NAME: ISHITA JINDAL

REG. NO.: 21BCB0061

EMAIL: ishitajindal1203@gmail.com

Python Weekly Assignment – 20-Oct-24

1. Analyzing Student Performance

A school administrator wants to analyze students' scores from a file that contains records of students and their exam results in the format name,score. Unfortunately, sometimes the file might be missing, corrupted, or contain invalid data. Write a program that reads the file, calculates the average score, and lists students who scored above average. Ensure proper handling of missing files and malformed data.

Ans.

```
import os
from statistics import mean

def validate_score(score_str):
    try:
        score = float(score_str)
        if 0 <= score <= 100:
            return score
        print(f"Score {score} is outside valid range (0-100)")
        return None
    except ValueError:
        print(f"Invalid score format: {score_str}")
        return None

def analyze_scores(filename):
    if not os.path.exists(filename):
        print(f"File '{filename}' not found.")
```

```
    return

students = []
invalid_count = 0

try:
    with open(filename, 'r') as file:
        for line_num, line in enumerate(file, 1):
            line = line.strip()
            if not line:
                continue

            parts = line.split(',')
            if len(parts) != 2:
                print(f"Line {line_num}: Wrong format - {line}")
                invalid_count += 1
                continue

            name, score_str = parts
            name = name.strip()
            if not name:
                print(f"Line {line_num}: Empty name")
                invalid_count += 1
                continue

            score = validate_score(score_str)
            if score is not None:
                students.append((name, score))

    if not students:
        print("No valid student records found in the file.")
```

```
            return

        scores = [score for _, score in students]
        avg_score = mean(scores)

        above_average = [(name, score) for name, score in students if score > avg_score]
        above_average.sort(key=lambda x: x[1], reverse=True)

        print("\nClass Statistics:")
        print(f"Class average: {avg_score:.2f}")
        print("\nTop Performers (above average):")
        for name, score in above_average:
            difference = score - avg_score
            print(f"{name}: {score:.1f} ({difference:+.1f} from average)")

    except PermissionError:
        print(f"Permission denied accessing file '{filename}'")
    except Exception as e:
        print(f"Unexpected error reading file: {str(e)}")

def main():
    filename = input("Enter the scores file name: ")
    analyze_scores(filename)

if __name__ == "__main__":
    main()
```

2. <u>Product Availability in a Store</u>

You work for an online store, and you need to help the operations team clean up their product list. They have a list of product IDs that contains duplicates due to system errors. Write a function that takes this list, removes duplicates, sorts the product IDs,

and returns the cleaned list. Make sure your function can handle an empty product list input.

Ans.

```
def clean_product_list(product_ids):
  try:
    unique_products = sorted(set(product_ids))
    if not unique_products:
      print("Warning: Empty product list provided")
      return []
    return unique_products
  except TypeError as e:
    print(f"Error: Invalid product ID found - {e}")
    return []

def get_product_ids():
  product_ids = []
  print("Enter product IDs (press Enter without input to finish):")

  while True:
    try:
      user_input = input("Enter product ID: ").strip()
      if not user_input:
        break
      product_id = int(user_input)
      if product_id <= 0:
        print("Error: Please enter positive numbers only")
        continue
      product_ids.append(product_id)
    except ValueError:
      print("Error: Please enter valid numeric IDs only")
```

```
            continue
    return product_ids

def main():
    product_ids = get_product_ids()
    cleaned_products = clean_product_list(product_ids)
    print(f"Original list: {product_ids}")
    print(f"Cleaned list: {cleaned_products}")

if __name__ == "__main__":
    main()
```

3. <u>Organizing Sales Data</u>

A small business owner has sales data in the form of tuples, each containing the customer's name and the amount they spent (e.g., ('Alice', 200)). Write a program that stores this data in a dictionary, where the customer's name is the key and the amount spent is the value. If a customer appears more than once, update their total spending. Print the customer data sorted by their names.

Ans.

```
def organize_sales_data():
    customer_sales = {}

    print("Enter customer sales data")
    print("Format: customer_name amount")

    while True:
        sale_input = input("\nEnter sale details: ").strip()

        if not sale_input:
```

```
            break

        try:
            name, amount = sale_input.rsplit(maxsplit=1)
            name = name.strip()
            amount = float(amount)

            if not name:
                print("Please enter a customer name")
                continue

            if amount <= 0:
                print("Amount must be greater than zero")
                continue

            if name in customer_sales:
                customer_sales[name] += amount
            else:
                customer_sales[name] = amount

        except ValueError:
            print("Please use format: customer_name amount")
            continue

    if customer_sales:
        print("\nCustomer Sales Summary:")
        print("-" * 40)
        print("Customer Name".ljust(25) + "Total Spent")
        print("-" * 40)

        for name in sorted(customer_sales.keys()):
```

```
        print(f"{name[:24].ljust(25)}${customer_sales[name]:.2f}")

    print("-" * 40)
    total = sum(customer_sales.values())
    print(f"Total Sales:".ljust(25) + f"${total:.2f}")
  else:
    print("\nNo sales data was entered.")

if __name__ == "__main__":
  organize_sales_data()
```

4. <u>Saving User Preferences</u>

A mobile app allows users to customize settings like theme (dark/light mode), language, and notification preferences. Write a program that saves a user's preferences using the pickle module and retrieves them when needed. Handle cases where the preferences file is missing or corrupted.

Ans.

```
import pickle
import os

def save_preferences(preferences):
  try:
    with open('user_prefs.dat', 'wb') as file:
      pickle.dump(preferences, file)
    print("Preferences saved successfully!")
  except Exception as e:
    print(f"Unable to save preferences: {e}")

def load_preferences():
  if not os.path.exists('user_prefs.dat'):
```

```
        print("No saved preferences found.")
        return get_default_preferences()

    try:
        with open('user_prefs.dat', 'rb') as file:
            return pickle.load(file)
    except Exception as e:
        print(f"Error loading preferences: {e}")
        return get_default_preferences()

def get_default_preferences():
    return {
        'theme': 'light',
        'language': 'English',
        'notifications': True
    }

def get_user_choice(prompt, options):
    while True:
        print(f"\n{prompt}")
        for i, option in enumerate(options, 1):
            print(f"{i}. {option}")

        choice = input("\nEnter your choice: ").strip()
            print("Please enter a number")

def manage_preferences():
    preferences = load_preferences()

    while True:
        print("\nUser Preferences Menu:")
```

```
print("1. View current preferences")
print("2. Change theme")
print("3. Change language")
print("4. Toggle notifications")
print("5. Save and exit")

choice = input("\nSelect an option (1-5): ").strip()

match choice:
    case "1":
        print("\nCurrent Preferences:")
        for key, value in preferences.items():
            print(f"{key.title()}: {value}")

    case "2":
        preferences['theme'] = get_user_choice(
            "Select theme:",
            ['light', 'dark']
        )

    case "3":
        preferences['language'] = get_user_choice(
            "Select language:",
            ['English', 'Spanish', 'French', 'German']
        )

    case "4":
        preferences['notifications'] = not preferences['notifications']
        print(f"Notifications {'enabled' if preferences['notifications'] else 'disabled'}")

    case "5":
```

```
            save_preferences(preferences)
            break

        case _:
            print("Please select a valid option (1-5)")

if __name__ == "__main__":
    manage_preferences()
```

5. <u>Analyzing Employee Salaries</u>

A company's HR department maintains employee records in a CSV file, which includes details like employee name, department, and salary. You've been tasked with analyzing this data to calculate the total and average salary per department. Write a program that reads the CSV using pandas, computes the required data, and saves the results to a new CSV. Handle situations where the file is missing or contains invalid data.

Ans.

```
import pandas as pd
import os

def analyze_salaries(input_file, output_file):
    if not os.path.exists(input_file):
        print(f"Error: '{input_file}' not found.")
        return

    try:
        data = pd.read_csv(input_file)
```

```
        if 'department' not in data.columns or 'salary' not in data.columns:
            print("Error: Missing required columns 'department' or 'salary'.")
            return

        data['salary'] = pd.to_numeric(data['salary'], errors='coerce')
        data = data.dropna(subset=['salary'])

        salary_summary = data.groupby('department')['salary'].agg(['sum', 'mean']).reset_index()
        salary_summary.columns = ['Department', 'Total Salary', 'Average Salary']

        salary_summary.to_csv(output_file, index=False)

        print(f"Salary analysis complete. Results saved to '{output_file}'.")
        print(salary_summary)

    except pd.errors.EmptyDataError:
        print("Error: Input file is empty.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

input_file = 'employee_salaries.csv'
output_file = 'salary_analysis.csv'
analyze_salaries(input_file, output_file)
```

6. <u>Validating User Signups</u>

Your company's website allows users to sign up with their email addresses. Write a Python program that checks if the provided email addresses are valid using regular expressions. Make sure the emails follow the proper format (e.g., username@domain.com). Your program should filter out invalid emails from a given list of signups.

Ans.

```
import re

def is_valid_email(email):
    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
    return re.match(pattern, email)

def filter_valid_emails(email_list):
    return [email for email in email_list if is_valid_email(email)]

def get_user_emails():
    emails = []
    print("Enter email addresses:")

    while True:
        email = input("Enter email: ").strip()
        if not email:
            break
        emails.append(email)

    return emails

emails = get_user_emails()
valid_emails = filter_valid_emails(emails)

print("Valid emails:", valid_emails)
```

7. <u>Currency Conversion Calculator</u>

You're building a currency conversion tool for a travel website. The tool should take two user inputs: the amount to convert and the conversion rate. Implement a program

that handles cases where the user enters invalid data, such as non-numeric input or a conversion rate of zero, and provides appropriate error messages.

Ans.

```
def currency_conversion():
  while True:
    try:
      amount = float(input("Enter the amount to convert: "))
      if amount <= 0:
        print("Error: Amount must be greater than zero.")
        continue
      break
    except ValueError:
      print("Error: Please enter a valid numeric amount.")

  while True:
    try:
      rate = float(input("Enter the conversion rate: "))
      if rate <= 0:
        print("Error: Conversion rate must be greater than zero.")
        continue
      break
    except ValueError:
      print("Error: Please enter a valid numeric conversion rate.")

  converted_amount = amount * rate
  print(f"Converted amount: {converted_amount:.2f}")

currency_conversion()
```

8. <u>Movie Ratings Aggregation</u>

A movie streaming service collects user ratings for movies. Each movie can be rated on a scale of 1 to 10. Write a program that takes a list of movie ratings and uses list comprehension to filter out ratings below 5 (bad ratings) and return a new list of good ratings squared. Handle cases where no ratings are provided.

Ans.

```
def aggregate_movie_ratings(ratings):
  if not ratings:
    return "No ratings provided."

  good_ratings_squared = [rating**2 for rating in ratings if rating >= 5]

  return good_ratings_squared

def get_user_ratings():
  ratings = []
  print("Enter movie ratings (1-10):")

  while True:
    try:
      rating = input("Enter rating: ").strip()
      if not rating:
        break
      rating = int(rating)
      if 1 <= rating <= 10:
        ratings.append(rating)
      else:
        print("Error: Please enter a rating between 1 and 10 only.")
    except ValueError:
```

```
            print("Error: Please enter a valid numeric rating.")

    return ratings

ratings = get_user_ratings()
result = aggregate_movie_ratings(ratings)

print("Good ratings squared:", result)
```

9. <u>Extracting Contact Information</u>

A company stores client data in text files, and some of the records contain phone numbers in inconsistent formats, such as (123) 456-7890 or 123-456-7890. Write a program that reads a text file, uses regular expressions to extract all phone numbers in either format, and prints the list of valid phone numbers.

Ans.

```
import re

def extract_phone_numbers(filename):
    try:
        with open(filename, 'r') as file:
            content = file.read()

        phone_pattern = r'\(\d{3}\)\s?\d{3}-\d{4}|\d{3}-\d{3}-\d{4}'
        phone_numbers = re.findall(phone_pattern, content)

        if phone_numbers:
            print("Valid phone numbers found:")
            for number in phone_numbers:
                print(number)
```

```
        else:
            print("No valid phone numbers found.")

    except FileNotFoundError:
        print(f"Error: The file '{filename}' was not found.")
    except Exception as e:
        print(f"An error occurred: {e}")
```

10. <u>Removing Duplicate User Data</u>

A loyalty program has a list of customer records, each stored as a tuple with the customer's name and email address (e.g., ('John Doe', 'john@example.com')). Due to an import error, some customers are listed multiple times. Write a Python program that removes duplicate entries using a set and prints the unique list of customers.

Ans.

```
def remove_duplicate_customers(customers):
    unique_customers = list(set(customers))
    return unique_customers

def print_customers(customers):
    if not customers:
        print("No customers found.")
        return

    print("Unique customer list:")
    for customer in customers:
        print(customer)
```

```
def get_customer_records():
    customers = []
    print("Enter customer records (name, email). Type 'done' to finish:")

    while True:
        entry = input("Enter customer record: ").strip()
        if entry.lower() == 'done':
            break

        try:
            name, email = map(str.strip, entry.split(','))
            customers.append((name, email))
        except ValueError:
            print("Error: Please enter the record in the format 'Name, Email'.")

    return customers

if __name__==__main__:
    customers = get_customer_records()
    unique_customers = remove_duplicate_customers(customers)
    print_customers(unique_customers)
```

11. Product Inventory Analysis

Your company manages product inventory through a CSV file that contains product ID, name, and quantity available. Write a program using pandas to filter products with low stock (less than 10 units). Handle potential issues like a missing or malformed CSV file, or missing columns in the data.

Ans.

```
import pandas as pd
import os

def create_inventory_file(filename):
    print("Enter product details (product_id, name, quantity):")

    products = []

    while True:
        entry = input("Enter product record: ").strip()
        if entry.lower() == 'done':
            break

        try:
            product_id, name, quantity = map(str.strip, entry.split(','))
            product_id = int(product_id)
            quantity = int(quantity)
            products.append((product_id, name, quantity))
        except ValueError:
            print("Error: Please enter the record in the format 'product_id, name, quantity'.")

    if products:
        df = pd.DataFrame(products, columns=['product_id', 'name', 'quantity'])
        df.to_csv(filename, index=False)
        print(f"Product inventory saved to '{filename}'.")
    else:
        print("No products entered. Inventory file not created.")

def analyze_inventory(input_file):
    if not os.path.exists(input_file):
        print(f"Error: The file '{input_file}' was not found.")
```

```
        return

    try:
        data = pd.read_csv(input_file)

        if 'product_id' not in data.columns or 'name' not in data.columns or 'quantity' not in
data.columns:
            print("Error: Missing required columns 'product_id', 'name', or 'quantity'.")
            return

        low_stock_products = data[data['quantity'] < 10]

        if low_stock_products.empty:
            print("No products with low stock found.")
        else:
            print("Products with low stock:")
            print(low_stock_products[['product_id', 'name', 'quantity']])

    except pd.errors.EmptyDataError:
        print("Error: The input file is empty.")
    except pd.errors.ParserError:
        print("Error: The input file is malformed.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

if __name__==__main__:
    filename = 'product_inventory.csv'
    create_inventory_file(filename)
    analyze_inventory(filename)
```

12. Statistical Analysis for a Sports Team

A sports analyst wants to analyze the performance statistics of players on a team. Each player's performance over the season is recorded as an array of scores. Write a program that generates a large array of player scores using numpy, and calculates the mean, median, variance, and standard deviation of the players' performance.

Ans.

```
import numpy as np

def generate_player_scores(num_players, num_games):
    return np.random.randint(0, 101, size=(num_players, num_games))

def calculate_statistics(scores):
    mean_scores = np.mean(scores, axis=1)
    median_scores = np.median(scores, axis=1)
    variance_scores = np.var(scores, axis=1)
    std_dev_scores = np.std(scores, axis=1)

    return mean_scores, median_scores, variance_scores, std_dev_scores

def main():
    num_players = int(input("Enter the number of players: "))
    num_games = int(input("Enter the number of games: "))

    player_scores = generate_player_scores(num_players, num_games)
    mean_scores, median_scores, variance_scores, std_dev_scores =
calculate_statistics(player_scores)

    for i in range(num_players):
        print(f"Player {i + 1}:")
```

```
        print(f"  Mean Score: {mean_scores[i]:.2f}")
        print(f"  Median Score: {median_scores[i]:.2f}")
        print(f"  Variance: {variance_scores[i]:.2f}")
        print(f"  Standard Deviation: {std_dev_scores[i]:.2f}")
        print()

if __name__ == "__main__":
    main()
```

13. <u>Managing Task Lists</u>

A task management system allows users to create and store to-do lists. Write a Python program that stores a user's list of tasks using pickle, allowing them to save and retrieve their tasks later. Ensure proper exception handling if the data file becomes corrupted or is missing.

Ans.

```
import pickle
import os

def save_tasks(task_list, filename='tasks.pkl'):
    try:
        with open(filename, 'wb') as file:
            pickle.dump(task_list, file)
            print("Tasks saved successfully.")
    except Exception as e:
        print(f"Error saving tasks: {e}")

def load_tasks(filename='tasks.pkl'):
    if not os.path.exists(filename):
```

```
        print("No existing task file found")
        return []

    try:
        with open(filename, 'rb') as file:
            return pickle.load(file)
    except (pickle.UnpicklingError, EOFError, FileNotFoundError) as e:
        print(f"Error loading tasks: {e}")
        return []
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
        return []

def display_tasks(task_list):
    if not task_list:
        print("No tasks in your list")
    else:
        print("\nYour tasks:")
        for i, task in enumerate(task_list, start=1):
            print(f"{i}. {task}")

def add_task(task_list):
    task = input("Enter a new task: ").strip()
    if task:
        task_list.append(task)
        print(f"Task '{task}' added.")
    else:
        print("Task cannot be empty.")

def remove_task(task_list):
    display_tasks(task_list)
```

```
    try:
        task_num = int(input("Enter the number of the task to remove: "))
        if 1 <= task_num <= len(task_list):
            removed_task = task_list.pop(task_num - 1)
            print(f"Task '{removed_task}' removed.")
        else:
            print("Invalid task number.")
    except ValueError:
        print("Please enter a valid number.")

def main():
    task_list = load_tasks()

    while True:
        print("\nTask Manager Menu:")
        print("1. View tasks")
        print("2. Add a task")
        print("3. Remove a task")
        print("4. Save and exit")

        choice = input("Choose an option (1-4): ").strip()

        if choice == '1':
            display_tasks(task_list)
        elif choice == '2':
            add_task(task_list)
        elif choice == '3':
            remove_task(task_list)
        elif choice == '4':
            save_tasks(task_list)
            break
```

```
        else:
            print("Invalid choice. Please select a valid option.")

if __name__ == "__main__":
    main()
```

14. <u>Social Media Post Analysis</u>

A social media platform needs to analyze hashtags used in posts. Write a Python program that extracts all unique hashtags from a given post using regular expressions. Ensure that the hashtags only contain letters and numbers (e.g., #Python3) and print them in a sorted list.

Ans.

```
import re

def extract_hashtags(post):
    hashtags = re.findall(r'#\w+', post)
    unique_hashtags = sorted(set(hashtags))
    return unique_hashtags

def main():
    post = input("Enter your social media post: ").strip()
    hashtags = extract_hashtags(post)

    if hashtags:
        print("\nUnique hashtags in the post:")
        for hashtag in hashtags:
            print(hashtag)
    else:
        print("No valid hashtags found in the post.")
```

```
if __name__ == "__main__":
    main()
```