	Design interactive applications using Python libraries for game development, data visualization and data processing by utilizing Pygame, Matplotlib, Pygal and API integration.
CO4	Build dynamic web applications using the Django framework by implementing models, templates, views, forms, authentication and database management.
CO5	Deploy and evaluate Python applications by applying software testing, version control, debugging and web deployment techniques using Git, Bootstrap and Heroku.

Instructional Method:

The course delivery method will depend on the content requirements and students' needs. The teacher, in addition to the conventional teaching method by the blackboard, may also use any of the tools, such as demonstration, role play, quizzes, brainstorming, MOOCs, etc.

From the content, 10% of topics are suggested for flipped-mode instruction.

Students will utilise supplementary resources, including online videos, NPTEL/SWAYAM videos, e-courses, and Virtual Laboratories.

The internal evaluation will be done on the basis of the CCE-Continuous and Comprehensive Evaluation.

SEE: Semester End Evaluation will be conducted at the end of each module for the evaluation of the performance of students in the laboratory.

Reference Books

[1] "Python Crash Course" by Eric Matthes
[2] "Python Programming: An Introduction to Computer Science" by John Zelle
[3] "Automate the Boring Stuff with Python" by Al Sweigart

Suggested Assessment Guideline

Module1: Python Programming Foundations and Problem-Solving

Develop a Python-based Restaurant or Small-Shop Order Management and Billing System

Criteria	Description	Marks
Problem Understanding & Core Implementation	Develop a functional order management system that displays the menu, accepts multiple customer orders with quantities, stores the order using appropriate data structures, performs basic calculations, and displays the correct order correctly	5
Billing Implementation, Validation & Result Verification	Complete the Restaurant/Small-Shop Order Management and Billing System by generating a professional bill with item-wise details, subtotal, tax/discount (if applicable), grand total, order modification features, proper input validation, and successful testing and execution without errors	5
Total		10

Develop a Complete Real-World Python Application.

Criteria	Description	Marks
Problem Understanding & Core Application Development	Develop the core functionality of the selected real-world Python application by implementing user interaction, data entry, data storage, calculations, decision-making, and displaying meaningful results using appropriate Python concepts	5
Advanced Feature Implementation, Testing & Debugging	Complete the application by implementing advanced features such as structured data management, efficient data processing, input validation, error handling, testing, debugging, and ensuring the application executes successfully with all required functionalities.	5
Total		10

Module2: Intermediate Python Programming, Functions, Oop, Files, Exceptions and Testing

Develop an Object-Oriented Inventory Management System

Criteria	Description	Marks
OOP Concept Understanding & Core Implementation	Develop the core Inventory Management System using Object-Oriented Programming by creating classes, objects, constructors, attributes, methods, and basic inventory operations with a functional user interface.	5
Advanced OOP, Data Persistence, Exception Handling & Testing	Complete the system by implementing inheritance, file handling with JSON for data persistence, exception handling, modular programming, complete interactive functionality, and	5

	successfully demonstrating, testing and modify/explain the application	
Total		10

Develop a Complete Client-Ready Python Application

Criteria	Description	Marks
Problem Understanding & Core Application Development	Develop the core client-ready Python application by implementing user interaction, structured data management functions, modular program structure, classes, objects, constructors, attributes, methods, and the primary application workflow	5
Advanced Implementation, Testing & Client Requirement Handling	Complete the application by implementing inheritance, file handling with JSON for data persistence, exception handling, unit testing, incorporating the final change request, and demonstrating a fully functional client-ready application	5
Total		10

Module3: Object-Oriented Game Application Development Using Pygame

Develop an Alien Fleet Shooting Application using Pygame

Criteria	Description	Marks
Game Concept Understanding & Core Implementation	Develop the core Alien Fleet Shooting application by implementing bullet creation, firing, movement, multiple bullet management, and generating a properly aligned alien fleet with basic gameplay interaction.	5
Game Object Management, Fleet Control & Modification	Complete the application by implementing bullet lifecycle management, bullet limits, fleet movement, boundary detection, direction changes, and successfully demonstrating the application with examiner-requested modifications.	5
Total		10

Develop and Demonstrate a Complete Alien Invasion-Style Pygame Game

Criteria	Description	Marks
Game Architecture & Core Gameplay Implementation	Develop the core Alien Invasion-style Pygame game by implementing the game architecture, game loop, ship movement, bullet system, alien fleet generation, movement, and collision detection with a playable game	5
Game State, Scoring, Difficulty & Result Demonstration	Complete the game by implementing game states, remaining lives, game reset, play button, scoring system, high score, levels, increasing difficulty, and successfully demonstrating the game with examiner-requested modifications	5
Total		10

Module4: Data Generation, Visualization and Api-Based Data Applications

Students are given an unfamiliar real-world dataset. Develop a Python Data Analysis Application that converts the raw data into meaningful information and visualizations.

Criteria	Description	Marks
Data Understanding, Cleaning & Core Analysis	Develop the core Python Data Analysis application by loading the given CSV/JSON dataset, identifying relevant fields, cleaning and transforming the data, performing basic statistical analysis, and identifying key insights such as highest and lowest records.	5
Data Visualization, Modification & Result Interpretation	Complete the application by generating meaningful data visualizations, handling examiner-requested modifications, and successfully demonstrating a fully functional data analysis application that converts raw data into actionable information	5
Total		10

Develop a complete API-powered Data Analysis and Visualization Application

Criteria	Description	Marks
API Integration, Data Processing & Statistical Analysis	Develop the core API-powered application by fetching data from the given API, validating the response, extracting and transforming JSON data, and performing meaningful statistical analysis on the retrieved dataset.	5
Visualization, Error Handling & Application Demonstration	Complete the application by generating meaningful visualizations, implementing robust API error and invalid-data handling, and successfully demonstrating the application with examiner-requested modifications	5
Total		10

Module5: Django web application development, authentication, version control and deployment

Develop a multi-user Django web application in which registered users can create and manage their own records

Criteria	Description	Marks
Django Architecture, CRUD & User Management Implementation	Develop the core multi-user Django web application by creating the database model, implementing complete CRUD functionality, forms with validation, and user registration and login features.	5
Authentication, Authorization, Ownership & Modification Handling	Complete the application by implementing authentication and authorization, demonstrating separate data ownership for multiple users, incorporating examiner-requested modifications, and successfully executing and demonstrating the fully functional application.	5
Total		10

Develop and deploy a complete Django web application for a real-world use case

Criteria	Description	Marks
Django Architecture, Database & CRUD Implementation	Develop the core Django web application by configuring the project architecture, creating models, database and admin, implementing URLs, views, dynamic templates, forms with validation, and complete CRUD functionality	5
Authentication, Deployment, Version Control & Application Demonstration	Complete the application by implementing authentication, authorization, user ownership, improving styling and usability, maintaining the project using Git with meaningful commits, deploying the application, and successfully demonstrating it with examiner-requested modifications and architecture explanation	5
Total		10