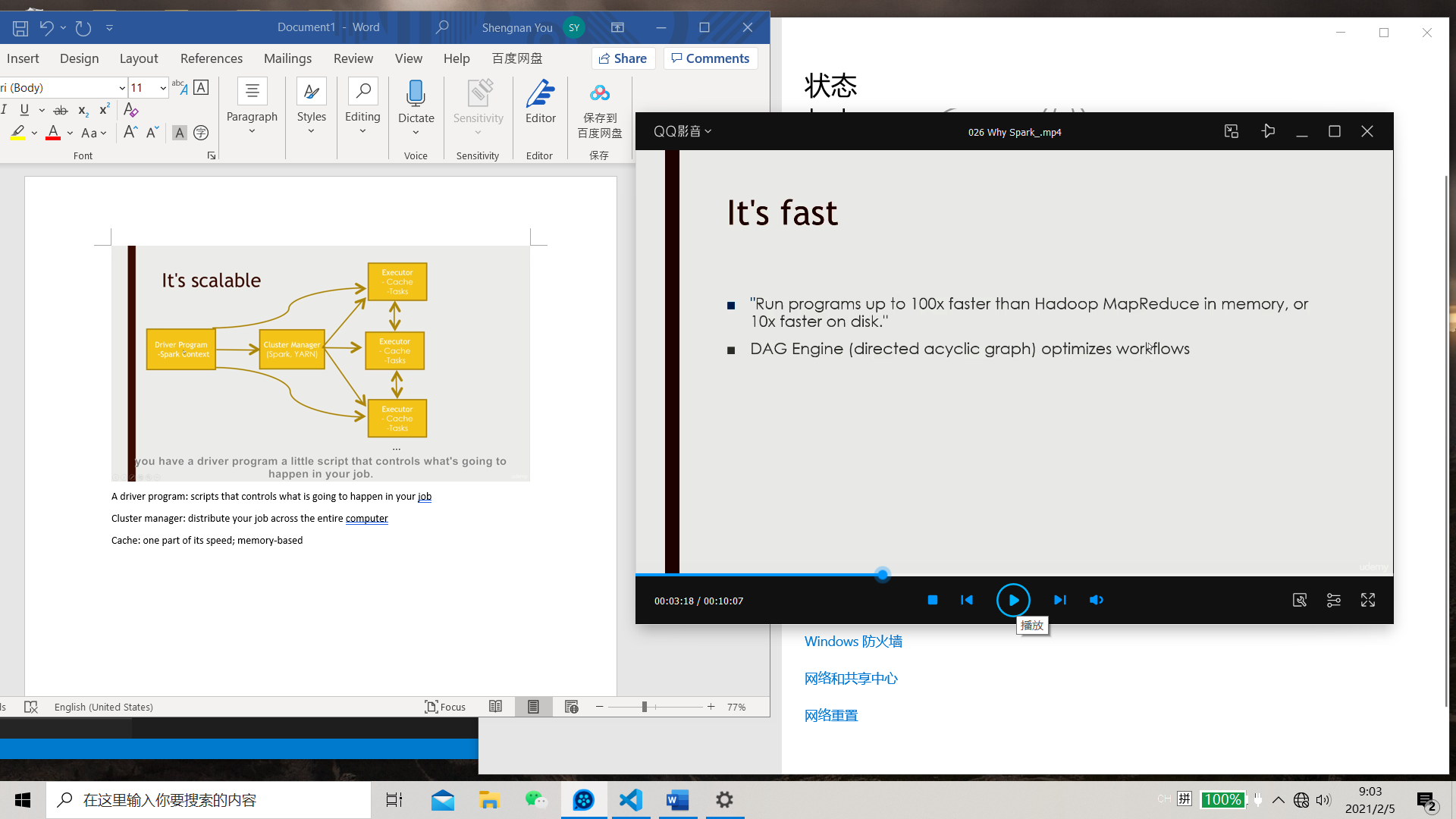


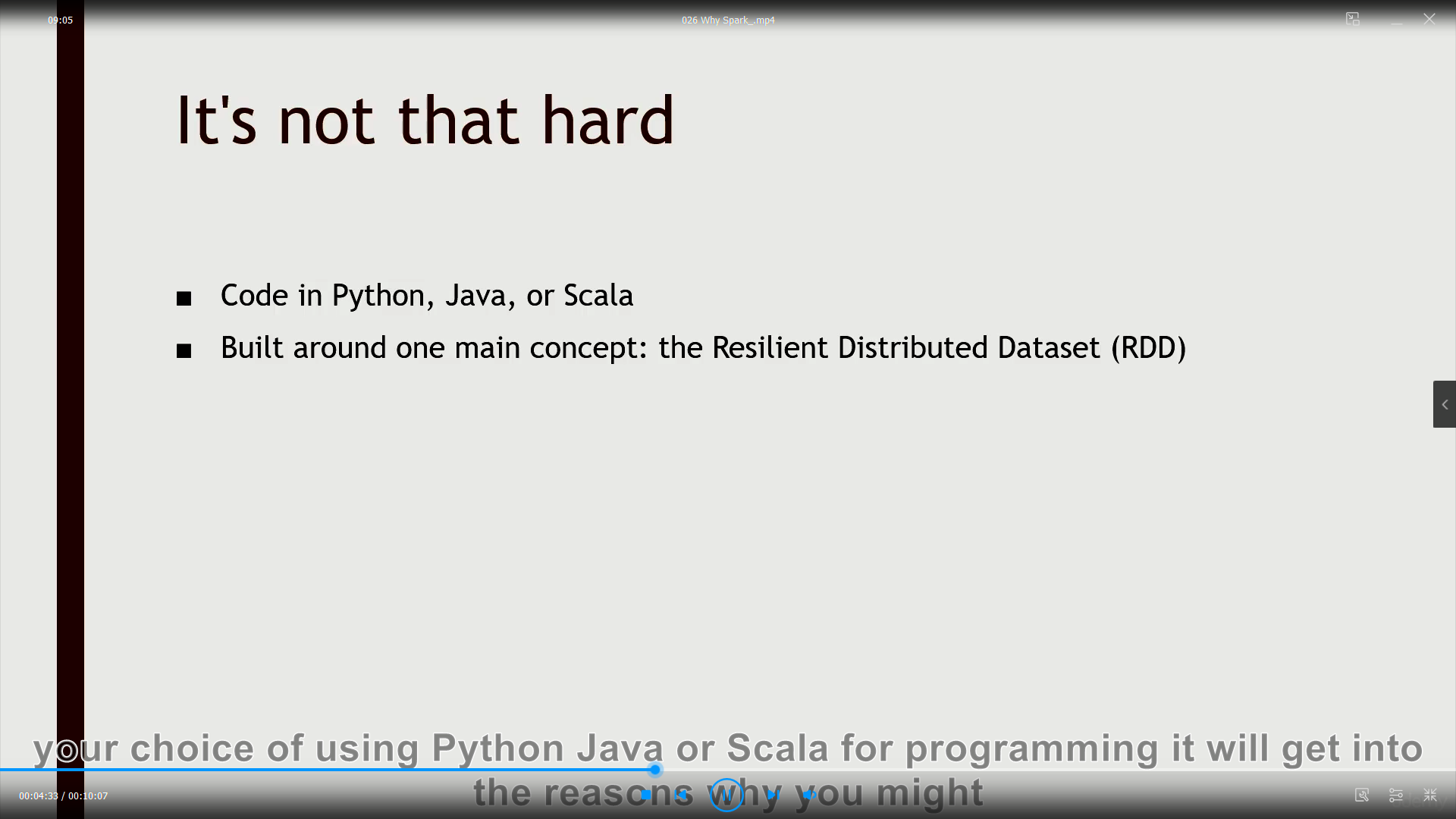
A driver program: scripts that controls what is going to happen in your job

Cluster manager: distribute your job across the entire computer

Cache: one part of its speed; memory-based



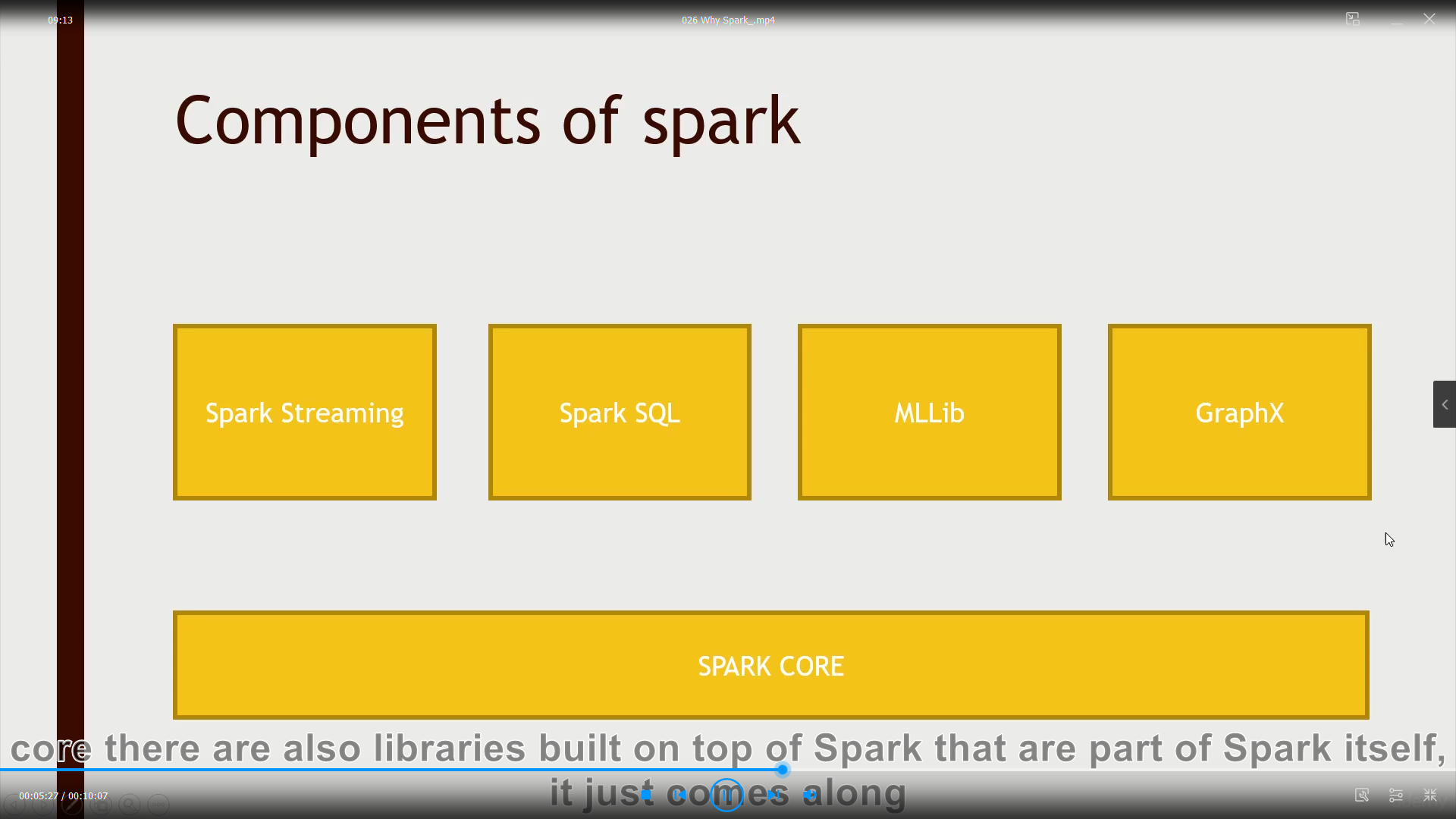
Directed acyclic graph: work backwards from the end of the result and figure out the fastest way to do it



An object that represents a data set.

There are various functions that you can call on that RDD object to transform it or to reduce it or analyze it

Spark 2 they have a data set built on RDD and is sql-focused

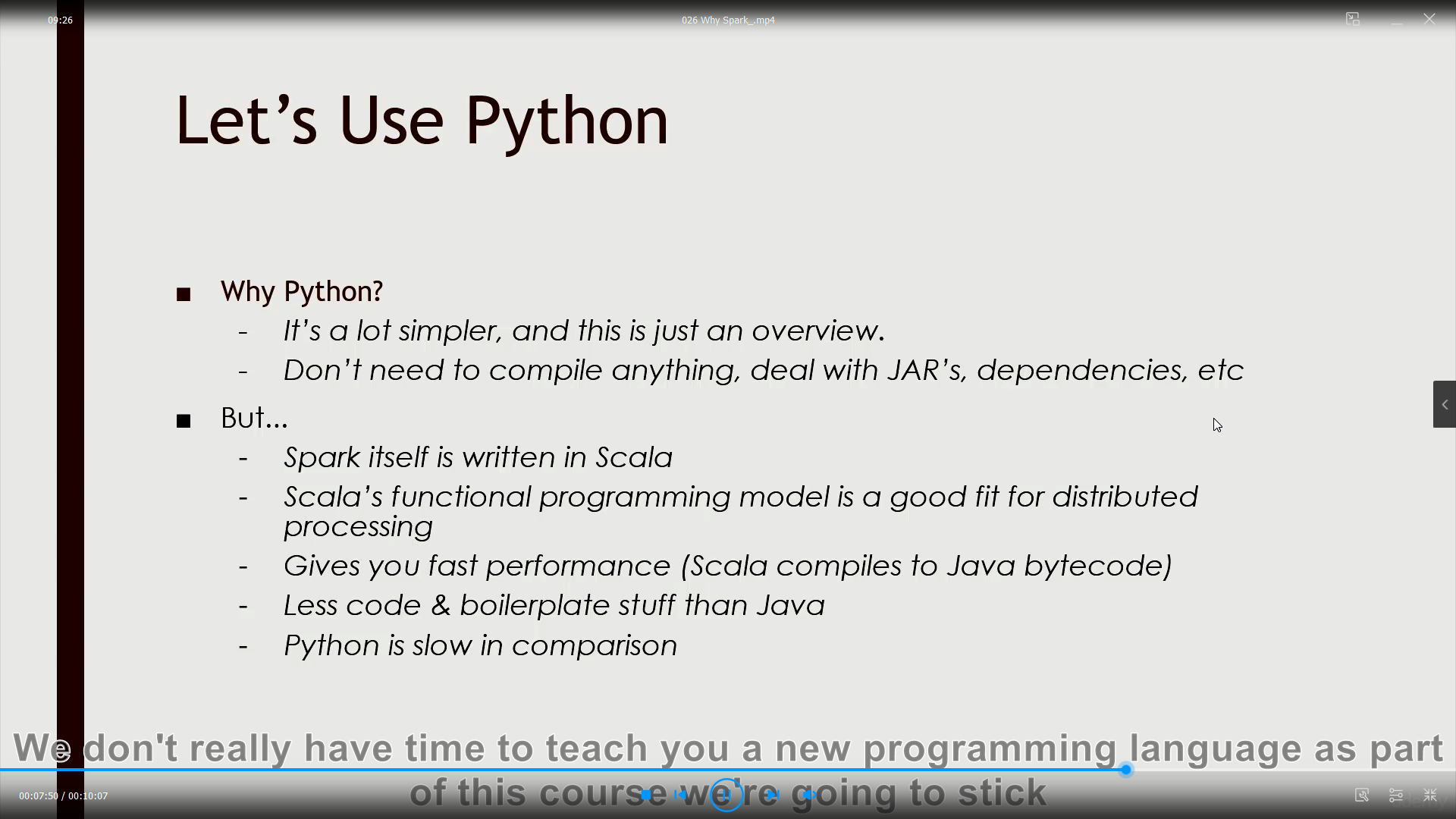


Spark streaming: Instead of doing batch processing of data you can actually input data in real time. A fleet of web servers producing web logs, then that data can be ingested and produced. The you can analyze it and output that result to database or some NoSQl data store

Spark sql: sql interface, write sql query or sql-like functions (Spark2.0)

Mllib: entire machine learning and data mining tool that you can run on datasets.

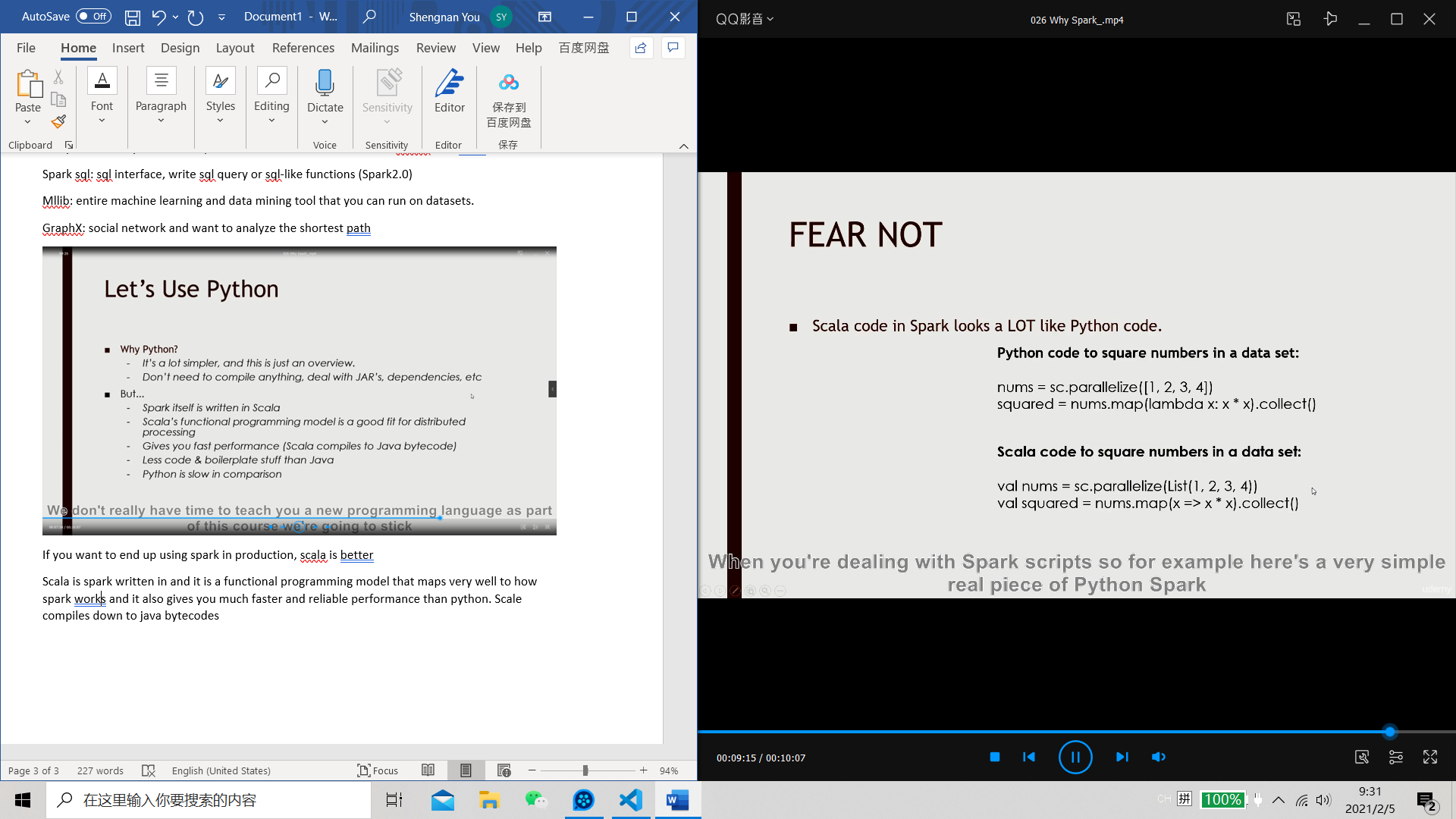
GraphX: social network and want to analyze the shortest path

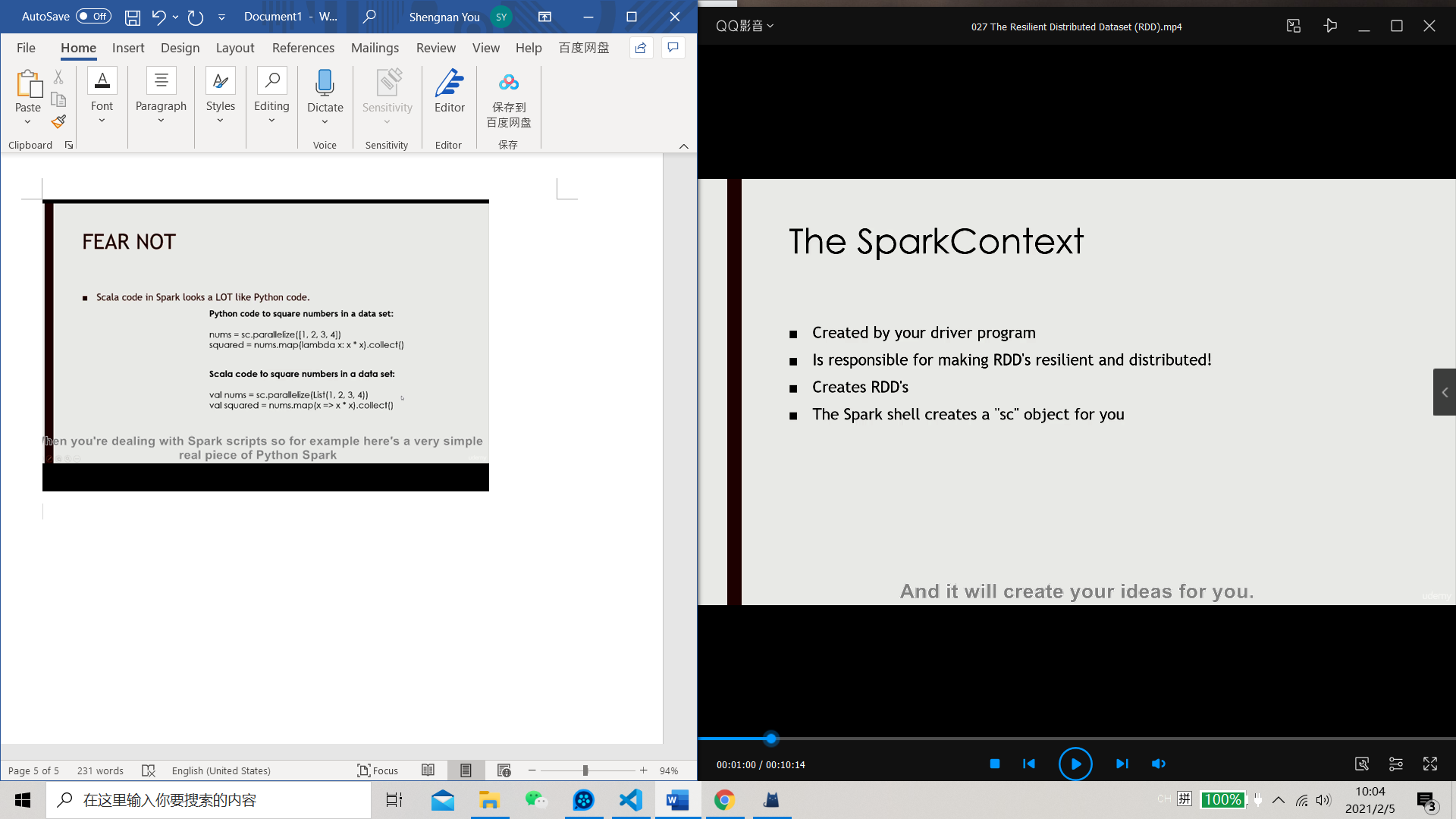


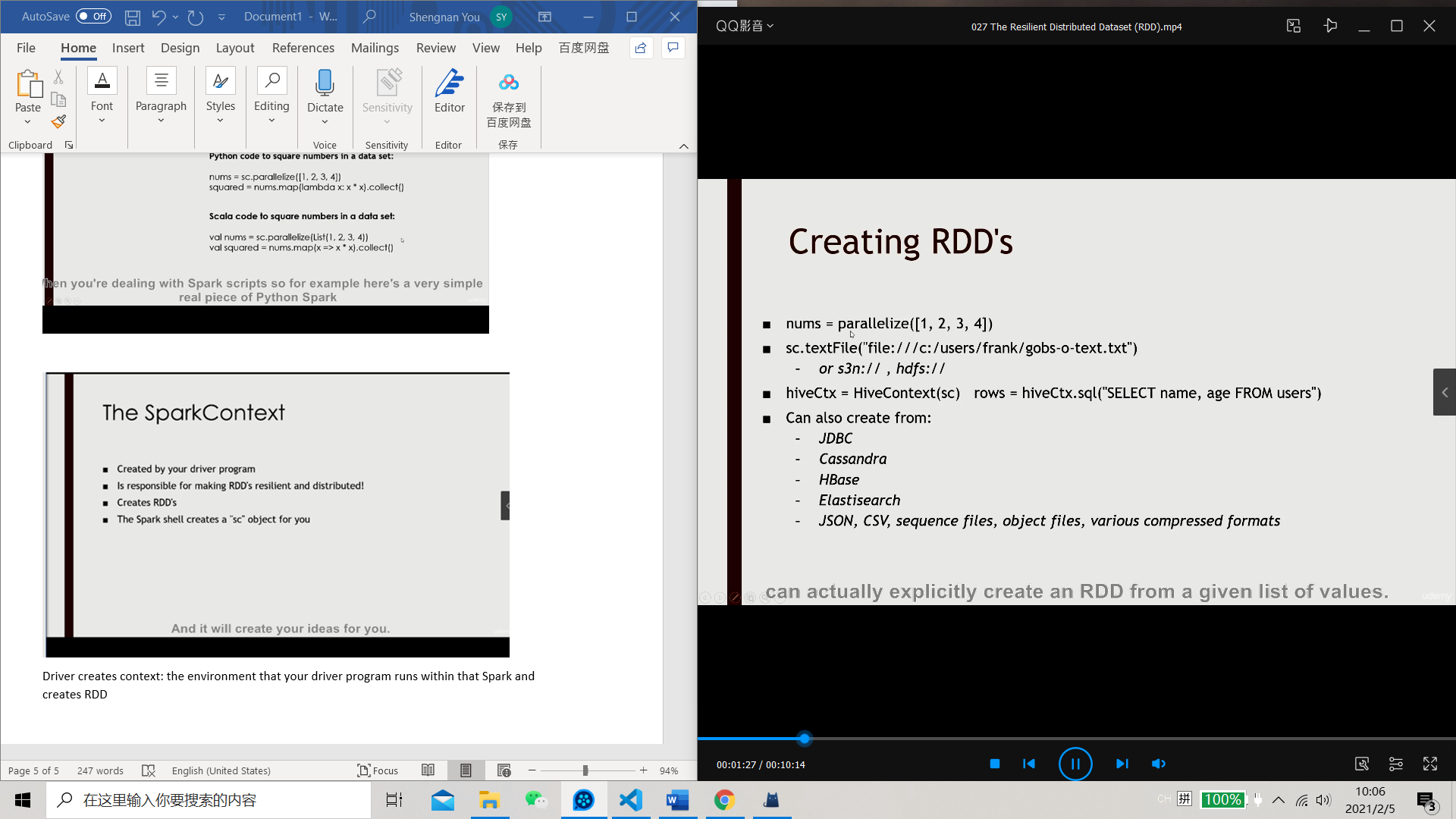
If you want to end up using spark in production, scala is better

Scala is spark written in and it is a functional programming model that maps very well to how spark works and it also gives you much faster and reliable performance than python. Scale compiles down to java bytecodes

RDD: resilient distributed dataset







data = [1, 2, 3, 4, 5]

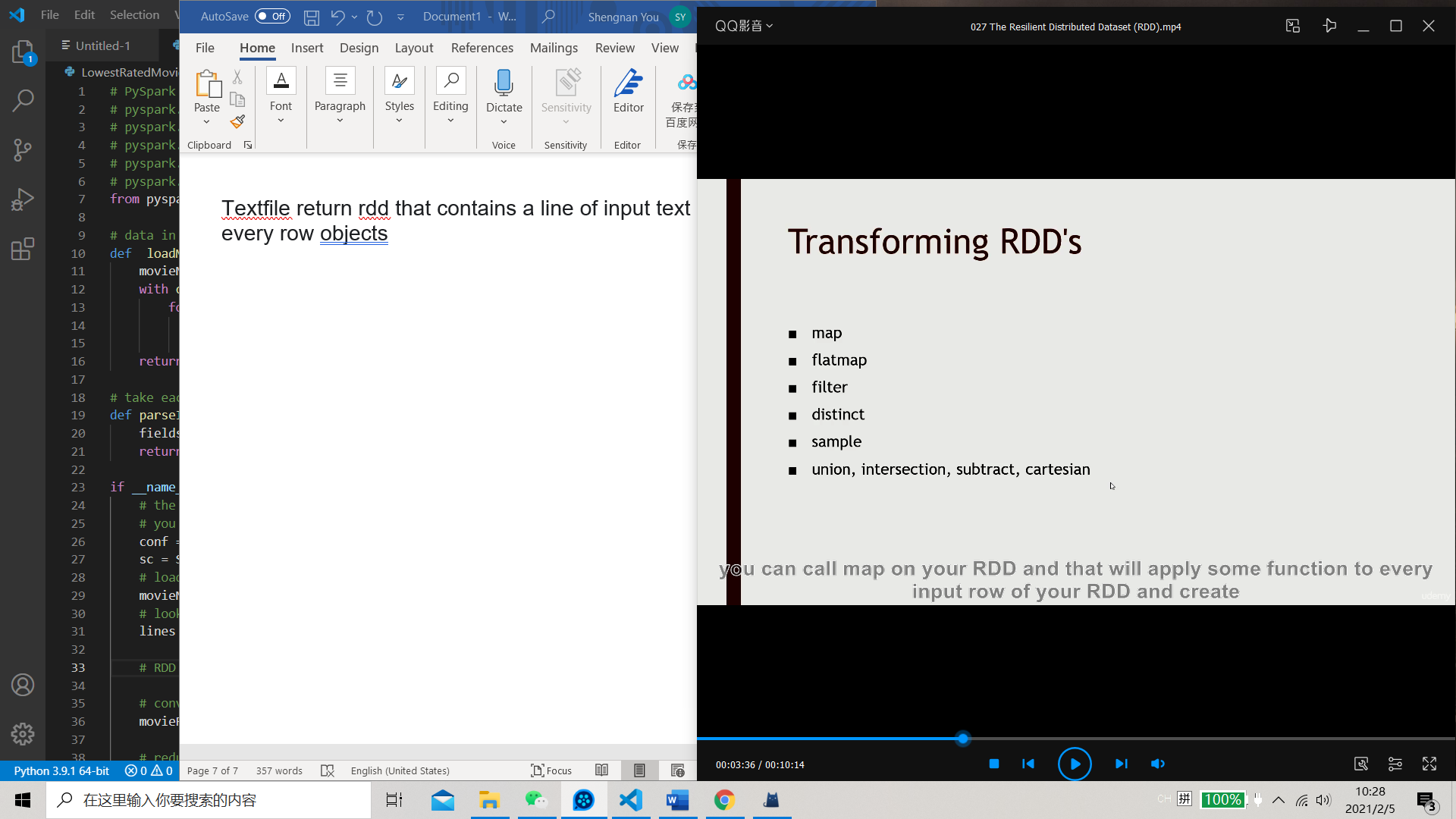
distData = sc.parallelize(data)

Once created, the distributed dataset (distData) can be operated on in parallel. For example, we can call distData.reduce(lambda a, b: a + b) to add up the elements of the list.

One important parameter for parallel collections is the number of *partitions* to cut the dataset into. Spark will run one task for each partition of the cluster. Typically you want 2-4 partitions for each CPU in your cluster. Normally, Spark tries to set the number of partitions automatically based on your cluster. However, you can also set it manually by passing it as a second parameter to parallelize (e.g. sc.parallelize(data, 10)).

Textfile return rdd that contains a line of input text on every row objects

Map:

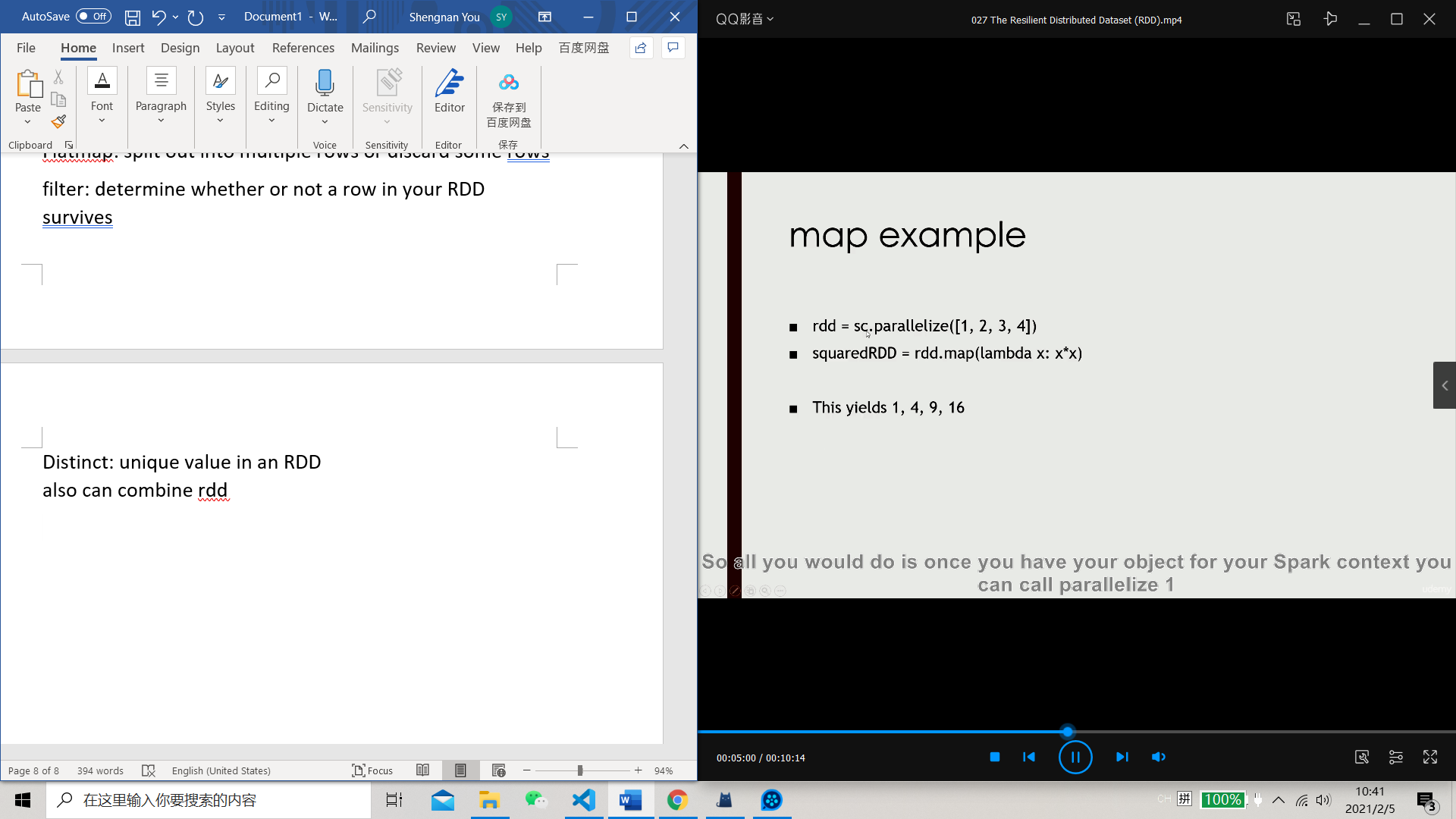


Map: a row to row transformation

Flatmap: split out into multiple rows or discard some rows

filter: determine whether or not a row in your RDD survives

Distinct: unique value in an RDD  
also can combine rdd

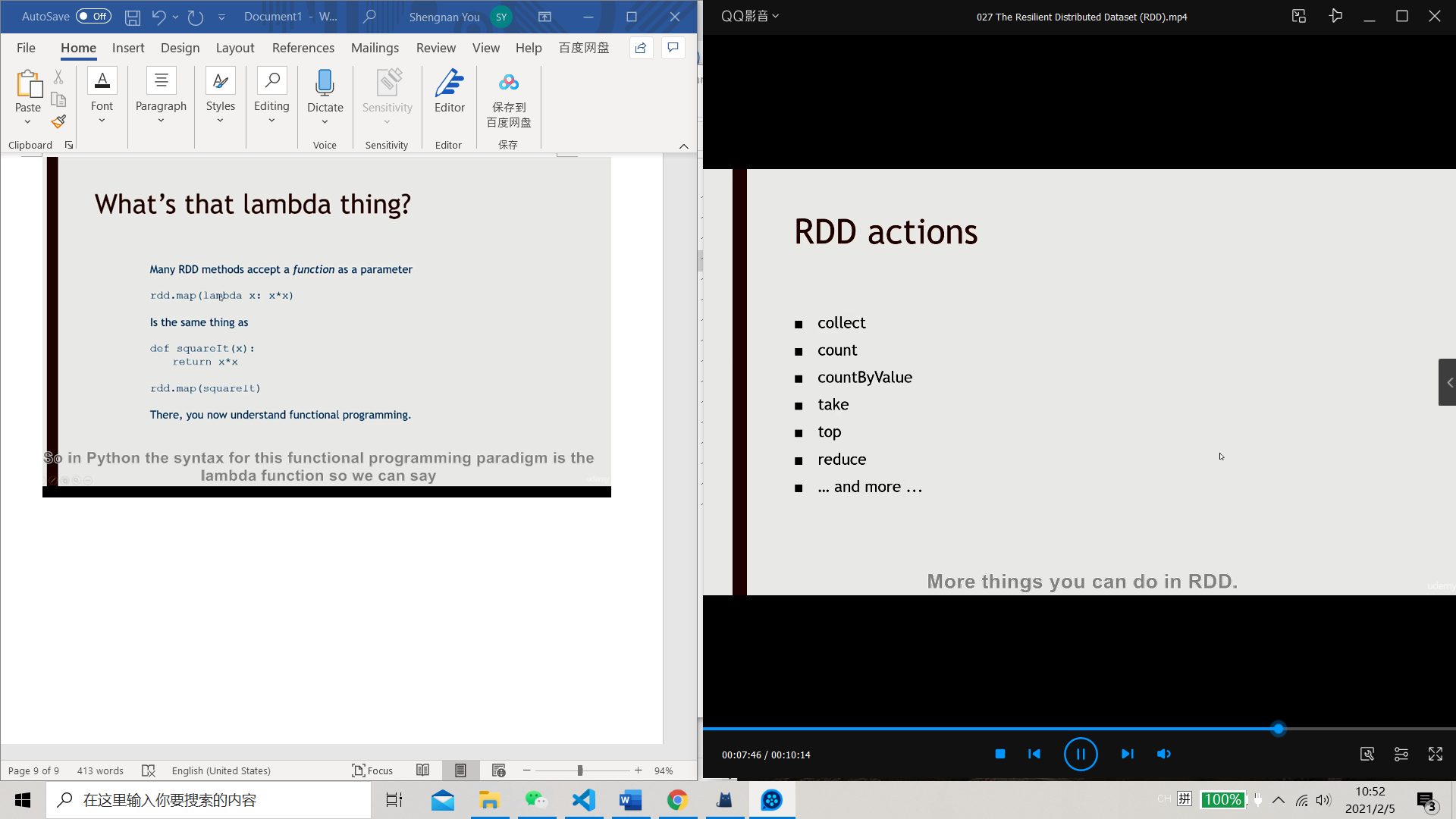


Take each input row from rdd return x\*x as a new rdd

Functional Programming: specifying a function as parameter



Reduce:

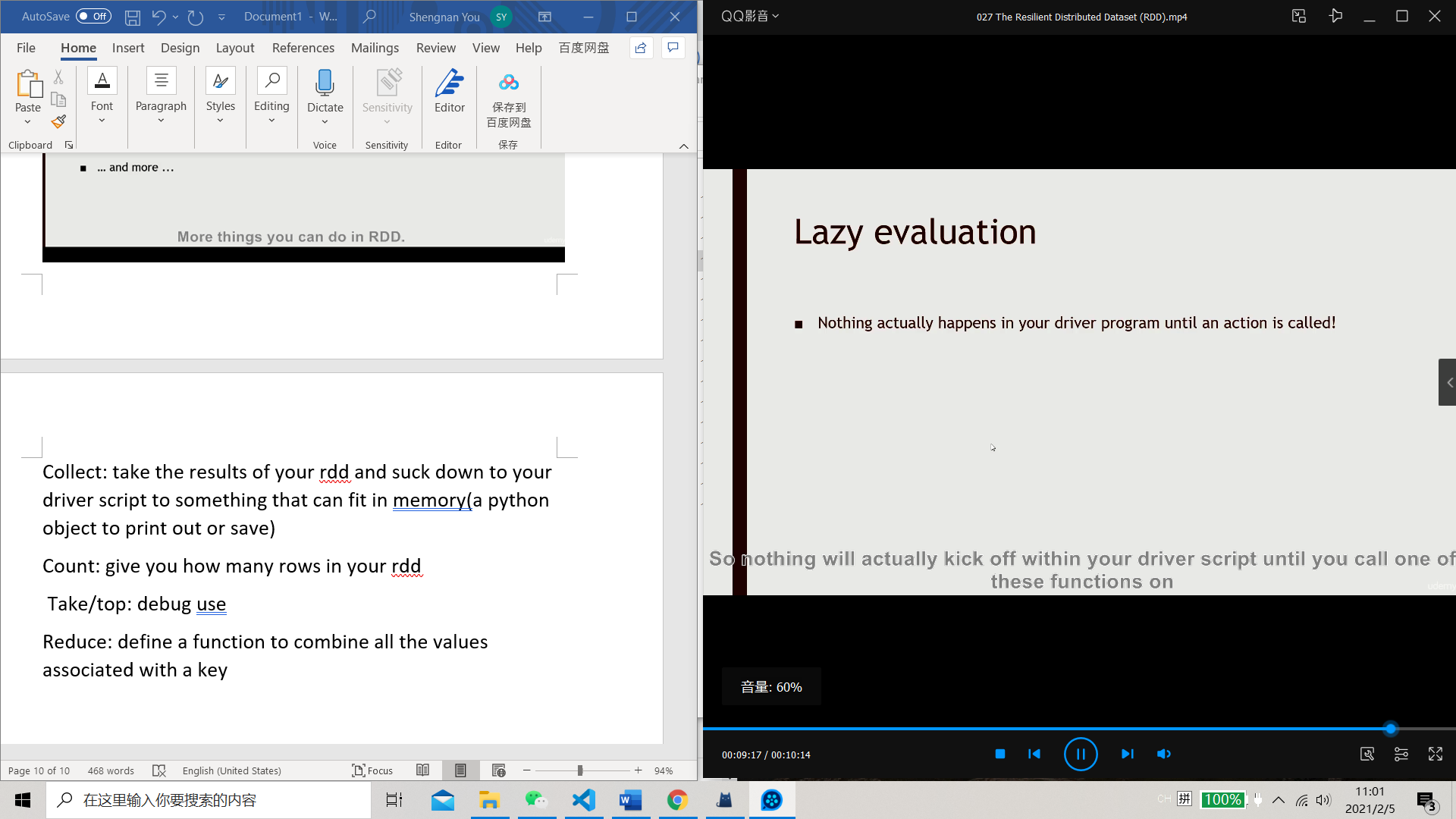


Collect: take the results of your rdd and suck down to your driver script to something that can fit in memory(a python object to print out or save)

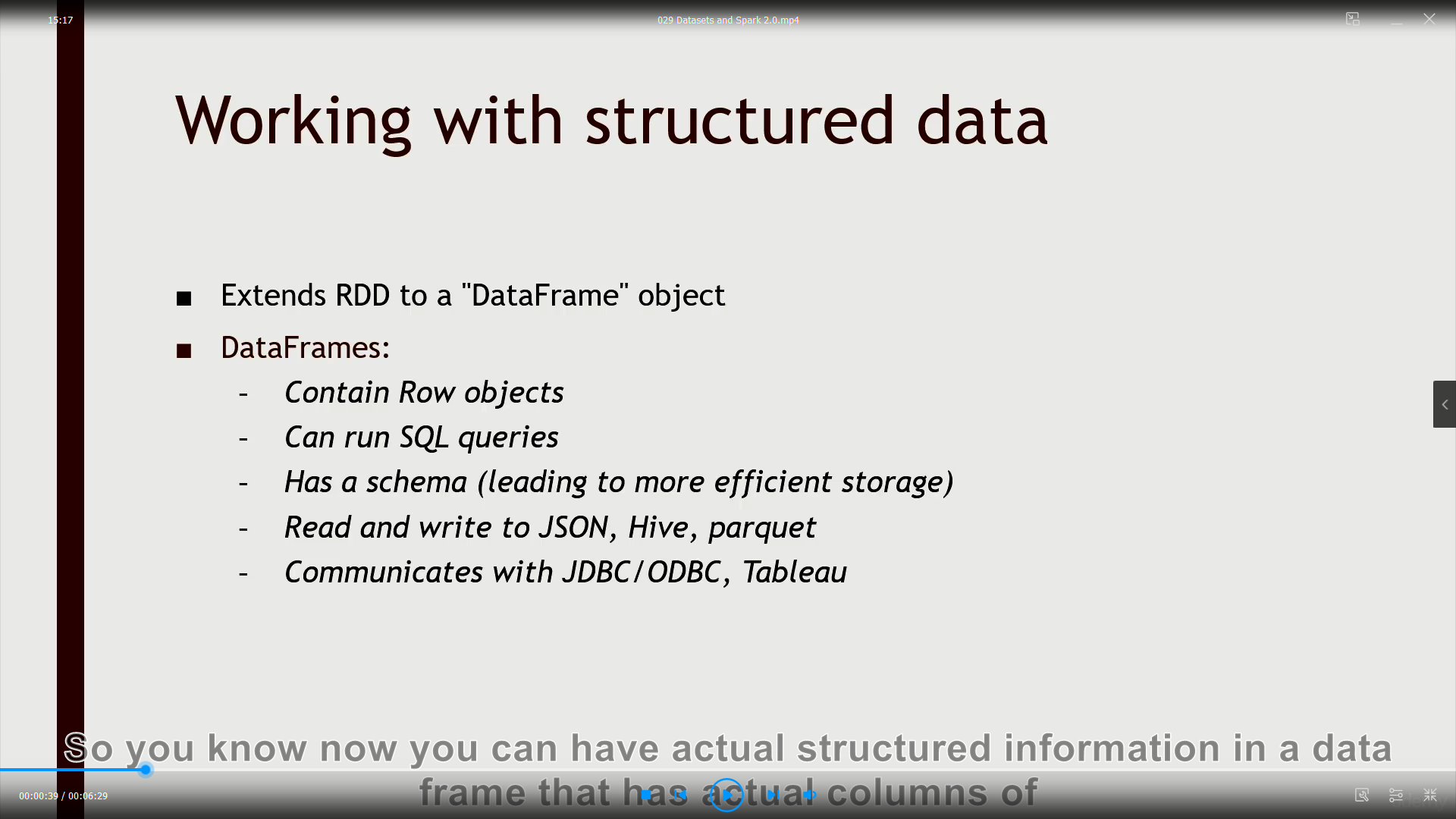
Count: give you how many rows in your rdd

Take/top: debug use

Reduce: define a function to combine all the values associated with a key



Nothing happens until actions are called (could be confusing)



Does not have much information about the types of the stuff



