1. The Smart Attendance Logger

The Goal: Manage a classroom attendance list and ensure no duplicate entries are recorded if a student taps their ID card twice.

- **Concepts:** `set`, `function`, `while`.
- **The Problem:** Create a function `mark_attendance()` that keeps asking for student names until the user types "STOP". Store names in a **Set** to automatically prevent duplicates.
- **Input:** Multiple strings (names) entered one by one.
- **Output:** The final count of unique students and the set itself.
- **Hint:**

> ***Expected Flow:*** *Start an empty* `set` *Use a* `while True` *loop to take* `input()` `if` *name is "STOP",* `break` `else` *add name to set Finally, print* `len(attendance_set)`.

---

2. The Restaurant Bill Splitter

The Goal: Calculate how much each person owes, including a tax and a custom tip percentage.

- **Concepts:** `functions`, `default values`, `list`.
- **The Problem:** Write a function `split_bill(items_prices, people, tax=0.05, tip=0.10)`. `items_prices` will be a **List** of decimals (floats).
- **Input:** A list like `[25.0, 15.0, 10.0]` and an integer for `people`.
- **Output:** A single float (the amount per person).
- **Hint:**

> ***Expected Flow:*** *Calculate* `sum()` *of the list Add tax and tip amounts Divide the grand total by* `people` `return` *the result.*

---

3. The Digital Wardrobe Organizer

The Goal: Check if a specific outfit is available or if parts of it are missing.

- **Concepts:** `multiple if`, `list`, `input`.
- **The Problem:** You have a list `wardrobe = ["Blue Jeans", "White Shirt", "Black Jacket"]`. Ask the user for 3 items they want to wear. Check each item individually against the list.
- **Input:** Three separate `input()` strings (e.g., "Red Tie", "White Shirt", "Socks").
- **Output:** A "Missing" warning for every item not found in the list.
- **Hint:**

> ***Expected Flow:*** *Take 3 inputs Use three* ***separate*** `if` *statements (not* `elif`*) to check if each input is* `not in` *the* `wardrobe` *list Print "Missing [Item]" for each failure.*

---

4. The E-Commerce Discount Engine

The Goal: Apply different discount rates based on the total purchase amount.

- **Concepts:** `if-elif-else`, `functions`.
- **The Problem:** Create a function `apply_discount(total)`.
- If total > $500: 20% off.
- If total > $200: 10% off.

- Else: No discount.
- **Input:** A single number (the total bill).
- **Output:** The final price after the discount is subtracted.
- **Hint:**

> ***Expected Flow:*** *Receive the* `total` *Use an* `if-elif-else` *chain to determine the discount percentage Calculate the new price* `return` *it.*

---

5. The Contact Book Search

The Goal: Look up a phone number using a person's name.

- **Concepts:** `dict` , `input` , `if-else` .
- **The Problem:** Create a **Dictionary** of 5 friends (Name: Number). Ask the user for a name to search.
- **Input:** A string (Name).
- **Output:** The phone number if found, or "Contact not found".
- **Hint:**

> ***Expected Flow:*** *Define the* `dict` *Get* `input()` *Use* `if name in phonebook:` *to check for the key Print the value (number) or the error message.*

---

6. The Travel Itinerary Builder

The Goal: Print a numbered list of travel destinations for a user.

- **Concepts:** `for` loop, `list` , `enumerate` .
- **The Problem:** Given a list `cities = ["Tokyo", "Paris", "New York", "London"]` , print them as a numbered list.
- **Input:** None (Uses the hardcoded list).
- **Output:** A vertically formatted list (e.g., "1. Tokyo").
- **Hint:**

> ***Expected Flow:*** *Use a* `for` *loop with* `enumerate(cities, start=1)` *Print the index and the city name on the same line.*

---

7. The ATM Pin Validator

The Goal: Lock the user out after 3 incorrect attempts.

- **Concepts:** `while` , `if-else` , `break` .
- **The Problem:** Set `correct_pin = "1234"` . Allow the user 3 tries to enter the right PIN.
- **Input:** Up to 3 string inputs.
- **Output:** "Welcome" on success, or "Card Blocked" after the 3rd fail.
- **Hint:**

> ***Expected Flow:*** *Set* `attempts = 0` *While* `attempts < 3` *: Take PIN input* `if` *correct, print "Welcome" and* `break` `else` *increment* `attempts` *. If* `attempts` *reaches 3, print "Card Blocked."*

---

8. The Movie Ticket Age Checker

The Goal: Categorize a movie-goer and assign a price.

- **Concepts:** `tuple` , `if-elif-else` .
- **The Problem:** Use a **Tuple** to store fixed prices: `PRICES = (0, 10, 15)` (Free, Child, Adult). Ask for age and determine which index from the tuple to use.
- **Input:** An integer (Age).
- **Output:** The price fetched from the tuple.
- **Hint:**

> ***Expected Flow:*** *Get age Use* `if-elif-else` *to decide the index (0, 1, or 2) Print the value from* `PRICES[index]` .

---

9. The Grocery Inventory Update

The Goal: Update the quantity of items in a store after a shipment arrives.

- **Concepts:** `dict` , `for` loop.
- **The Problem:** Given `inventory = {"Apples": 50, "Bananas": 20}` , a shipment of 10 more of each arrives. Update the dictionary using a loop.
- **Input:** None.
- **Output:** The updated dictionary.
- **Hint:**

> ***Expected Flow:*** *Loop through* `inventory.keys()` *For each item, add 10 to the current value Print the dictionary after the loop finishes.*

---

10. The Flexible Event Invitation

The Goal: Create a function that can accept any number of guests and specific event details.

- **Concepts:** `functions` , `*args` , `**kwargs` .
- **The Problem:** Write `create_invite(event_name, *guest_names, **details)` .
- **Input:** One string, followed by multiple guest names, followed by named details (e.g., `Time="6pm"`).
- **Output:** A printed invitation showing the event, guests, and details.
- **Hint:**

> ***Expected Flow:*** *Print the* `event_name` *Use a* `for` *loop to iterate through* `*args` *(guests) Use* `.items()` *to loop through* `**details` *(extra info) and print them.*
