1. Interactive Challenge

Your Turn!

- Define an enum `Role` for staff roles (Doctor, Nurse, Admin).
- Create an interface `Staff` with fields for `id`, `name`, and `role`.
- Create an array of staff members using the interface and enum.
- Write a function that prints a summary of all staff, showing their name and role.

2. Interactive Challenge

Your Turn!

- Define a type `Profile` with `username` (string), `bio` (string or null), and optional `avatarUrl` (string).
- Create two profiles: one with a null bio and no avatar, and one with both fields set.
- Write a function `showProfile` that prints the username, a default message if bio is null, and a default avatar if `avatarUrl` is undefined.

3. Interactive Challenge

Your Turn!

1. **Define** a `CustomerID` alias for `string`.
2. **Create** a `Customer` object alias with `id: CustomerID`, `name: string`, and optional `email?: string`.
3. **Implement** a `processOrder` function type alias that accepts `orderId: number` and a callback `(status: OrderStatus) => void`.
4. **Use** the `Container<T>` generic to wrap a `Customer` object.

4. Interactive Challenge / Mini-Project

Implement four small functions to practice each decision-making construct:

1. **`checkSign(num: number): void`**
   Use an **`if`** statement to log whether `num` is positive.

2. **`evenOrOdd(num: number): void`**
   Use an **`if…else`** to log whether `num` is even or odd.

3. **`getGrade(score: number): string`**
   Use an **`if…else if…else`** ladder to return a letter grade:

   - `score ≥90 → "A"`
   - `score ≥80 → "B"`
   - `score ≥70 → "C"`
   - `score ≥60 → "D"`
   - otherwise `"F"`

4. **`provideFeedback(grade: string): void`**
   Use a **`switch`** to log a feedback message for each grade (`"A"`… `"F"`), with a `default` for any unexpected value.

5. Interactive Challenge / Mini-Project

Your Turn!

1. Add a counter for each transaction type (`checkout` , `return` , `priority` , `cancelled`) using a **`for`** loop and an object.

2. Use a **`while(true)`** infinite loop with a **`break`** condition when a new priority transaction arrives.

3. Modify the **`do…while`** loop to handle a dynamic queue (an array you can `push` new returns into).

4. Use **`for…in`** to reset all inventory counts to zero.

5. Display visitor names in reverse order using a **`for`** or **`for…of`** loop.

6. Interactive Challenge

Your Turn!

1. Call `displayMember` for two members: one with email, one without.

2. Use `calculateFines` to sum fines: 5, 10, 2.5.

3. Compute a membership fee for $100 with default discount, then with 20%.

4. Greet visitors “Alice” and “Bob” using both `vipGreet` and `consoleGreet`.

5. Compute `factorial(5)`.

6. Generate a text report and a JSON report for an array of sample objects (e.g., `{ title: "1984" }`).

7. Interactive Challenge / Mini-Project

Your Turn!

1. **`describePerson`**
    - Required: `name: string`
    - Optional: `age?: number`
    - Print `"Name: <name>, Age: <age>"` or `"Name: <name>, Age: Unknown"`.
2. **`calculatePrice`**
    - Required: `basePrice: number`
    - Default: `discount: number = 0.1`
    - Return price after discount.

3. Test calls:

```
describePerson("Eve");
describePerson("Frank", 28);
console.log(calculatePrice(100));      // 90
console.log(calculatePrice(100, 0.2)); // 80
```