# Angular Observable, Subscribe, RxJS Operators & Signals - Cheat Sheet

## 1. What is an Observable?

An Observable is like a stream of data that can be observed over time. It is often used for HTTP calls, event handling, and user interactions in Angular.

## 2. What is subscribe()?

subscribe() is a method used to listen to the Observable. It takes up to three callback methods:  
- next(): When data is received  
- error(): When an error occurs  
- complete(): When the stream completes

### Example:

this.bookService.getAllBooks().subscribe({  
 next: (data) => console.log('Data:', data),  
 error: (err) => console.error('Error:', err),  
 complete: () => console.log('Done!')  
});

## 3. Common RxJS Operators

- map(): Transform data (e.g., change a book title)  
- filter(): Show only certain items (e.g., price > 100)  
- tap(): Perform side effects like logging  
- catchError(): Handle errors gracefully  
- switchMap(): Cancel previous request and use the latest  
- mergeMap(): Run multiple requests in parallel  
- concatMap(): Run requests sequentially

### Example with pipe():

this.bookService.getAllBooks()  
 .pipe(  
 tap(() => console.log('Fetching...')),  
 map(data => data.filter(book => book.price > 100)),  
 catchError(error => {  
 this.errorMessage = 'Something went wrong';  
 return of([]);  
 })  
 )  
 .subscribe(data => this.books = data);

## 4. What are Signals?

Signals are a reactive primitive in Angular (v16+) used to store and track state that automatically updates the UI. They are an alternative to local variables and subscriptions.

### Signal Example:

count = signal(0);  
increment() {  
 this.count.set(this.count() + 1);  
}

### HTML:

<button (click)="increment()">Click Me</button>  
<p>You clicked {{ count() }} times.</p>