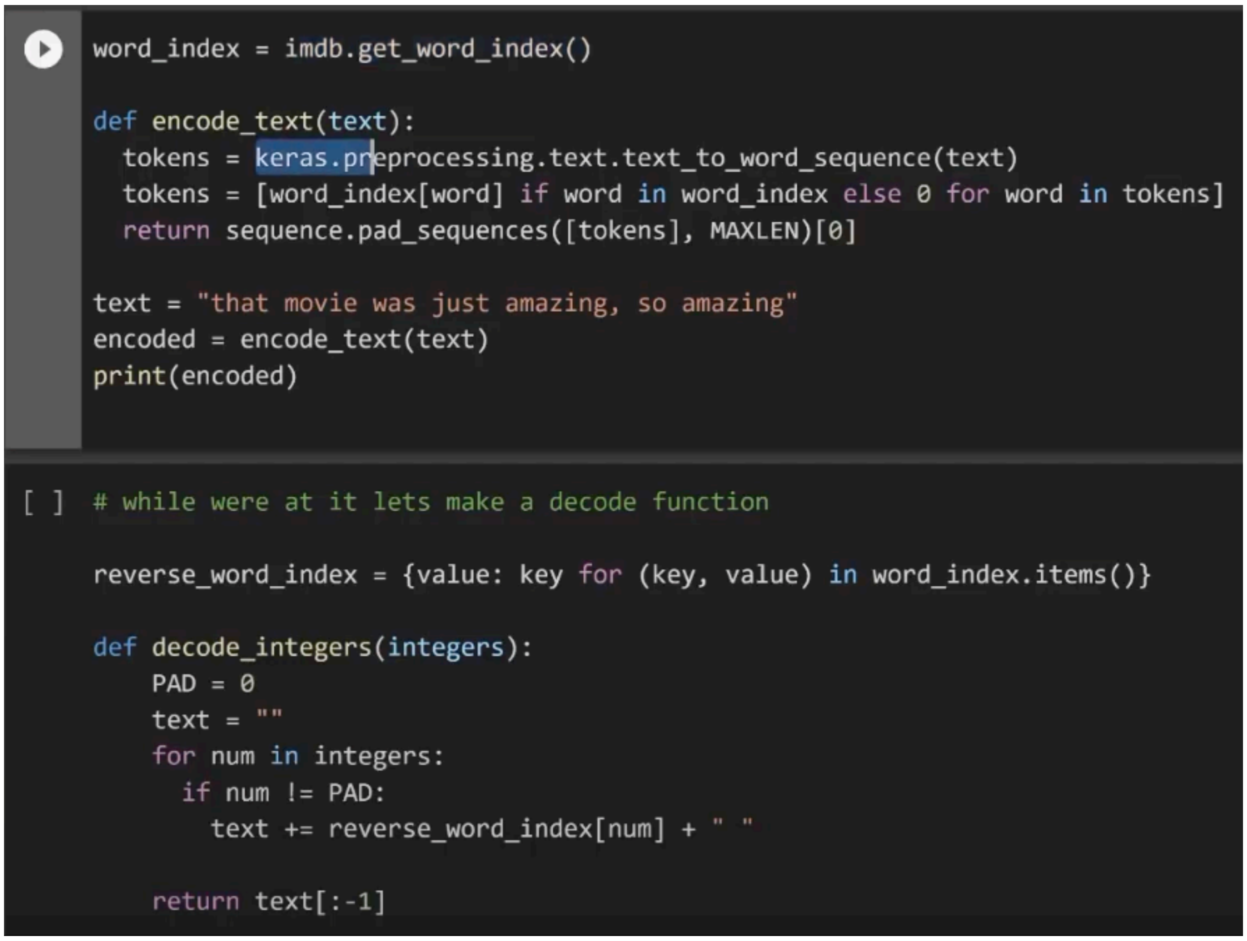
- -> the model has been trained
- ***using the model to make predictions***
    - ***pre-processing the data***
        - -> the data which is input into the model needs to be in the same format as the data which was used to train it
        - -> he's loaded in movie reviews from imdb
            - ***then keras preprocessing***
                - turning the text into individual words / tokens

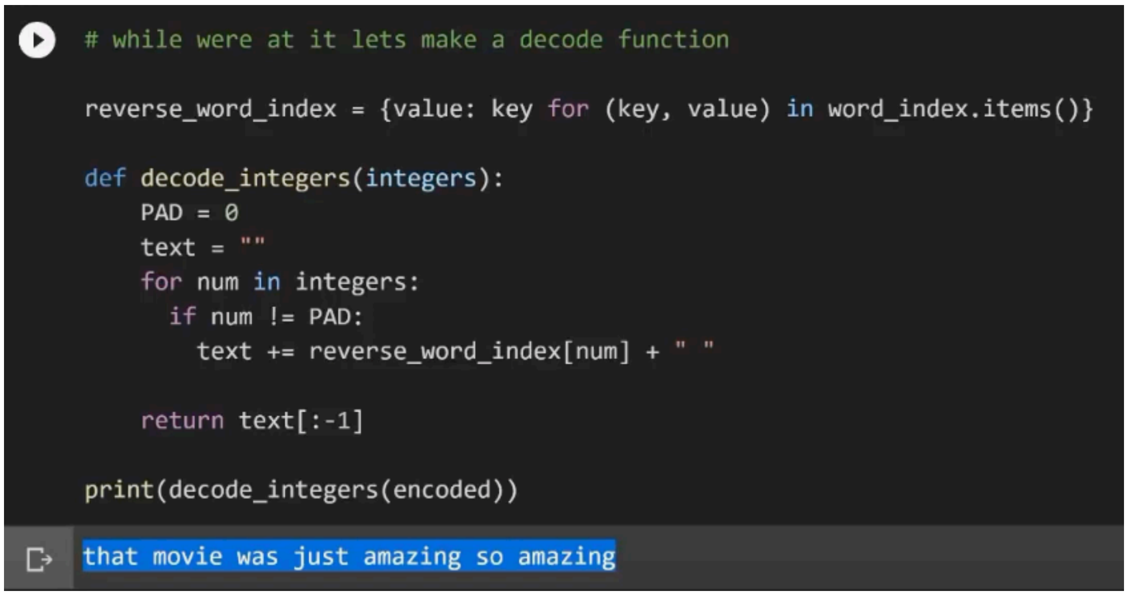


```
word_index = imdb.get_word_index()

def encode_text(text):
  tokens = keras.preprocessing.text.text_to_word_sequence(text)
  tokens = [word_index[word] if word in word_index else 0 for word in tokens]
  return sequence.pad_sequences([tokens], MAXLEN)[0]

text = "that movie was just amazing, so amazing"
encoded = encode_text(text)
print(encoded)
```

```
# while were at it lets make a decode function

reverse_word_index = {value: key for (key, value) in word_index.items()}

def decode_integers(integers):
    PAD = 0
    text = ""
    for num in integers:
      if num != PAD:
        text += reverse_word_index[num] + " "

    return text[:-1]
```

                - -> the function he's defined is to process the data which we're importing to run the model on
            - ***-> then iterating through the different tokens***
                - -> iterating through the words in the reviews which are being imported and if they're not in our training model then setting them equal to 0
                - -> so the only words which are in the reviews which are passed into the models to make predictions are words it actually knows <- we're removing the words it doesn't know
                - -> returning the first index of the pad sequence
                - -> a list of lists
            - ***-> the output of this cell***
                - -> it's returning the movie review in an array with a lot of zeros - these are the words which the function we defined removed - because they weren't in the training data
            - ***-> making a decode function which goes from number to text***
                - -> to go from the numbers to the words
                - -> i.e at the moment we have numbers used to represent words -> we want a function which can convert those numbers back into words
                - ***-> it's defining a function***
                    - -> if the number isn't 0
                    - -> then add the reverse lookup and return everything apart from the last

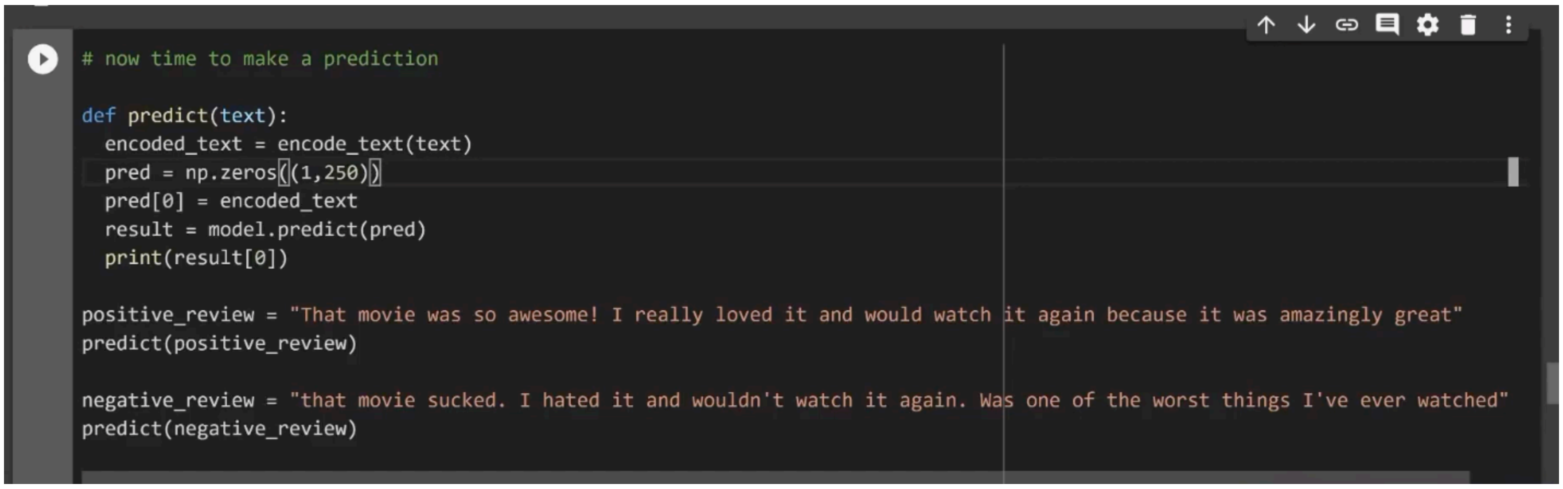
space which it would have added



```
# while were at it lets make a decode function

reverse_word_index = {value: key for (key, value) in word_index.items()}

def decode_integers(integers):
    PAD = 0
    text = ""
    for num in integers:
      if num != PAD:
        text += reverse_word_index[num] + " "

    return text[:-1]

print(decode_integers(encoded))
```

```
that movie was just amazing so amazing
```

- ***-> defining the predict function***



```
# now time to make a prediction

def predict(text):
  encoded_text = encode_text(text)
  pred = np.zeros((1,250))
  pred[0] = encoded_text
  result = model.predict(pred)
  print(result[0])

positive_review = "That movie was so awesome! I really loved it and would watch it again because it was amazingly great"
predict(positive_review)

negative_review = "that movie sucked. I hated it and wouldn't watch it again. Was one of the worst things I've ever watched"
predict(negative_review)
```

  - -> the argument is the text
  - -> then turning it into numbers, removing the spaces
  - -> then creating a blank numpy array which is full of zeros
  - -> the shape which the model expects is something by 250
  - -> then inserting the entry into that text
  - -> then printing the result
- ***-> then testing the function***
  - -> the values it's returning are the percentage positivity that the review is
  - -> the presence of certain words makes a huge difference -> especially for shorter reviews
  - -> e.g removing the word "awesome" decreases the positivity score of the model by 10%
  - -> he's removing certain words to see how the score changes
  - -> before you make a prediction with your own review, you should use the encodings from the training dataset to encode the review you are working with