1) Defining collection pipeline
   In this part author wrote general concepts of collection pipelines. Simply defined the data types that the functions operating on. Also, it shows us the concept of functional programming is like a pipeline. So the data just pass through these pipelines and these pipelines are just like the real ones, means we can connect them.
2) Exploring more pipelines and operations
   In this part author gives some examples of lambda expressions and functional programming. These examples are mostly in Ruby or time to time in Clojure. These examles contains; map, groupBy and a combined operation of multiple pipeline operations.
3) Alternatives
   In this part author discuss the alternatives of functional programming. Simply we can define almost all the operations by just using OOP concepts. However, it is generally more simple and clear if we do it by FP. Some of these methods are; loops and comprehensions.
4) Nested Operator Expressions
   Here we mainly see that these pipeline operations can be used as nested operations.
5) Laziness
   In this part we mainly learn about the efficiency of functional programming. Some times, functions no need to operate on all the elements, in this case, we just operate the necessary part of the data. This can be detected at the beginging of pipline so that we can prevent the overload or bottleneck.
6) Parallelisim
   We see that pipelines usually work with parallel invocations. So that it is mostly not problem to work with multiple cores.
7) Immutability
   Immutability is an important concept to prevent the side efects in programming and have a secure multithreading programs. We see that pipelines usually works with immutable objects. However, it is also ok if we use the mutable objects too.
8) Debugging
   Some techniques about debugging in pipelines. I believe print statements at the middle of pipeline would be the easiest and most clear way to do that.
9) When to use it
   We learn that pipelines are mostly good in the situation where we have a single collection input and a single output. We can use it in other positions but we have keep watch over the usage of it to harness the advantages of functional programming efficiently.