Week 4: Industrial Training Report

Overview

Week 4 of the industrial training at **Auribises Technologies Pvt. Ltd.** focused on advanced Python concepts including recursion, linked lists, stacks, containers, data structures, file handling, and modules. The week emphasized practical applications through small projects and exercises, allowing students to understand how data is stored, processed, and persisted in real-world scenarios.

Day 16: Recursion, Function Operations, and String Formatting

The sixteenth day started with **function memory operations**, exploring **reference copy** and **hash codes** to understand how functions affect data in memory. We introduced **recursion**, which uses the memory stack to perform repeated operations, especially in **non-linear data structures** like trees and graphs. We also discussed the difference between **loops (linear approach)** and **recursion (non-linear approach)**.

Topics Covered:

- Reference copy and hash code usage in functions
- Recursion vs. loops
- Factorial and Fibonacci series using recursion
- String formatting using `.format()`

Example 1: Factorial Using Recursion

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)

print(factorial(5))  # Output: 120
```

Example 2: Fibonacci Series Using Recursion

```
def fibonacci(n):
    if n <= 1:
        return n
    else:
        return fibonacci(n-1) + fibonacci(n-2)

for i in range(7):
    print(fibonacci(i), end=' ')
```

Day 17: Linked Lists and Stack Implementation

Day 17 focused on **linked lists** (singly, doubly, circular doubly) and their applications in Python. We practiced adding and removing items from a playlist and a car collection using a **circular doubly linked list**. Additionally, we implemented **stacks** using linked lists, practicing **push, pop, and iteration** operations.

Topics Covered:

- Singly and doubly linked lists
- Circular doubly linked list as a stack
- Stack operations (LIFO)
- Shopping cart assignment using circular doubly linked list

Example: Adding Products to a Circular Doubly Linked List (Shopping Cart)

```
class Node:
    def __init__(self, product):
        self.product = product
        self.next = None
        self.prev = None

class CircularDoublyLinkedList:
    def __init__(self):
        self.head = None

    def add_product(self, product):
        new_node = Node(product)
        if not self.head:
            self.head = new_node
```

```
            new_node.next = new_node.prev = new_node
        else:
            tail = self.head.prev
            tail.next = new_node
            new_node.prev = tail
            new_node.next = self.head
            self.head.prev = new_node

cart = CircularDoublyLinkedList()
cart.add_product("Laptop")
cart.add_product("Smartphone")
```

Day 18: Containers and Data Properties

Day 18 explored **single and multi-value containers** in Python. We studied integers, floats, booleans, tuples, lists, strings, sets, and dictionaries. Operations like **indexing, negative indexing, slicing, concatenation, multiplicity, and membership testing** were applied to these containers.

Topics Covered:

- Single value containers: int, float, bool
- Multi-value containers: list, tuple, string, set, dictionary
- Properties: indexing, negative indexing, slicing, concatenation, multiplicity, membership
- Practice using 1D and 2D lists

Example 1: String Slicing and Concatenation

```
email = "john@example.com"
username = email[:4]  # 'john'
domain = email[5:]     # 'example.com'
full_email = username + "@" + domain
```

Example 2: Multiplicity and Membership

```
data = [1, 2, 3]
multiple_data = data * 3  # [1, 2, 3, 1, 2, 3, 1, 2, 3]
print(10 in data)  # False
```

Day 19: Tuples, Sets, Dictionaries, and Data Persistence

Day 19 involved applying container properties to **tuples, sets, and dictionaries**. We practiced operations like insertion, deletion, uniqueness, and iteration. We also explored **data persistence**, which allows saving data for long-term storage and retrieval.

Topics Covered:

- Tuple operations
- Set operations: add, remove, union, intersection
- Dictionary operations: keys, values, insertion, deletion, iteration
- Persistent storage and data parsing

Example 1: Dictionary Iteration

```
attendance = {"Alice": 90, "Bob": 85, "Charlie": 92}
for student, marks in attendance.items():
    print(f"{student}: {marks}%")
```

Example 2: Set Operations

```
A = {1, 2, 3}
B = {3, 4, 5}
print("Union:", A | B)
print("Intersection:", A & B)
```

Day 20: File Handling and Modules

The final day focused on **file handling** and Python **modules**. We worked on creating and reading plain text files (`.txt`, `.json`, `.py`) and saving structured data like **visitor logs** into CSV files. We also learned how to organize Python code into modules for reusability.

Topics Covered:

- File handling: open, read, write, append, using `with` keyword
- Limitations: cannot process `.pdf`, `.docx`, `.pptx`
- Visitor Log Book project using OOP and CSV
- Modules: creating and importing `.py` files for reusable code

Example: Writing Visitor Details to CSV

```
import csv

visitors = [["John", "10:00 AM"], ["Alice", "10:30 AM"]]
with open('visitors.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(["Name", "Time"])
    writer.writerows(visitors)
```

Example: Importing a Module

```
# math_operations.py

def add(a, b):
    return a + b

# main.py
import math_operations
print(math_operations.add(5, 10))  # Output: 15
```

Summary

Week 4 provided a comprehensive understanding of recursion, linked lists, stacks, containers, data structures, file handling, and modules. Students gained practical experience with circular

doubly linked lists, stacks, shopping cart applications, and visitor log projects, along with understanding the difference between temporary file storage and persistent databases. The exercises reinforced key concepts and prepared learners for real-world programming scenarios.