Data Analytics with Python

Assignment 3: Advanced Python Programming

Objective:

By the end of this assignment, you will gain a solid understanding of advanced Python programming concepts, including object-oriented programming, exception handling, and utilizing external libraries for data analysis.

Task 1: Deep Dive into Object-Oriented Programming

Instructions:

- Extend the Employee class from Assignment 2 by adding the following:
  - A method give_raise(self, amount) that increases the employee's salary by the specified amount.
  - A method change_position(self, new_position) that changes the employee's position.
- Create an object employee2 of the Employee class, with the name "John Smith", position "Analyst", and salary 65000.
- Give employee2 a raise of 5000 and change his position to "Senior Analyst". Use the display_info() method to verify the changes.

Expected Output:

- Updated employee details displayed.

Task 2: Handling Exceptions

Instructions:

- Write a function divide_numbers(a, b) that takes two arguments and returns the result of dividing a by b.
- Use a try and except block to handle the ZeroDivisionError. If the error occurs, print "Cannot divide by zero!".
- Test the function with a = 10 and b = 0 to verify that the exception handling works.

Expected Output:

- Error message displayed when attempting to divide by zero.

Task 3: Working with External Libraries (NumPy and Pandas)

Instructions:

- Import the numpy library and create a 2D array matrix with the following values: [[1, 2, 3], [4, 5, 6], [7, 8, 9]].
- Use numpy to calculate and print the sum of all elements in the array.
- Import the pandas library and create a DataFrame df with the following data:

Name	**Age**	**Department**
John	28	HR
Jane	34	IT
Doe	22	Marketing
Smith	45	Sales

- Print the DataFrame and calculate the average age of the employees.

Expected Output:

- Sum of elements in the matrix displayed.
- DataFrame displayed along with the average age.

Task 4: Working with JSON Data

Instructions:

- Create a Python dictionary employee_data with the following structure:

python

Copy code

```
employee_data = {
  "employees": [
    {"name": "John", "position": "Manager", "salary": 80000},
    {"name": "Jane", "position": "Engineer", "salary": 75000},
    {"name": "Doe", "position": "Analyst", "salary": 65000}
  ]
}
```

- Import the json module and convert the dictionary to a JSON string.

- Save the JSON string to a file named employee_data.json.
- Read the JSON data from the file and convert it back to a dictionary. Print the dictionary to verify the content.

Expected Output:

- JSON data saved and read back correctly, with the dictionary content displayed.

Task 5: Advanced Data Handling with Pandas

Instructions:

- Load the employee_data.json file into a Pandas DataFrame.
- Add a new column bonus to the DataFrame, with a bonus of 10% of the salary for each employee.
- Print the updated DataFrame.

Expected Output:

- DataFrame with the bonus column displayed.

Task 6: Visualization with Matplotlib

Instructions:

- Import the matplotlib.pyplot library.
- Create a bar chart that shows the salaries of employees from the df DataFrame created in Task 3.
- Label the axes and give the chart a title "Employee Salaries".
- Display the chart.

Expected Output:

- Bar chart displaying employee salaries.

Task 7: Handling Dates and Times with datetime

Instructions:

- Use the datetime module to create a variable current_time that stores the current date and time.
- Print current_time in the format YYYY-MM-DD HH:MM:SS.
- Calculate the date 10 days from today and print it.

Expected Output:

- Current date and time displayed in the specified format.
- Date 10 days from today displayed.