# Design patterns in 3rd party libraries

## Angular: (Flux)

- Observable Design Pattern:
  - Observable design pattern is used almost everywhere when working with Angluar and Rxjs libraries. The observables are used in offering a direct flow of data from the javascript/typescript logic to the html components, allowing the display to change dynamically.
  - Link to documentation: https://rxjs-dev.firebaseapp.com/guide/observable
  - Link to design pattern: https://www.tutorialspoint.com/design_pattern/observer_pattern.htm

- Iterator Design Pattern:
  - Iterator design pattern is commonly used when iterating over objects in different types of collections. This can be found in equal amounts everywhere in every technology that is based on Object Oriented Programming.
  - Link to design pattern: https://www.tutorialspoint.com/design_pattern/iterator_pattern.htm

- State Design Pattern:
  - State design pattern is commonly used in Angular for saving values from one instance of the User Interface to another. The state usually contains the most relevant and valuable data from which the typescript+HTML can generate most of the content of the UI.
  - Link to documentation: https://ngrx.io/guide/store
  - Link to design pattern : https://www.tutorialspoint.com/design_pattern/state_pattern.htm

- Template Design Pattern:
  - Templates are commonly used when working with class type objects. Such examples would vary from classic arrays to the commonly used in angular Subjects(or Behaviour Subjects)
  - Example of a component based on template: https://www.learnrxjs.io/learn-rxjs/subjects/behaviorsubject
  - Link to documentation: https://www.tutorialspoint.com/design_pattern/template_pattern.htm

- Chain of Responsability Design Pattern:
    - When working with RXJS Observables, most of the time pipes are a necessity for processing data from the selected sources. In those pipes there are functions such as ‘switchMap’, which transfer the responsibility from the previous sources to the newly created source of data from the switchMap. The switchmap takes as input the previous stream and returns a new stream, decoupling the previous stream from the afterward logic.
    - Link to a functionality example that uses chain of responsibility: https://www.learnrxjs.io/learn-rxjs/operators/transformation/switchmap
    - Link to design pattern: https://www.geeksforgeeks.org/chain-responsibility-design-pattern/

# Django: (Model View Template)

- Template Design Pattern
  - Same as described in the Angular section above, most if not all Object Oriented Programming programming languages use the Template Design Pattern to implement flexibility in their data structures.
  - Link to the design pattern: https://www.tutorialspoint.com/design_pattern/template_pattern.htm
- Observer
  - Signals is a component from Django framework, based on Observer pattern that allows decoupled applications to get notified when certain events occur.
- Decorators
  - In django, there are multiple use cases where decorators are used, from which the common are the http decorators that can restrict access to views based on the request method.
  - Link to the documentation: https://docs.djangoproject.com/en/3.1/topics/http/decorators/
  - Link to design pattern: https://www.tutorialspoint.com/design_pattern/decorator_pattern.htm

# Pytorch: ( Iterative programming )