School of Computer Science

P.G.D.C.A. (Post Graduate Diploma in Computer Applications) Programme

Subject Code & Name: PGDCA203 - Practical based on PGDCA201 and PGDCA202

Teaching And Evaluation Scheme:

Teaching Scheme				Examination Scheme			
Credits				Component Weightage			
				CCE		SEE	
L	T	P	Total	TH	PWE	TH	PWE
-	-	4	4	-	50%	-	50%

Programme Name	P.G.D.C.A.
Semester	2
Course Code	PGDCA203
Course Title	Practical based on PGDCA201 and PGDCA202
Course Content Type (Th. / Pr.)	Pr.
Course Credit	4
Sessions + Lab Per Week	8
Total Teaching/ Lab Hours	120
*** 2 Lab = 1 session**	

Learning Objectives
Upon successful completion of this course, students will be able to: • Understand the fundamental concepts of Python programming including syntax, variables, data types, operators, and control structures. • Apply problem-solving techniques using conditional, iterative, and functional programming constructs.

- Develop modular and reusable programs using functions, recursion, and advanced Python features.
- Utilize Python collections and NumPy libraries for efficient data manipulation and numerical computation.
- Analyse real-world datasets using statistical functions and basic visualization techniques.
- Design and implement structured, readable, and maintainable Python applications.
- Evaluate alternative programming approaches and select appropriate data structures and algorithms for solving computational problems.
- Create practical Python-based solutions for scientific, business, and data-oriented applications.
- Understand the principles of Object-Oriented Programming and differentiate them from Procedure-Oriented Programming approaches.
- Apply C++ language constructs, control structures, functions, arrays, and strings to develop simple programs for solving computational problems.
- Develop object-oriented solutions using classes, objects, constructors, destructors, inheritance, and polymorphism.
- Implement dynamic program behaviour through function overloading, operator overloading, virtual functions, and runtime polymorphism.
- Utilize exception handling and file handling mechanisms to create robust and efficient C++ applications.

Learning Outcomes

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

- Explain the fundamental concepts of Python programming including syntax, variables, data types, input-output operations, type conversion, and string manipulation techniques.
- Develop Python programs using arithmetic, logical, relational, and bitwise operators along with conditional and iterative control structures to solve computational problems.
- Utilize Python collections (Lists, Tuples, Sets, Dictionaries) and NumPy library functions for efficient data storage, manipulation, and statistical analysis.
- Analyse and implement modular programming concepts using user-defined functions, recursion, lambda expressions, decorators, and higher-order functions.
- Evaluate alternative programming approaches and select appropriate Python constructs, libraries, and data structures for solving real-world problems effectively.
- Design and develop structured, efficient, reusable, and maintainable Python applications by integrating multiple programming concepts and librariesExplain the concepts of Object-Oriented Programming, C++ program structure, data types, operators, arrays, strings, and control structures.

- Construct C++ programs using functions, classes, objects, constructors, destructors, and access specifiers for modular software development
- Apply inheritance, function overloading, operator overloading, and polymorphism techniques to model real-world problems.
- Implement exception handling and file handling operations for developing reliable and maintainable applications.
- Analyse and design object-oriented solutions by selecting appropriate OOP concepts and programming constructs for problem solving.

CLO–PLO Mapping Matrix:

CLO/ PLO	PLO1	PLO2	PLO3	PLO4	PLO5	PLO6
CLO1	3	1	–	1	1	1
CLO2	2	3	2	3	2	1
CLO3	2	3	2	2	3	2
CLO4	2	3	3	3	2	2
CLO5	1	2	2	3	3	3
CLO6	2	3	3	2	3	3

L:: Lecture, **T::** Tutorial, **P::** Practical
CCE:: Continuous & Comprehensive Evaluation
(CCE theory includes Mid Semester Examination, Assignment, MCQ Quizzes, Seminar, Reflective Notes, Class Participation, Case Analysis and Presentation, Slip Tests (Announced/ Surpirsed), Attendance etc. or any combination of these)
PWE: Practical Work Examination
(PWE includes Laboratory Practical Work, Project Work, Viva simulation exercise work etc.)
SEE:: Semester End Evaluation

Sample practical exercises based on PGDCA201- Programming in Python

UNIT-1:

1. Write a Python program to display "Hello World".
2. Write a Python program to print your personal information (Name, Course, College, Mobile No.).
3. Write a program to declare and display variables of different data types.
4. Write a program to accept user input and display it.
5. Write a program to assign multiple values to multiple variables and to assign one value to multiple variables.
6. Demonstrate dynamic variable declaration in Python.
7. Write a program to calculate the sum of two numbers entered by the user.
8. Write a program to swap two variables without using a third variable.

9. Write a program to identify the data type of different variables using type().
10. Write a program demonstrating integer, float, and complex numbers.
11. Write a program to perform arithmetic operations on complex numbers.
12. Write a program to demonstrate Boolean values and logical operations.
13. Write a program to create and display a set datatype.
14. Write a program to demonstrate type casting using int(), float(), and str().
15. Write a program to access individual characters of a string using indexing.
16. Write a program to perform string slicing operations.
17. Write a program to demonstrate negative indexing in strings.
18. Write a program to find the length of a string using appropriate method.
19. Write a program to concatenate two strings.
20. Write a program demonstrating string methods: upper(), lower(), replace(), split(), count(), and join().

UNIT – II:

1. Perform arithmetic operations on two numbers.
2. Find the largest of two numbers.
3. Find the largest of three numbers.
4. Check whether a number is positive, negative, or zero.
5. Check whether a number is even or odd.
6. Check voting eligibility.
7. Prepare Marksheet of 12th Standard.
8. Implement a simple calculator using conditional statements.
9. Check whether a year is a leap year.
10. Find the smallest of three numbers.
11. Generate numbers from 1 to 10 using loop.
12. Find the sum of first N natural numbers using while loop.
13. Generate multiplication tables using loop.
14. Generate numbers from 1 to 20 using for.
15. Generate multiplication tables using for.
16. Demonstrate range() function.
17. Demonstrate pass statement.
18. Demonstrate else with for loop.
19. Demonstrate nested for loops by printing patterns.
20. Generate Fibonacci series using loops.

UNIT – III:

1. Create and display a list.
2. Access list elements using indexing.
3. Perform various list slicing operations.
4. Perform Add, Insert, Remove, Delete, Sort and Reverse elements in a list using appropriate end().
5. Count occurrences of an element in the list.
6. Perform Copy operation on a list.
7. Create and display tuples.
8. Access tuple elements.
9. Use count() and index() on tuples.

10. Convert a list into a tuple.
11. Create a set and display elements.
12. Add and Remove elements to a set.
13. Perform union of two sets.
14. Demonstrate update() method.
15. Create dictionary and Add and Access a dictionary elements.
16. Remove elements using pop().
17. Copy dictionaries.
18. Create a NumPy array from a list.
19. Calculate Mean, Median, Mode, Standard Deviation and Variance using NumPy.
20. Create a simple data visualization (Bar Chart/Line Chart) using Python.

UNIT – IV:

1. Define and call a simple function.
2. Create a function with parameters.
3. Create a function with return value.
4. Demonstrate local variables.
5. Demonstrate global variables.
6. Create a function with default arguments.
7. Create a function using keyword arguments.
8. Demonstrate *args and **kwargs with proper example.
9. Write a recursive function to calculate factorial.
10. Write a recursive function for Fibonacci series.
11. Create a function to check prime numbers.
12. Create a function to calculate simple interest.
13. Create a function to find maximum of three numbers.
14. Create a lambda function for addition.
15. Create a lambda function for square of a number.
16. Use map() to square list elements.
17. Use map() to convert strings to uppercase.
18. Use filter() to find even numbers.
19. Use filter() to find prime numbers.
20. Use reduce() to calculate product of list elements.

Sample Practical exercises based on PGDCA202 - OOP using C++

1. Write a C++ program to display personal information like name, phone number, email id
2. Write a C++ program to perform basic arithmetic operations on two numbers entered by the user.
3. Write a C++ program to generate multiplication tables from 5 to 10 using loops.
4. Write a C++ program to find the largest, smallest, and average of three numbers.
5. Write a C++ program to convert temperature between Celsius and Fahrenheit.
6. Develop a class TemperatureConverter to perform temperature conversions using member functions.
7. Write a C++ program to swap two numbers using call-by-reference.

8. Write a C++ program to dynamically allocate memory for storing and displaying student marks using the new operator.
9. Write a C++ program to generate different number and star patterns using nested loops.
10. Write a C++ program to calculate Simple Interest, Compound Interest.
11. Develop a Student Result Processing System using arrays to calculate total marks, percentage, and grade.
12. Write a C++ program to maintain employee records and calculate salary details using structures.
13. Develop a utility billing system to calculate charges based on user consumption slabs.
14. Write a function to input a matrix of user-defined dimensions.
15. Write a C++ program to perform matrix addition and subtraction using functions.
16. Write a C++ program to demonstrate default arguments through matrix operations.
17. Write a C++ program to demonstrate function overloading for area calculation of different shapes.
18. Write a macro to find the maximum of three values.
19. Implement an inline function to find the maximum of three values and compare it with a macro.
20. Write a function to calculate powers of numbers using default arguments.
21. Demonstrate function overloading by implementing power functions for integer and floating-point values.
22. Design a class BankAccount to perform account creation, deposit, withdrawal, and balance inquiry operations.
23. Extend the BankAccount class to manage records of multiple customers.
24. Design a class Vector to create, update, scale, and display vectors.
25. Extend the Vector class to perform vector addition and subtraction using objects as arguments.
26. Use a friend function to add distances represented in different units.
27. Develop a program to display converted distance values in user-selected units.
28. Write a program demonstrating the use of constructors, parameterized constructors, copy constructors, and destructors.
29. Design a banking system with Savings and Current Accounts using inheritance.
30. Implement account operations such as deposit, withdrawal, interest calculation, and penalty deduction using derived classes.
31. Create a Shape hierarchy using virtual functions to calculate areas of Triangle and Rectangle.
32. Extend the Shape hierarchy by adding a Circle class and demonstrate runtime polymorphism.

33. Demonstrate the role of virtual functions by comparing outputs with and without overriding in derived classes.
34. Write a C++ program to store and display product information in a formatted tabular report using stream manipulators.
35. Modify the report generation program to display formatted output with custom fill characters and alignment.
36. Write a program to read and display contact details from a text file.
37. Create a formatted directory report showing names and contact numbers in aligned columns.
38. Develop a class-based Contact Management System that stores records in a file.
39. Write a C++ program to demonstrate exception handling for invalid user inputs.
40. Develop a program that handles divide-by-zero, invalid data type, and file access exceptions using multiple catch blocks.