1. Intent:

* An iterator is a behavioral design pattern that allows you to traverse the elements of a collection without exposing its underlying representation (such as a list, stack, or tree).
* "To provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation." (Gang of Four)

1. Motivation\_1:

* Imagine building a website for a store
* There will be different types of goods, hence, different ways to store the data
* For example:
  + Books are stored by genre
  + Clothes are stored by placing them in an array (for example, Clothes clothes[n] to store clothes that always have a size, type, and materials, and can be the same as others). E.g., Jacket 1 has the same size S as Jacket 2
  + Electronics have more things to deal with, and usually they are unique, so a hash map is a better choice for fast looking.
* Because of that, if users use a naive solution, they need to know exactly what the program looks like, to at least call the products correctly
* With an iterator pattern, they just need to ask the program to print itself

1. Motivation\_2: Send message

* Imagine that you are managing a server, more specifically, Meta, which is the home to Facebook, Instagram, and WhatsApp
* But the users' data is stored differently in these applications
* Facebook users are stored in a stack
* Instagram users are stored in a vector
* TikTok (WhatsApp) users are stored in an unordered\_map
* There is no specific reason for this choice, but it is rather an example that the data might be stored differently
* So what if the server wants to say hello to every user from those social networks?
* They can do it based on the data type of the user, with 3 different for loops to send the messages
* But there is a better way to do it, with an iterator pattern, we can send messages to them with the same syntax, or generally, we said that the iterator pattern allows you to traverse all of them uniformly

1. Applicability:
   1. Complex data structure:

* Hide complexity from clients
* Just show the simple methods of accessing the collection elements
  1. Traverse data in multiple ways
* Reduce time and effort for a data structure
* Making the code cleaner and leaner
  1. Traverse different types of data
* As an example, the iterator pattern provides a solution to traverse uniformly
* Still works if new, various kinds are added to these interfaces

https://refactoring.guru/design-patterns/iterator