**CHAPTER-1**

**DAY TO DAY ACTIVITIES**



**Internship Program on Python for BE-3rd Sem students**

**From 09th to 28th September 2024 (During 3rd semester vacations).**

**Student Name:Yamini D USN No: 3BR23CD106 Branch: CSE-DS**

|  |  |  |  |
| --- | --- | --- | --- |
| **INDEX PAGE** | | | |
| **Day** | **Date** | **Content Covered** | **Signature of the** |
| **faculty in-charge** |
| **1** | **09.09.24** | **Introduction to Python, Setup & Installation, First Python Program, Variables, Data Types, and Basic I/O** |  |
| **2** | **10.09.24** | **Control Structures: If-else, Loops,**  **Functions and Modules** |  |
| **3** | **11.09.24** | **Lists, Tuples, and Dictionaries, File Handling** |  |
| **4** | **12.09.24** | **Exception Handling, Practice exercises on Python basics** |  |
| **5** | **13.09.24** | **Introduction to OOP, Classes, and Objects** |  |
| **6** | **14.09.24** | **Inheritance, Polymorphism, and Encapsulation** |  |
| **7** | **15.09.24** | **Abstract Classes and Interfaces** |  |
| **8** | **17.09.24** | **Practice exercises on OOP concepts** |  |
| **9** | **18.09.24** | **Introduction to DSA, Arrays, and Linked Lists** |  |
| **10** | **19.09.24** | **Introduction to DSA, Array, and Linked List** |  |
| **11** | **20.09.24** | **Introduction to Stack & Queue** |  |
| **12** | **21.09.24** | **Practice Exercise on basic concept (Reduce, Lambda function, List Comprehension)** |  |
| **13** | **23.09.24** | **Introduction to Tree Data Structure** |  |
| **14** | **24.09.24** | **Introduction to Graph Data Structure** |  |
| **15** | **25.09.24** | **Searching Algorithms and Project Building & Persentations** |  |
| **16** | **26.09.24** | **Project Building & Persentations** |  |
| **17** | **27.09.24** | **Project Building & Persentations** |  |
| **18** | **28.09.24** | **Project Building & Persentations** |  |

**COMPANY PROFILE**

**Company Name: EZ Trainings and Technologies Pvt. Ltd.**

**Introduction:**

EZ Trainings and Technologies Pvt. Ltd. is a dynamic and innovative organization dedicated to providing comprehensive training solutions and expert development services. Established with a vision to bridge the gap between academic learning and industry requirements, we specialize in college trainings for students, focusing on preparing them for successful placements. Additionally, we excel in undertaking development projects, leveraging cutting-edge technologies to bring ideas to life.

**Mission:**

Our mission is to empower the next generation of professionals by imparting relevant skills and knowledge through specialized training programs. We strive to be a catalyst in the career growth of students and contribute to the technological advancement of businesses through our development projects.

**Services:**

**College Trainings:**

* Tailored training programs designed to enhance the employability of students.
* Industry-aligned curriculum covering technical and soft skills.
* Placement assistance and career guidance.

**Development Projects:**

* End-to-end development services, from ideation to execution.
* Expertise in diverse technologies and frameworks.
* Custom solutions to meet specific business needs.

**Locations:** Hyderabad | Delhi NCR

At EZ Trainings and Technologies Pvt. Ltd., we believe in transforming potential into excellence

# ABSTRACT

# ### Abstract

# This document presents a Python-based Disk Space Management Tool designed to monitor and manage disk space usage on a given file system path. The tool includes a class, DiskSpaceManager, which provides methods to retrieve disk usage statistics, analyze disk space utilization, and suggest potential cleanup actions to maintain optimal disk space.

# The primary functionalities include calculating total, used, and free disk space, determining the percentage of disk space utilized, and offering actionable suggestions when disk usage exceeds certain thresholds.

# To ensure reliability and correctness, the implementation is complemented by a suite of unit tests using Python's unittest framework. These tests validate the proper functioning of the disk usage retrieval and the generation of cleanup recommendations.

# The tool is designed to be modular, easy to integrate into larger systems, and capable of running independently for monitoring and management purposes.

# The Disk Space Management Tool is particularly useful for system administrators and developers who need to automate the monitoring of disk space usage and ensure that systems do not run out of disk space, potentially avoiding critical failures.

# The tool can be expanded and customized based on specific needs, making it a flexible solution for disk space management tasks.

# INTRODUCTION OF THE PROJECT

Disk space management is a critical aspect of maintaining the health and efficiency of computer systems, especially in environments where large amounts of data are generated and stored. As storage needs grow, so does the potential for disk space to become a limiting factor, leading to degraded system performance or even failure if the disk runs out of space. Effective disk space management ensures that sufficient storage is available for system operations and data storage, helping to prevent system crashes and data loss.

This document introduces a Python-based Disk Space Management Tool designed to automate the process of monitoring and managing disk space. The tool provides a straightforward approach to retrieving disk usage statistics, analyzing disk space utilization, and offering actionable cleanup suggestions when necessary. The solution is implemented in the form of a class, DiskSpaceManager, which encapsulates the core functionalities required for disk space management.

Key features of the tool include:

1. \*Disk Usage Retrieval\*: The tool provides detailed information about total, used, and free disk space on a given file system path.

2. \*Disk Space Analysis\*: It analyzes the disk space usage by calculating the percentage of space utilized, offering insights into the current state of the disk.

3. \*Cleanup Suggestions\*: Based on the analysis, the tool suggests potential actions to free up disk space when usage exceeds defined thresholds or when free space falls below a certain level.

The implementation also includes unit tests to verify the correctness of the functionality provided by the DiskSpaceManager class. This ensures that the tool behaves as expected under various conditions.

The Disk Space Management Tool is intended for use by system administrators, developers, and other IT professionals who need to maintain optimal disk space usage on their systems. It can be easily integrated into existing monitoring frameworks or used as a standalone utility.

# MODULE DESCRIPTION

### Description

The Python-based Disk Space Management Tool is a utility designed to automate the monitoring and management of disk space on a given file system path. The tool is implemented using a class, DiskSpaceManager, which encapsulates the core functionalities necessary for retrieving disk usage information, analyzing disk space utilization, and suggesting cleanup actions to maintain optimal disk usage.

#### Key Components:

1. \*DiskSpaceManager Class\*:

- \*\*Initialization (\_\_init\_\_ Method):

- The class is initialized with a specified file system path.

- It uses Python's shutil.disk\_usage() function to retrieve and store disk space statistics for the provided path, including total, used, and free space.

- \*\*get\_disk\_usage Method\*\*:

- This method returns a dictionary containing the current disk usage statistics (total, used, and free).

- It provides a simple way to access the raw disk space data for monitoring purposes.

- \*\*analyze\_disk\_space Method\*\*:

- This method calculates the percentage of disk space used.

- It returns a dictionary that includes the total space, used space, free space, and the calculated usage percentage.

- This analysis helps in understanding how much of the disk space is currently occupied and how much is still available.

- \*\*suggest\_cleanup\_actions Method\*\*:

- Based on the analysis of disk space, this method provides recommendations to optimize disk usage.

- If the usage percentage exceeds 80%, it suggests removing unnecessary files.

- If the available free space is less than 10 GB, it suggests freeing up space to avoid running out of disk.

- If the disk space usage is within acceptable limits, it indicates that the disk space usage is optimal.

- The method returns a list of suggestions that can be used to guide cleanup efforts.

2. \*Unit Testing (TestDiskSpaceManager Class)\*:

- The tool includes a suite of unit tests to ensure the correct functionality of the DiskSpaceManager class.

- \*\*test\_disk\_usage Method\*\*: Verifies that the get\_disk\_usage method returns a dictionary containing the expected keys (total, used, and free).

- \*\*test\_cleanup\_suggestions Method\*\*: Checks that the suggest\_cleanup\_actions method returns at least one suggestion, ensuring that the method produces valid output under different conditions.

3. \*\*Main Guard (\_\_main\_\_):

- The main section of the script is responsible for executing the unit tests and providing an example usage of the DiskSpaceManager.

- It initializes a DiskSpaceManager object for the root path (or any specified path), retrieves and displays the disk usage details, performs disk space analysis, and prints the cleanup suggestions.

#### Usage:

The Disk Space Management Tool is designed to be flexible and easy to integrate into existing systems. It can be used as a standalone utility for monitoring disk space or as part of a larger system administration toolkit. By automating the process of disk space analysis and providing actionable insights, the tool helps to ensure that systems maintain optimal performance and avoid issues related to insufficient disk space.

This tool is particularly useful for system administrators, IT professionals, and developers who manage systems with significant storage requirements. It provides a proactive approach to disk space management, enabling timely interventions to prevent potential problems related to disk space exhaustion

# ALGORITHM

Certainly! Here's an algorithm that describes the logic and steps involved in your DiskSpaceManager Python code.

### \*Algorithm for DiskSpaceManager\*

1. \*Initialization\*:

- \*Input\*: A file system path (e.g., "/").

- \*Process\*:

1. Use the shutil.disk\_usage(path) function to retrieve the total, used, and free disk space for the given path.

2. Store the retrieved values (total, used, free) as attributes of the DiskSpaceManager class instance.

- \*Output\*: An initialized DiskSpaceManager object with disk usage data for the specified path.

2. \*Get Disk Usage\*:

- \*Input\*: None.

- \*Process\*:

1. Create a dictionary containing the total, used, and free disk space values stored in the class instance.

- \*Output\*: A dictionary with the disk usage information:

- 'total': Total disk space.

- 'used': Used disk space.

- 'free': Free disk space.

3. \*Analyze Disk Space\*:

- \*Input\*: None.

- \*Process\*:

1. Calculate the disk usage percentage using the formula:

\[

\text{{usage percentage}} = \left(\frac{{\text{{used}}}}{{\text{{total}}}}\right) \times 100

\]

2. Create a dictionary containing:

- 'total\_space': Total disk space.

- 'used\_space': Used disk space.

- 'free\_space': Free disk space.

- 'usage\_percentage': Calculated disk usage percentage.

- \*Output\*: A dictionary containing the disk space analysis information.

4. \*Suggest Cleanup Actions\*:

- \*Input\*: None.

- \*Process\*:

1. Call analyze\_disk\_space() to get the disk usage analysis.

2. Initialize an empty list suggestions to store cleanup recommendations.

3. If the usage percentage is greater than 80%, add a suggestion to consider removing unnecessary files.

4. If the free space is less than 10 GB, add a suggestion to free up some space to avoid running out of disk.

5. If no suggestions are added, add a suggestion stating that disk space usage is optimal.

- \*Output\*: A list of suggestions or recommendations to manage disk space better.

5. \*Unit Tests\*:

- \*Setup\*:

1. Initialize a DiskSpaceManager object with the root path (e.g., "/").

- \*Test Disk Usage\*:

1. Call get\_disk\_usage() and verify that the returned dictionary contains keys 'total', 'used', and 'free'.

- \*Test Cleanup Suggestions\*:

1. Call suggest\_cleanup\_actions() and verify that the returned list contains at least one suggestion.

6. \*Execution (Main Guard)\*:

- \*Input\*: None.

- \*Process\*:

1. Execute the unit tests.

2. Initialize a DiskSpaceManager object with the root path (e.g., "/").

3. Retrieve and print the disk usage details using get\_disk\_usage().

4. Perform disk space analysis using analyze\_disk\_space() and print the results.

5. Retrieve and print cleanup suggestions using suggest\_cleanup\_actions().

This algorithm provides a clear, step-by-step guide to understanding the logic behind the DiskSpaceManager code, from initialization to suggesting cleanup actions and testing the functionality.

# SOURCE CODE

# import os

# import shutil

# import unittest

# class DiskSpaceManager:

# def \_init\_(self, path):

# self.path = path

# self.total, self.used, self.free = shutil.disk\_usage(path)

# def get\_disk\_usage(self):

# 

# return {

# 'total': self.total,

# 'used': self.used,

# 'free': self.free

# }

# def analyze\_disk\_space(self):

# 

# usage\_percentage = (self.used / self.total) \* 100

# return {

# 'total\_space': self.total,

# 'used\_space': self.used,

# 'free\_space': self.free,

# 'usage\_percentage': usage\_percentage

# }

# def suggest\_cleanup\_actions(self):

# 

# suggestions = []

# usage\_info = self.analyze\_disk\_space()

# if usage\_info['usage\_percentage'] > 80:

# suggestions.append("Consider removing unnecessary files.")

# if usage\_info['free\_space'] < 10 \* (1024 \*\* 3):  # If less than 10 GB is free

# suggestions.append("Free up some space to avoid running out of disk.")

# 

# if not suggestions:

# suggestions.append("Disk space usage is optimal.")

# 

# return suggestions

# class TestDiskSpaceManager(unittest.TestCase):

# def setUp(self):

# self.manager = DiskSpaceManager("/")

# def test\_disk\_usage(self):

# usage = self.manager.get\_disk\_usage()

# self.assertIn('total', usage)

# self.assertIn('used', usage)

# self.assertIn('free', usage)

# def test\_cleanup\_suggestions(self):

# suggestions = self.manager.suggest\_cleanup\_actions()

# self.assertTrue(len(suggestions) > 0)

# if \_\_name\_\_ == "\_main\_":

# unittest.main()

# 

# 

# 

# if \_\_name\_\_ == "\_main\_":

# 

# manager = DiskSpaceManager("/")

# disk\_usage = manager.get\_disk\_usage()

# print("Disk Usage Details:")

# print(disk\_usage)

# analysis = manager.analyze\_disk\_space()

# print("\nDisk Space Analysis:")

# print(f"Total Space: {analysis['total\_space']} bytes")

# print(f"Used Space: {analysis['used\_space']} bytes")

# print(f"Free Space: {analysis['free\_space']} bytes")

# print(f"Usage Percentage: {analysis['usage\_percentage']:.2f}%")

OUTPUT

Disk Usage Details:

{'total': 512110190592, 'used': 324558110720, 'free': 187552079872}

Disk Space Analysis:

Total Space: 512110190592 bytes

Used Space: 324558110720 bytes

Free Space: 187552079872 bytes

Usage Percentage: 63.39%

**CONCLUSION**

**### Conclusion**

**Effective disk space management is crucial for maintaining the performance, stability, and reliability of computer systems, especially in environments where data storage demands are continuously increasing. The Python-based Disk Space Management Tool presented in this document provides a practical and automated solution for monitoring and managing disk space usage on any given file system path.**

**The tool’s core functionalities—disk usage retrieval, disk space analysis, and cleanup action suggestions—are encapsulated within the `DiskSpaceManager` class, offering a modular and easy-to-use interface. By automating these processes, the tool helps system administrators and IT professionals prevent disk space-related issues, such as system slowdowns or failures due to insufficient storage.**

**The inclusion of unit tests ensures that the tool operates as expected, providing confidence in its accuracy and reliability. Additionally, the example usage provided in the main section of the script demonstrates how the tool can be easily integrated into daily operations or larger system monitoring frameworks.**

**In summary, this Disk Space Management Tool not only simplifies the task of disk space monitoring but also provides actionable insights that can prevent critical issues before they occur. It is an invaluable utility for anyone responsible for maintaining the health and efficiency of computer systems, offering a proactive approach to disk space management that can adapt to a wide range of environments and use cases.**

**REFERENCE**

* **GOOGLE**
* **CHAT GPT**
* **META AI**