* Exploring Angular Forms
  + Forms in Angular
    - Note that the **mat-form-field > input** element that we use to allow users to filter data is a way for a user to interact
    - But, if you look at the generated HTML, you will only see an **input** element (no **form** element)
    - This is because you don’t need form data to submit this request; it is done via an async JS XMLHttpRequest (XHR) (i.e. AJAX)
    - This doesn’t require a form container to handle data encoding and transmission
  + Reasons to Use Forms
    - Even if we don’t need a form to handle encoding/transmission, there are other advantages
    - Specifically, there are disadvantages to using a single input element and a text string
      * Can’t keep track of global form state
      * Can’t easily display an error message to users if a form is invalid
      * We are not validating the data in any way
    - Note that we could add these features manually in our Components and Angular directives like \*ngIf and \*ngFor, but there are easier ways
    - Specifically, Template-Driven Forms and Client-Driven Forms (aka Reactive Forms)
  + Template-Driven Forms
    - These forms are defined mainly in the Template for your component (big surprise)
    - They involve defining a form element and using the ngForm module
    - You define inputs in the form and define attributes (per normal) and use the [()] two-way property binding syntax to ensure the forms are responsive
    - Note that [(ngModel)]=”city.Name” is really Angular shorthand for [ngModel]=”city.Name” and (ngModelChanged)=”city.Name = $event”
    - Pros
      * These forms are easy to write (especially from an HTML knowledge standpoint)
      * These forms are easy to read and understand if you have a decent background in HTML
    - Cons
      * These require a lot of HTML which is difficult to maintain and more error-prone than pure TS
      * These forms cannot be unit tested; they require E2E tests with a browser to e.g. verify the validation logic
      * Readability will quickly drop as you add more and more validation and logic
    - Generally, Template-Driven is a good option when the forms are small and the validation logic is light
  + Model-Driven/Reactive Forms