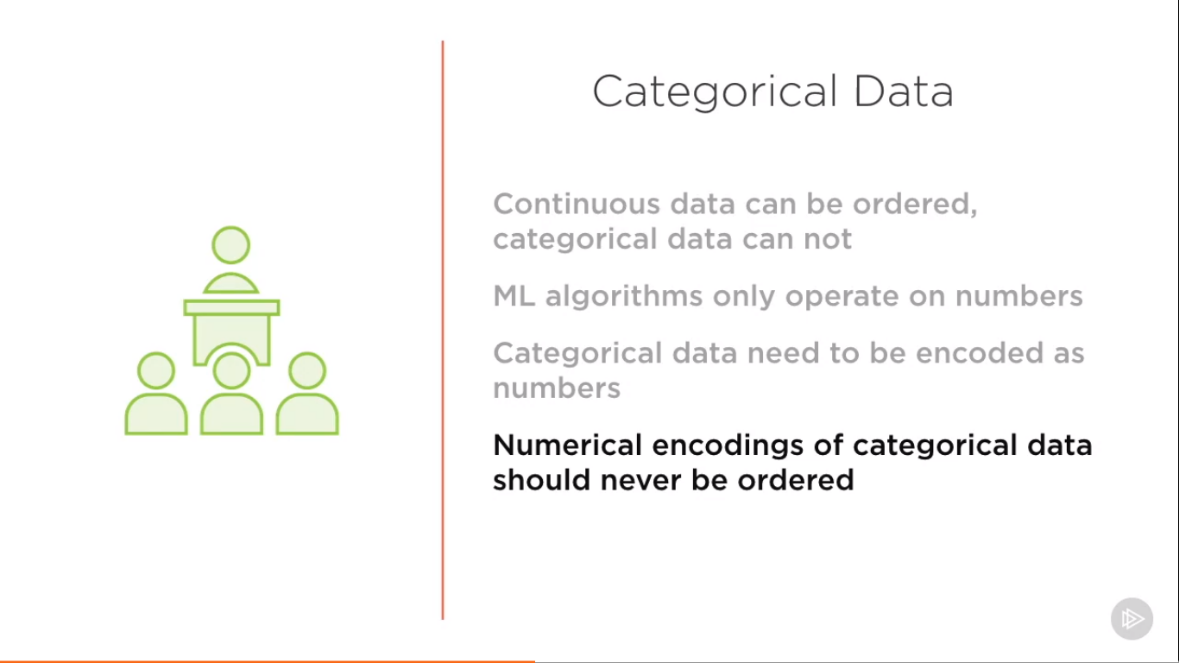
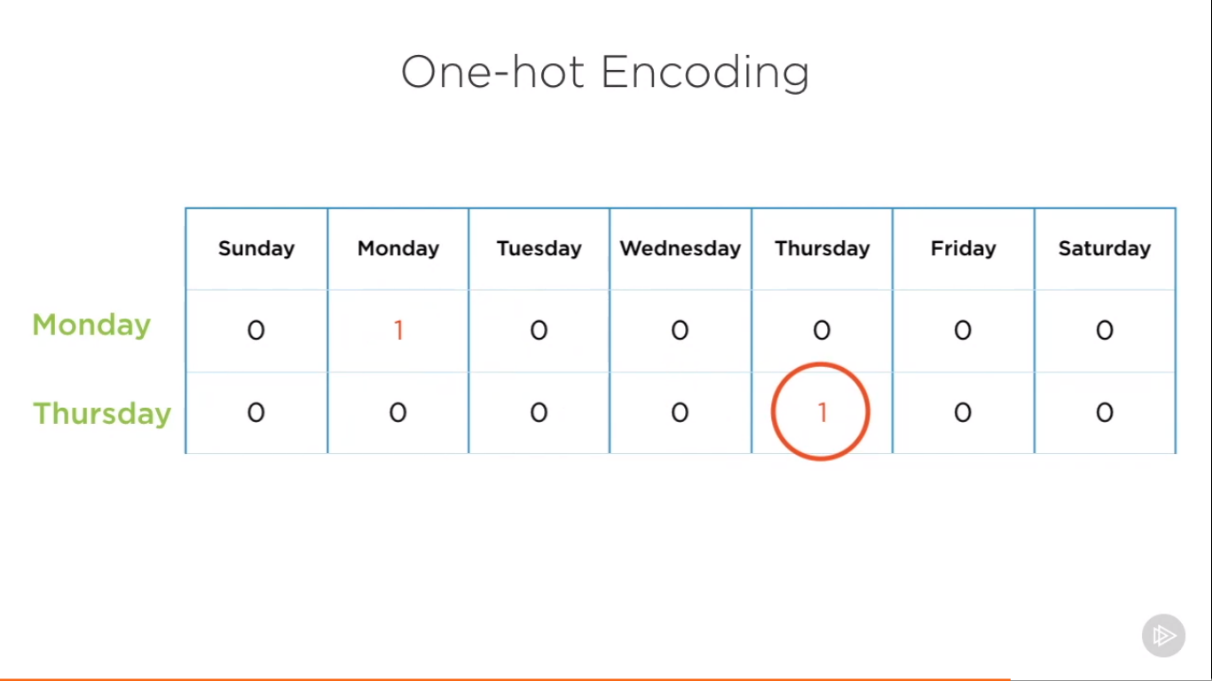
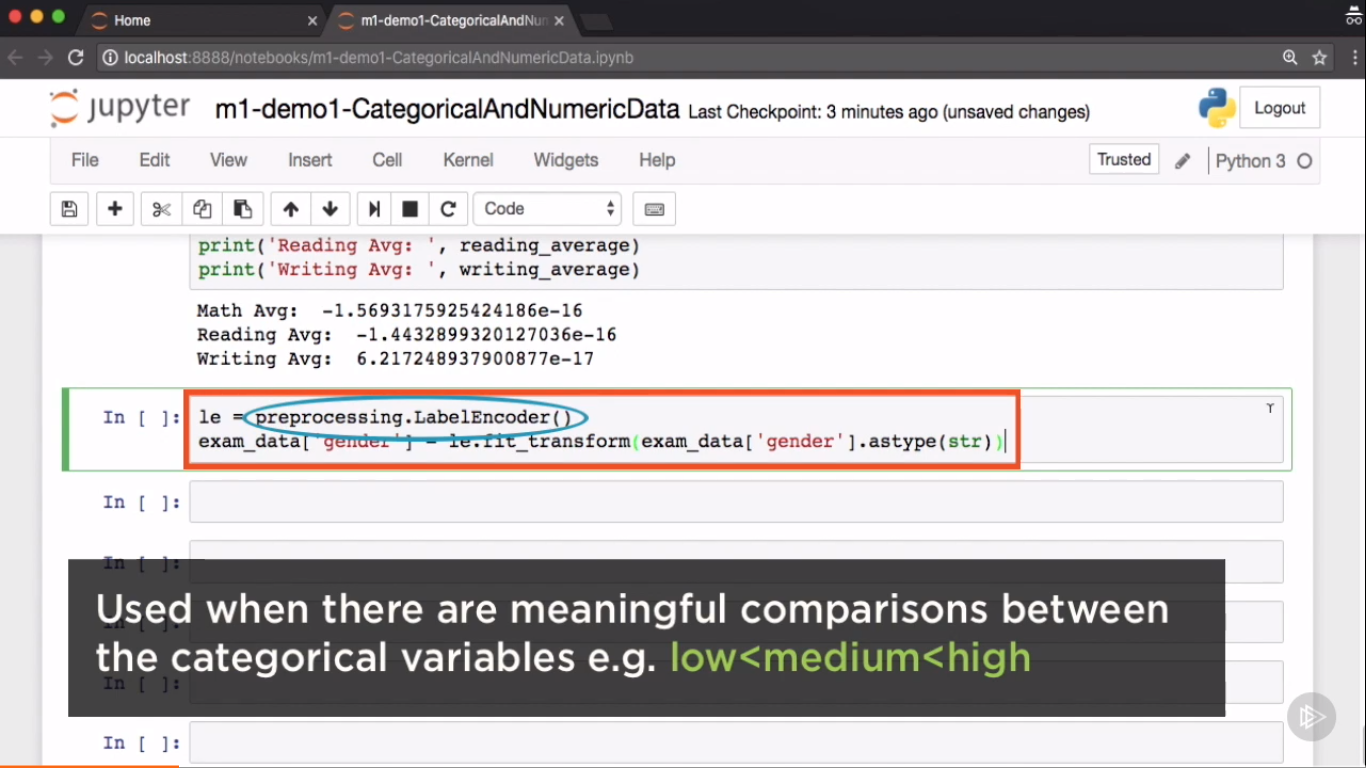
**Categorical Data and One Hot Encoding:**

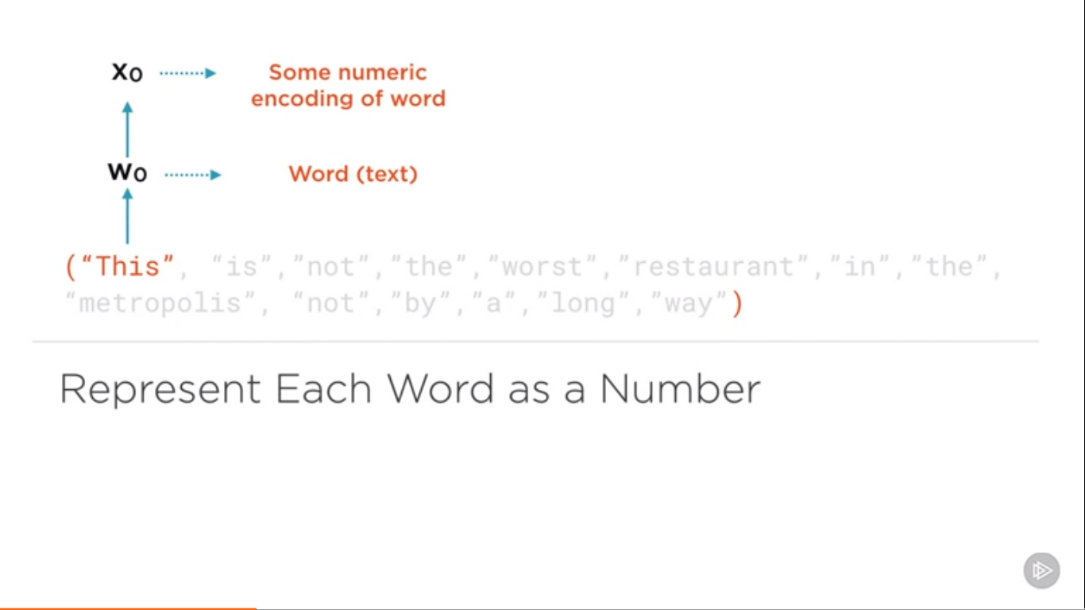


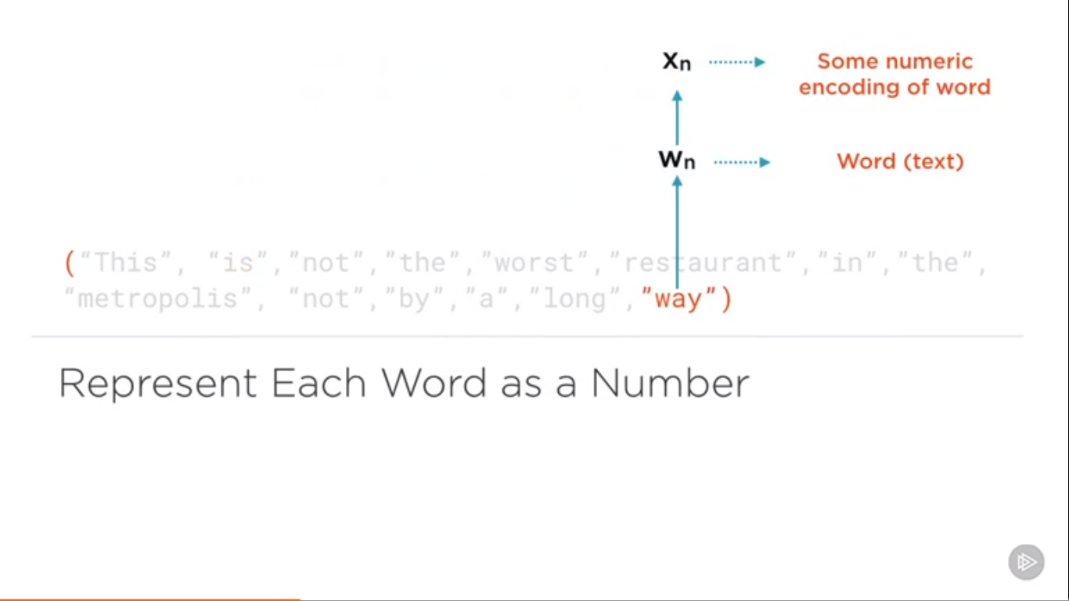


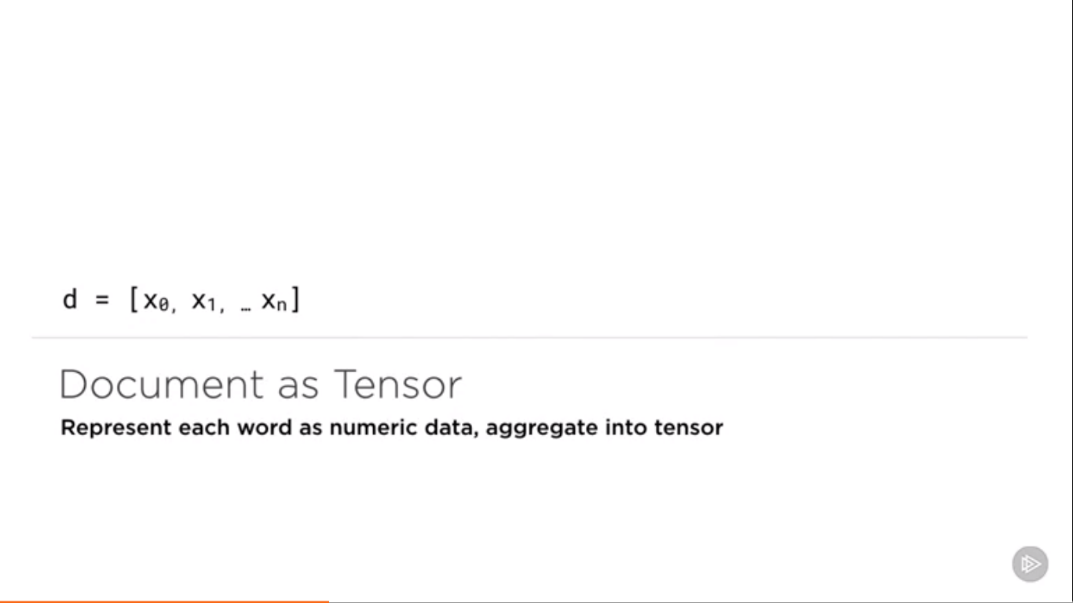
**Representing Numeric Data in Categorical Form:**

****

**Encoding Text Data in Numeric Form:**

****

****

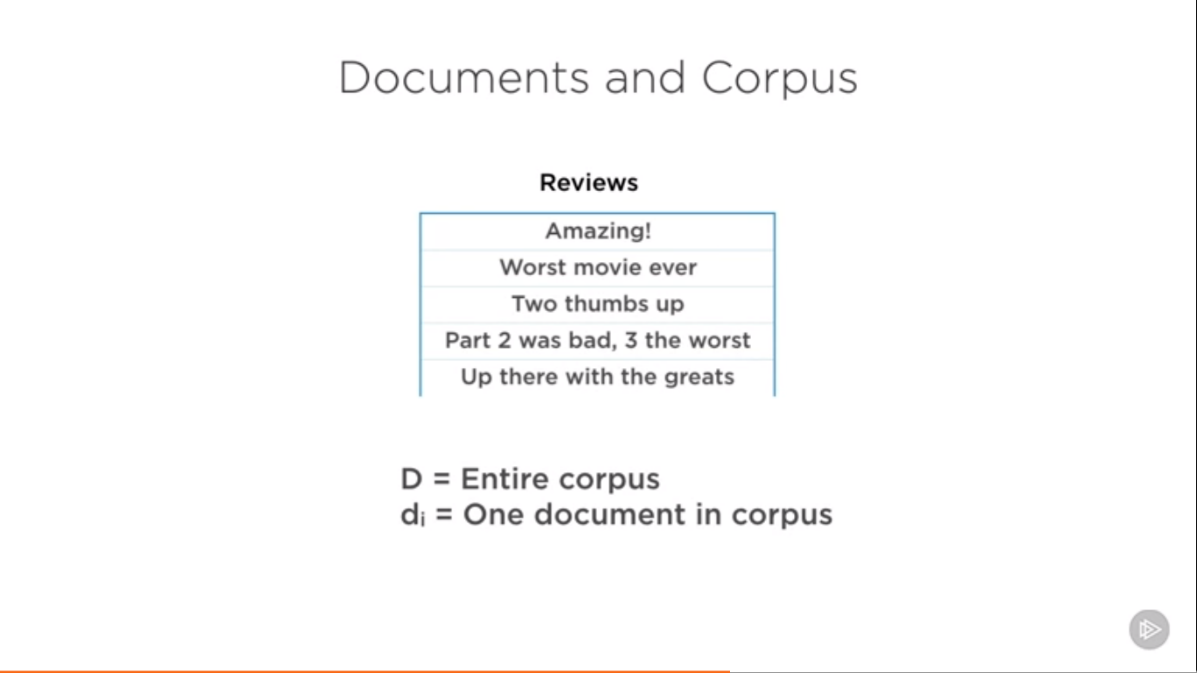
****

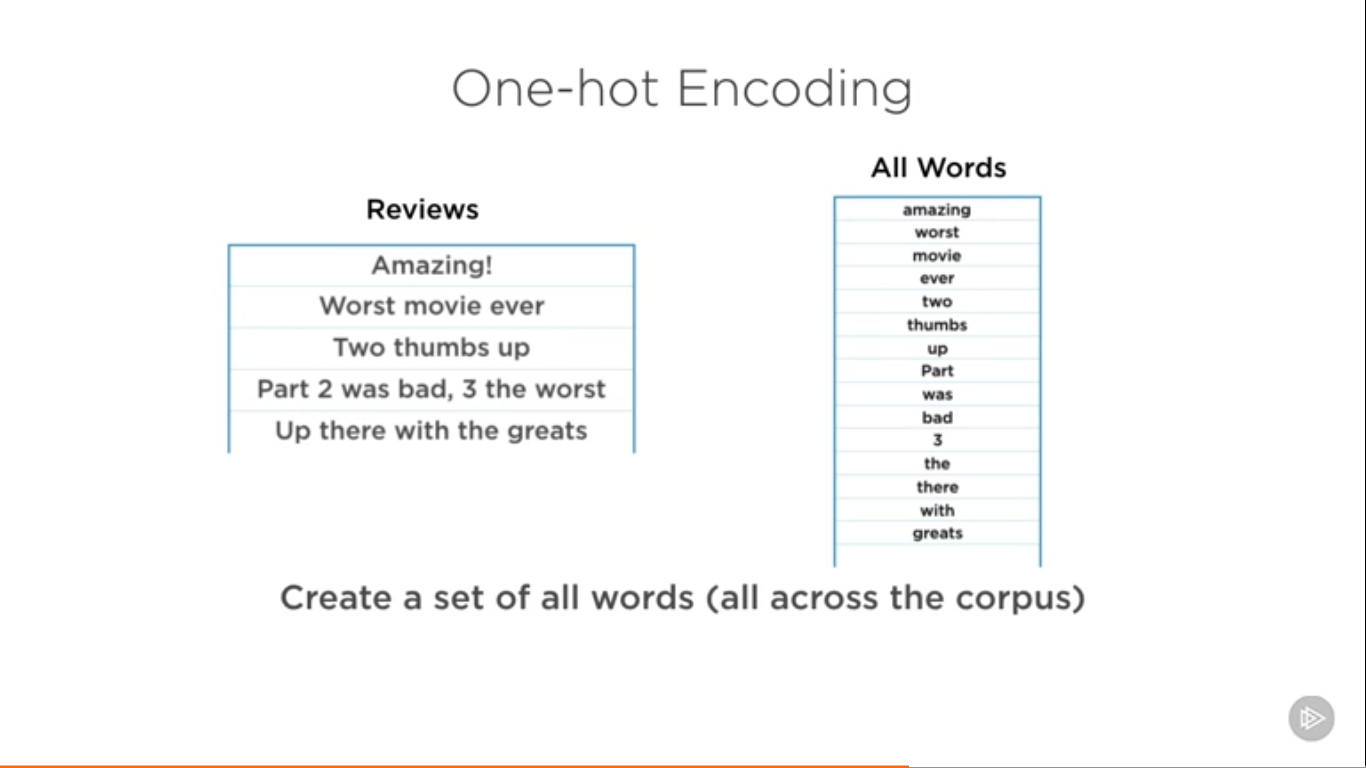
How best can words be represented as numeric data?

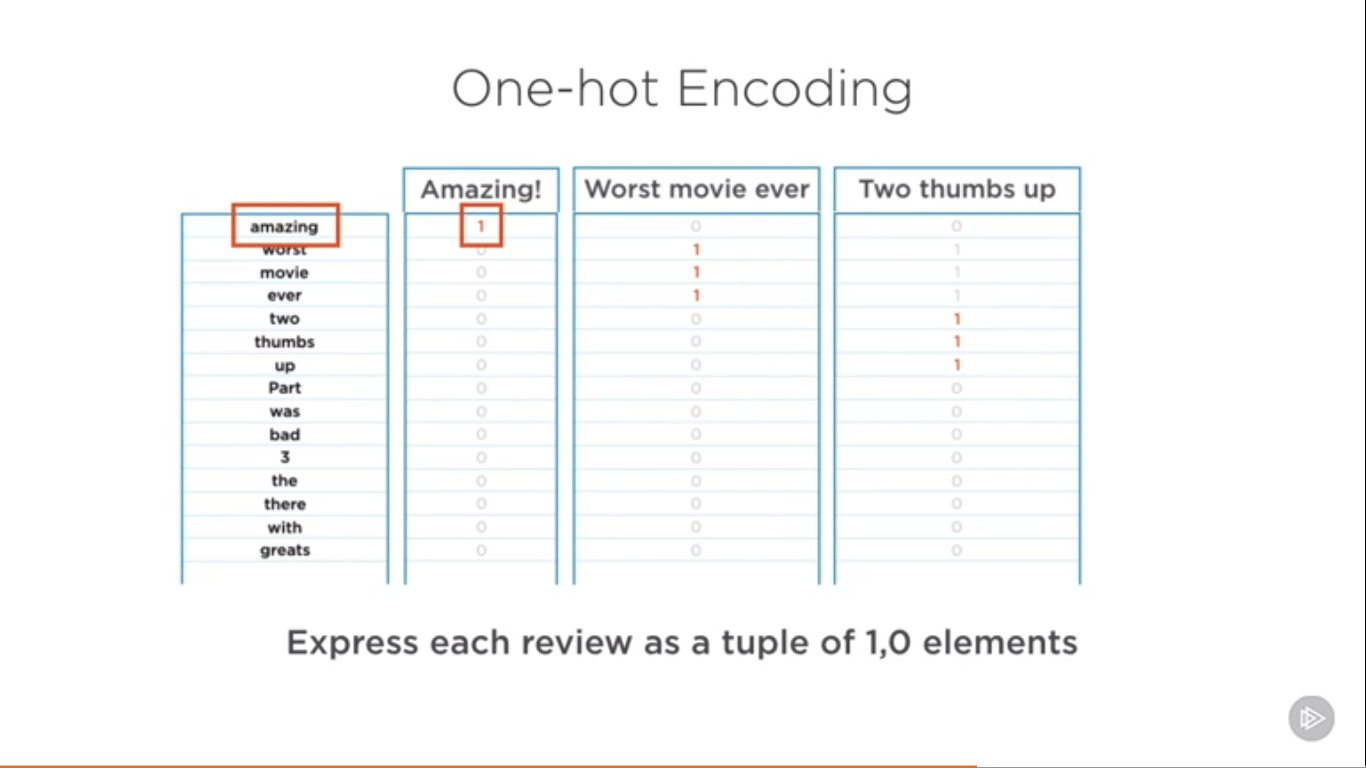
**Word Embedding**

* One-hot
* Frequency based
* Prediction based (used in DL specially in Neural network)

**One Hot**

****

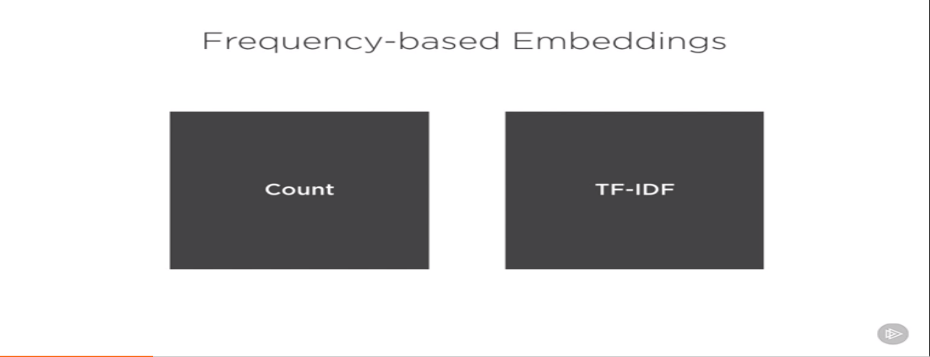
****

****

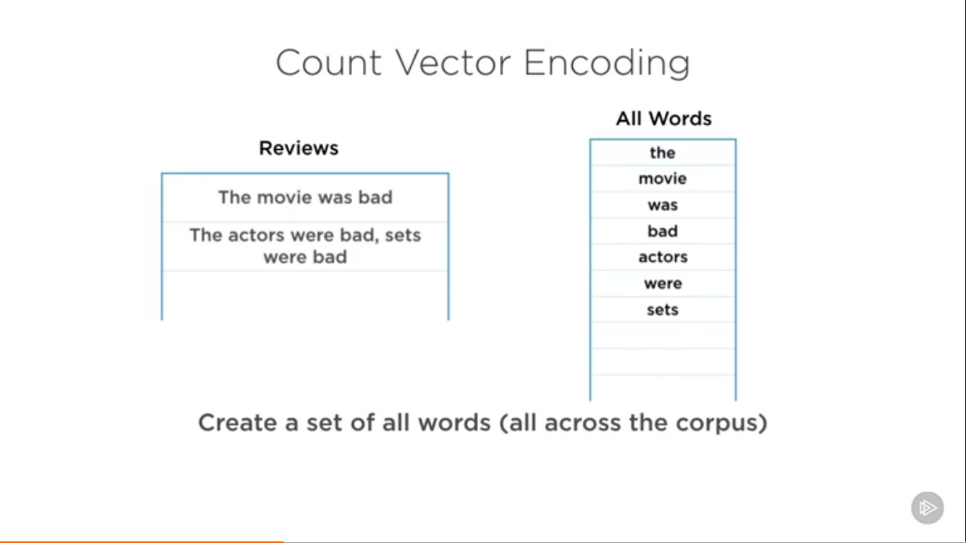
**Flaws of One hot encoding:**

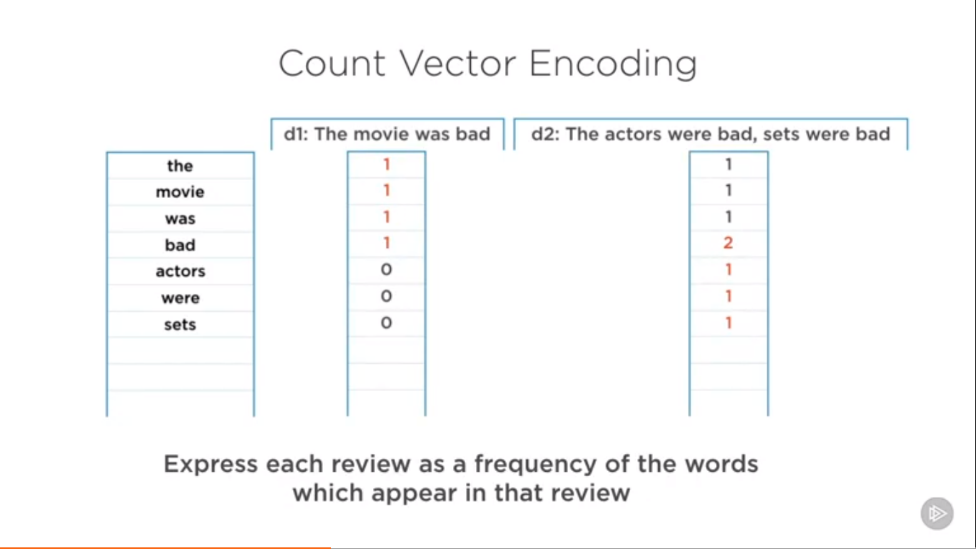
* Large vocabulary – enormous feature vectors
* Unordered – Lost all context
* Lost – Lost Frequency info
* One hot encoding does not capture any semantic information or relationship with words

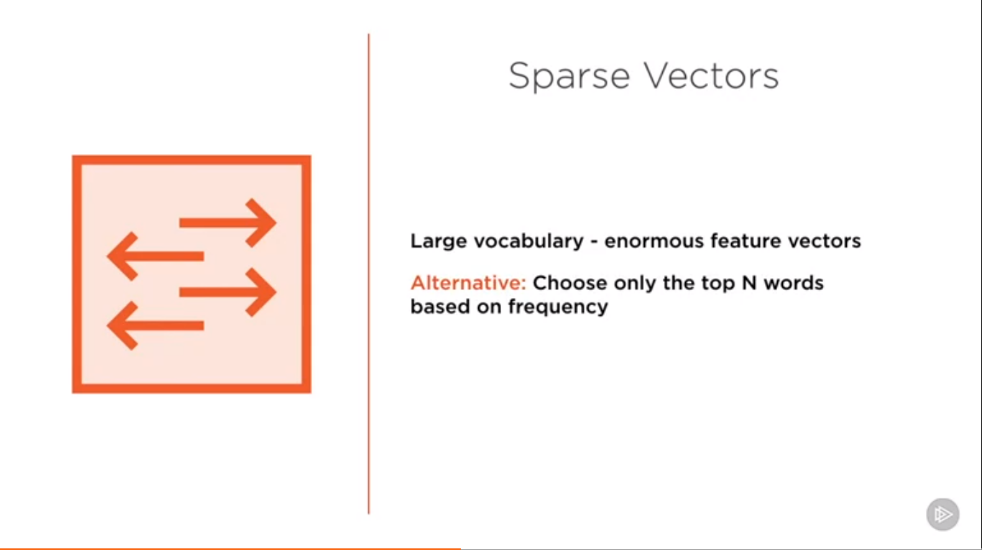
**Frequency Based Embedding**

****

Count captures how often a word occurs in a document i.e. the count or the frequency.







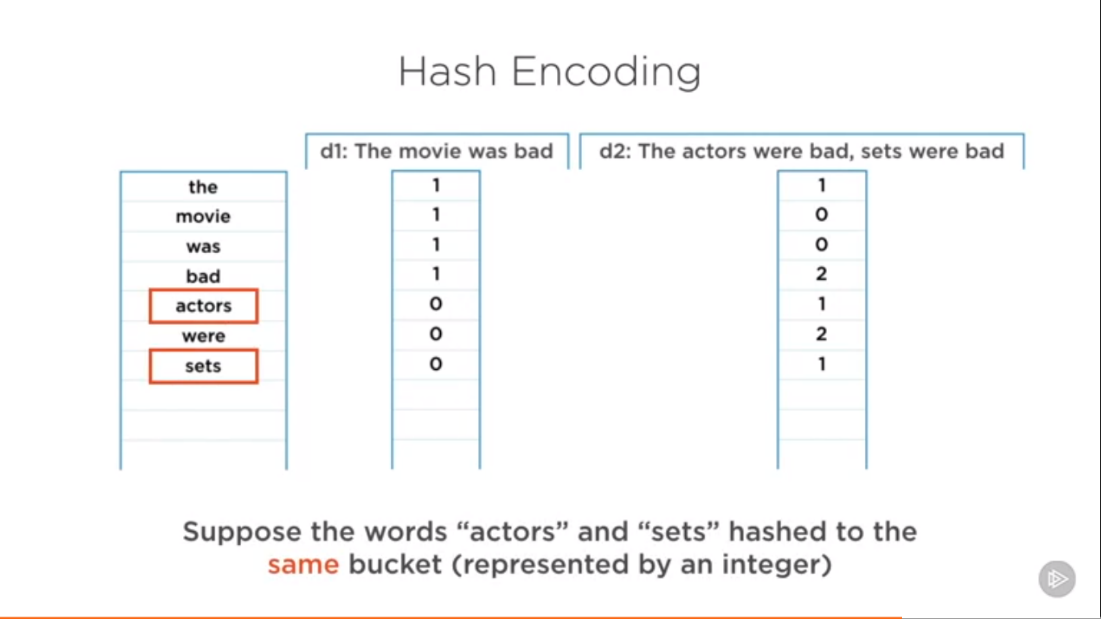
**Flaws:**

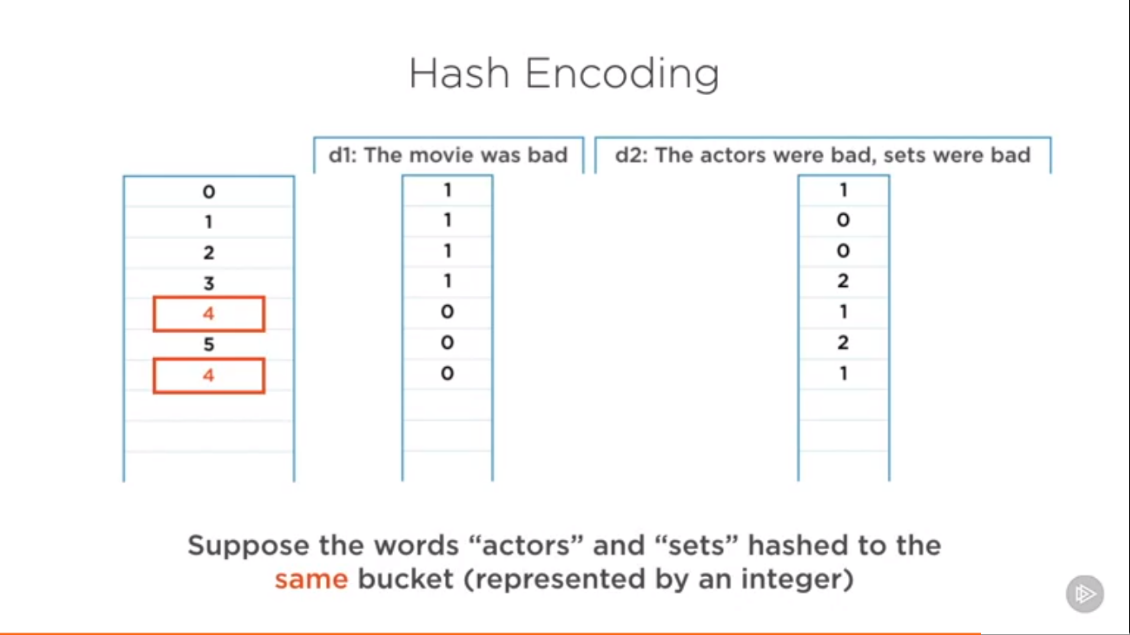
* Large vocabulary – enormous feature vectors
* Unordered – lost all context
* Semantics and word relationships lost

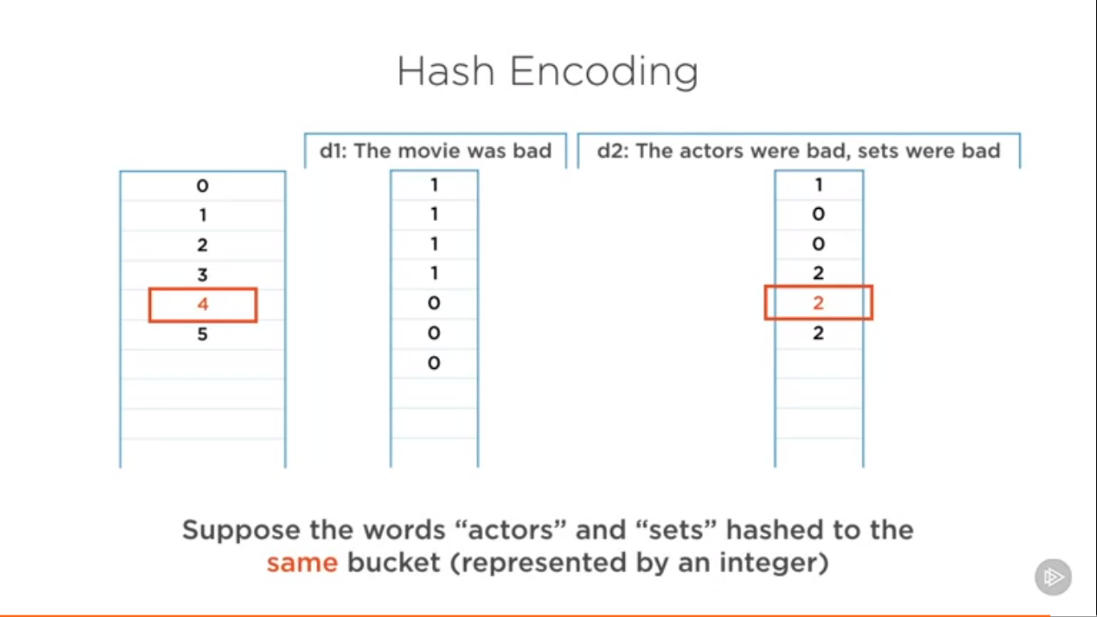
Solution for Large Vocabulary

* Hash words to buckets to have a fixed vocabulary size.
* Choose enough buckets so that collision is rare

Ex



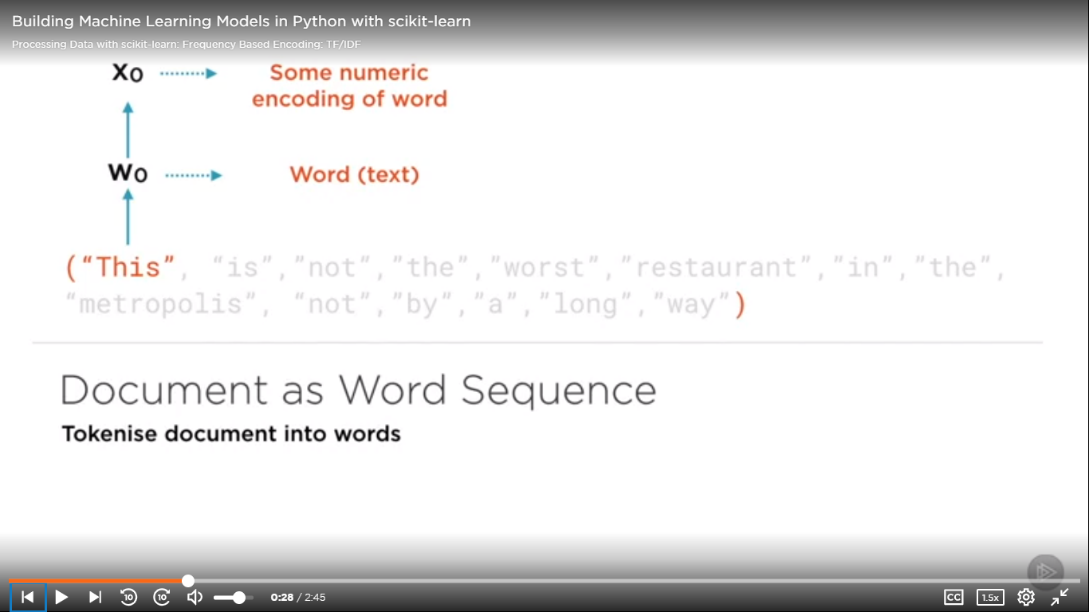




So we lost some info, now we don’t know that these two are different words

**Working of TF/IDF Algorithm:**

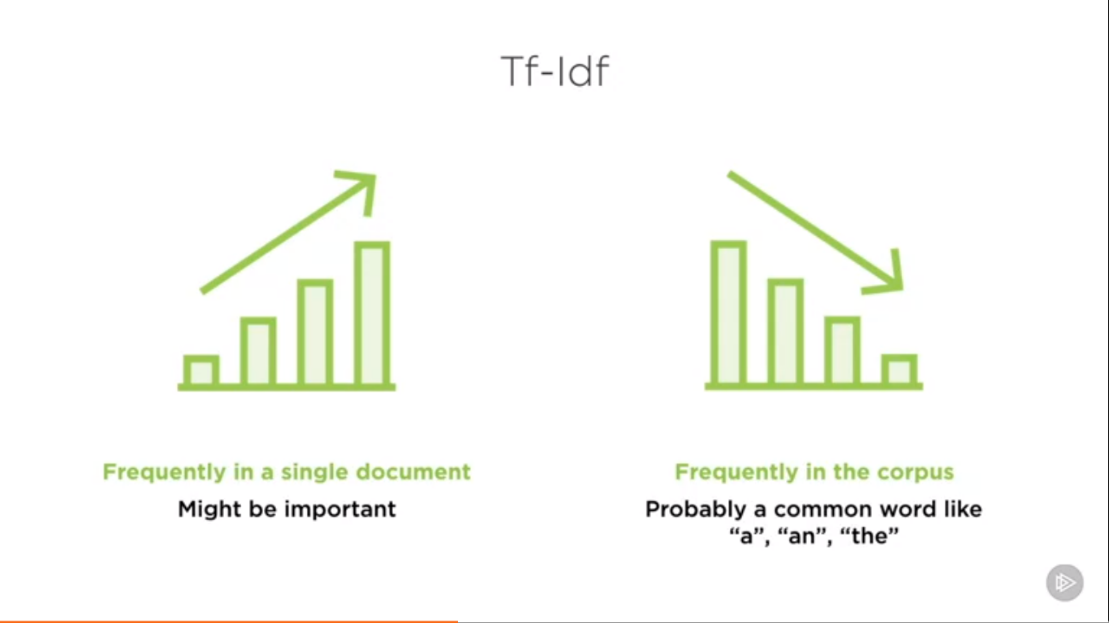
It captures two pieces of info – How often a word occurs in a Document as well as the in the entire corpus.

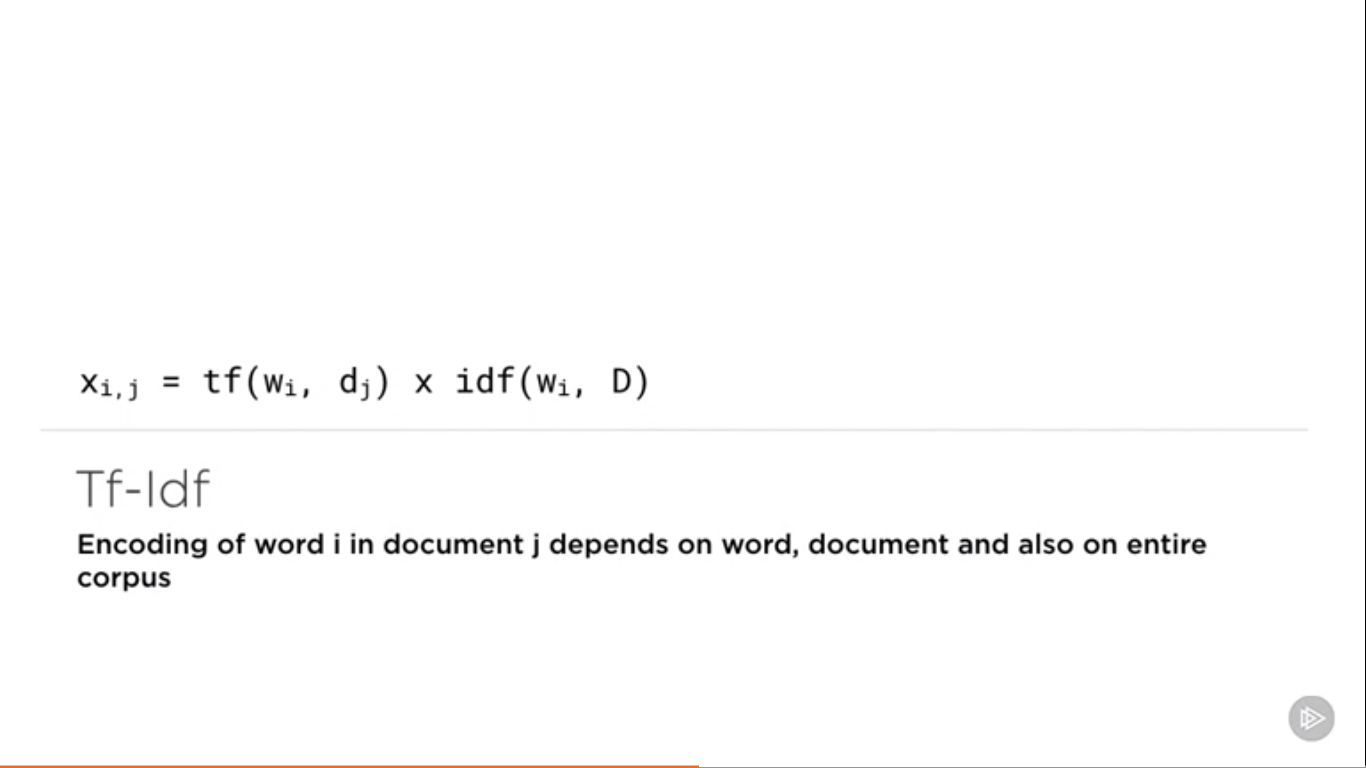


**xi  = tf (wi) \* idf(wi)**

**Tf = Term Frequency**

**Idf = Inverse Document frequency**

****

****

**Tf = Term Frequency**

Measure of how **frequently** word i occurs in document j

**Idf = Inverse Document Frequency**

Measure of how **infrequently** word i occurs in corpus D

**Tf – Idf**

High weight for word i in document j if word occurs a lot in this document, but rarely elsewhere

**Important Advantages:**

* Feature vector much more tractable in size
* Frequency and relevance captured

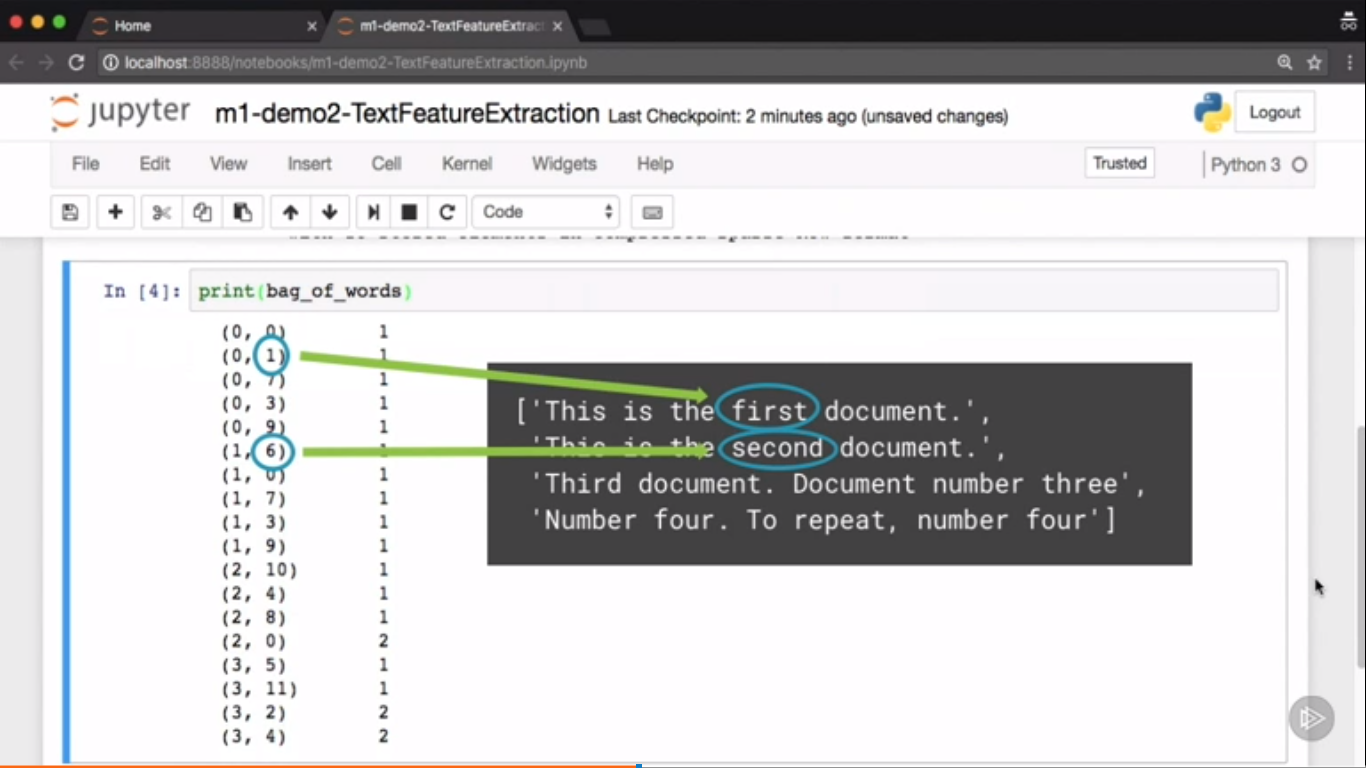
**One big drawback**

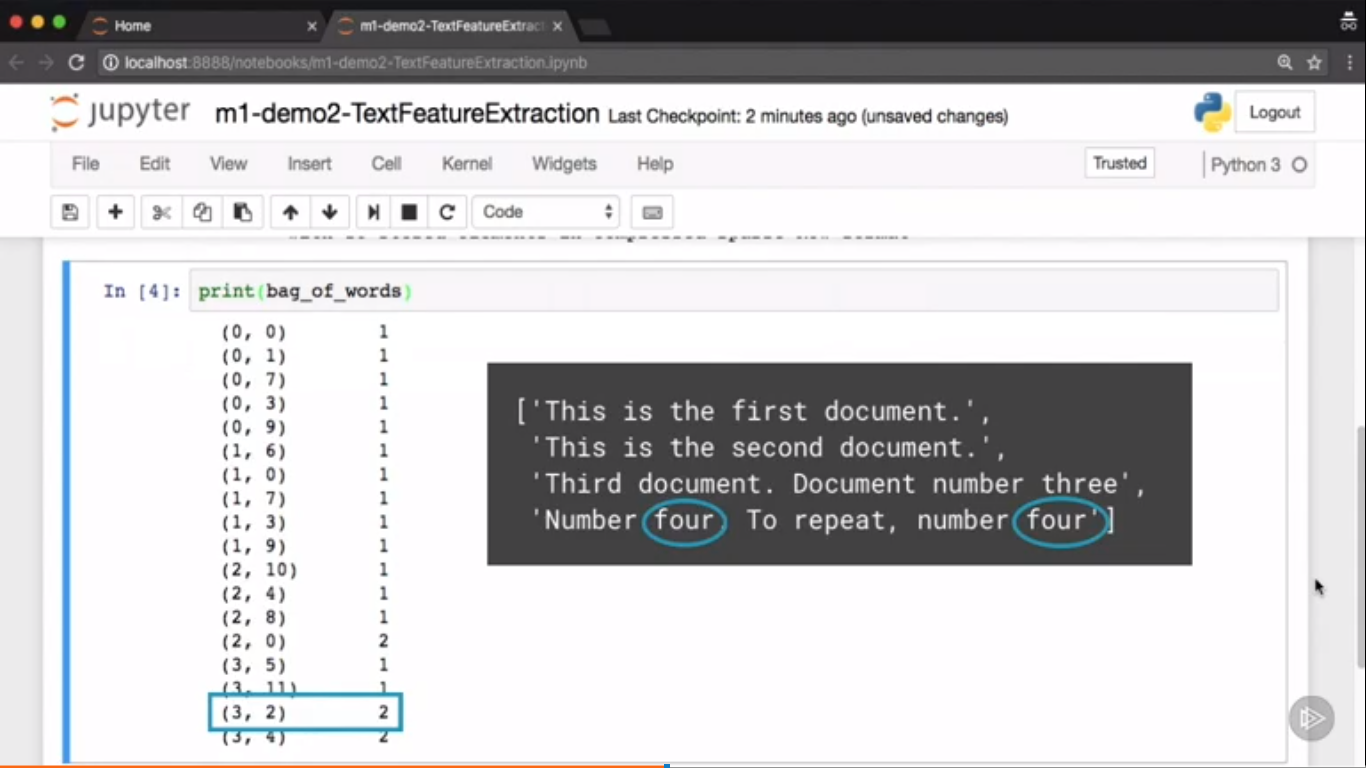
* Context still not captured

**Demo: CountVectorizer, TfidfVectorizer, HashVectorizer:**

**vectorizer.fit(<data>) –** Generate Unique IDs for words in Corpus.

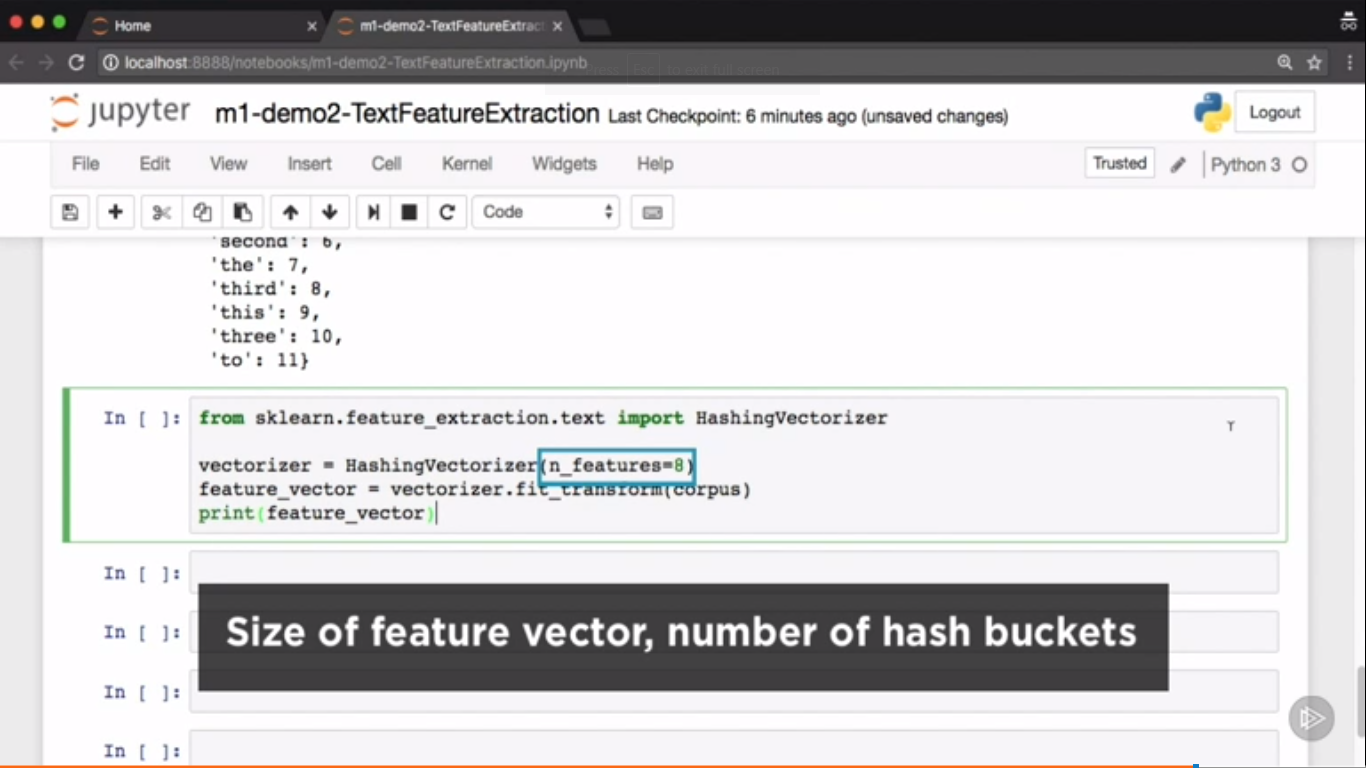
**vectorizer.transform(<data>)** – Assign the generated IDs to Corpus.

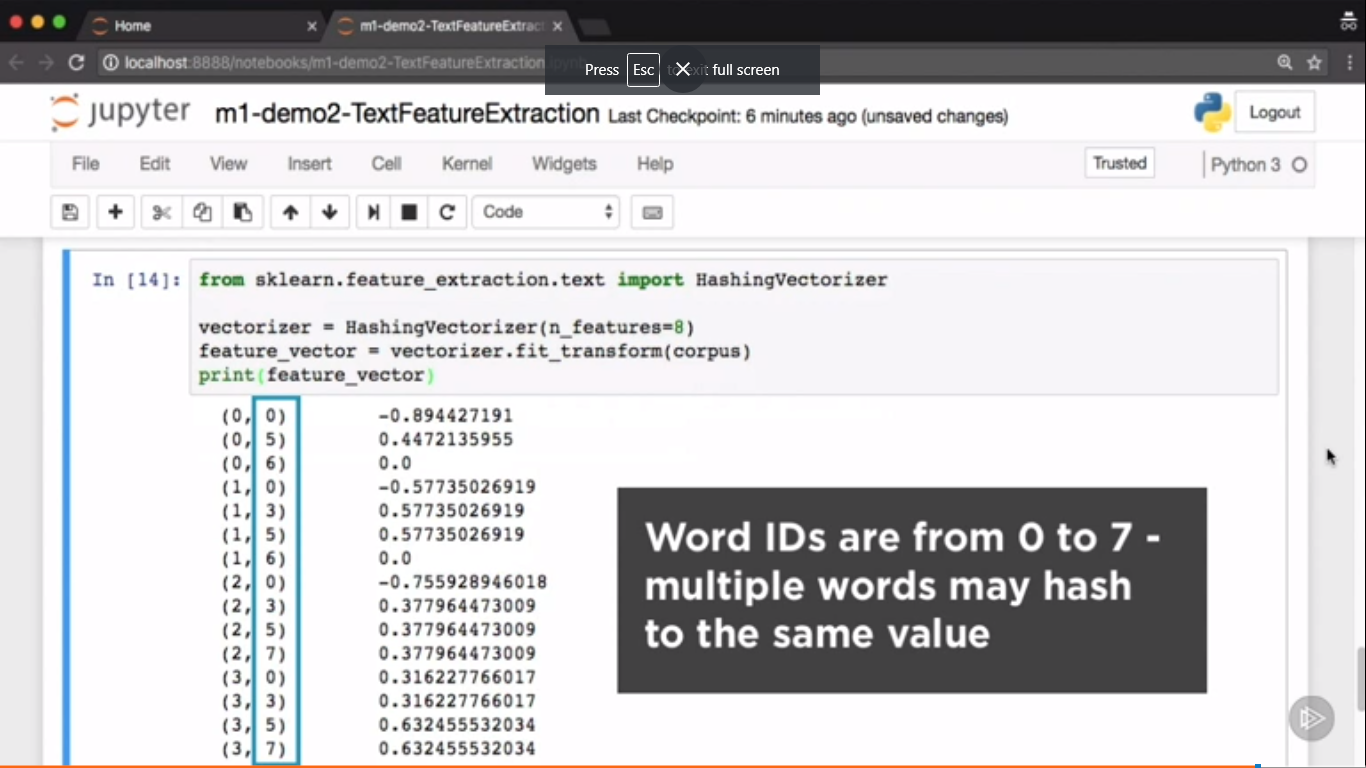




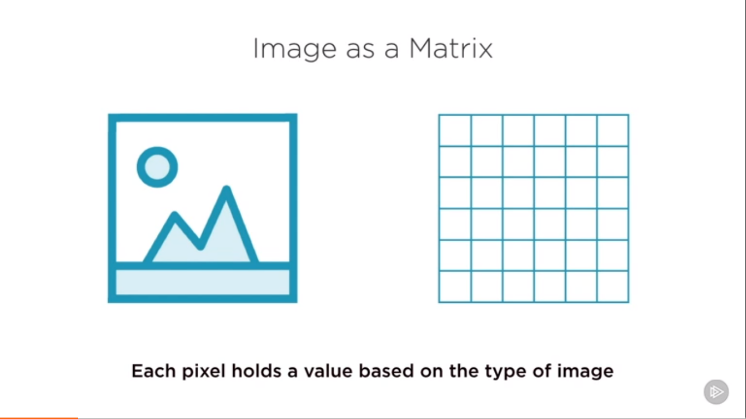
Typically the most occurred words usually get the lower ID

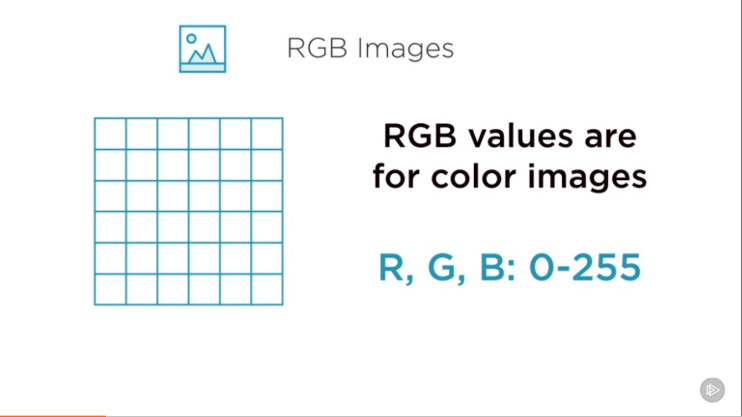
**HashingVectorizer**

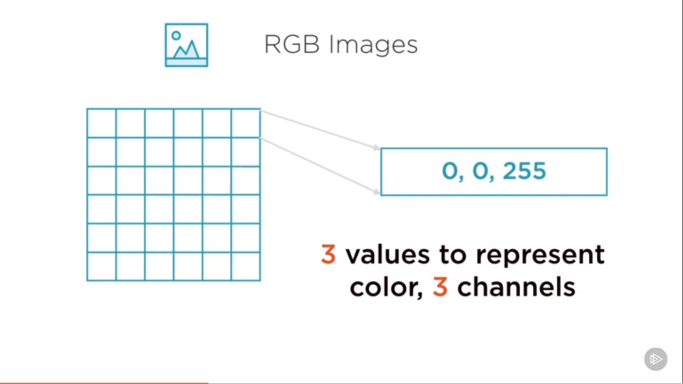
****

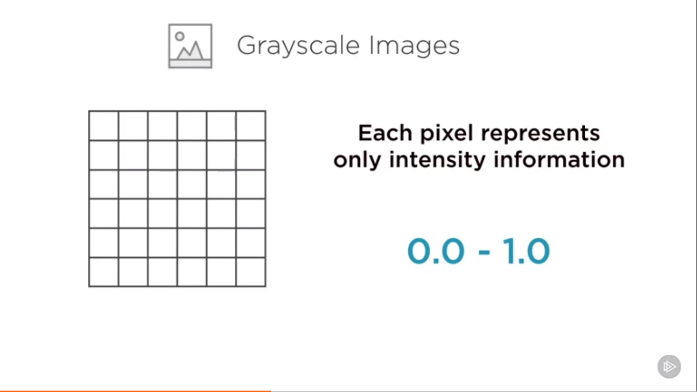
****

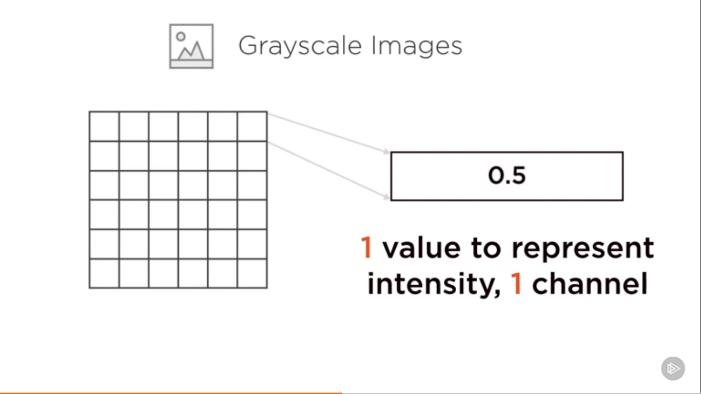
**Representing Images in Numeric Form:**

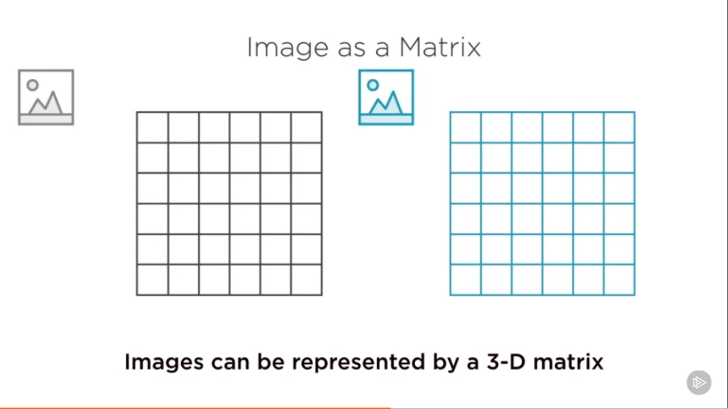
****

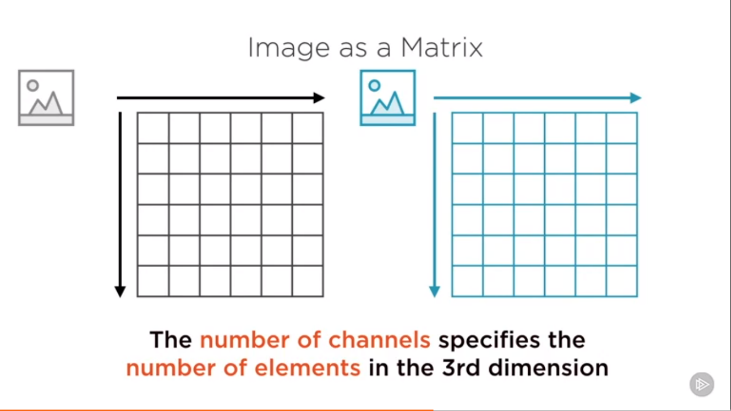
****

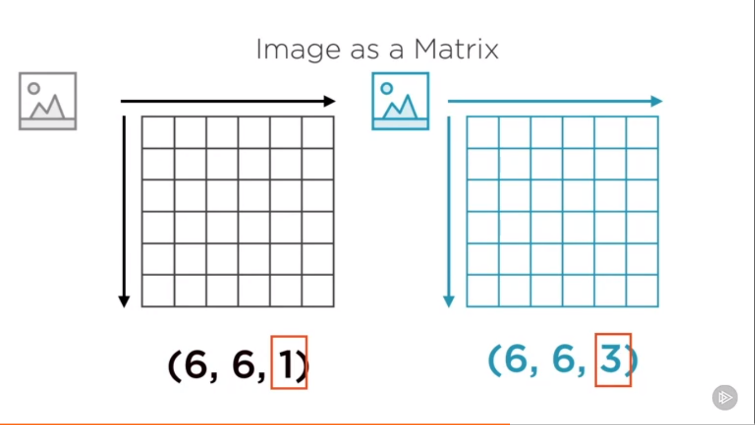
****

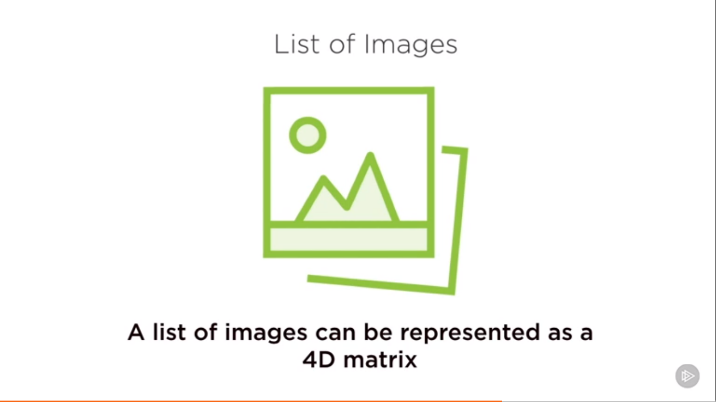
****

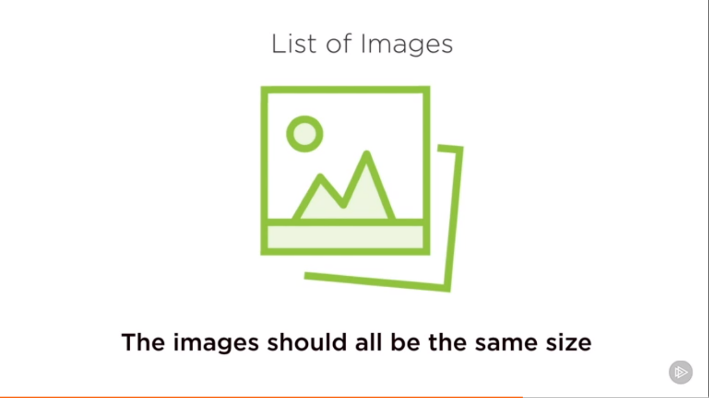
****

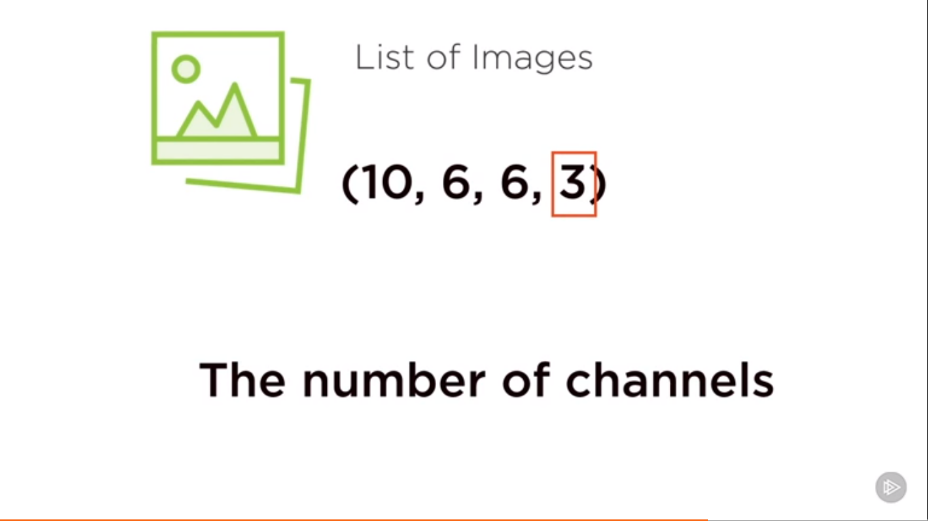
****

****

****

****

****

****

****

****

**Demo Extracting Features from image:**

Use the OpenCV library for image processing