Algorithm Exercise – MatchJob (Detailed Version)

1. Objective

You will create a Python program that helps a candidate find job offers that match their skills.

This exercise focuses on algorithmic thinking, not UI or databases.

You must write clean, readable Python code that follows clear logical steps.

2. What You Will Learn

- Working with lists and dictionaries
- Writing loops and conditions
- Comparing two lists of data
- Sorting with multiple criteria
- Designing small algorithms step by step

3. Data Description

Candidate: A list of skills.

Example: ['python', 'sql', 'git']

Job Offer:

- a title (string)
- a list of required skills

Example: Backend Junior → ['python', 'git', 'docker']

4. Business Rules (Very Important)

Rule 1 – Compatibility

A job offer is compatible if the candidate has at least TWO required skills from the offer.

If not, ignore the offer completely.

Rule 2 – Score

The score equals the number of skills in common between the candidate and the offer.

Example: candidate ['python','sql','git'] and offer ['python','git'] → score = 2.

Rule 3 – Sorting

Sort compatible offers by:

1. Highest score first

2. If scores are equal → alphabetical order of job title

Rule 4 – Output

For each compatible offer display:

- Job title

- Score

- List of matching skills

5. Example

Candidate: ['python', 'sql', 'git']

Offers:

Backend Junior → ['python', 'git', 'docker']

Data Intern → ['sql', 'excel']

Fullstack → ['python', 'sql', 'react']

Compatible offers: Backend Junior (2), Fullstack (2)

Sorted result:

Backend Junior – score 2 – python, git

Fullstack – score 2 – python, sql

6. Suggested Steps to Solve

Step 1 – Write a function that finds common skills between two lists.

Step 2 – Write a function that checks if an offer is compatible.

Step 3 – Write a function that computes the score.

Step 4 – Filter all compatible offers.

Step 5 – Sort the offers.

Step 6 – Display the result clearly.

7. Bonus Ideas

- Return only the TOP 3 offers
- Ignore uppercase/lowercase differences (Python vs python)
- Add an optional skill list with bonus points
- Add candidate preferred location later

8. Advice from a Software Craftsmanship Trainer ■

Write small functions.

Test your code with simple examples first.

Print intermediate results to understand your algorithm.

Try to write one rule at a time, exactly like Test■Driven Development.