Faculty of Technology and Engineering

U & P U. Patel Department of Computer Engineering

Date: 03 / 01 / 2025

Practical List

Academic Year	:	2024-25	Semester	:	4th
Course code	:	CE269	Course name	:	Programming in Python

Sr. No.	Aim	Hours	CO
1.	To develop environment configuration skills by setting up Python, Anaconda, and virtual environments for efficient project dependency management. **Installation** • Install Python and verify its installation using the command prompt or terminal. • Install Anaconda and validate it using system commands. **Configuration** • Set up environment variables (if required) to access Python and Conda from the command line. • Verify the versions of Python and Anaconda installed on the system. **Virtual Environment** • Create a new virtual environment using both Conda and venv methods. • Activate and deactivate the created virtual environment. • Install a specific package (like NumPy) in the virtual environment and verify its installation. • List installed packages in the virtual environment. **Project Isolation** • Understand the need for project isolation and how virtual environments ensure it. • Demonstrate the impact of package version isolation using an example.	2	1
2.	**a) Implement following operation using python tuple concept.** **Tuple operation** • Create tuples with different data types (integer, float, string, and mixed). • Access tuple elements using positive and negative indices.	3	2

- Perform tuple slicing to extract specific portions of the tuple.
- Count occurrences of an element and find the index of an element in a tuple.
- Use built-in functions like len(), max(), min(), and sum() with tuples.
- Write a program to count and print distinct elements from a tuple.
- Convert a list to a tuple and vice versa.
- Demonstrate unpacking of tuples into individual variables.

b) Implement following operation using Python List concept.

List Operation

- Create a list of integers, strings, and mixed data types.
- Access elements using indices, perform slicing, and update list elements.
- Add and remove elements using append(), insert(), remove(), and pop() methods.
- Concatenate and repeat lists using operators.
- Create a list of squares of the first 10 natural numbers using list comprehension.
- Filter even numbers from a list using list comprehension.
- Demonstrate sorting, reversing, and copying lists.
- Write a program to remove duplicates from a list.

c) Implementing following operation using python dictionaries concept.

Dictionary Operation:

- Create a dictionary to store key-value pairs.
- Access, update, and delete dictionary elements using keys.
- Use dictionary methods like keys(), values(), and items().
- Add a new key-value pair and remove an existing key-value pair.
- Create a nested dictionary to store student details (like name, age, and marks).
- Access and update elements in a nested dictionary.
- Merge two dictionaries using update().
- Write a program to sort a dictionary based on its values.

d) Implementing following operation using python set concept.

Set Operation:

- Create a set to store unique elements.
- Add elements to a set.
- Remove elements from a set.
- Combine elements from two sets.
- Find common elements between two sets.
- Find elements present in one set but not in another.
- Find elements present in either of the sets but not in their intersection.

- Check if one set is a subset of another.
- Check if one set is a superset of another.
- Remove all elements from a set.

Assignment Task for Students
Use tuples in practical data handling scenarios.

- Write a function that takes a tuple as an argument and returns the tuple with all duplicates removed.
- Create a list of tuples representing student names and marks, and sort the list by marks.
- Write a program to count the frequency of elements in a tuple using Counter from the collections module.
- Implement a tuple-based record system where each tuple represents a record (ID, Name, and Marks) and perform search operations.

Apply list concepts to real-world scenarios.

- Implement a program to generate a list of prime numbers within a given range.
- Flatten a nested list using recursion.
- Write a program to find the second largest element from a list without using built-in functions.
- Use a list to manage a task queue, where tasks are added, removed, and processed sequentially.

Apply dictionary concepts to real-world scenarios.

- Write a program to count the frequency of each word in a string and store it in a dictionary.
- Implement a simple phonebook application using a dictionary where users can add, delete, and search for contacts.
- Create a dictionary of students and their grades. Write a program to filter students who scored more than a specific mark.
- Write a program to convert a list of tuples (key-value pairs) into a dictionary and vice versa.

Apply set concepts to real-world scenarios.

- Write a program to remove duplicate elements from a list using a set.
- Write a program to find the symmetric difference of multiple sets.
- Write a program to find the intersection of two sets where one set has even numbers and another has prime numbers.
- Write a program to find the union of two sets where one set has odd numbers and another has multiple of 9's.
- Write a program to find the difference of two sets where one set has positive numbers and another has negative numbers.

Capstone Project1: College Event Management System
Objective: Apply tuple, list, and dictionary concepts to manage participants and event details.

	Task: • Store event information as a dictionary where the event name is the key and the value is a list of participant tuples. • Each tuple contains (Participant Name, Contact Number, Department, Participation Status). • Write a program to: 1. Display the list of participants for a specific event. 2. Search for a participant by name and display their event details. 3. Mark a participant as "Attended" or "Not Attended". 4. Generate a summary of total participants in each event. **Capstone project2: Online Food Delivery System** **Objective:** Design an online food delivery system using list, tuple, dictionary, and set concepts. **Task:** • Store menu items as a dictionary where item names are keys and (Price, Category) is the value. • Create a list of orders where each order is a tuple (Order ID, Customer Name, Item List, Total Bill). • Use a set to store unique customer names who have placed orders. • • Write a program to: 1. Allow users to place an order by selecting items from the menu. 2. Generate a bill for the customer and store it in the list of orders. 3. Display the total revenue generated from all orders. 4. Display the list of unique customers who have placed orders.		
3.	**Function and loop in python:** **Basics of function** • Define a simple function that takes inputs and returns an output. • Define a function with positional, keyword, default, and variable-length arguments • Define a function that returns the multiple values using tuple. • Define an anonymous function using lamba keyword • Define a function inside another function. • Create and use decorators to modify the behavior of functions. • Define a function that calls itself to solve a problem recursively. • Define functions that take other functions as arguments or return functions as results. • Add docstrings to functions to document their purpose and usage. • Use type annotations to specify the expected types of function arguments and return values. **Basics of loops** • Iterate over a sequence (list, tuple, string, or range) using a for loop.	2	1, 2

	• Repeat a block of code as long as a condition is true using a while loop. • Use loops inside other loops to handle multi-dimensional data structures. • Use break, continue, and pass to control the flow of loops. • Use the enumerate function to get both the index and value while iterating over a sequence. • Use the range function to generate a sequence of numbers for iteration. • Iterate over the key-value pairs of a dictionary using a for loop. • Use list comprehensions to create new lists by applying an expression to each item in an existing list. **Assignment task for the students:** • Write a program to create a basic calculator with functions for addition, subtraction, multiplication, and division. • Write a recursive function to calculate the factorial of a number. • Write functions to perform various list operations such as finding the maximum, minimum, sum, and average of a list of numbers. • Write a function that generates the Fibonacci sequence up to a given number of terms using a for loop. • Write functions to add, update, and delete key-value pairs in a dictionary, merge two dictionaries, and display the dictionary contents using loops. • Write a program to create a simple to-do list application that allows users to add, remove, and view tasks. • Write a program that accepts a list of numbers and returns a new list containing only the even numbers. • Write a program that finds the largest and smallest numbers in a list without using built-in functions like max() and min().		
4.	**Class, Objects and Inheritance:** • Create a class with attributes and methods. • Instantiate an object from a class. • Access and modify the attributes of an object. • Call methods defined in a class using an object. • Access and modify the attributes of an object using getter and setter methods. • Create a subclass that inherits from a superclass. • Override methods in a subclass to provide specific implementations. • Call methods from the superclass using the super() function. • Define and use class variables that are shared among all instances of a class. • Define and use instance variables that are unique to each object. • Create a class that inherit from multiple superclass	3	4

	• Understand and use the method resolution order to determine the order in which base classes are searched. **Assignments for the students:** • Create a Student class with attributes for name, age, and grades, and methods to calculate the average grade and display student information. • Create a BankAccount class with attributes for account number, balance, and account type, and methods to deposit, withdraw, and display account information. • Create a Person superclass with attributes for name and age, and a Student subclass that inherits from Person and adds an attribute for student ID and a method to display student information.		
5.	**Polymorphism, Encapsulation, and Method & Attributes:** • Use private attributes and provide public getter and setter methods to access them. • Demonstrate polymorphism by defining a common interface and implementing it in multiple classes. • Override methods in a subclass to provide specific implementations. • Define multiple methods with the same name but different parameters to handle different types of inputs. • Define class methods using the @classmethod decorator to operate on class-level data. • Define static methods using the @staticmethod decorator for utility functions that do not depend on instance or class data. • Use the @property decorator to create getter and setter methods for attributes. **Assignment task for the students:** • Create a Shape superclass with a method to calculate the area, and Circle and Rectangle subclasses that override the area calculation method to demonstrate polymorphism. • Create a Book class with private attributes for title, author, and ISBN, and public getter and setter methods to access and modify these attributes, demonstrating encapsulation. **Capstone Project: Online Bookstore System** **Objective**: Design an online bookstore system using Object-Oriented Programming (OOP) concepts such as class, object, inheritance, polymorphism, encapsulation, and method & attributes. **Task Description:** • **Store books as objects**: Create a class to represent books with attributes like title, author, genre, and price.	4	4

	• **Store customer orders**: Use another class to represent customer orders, where each order contains customer details, list of ordered books, and total bill. • **Use inheritance for different types of books**: Use inheritance to model different categories of books (e.g., E-books, Printed books). • **Use polymorphism**: Implement polymorphism to calculate the total price for different types of books (printed vs. e-books). • **Encapsulation**: Implement encapsulation to protect sensitive data like customer payment details and order history. • **Attributes and Methods**: Define appropriate methods to allow customers to place orders, view book catalog, and view the total revenue of the bookstore.		
6.	**Modules and Packages:** • Import a module and use one of its functions to perform a simple operation (Math and sqrt). • Use a module to work with dates and times. • Create and use a custom module. • Work with a package in Python. • Check if a package is installed and use it. • Use the pip command to install a third-party package. • Use the sys module to manipulate the module search path. • Reload a module after making changes to it. • Create a package with an __init__.py file and use its submodules. • Check the version of an installed package using pip. • List all installed packages using pip. • Uninstall a package using pip. **Assignment tasks for the students:** • Create a custom Python module with at least two functions (e.g., one that performs basic arithmetic operations and one that checks if a number is even or odd). Import and use this module in your main script. • Import the datetime module and use it to display the current date and time, as well as format it in a readable way. • Create a custom Python module that contains two functions: calculate_area for calculating the area of different shapes (circle, rectangle, triangle) and is_prime for checking if a number is prime, then import and use this module in your main script to perform calculations.	2	1, 2
7.	**File Operations:** • Open a file in read, write, or append mode using the open function. • Read the entire content of a file using read(), readline(), readlines(). • Write data to a file using write() and writelines() • Append data to the end of a file using write() in append mode.	2	3

	• Perform the different file operations (close, tell, seek, os.path.exists, os.remove, os.rename, os.path) **Assignment for the students:** • Write a program that creates a new file, writes some content to it, closes the file, and then reopens it to read and display the content. • Write a program that reads a list of numbers, inserts odd numbers into a file named odd_numbers.txt, and even numbers into a file named even_numbers.txt. • Write a program that creates a new file, writes some content to it, closes the file, and then reopens it to read and display the content. • Write a program that reads a text file and prints any 5 words from the file. • Write a program that generates a triangle pattern of 5 rows and saves the pattern into a file named triangle.txt.		
8.	**Exception Handling:** • Use a try-except block to catch and handle exceptions. • Use multiple except blocks to handle different types of exceptions. • Use an else block to execute code if no exceptions are raised. • Use a finally block to execute code regardless of whether an exception is raised. • Use the raise statement to raise an exception manually. **Assignments for the students:** • Write a program to define a custom exception and raise it in specific scenarios, then handle it using a try-except block. • Write a program to access an element in a list and handle an IndexError if the index is out of range. Display a user-friendly error message and prompt the user to enter a valid index. • Write a program to convert a string to an integer and handle a ValueError if the string is not a valid number. Display a user-friendly error message and prompt the user to enter a valid number.	2	1
9.	**Database Connection:** • Establish a connection to an SQLite database. • Create a cursor object for executing SQL commands. • Execute SQL commands like CREATE TABLE, INSERT, SELECT, UPDATE, and DELETE. • Use cursor.fetchone(), cursor.fetchall(), and cursor.fetchmany() to retrieve query results. • Use connection.commit() to save changes made to the database. • Use connection.close() to close the database connection. • The methods use try and except blocks to handle database-related exceptions.	3	3

	Assignments for the students: • Create a student management system that: 1. Allows adding new students 2. Updates existing student records 3. Deletes student records 4. Displays all students 5. Searches for students by various criteria **Capstone Project: Online Bookstore System** Enhance the basic online bookstore system to include database integration, file handling, exception handling, and an interactive interface. **Task Description:** • **Database Integration**: Use SQLite to store and manage book inventory, customer details, and order records. • **File Operations**: Provide functionality to export and import book inventory and customer orders to/from text files. • **Modules and Packages**: Organize the system into reusable modules and packages for database operations, file handling, and utility functions. • **Exception Handling**: Handle errors related to database operations, file handling, and invalid user inputs gracefully. • **Interactive Interface**: Create a menu-driven program that allows users to browse books, place orders, view order history, and manage inventory.		
10.	**Numpy Operations:** • **np.array()**: Create a NumPy array from a list or tuple. • **np.zeros()**: Create an array filled with zeros. • **np.ones()**: Create an array filled with ones. • **np.arange()**: Create an array with a range of values. • **np.linspace()**: Create an array with evenly spaced values over a specified interval. • **np.reshape()**: Change the shape of an array without changing its data. • **np.flatten()**: Convert a multi-dimensional array into a 1D array. • **np.transpose()**: Transpose an array (swap rows and columns). • **np.concatenate()**: Join two or more arrays along an axis. • **np.split()**: Split an array into multiple sub-arrays. • **np.vstack()**: Stack arrays vertically (row-wise). • **np.hstack()**: Stack arrays horizontally (column-wise). • **np.append()**: Append values to the end of an array. • **np.delete()**: Delete elements from an array along a specified axis. • **np.insert()**: Insert values into an array at specified positions. • **arr[i]**: Access an element of an array at index i. • **arr[i:j]**: Slice an array from index i to j (exclusive). • **arr[start:end:step]**: Slice an array with a step between start and end.	2	2

	• **arr[:, :]**: Access all elements along rows and columns (for 1D, 2D, and 3D arrays). **Assignment for the students:** **https://www.kaggle.com/code/utsav15/100-numpy-exercises**		
11.	**Hello world in Django:** • Set Up the Django Project • Create a Django App • Define a URL Pattern • Create a View • Create a Template • Run the Development Server **Assignment for the students:** • Create a basic calculator which include only basic operations of addition, multiplication, division and subtraction.	2	4
12.	**Crud operations using django (Book Management System)** • Set Up the Django Project • Define a Book model with fields for title, author, publication date, ISBN, and genre. • Create View: Create a view to display a form for adding a new book and handle the form submission to save the book record. • Read View: Create a view to display a list of all books and a view to display the details of a single book. • Update View: Create a view to display a form for editing an existing book and handle the form submission to update the book record. • Delete View: Create a view to handle the deletion of a book record. • Create templates for the following: 1. A form to add a new book. 2. A form to edit an existing book. 3. A list view to display all books. 4. A detail view to display the details of a single book. 5. A confirmation page for deleting a book. • Define URL patterns for the create, read, update, and delete views. • Map the URL patterns to the corresponding views in the app's urls.py file. • Create a Django form for the Book model to handle input validation and display errors. • Use the form in the create and update views. • Register the Book model with the Django admin interface to manage book records through the admin panel.	3	4