README — Project 77: Internship Applicants Filter

Overview

This README explains Project 77 — Internship Applicants Filter. The project demonstrates basic Python data handling across several types: integers, strings, booleans, lists, arrays, and dictionaries. Each section contains a short description of the task, the approach used, and sample outputs. The goal is to process a small, realistic set of applicant data and present clear, reproducible results.

Prerequisites

You will need Python 3.8+ installed. The project uses only the Python standard library, except for optional tools if you want to export data (none required).

No external packages are strictly necessary. If running the provided .py scripts, ensure you have access to a terminal or an IDE (VS Code, PyCharm, etc.).

Files Included

- `project77_applicants.py` — (suggested) main script that runs all sections and prints results.
- `Project77_Results.docx` — (optional) human-readable report with outputs and explanations.
- `README.docx` — this document.

How to Run

1. Save `project77_applicants.py` in a folder.
2. Open a terminal and navigate to that folder.
3. Run: `python project77_applicants.py`

The script prints screenshot-style outputs for each section. You can edit the sample data inside the script to test different scenarios.

Section Details

1) Integers — Totals and Statistics

Purpose: Work with integer values that represent applicant scores, counts, or quantities.

What to do:
- Provide a list of integer scores (e.g., [85, 90, 72, 88, 95, 60]).
- Compute: total (sum), average, minimum, maximum.

Notes: Average is computed as `total / number_of_items`. The sample output in the project is:
Total: 490
Average: 81.67

Minimum: 60
Maximum: 95

2) Strings — Formatted Report
Purpose: Create human-readable summaries using formatted strings (f-strings).

What to do:
- Use at least two f-strings that summarize totals and averages.
Example f-strings:
f'Internship Applicants Report — Total Applicants: {len(scores)}; Total Score: {total}'
f'Average Score across applicants stands at {average:.2f}.'

These sentences help produce a polished textual report for stakeholders.

3) Booleans — Threshold Check
Purpose: Apply a threshold and decide whether performance is acceptable.

What to do:
- Define a threshold variable (e.g., threshold = 80).
- Use a compound boolean condition to determine status, for example:
`if average > threshold and maximum > 90:` then print 'Above Standard' else 'Below Standard'.

Notes: Compound conditions allow more nuanced checks (e.g., average must be high AND top performer must exceed a value).

4) Lists — Manage Applicant Names
Purpose: Maintain and modify a list of applicant names or items.

What to do:
- Start with a sample list: ['Alice', 'Bob', 'Clara', 'David', 'Eva'].
- Add a new applicant (e.g., 'Frank').
- Remove an applicant by condition (e.g., remove 'Bob' if present).
- Show the list before and after modifications, then sort it and display the sorted list.

Sample output:
Original Applicants: ['Alice', 'Bob', 'Clara', 'David', 'Eva']
After adding 'Frank' and removing 'Bob': ['Alice', 'Clara', 'David', 'Eva', 'Frank']
After sorting: ['Alice', 'Clara', 'David', 'Eva', 'Frank']

5) Arrays — Fixed-size Numeric Subset
Purpose: Demonstrate the `array` module for compact, typed numeric storage.

What to do:

- Create an array.array('i', [85, 90, 72]) representing a fixed subset of scores.
- Compute `sum(array)` and compare it to the full list's total.

Notes: Arrays are useful when you want a typed, memory-efficient structure. In this project the array sum is 247 while the full list total is 490.

6) Dictionaries — Records and Updates
Purpose: Store applicant records as dictionaries and perform updates.

What to do:
- Create a list of dictionaries, each with `id`, `name`, and `value`.
- Update one record (e.g., change Clara's score from 72 to 80).
- Remove another record (e.g., delete Bob).
- Compute the total of the `value` field across remaining records.

Sample result:
[{'id':1, 'name':'Alice', 'value':85}, {'id':3, 'name':'Clara', 'value':80}]
Total of values: 165