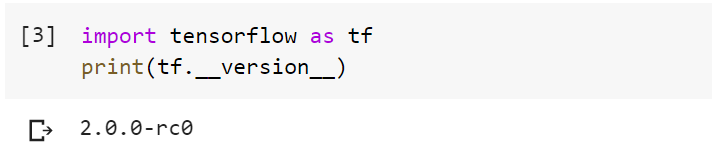
Environment set up

In this LiveProject, we will use Tensorflow 2.0 version for our work. This lab was created and validated using Tensorflow for GPU version 2.0.0-rc0, running in Google colab.

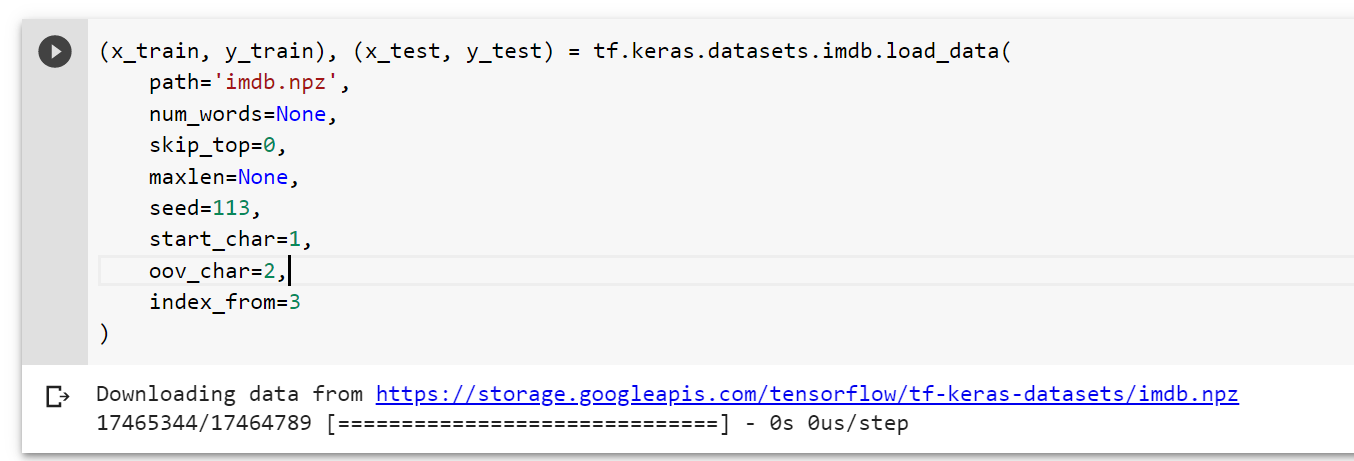


Check the Tensorflow version installed.

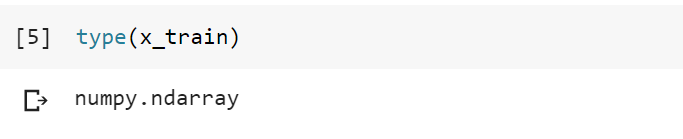


### **Load IMDB data**

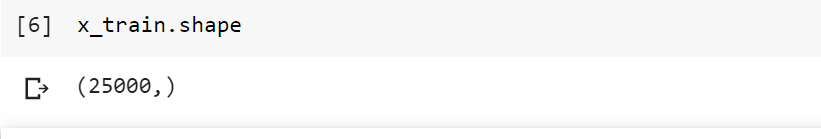
Load the IMDB review data as numpy array. The dataset is nicely split into training and test, and then into data (x) and label (y).

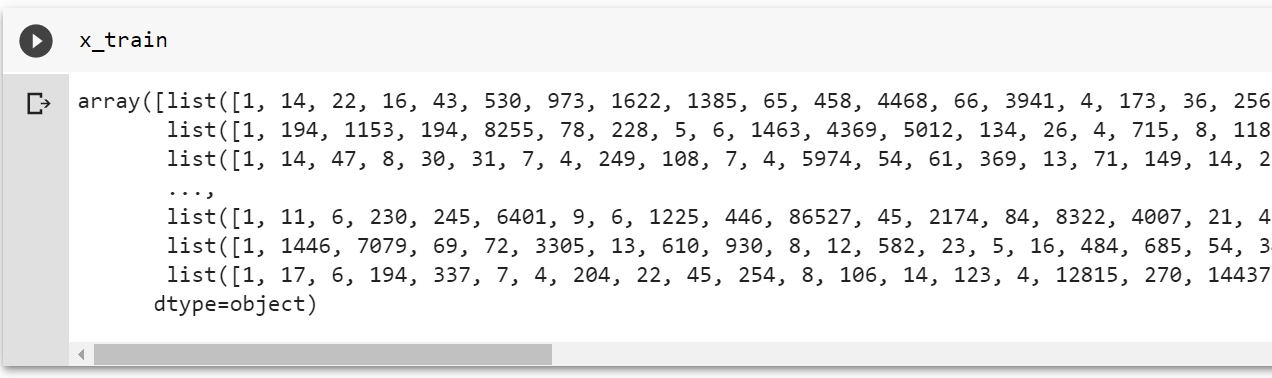


Examine the data type with type command.

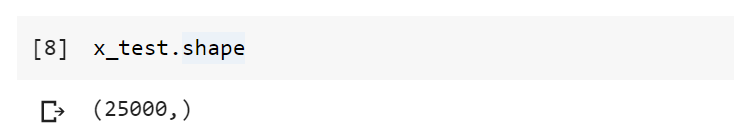


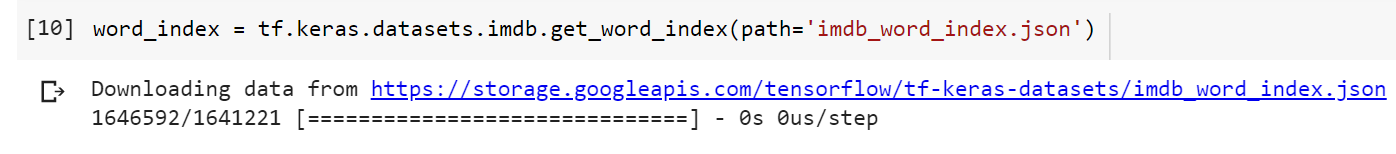
Examine data structure with numpy's shape command

Let us take a look at the content.

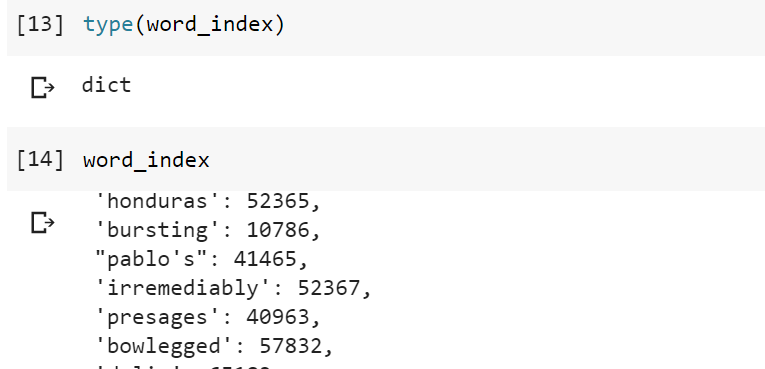


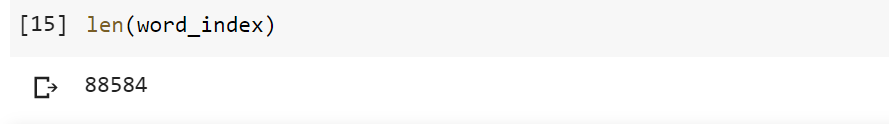
It appears each element is a in the numpy array is a list of integers. This suggests that each integer encodes a word, which requires a dictionary in order to map it back to actual word.

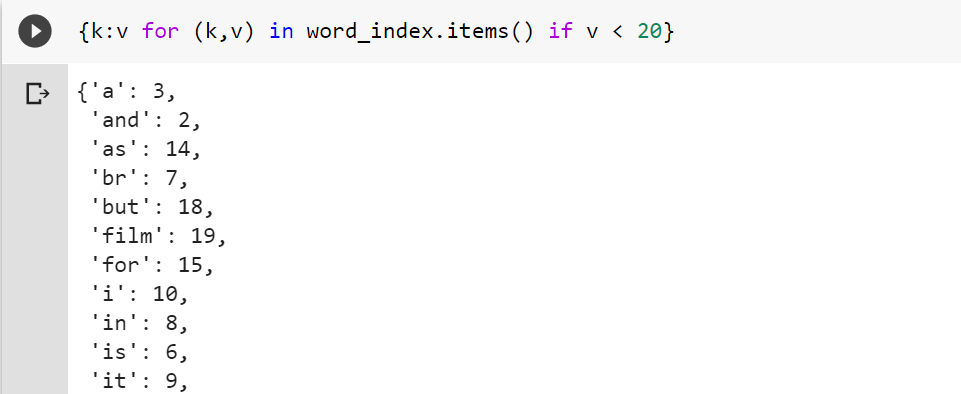
Let us load the word index provided by the dataset.



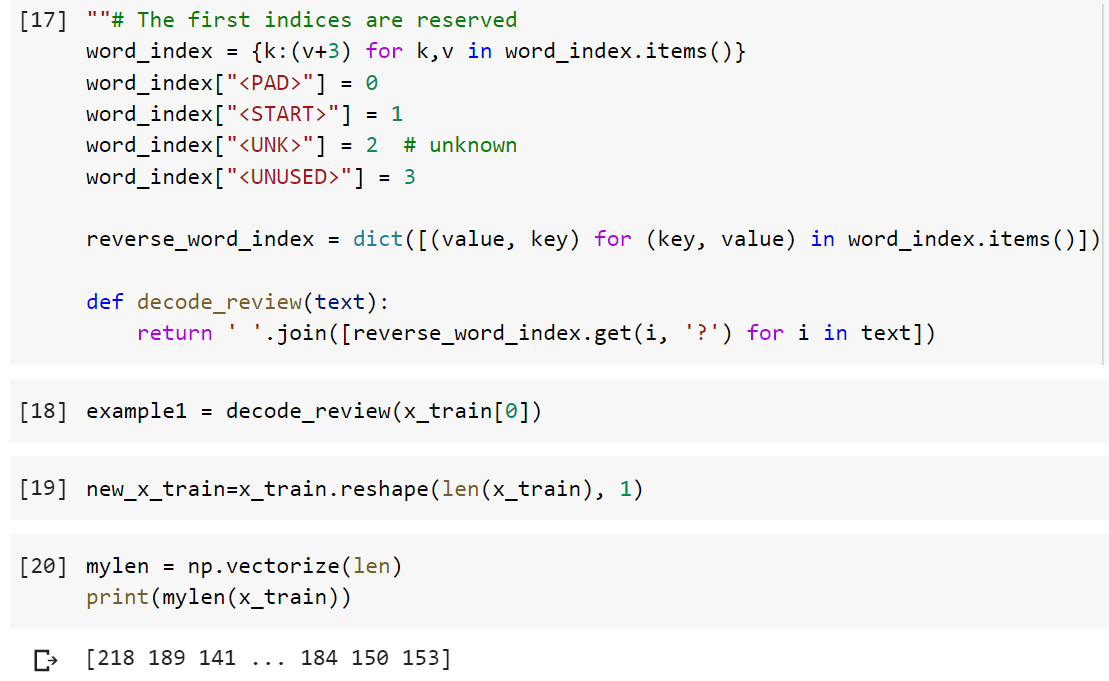
The word index is a type of data structure known as dictionary, which is a key-value pair. Later we will use this as a basis to map integers back to words.



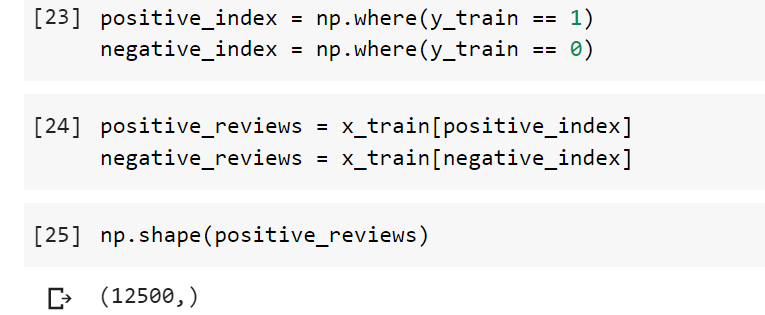
We can also find out how many unique words this dictionary contains. Now we know there are 88584 unique words (tokens) in the imdb dataset. Each word has a unique number associated with it. This structure is known as key-value pair. Therefore, there are 88584 key-value pairs, organized as a dictionary in Python datra structure. As examples, below are a few words (tokens) in this dictionary.



Now lets add a few special words for use later. This is a common practice for NLP problem, where it is important to bring a consistency to text strings by giving it a start, a token to handle words outside this dictionary, as well as padding to ensure all text data have same length. We bump original words by three positions, and appended the following new words to the dictionary. We also reverse the key-value relationship and created a new dictionary for reverse lookup. In addition, we created a function decode\_review to convert data from integer into words.



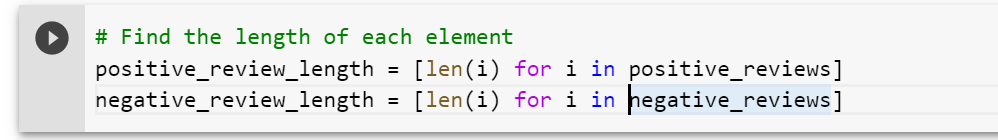
Now find the index for positive and negative reviews

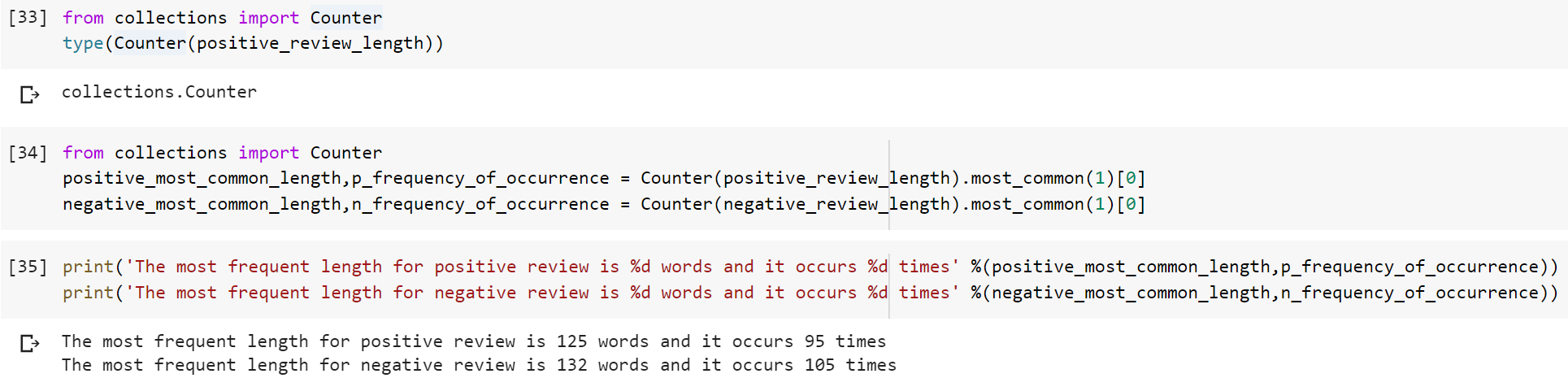


### **Basic discovery**

Let us try to answer the following question: How is length different between positive and negative reviews?

This is important because we want ot know if the amount of information contained in the review are different depending on types of reviews.



We will use python's Counter to perform frequency counting. 

Lets make a figure to visualize distributions of review length

