1. In the WikipediaPopular class, it would be much more interesting to find the page that is most popular, not just the view count (as we did with Spark). What would be necessary to modify your class to do this? (You don't have to actually implement it.)

Update WikipediaMapper class to product text value in format {viewcount,title}.

Update “WikipediaReducer” class to iterate through all the titles and view counts for each time stamp

1. An [RDD](https://spark.apache.org/docs/3.4.1/api/python/reference/api/pyspark.RDD.html#pyspark.RDD) has many methods: it can do many more useful tricks than were at hand with MapReduce. Write a sentence or two to explain the difference between .map and .flatMap. Which is more like the MapReduce concept of mapping?

.map tales a function and applies to each element of the RDD and returns exactly one output.

.flatmap takes a function and can return zero, one or many outputs.

.map is the most similar to MapReduce as they both utilize one to one mapping.

1. Do the same for .reduce and .reduceByKey. Which is more like the MapReduce concept of reducing?

.reduce ia a binary function that aggregates all elements into a single result

.reduceBykey is a binary function that combine elements with each key in a key-value pair

.reduceByKey is most similar with MapReduce as they both combine values associated with each key.

1. When finding popular Wikipedia pages, the maximum *number* of page views is certainly unique, but the most popular *page* might be a tie. What would your improved Python implementation do if there were two pages with the same highest number of page views in an hour? What would be necessary to make your code find *all* of the pages views the maximum number of times? (Again, you don't have to actually implement this.)

Remove the lambda function and create a new function called reduce\_max\_views. Run an if logic that will return both pages that have the same highest number.

def reduce\_max\_views(a, b):

if a[0] > b[0]:

return (a[0], [a[1]])

elif a[0] < b[0]:

return (b[0], [b[1]])

else:

return (a[0], a[1] + [b[1]])