# Homework 3: Fortune Cookie Jar

With this digital fortune cookie jar, the user can:

- Enter participant names
- The jar will assign a random fortune to each new name
- The jar continues until the user exits


**Example**:

- The user opens the jar
- The program prompts for a name (or a comma-separated list)
- The user enters: `Ava, Ben`
- The jar assigns two distinct fortunes - one to Ava, one to Ben
- The user can type `list` to see current assignments, or add more names; the session ends when the user chooses to exit

---

## Instructions

In this assignment, you will implement a **`FortuneCookieJar`** class with the following methods:

1. **`__init__(self, fortunes)`**
   Initialize a new **`FortuneCookieJar`** object.
   - Set the attribute **`fortune_slips`** to the **`fortunes`** argument. This is a list of possible fortunes a user could receive.
   - Set the attribute **`name_roster`** to an empty list to store the names in the order they were assigned a fortune.
   - Set the attribute **`dealt_indices`** to an empty list to store the indices of the fortunes already dealt to unique names.

2. **`__str__(self)`**
   Controls the string representation of **`FortuneCookieJar`**.
   - Return a single string with all fortunes in **`fortune_slips`**, joined by dashes (-).
   - If **`fortune_slips`** is empty, return an empty string `""`.

3. **`assign_fortune(self, name)`**
   Assign a fortune for **`name`**.
   - If **`name`** already exists in **`name_roster:`**
     - Find its position `pos`, fetch `fortune_slips[dealt_indices[pos]]`, and return: `"That name already has a fortune: <fortune>"`
     - ***Note:*** Do not modify **`name_roster`** or **`dealt_indices`**.
   - If **`name`** is new:
     - Build a list of available indices by iterating from `0` → `len(fortune_slips)-1` and excluding the values that already exist in `dealt_indices`.
     - If no indices remain, return:
       `"The jar is empty—no fortunes left to assign."`
       Do not change any lists.
   - Otherwise, choose a random available index (use Python's **`random`** module):
     - Append **`name`** to **`name_roster`**
     - Append the chosen index to **`dealt_indices`**
     - Return the assigned fortune as `"<fortune>"`

4. **`distribute_session(self)`**
   Control the interactive loop with the **`FortuneCookieJar`**.
   - On a new session, prompt exactly:
     `"Turn 1 - Enter a name (or a comma-separated list), or type 'list' or 'Done': "`
   - If user input is exactly **"`Done`"** (no other capitalization):
     - Print `"Goodbye! See you soon."` and stop prompting.
   - If user input is exactly **"`list`"** (no other capitalization):
     - Print each current assignment in order of **`name_roster`** as:
       `"<name>: <fortune>"`
   - Otherwise, treat input as names:
     - Split the line by comma, trim spaces for each entry, ignore any empty entries.
     - For each **`name`**, call **`assign_fortune(name)`** and print the returned string.
     - If the jar runs out of fortunes mid-line, remaining names should print the "jar is empty" message.
   - After handling the input, prompt the next turn with the updated number:
     - `"Turn <turn_number> - Enter a name (or a comma-separated list), or type 'list' or 'Done': "`
     - ***Tip:*** You could increment a counter in each loop, or compute `len(name_roster)+1`.

5. **`tally_distribution(self)`**
   Summarize fortune usage across unique names.
   - Create a frequency list of length **`len(fortune_slips)`** initialized to zeros.
   - For each index in **`dealt_indices`**, increment the corresponding frequency.
   - Build and return a list of strings:
     `"<number_of_times> - <fortune>"` where `<fortune>` is in *lowercase*.
   - Sort the returned list by `<number_of_times>` in descending order.
     (Ties should be in alphabetical order.)
   - If **`dealt_indices`** is empty, print `"Empty"` and return `[]`.

6. **`main()`**
   - Create a **`FortuneCookieJar`** object with a **`fortune_slips`** list such as:
     - Follow Your Inner Voice
     - Opportunity Knocks Softly
     - Trust the Process
     - Ask for Help
     - Change is Coming
     - Enjoy the Little Things
   - Start the interaction by calling **`distribute_session()`**
   - After exit, display the output of **`tally_distribution()`** in the terminal

---

## Sample Output

- Immediate exit: If the user types `Done` on Turn 1, `dealt_indices` is empty: program prints `"Empty"` and **`tally_distribution()`** returns `[]`.
- Duplicate name: Re-entering an existing name re-reports its fortune.
- No duplicates per session: Each new name receives a different fortune until the pool is empty; afterward, new names get the “jar is empty” message.
- `list` command: Prints `<name>: <fortune>` lines in the order names were assigned (using **`name_roster`** and **`dealt_indices`** only).

---

## Grading Rubric: 60 Points Total

- 5 pts: **`__init__`** correctly initializes **`fortune_slips`**, **`name_roster`**, and **`dealt_indices`**
- 5 pts: **`__str__`**

    - 3 pts: Joins non-empty **`fortune_slips`** with dashes
    - 2 pts: Returns `""` when **`fortune_slips`** is empty
- 5 pts: **`assign_fortune`** correctly re-reports an already assigned name
- 5 pts: **`assign_fortune`** selects from remaining fortune indices and appends the chosen index for new names
- 5 pts: **`assign_fortune`** appends new names to **`name_roster`** (kept aligned with **`dealt_indices`**)
- 5 pts: **`distribute_session`** starts with the specified prompt format
- 5 pts: **`distribute_session`** correctly implements comma-separated names
- 5 pts: **`distribute_session`** correctly implements the `list` command
- 5 pts: **`tally_distribution`** returns a formatted list with fortunes in lowercase
- 5 pts: **`tally_distribution`** sorts by frequency in descending order
- 2 pts: **`fortune_slips`** is properly defined and used in **`main()`**
- 2 pts: **`FortuneCookieJar`** is properly constructed and used in **`main()`**
- 2 pts: **`distribute_session`** is called correctly in **`main()`**
- 2 pts: **`tally_distribution`** is called and its output is displayed in **`main()`**

---

## Extra Credit: 6 Points

Create a **`jar_sanity_check()`** function to test outcomes.

- 1 pt: Correct **`tally_distribution`** behavior when **`dealt_indices`** is empty
- 2 pts: With
    - `fortune_slips = ['trust the process', 'ask for help', 'enjoy the little things']`
    - `dealt_indices = [2, 1, 2]`

      confirm counts and sort order
- 1 pt: Correct initial prompt from **`distribute_session()`**
- 1 pt: Correct re-report behavior from **`assign_fortune()`** for an existing name
- 1 pt: Confirm that comma-separated inputs assign fortunes without duplication and that list prints assignments in order

---

## Running Your Code

If you are having trouble running your code in VS Code, click the dropdown arrow in the top right corner of your VS Code window. Then, press “Run Python File.”

## Submission Instructions

Follow the instructions on Canvas to submit your git repo link by the due date and time.