**Summary –Day7**

**Name:Tejaswini Gokanakonda**

## **Roll no**:**DE142**

## **Date:13-11-2024**

### **Python Collections for Data Engineering**

### **1. Get Unique Values from a List using map and set**

* set(): Removes duplicate elements from a list.
* map(): Transforms items in a list.
* **Example:**

data = ["apple", "banana", "apple", "orange", "banana"] unique\_values = list(set(data)) print(unique\_values)

*# Output: ['banana', 'orange', 'apple']*

#### **2.Sort Python Lists using key**

* **Sorting with**key: Customizes sorting using a function.
* **Example:**

fruits = ["apple", "banana", "grape"] sorted\_fruits = sorted(fruits, key=len) print(sorted\_fruits)

*# Output: ['grape', 'apple', 'banana']*

#### **3. Overview of JSON Strings and Files**

* **JSON (JavaScript Object Notation)**: Lightweight data format for storing and exchanging data.
* **Example JSON String:**

json

{"name": "John", "age": 30, "city": "New York"}

* **JSON File Reading Steps**: Use json module in Python.

#### **4. Read JSON Strings to Python dicts or lists**

* json.loads(): Converts JSON string to Python dict or list.
* **Example:**

python

import json json\_string = '{"name": "Alice", "age": 25}' data = json.loads(json\_string) print(data['name'])

*# Output: Alice*

#### **5. Read JSON Schemas from File to Python dicts**

* json.load(): Reads JSON data from a file.
* **Example:**

python

with open('schema.json', 'r') as file: schema = json.load(file) print(schema)

#### **6. Processing JSON Data using Python**

* **Extracting values, iterating over data, filtering, etc.**
* **Example:**

python

data = {"people": [{"name": "Alice"}, {"name": "Bob"}]} for person in data["people"]: print(person["name"])

#### **7. Extract Details from Complex JSON Arrays using Python**

* Use **nested loops** and **list comprehensions**.
* **Example:**

python

data = {"employees": [{"name": "Tom", "skills": ["Python", "SQL"]}]} skills = [emp["skills"] for emp in data["employees"]] print(skills)

*# Output: [['Python', 'SQL']]*

#### **8. Sort Data in JSON Arrays using Python**

* sorted()**with**key for JSON arrays.
* **Example:**

python

data = [{"name": "John", "age": 35}, {"name": "Alice", "age": 30}] sorted\_data = sorted(data, key=lambda x: x["age"]) print(sorted\_data)

*# Output: [{'name': 'Alice', 'age': 30}, {'name': 'John', 'age': 35}]*

#### **9. Create Function to Get Column Details from Schemas JSON File**

* **Example Function:**

python

Copy code

def get\_column\_details(schema): return [col["name"] for col in schema["columns"]]

#### **10. Lists and Tuples in Python**

* **Lists**: Mutable, ordered collections.

python

fruits = ["apple", "banana"] fruits.append("orange")

* **Tuples**: Immutable, ordered collections.

python

colors = ("red", "blue", "green")

#### **11. Enriching Data using Numpy & Pandas**

* **Numpy**: Fast operations on arrays.

python

import numpy as np arr = np.array([1, 2, 3]) print(arr \* 2)

*# Output: [2 4 6]*

* **Pandas**: Dataframes for tabular data processing.

python

import pandas as pd df = pd.DataFrame({'Name': ['Tom', 'Jerry'], 'Age': [25, 30]}) print(df)

**Data Processing using Pandas DataFrame APIs**

#### **1. Pandas for Data Processing**

* **Pandas** is a powerful library for data manipulation and analysis. It uses data structures like Series (1D) and DataFrame (2D).
* **Key Features**: Easy handling of missing data, merging and joining datasets, reshaping data, and performing statistical operations.
* **Example:**

python

import pandas as pd *# Creating a DataFrame* df = pd.DataFrame({'Name': ['Alice', 'Bob'], 'Age': [25, 30]}) print(df)

**Output**:

markdown

Name Age 0 Alice 25 1 Bob 30

#### **2. Reading CSV Data using Pandas**

* CSV (Comma-Separated Values) is a common file format used for storing data.
* pd.read\_csv(): Reads a CSV file into a DataFrame.
* **Example:**

python

df = pd.read\_csv('data.csv') *# Assumes 'data.csv' exists* print(df.head()) *# Display the first 5 rows*

#### **3. Read Data from CSV Files to Pandas DataFrames**

* **Description**: DataFrames are table-like data structures. You can load data from files like CSV, Excel, etc.
* **Example:**

python

df = pd.read\_csv('employees.csv') print(df.head()) *# Displays the first 5 rows of the DataFrame*

#### **4. Filter Data in Pandas DataFrame using query**

* **Filtering**: Extracts rows based on conditions.
* query(): Allows for easy filtering with boolean expressions.
* **Example:**

python

*# Sample DataFrame*

df = pd.DataFrame({'Name': ['Alice', 'Bob', 'Eve'], 'Age': [25, 30, 35]}) *# Filter rows where Age is greater than 25* filtered\_df = df.query('Age > 25') print(filtered\_df)

**Output**:

Name Age 1 Bob 30 2 Eve 35

#### **5. Get Count by Status using Pandas DataFrame APIs**

* **Description**: Grouping data and counting occurrences for each group.
* **Example:**

python

df = pd.DataFrame({'Status': ['Active', 'Inactive', 'Active']}) count\_by\_status = df.groupby('Status').size() print(count\_by\_status)

**Output**:

mathematica

Status Active 2 Inactive 1 dtype: int64

#### **6. Get Count by Month and Status using Pandas DataFrame APIs**

* groupby()**with multiple columns**: Allows for hierarchical grouping.
* **Example:**

python

df = pd.DataFrame({'Month': ['Jan', 'Jan', 'Feb'], 'Status': ['Active', 'Inactive', 'Active']}) count\_by\_month\_status = df.groupby(['Month', 'Status']).size() print(count\_by\_month\_status)

**Output**:

mathematica

Month Status Feb Active 1 Jan Active 1 Inactive 1 dtype: int64

#### **7. Create DataFrames using Dynamic Column List on CSV Data**

* **Dynamic Columns**: Specify columns to load from a CSV file.
* **Example:**

python

cols = ['Name', 'Age'] df = pd.read\_csv('data.csv', usecols=cols) print(df)

#### **8. Performing Inner Join between Pandas DataFrames**

* **Inner Join**: Combines rows from two DataFrames where a common key exists.
* pd.merge(): Used for joining DataFrames.
* **Example:**

python

df1 = pd.DataFrame({'ID': [1, 2], 'Name': ['Alice', 'Bob']}) df2 = pd.DataFrame({'ID': [2, 3], 'Salary': [5000, 6000]}) joined\_df = pd.merge(df1, df2, on='ID', how='inner') print(joined\_df)

**Output**:

yaml

ID Name Salary 0 2 Bob 5000

#### **9. Perform Aggregations on Join Results**

* groupby()**with**agg(): Applies multiple aggregation functions.
* **Example:**

python

result = joined\_df.groupby('Name').agg({'Salary': 'sum'}) print(result)

**Output**:

markdown

Salary Name Bob 5000

#### **10. Sort Data in Pandas DataFrames**

* sort\_values(): Sorts data by one or more columns.
* **Example:**

python

df = pd.DataFrame({'Name': ['Alice', 'Bob'], 'Age': [30, 25]}) sorted\_df = df.sort\_values(by='Age') print(sorted\_df)

**Output**:

Name Age 1 Bob 25 0 Alice 30

#### **11. Writing Pandas DataFrames to Files**

* to\_csv(): Writes a DataFrame to a CSV file.
* **Example:**

python

df.to\_csv('output.csv', index=False)

#### **12. Write Pandas DataFrames to JSON Files**

* to\_json(): Converts and saves data to a JSON file.
* **Example:**

python

df.to\_json('data.json')