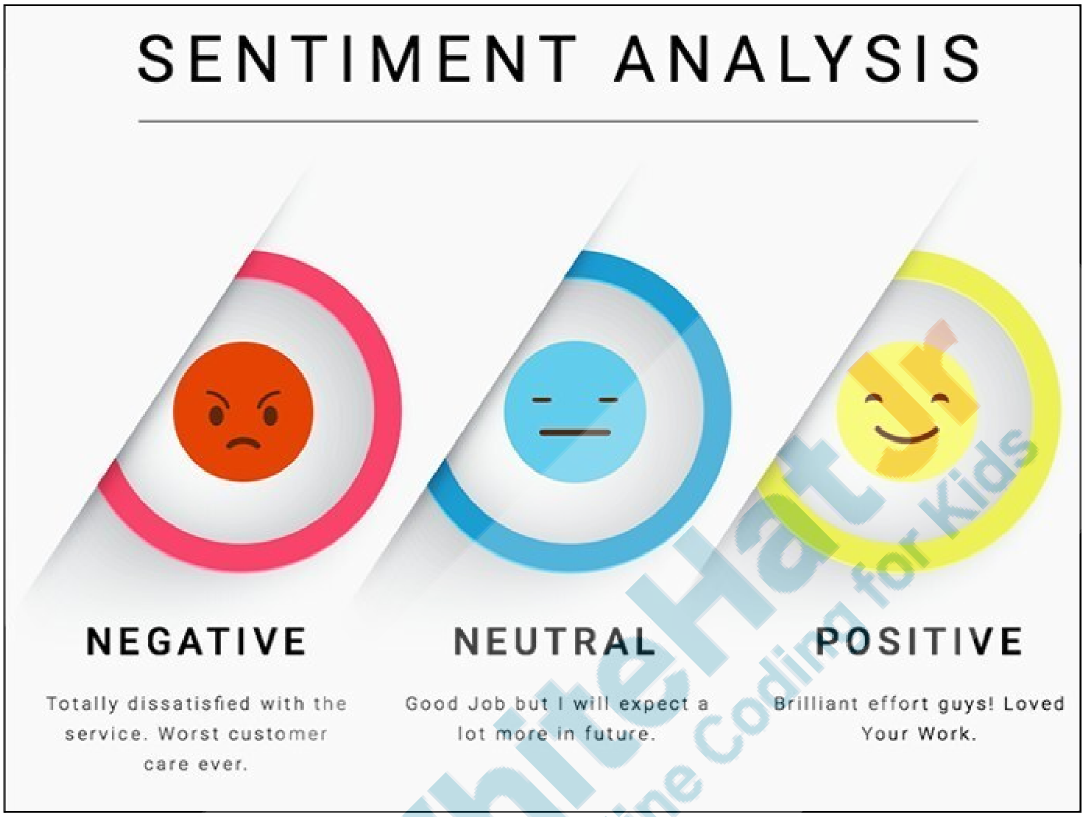
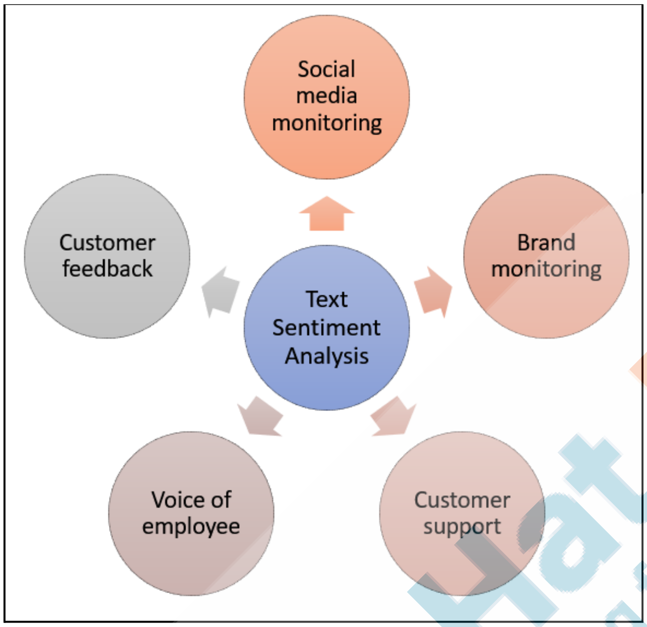
Topic	TEXT DATA TOKENIZATION	
Class Description	**The student will learn to preprocess text for sentiment analysis.**	
Class	**PRO C114**	
Class time	**45 mins**	
Goal	● Learn to preprocess text data ● To perform Tokenization and Padding	
Resources Required	● Teacher Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Smartphone ● Student Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen	
Class structure	**Warm-Up** **Teacher-led Activity 1** **Student-led Activity 1** **Wrap-Up**	**10 mins** **10 mins** **20 mins** **05 mins**
Credit	Tensorflow by Google Brain team. Keras by Google Engineer Francois Chollet. Kaggle: Emotion Detection from Text	
WARM-UP SESSION - 10 mins		
Teacher Starts Slideshow **Slide 1 to 4**		

Refer to speaker notes and follow the instructions on each slide.	
Teacher Action	**Student Action**
Hey <student's name>! It's great to see you! How are you? Can you tell me what we learned in the previous class? ***Note**: Encourage the student to give answers and be more involved in the discussion.* Amazing! Till now we know how computers see the world and recognize images and objects. We also learned how we can use the computer to classify images into different categories.	**ESR**: Hi, I am good! **ESR**: We learned to compile, train and test the model. We were able to detect the Chest X-Ray Images with Pneumothorax disease.
WARM-UP QUIZ Click on In-Class Quiz	
Continue WARM-UP Session **Slide 5 to 22**	
Have you seen spam message blockers on mobile phones? What do you think? How does it work?	**ESR**: Yes. **ESR:** Applications are used to block spam messages and block calls

Yes but How does an application know to block a particular message or call? Let’s play a game which is known as ‘**Guess the emotion**’. As you can see certain text is displayed on the screen you have to guess the emotion related to the text. Great! Do you know why you were able to get them right? You are able to get it right because your brain is trained to identify the emotions or sentiments behind these sentences. If I give the same text to the computer will it be able to guess the correct emotion of the same text? This process of analyzing the attitude, view or emotion behind a text is known as **Text Sentiment Analysis.**	**ESR:** The app has some data using which it can recognize spam messages and unwanted calls. **ESR:** No

 SENTIMENT ANALYSIS NEGATIVE Totally dissatisfied with the service. Worst customer care ever. NEUTRAL Good Job but I will expect a lot more in future. POSITIVE Brilliant effort guys! Loved Your Work.	
Analyzing sentiment related to text or speech is a human quality. If we want machines to behave like humans, they should be able to evaluate the sentiment related to text or speech. Thus, We'll be creating a machine learning model to analyze the emotions from any text. Sentiment analysis is used in many different areas such as: ● Social media monitoring. ● Customer support. ● Customer feedback. ● Brand monitoring and reputation management. ● Voice of employee etc.	**ESR:** Yes.



So, today we’ll be learning about processing text that machines can analyze. We’ll be creating a model to predict the sentiments related to text.

Tell me about some qualities that human beings posess?

Can we give this ability to computers?

ṁ

For this, we will be looking at a concept called NLP. NLP is a branch ...NLP is the branch of artificial intelligence or AI, concerned with giving computers the ability to understand the text and spoken words just like a human being.

NLP makes it possible for computers to

- read text
- hear speech
- interpret text and speech
- measure sentiment and determine which parts are important.

ESR: Humans listen and speak, play, read etc.

ESR: Yes we can give this ability to computers in order to make them perform tasks as humans do

Sentiment Analysis is a part of **Natural language processing (NLP).** NLP is used for: ● Translation Application ● Fake News Detection ● Classifying Emails ● Predicting Disease ● Error Detection ● IVR Application ● Sentiment Analysis ● Personal Voice Assistant	
Are you excited to learn about making a computer identify the emotions present in text? Let's get started.	**ESR:** Yes
Teacher Ends Slideshow	
TEACHER-LED ACTIVITY - 10 mins	
Teacher Initiates Screen Share	
<u>ACTIVITY</u> ● **Text Data Tokenization** ● **Padding the tokenized sequence**	
Teacher Action	**Student Action**
In previous classes, we **preprocessed** images to classify them right? Which operations did we perform on images to extract useful information from them?	**ESR:** Yes **ESR:** The steps involved: 1. Mapping each image with a label.

Note: Help the student to recollect how images are pixel values arrays(that we covered in earlier classes)

Great!

But there is a problem that we can face while dealing with text. We can have text written in paragraphs or in tabular format. Text contains punctuation marks with styling such as bold, italics etc. What do you think we can do with text to analyze it for different purposes?

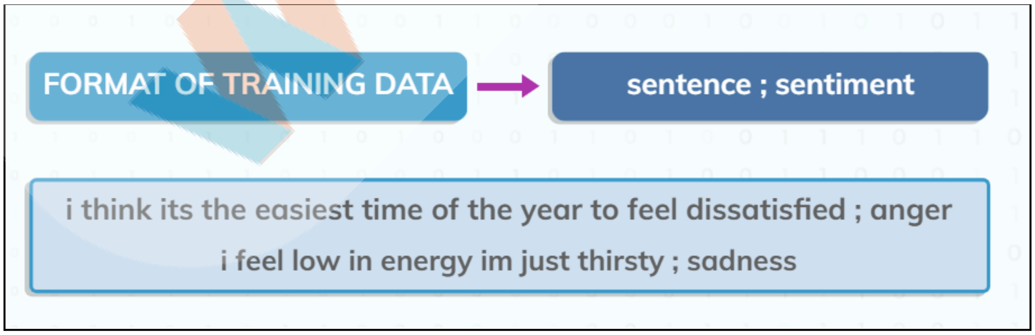
So, we need to filter this text and represent it in a format that is easier to understand and process. For this purpose, we'll be using a Python library known as "**Pandas**".

Note: *Open Teacher activity 1 to show the data file to the student.*

As you can see, sentiment is written along with each and every sentence separated by a semicolon. We have to train our model with this data so that it can predict sentiment or emotion related to any text given to it.

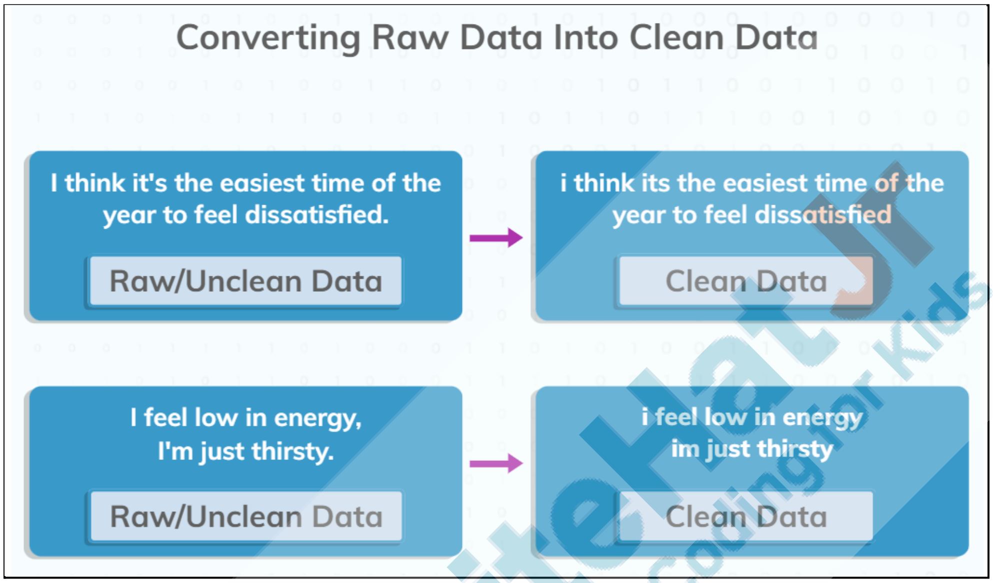
2. Converting all images to the same-sized array.
3. Augmenting image data

ESR: We can convert them into arrays of numbers using some operation.



i think its the easiest time of the year to feel dissatisfied ; anger
i feel low in energy im just thirsty ; sadness

The training data that is provided to you for today's project is clean data meaning it doesn't have any styling or punctuation marks. The text is in lowercase and it is provided in an MS Excel file.



Converting Raw Data Into Clean Data

I think it's the easiest time of the year to feel dissatisfied.

Raw/Unclean Data

→

i think its the easiest time of the year to feel dissatisfied

Clean Data

I feel low in energy, I'm just thirsty.

Raw/Unclean Data

→

i feel low in energy im just thirsty

Clean Data

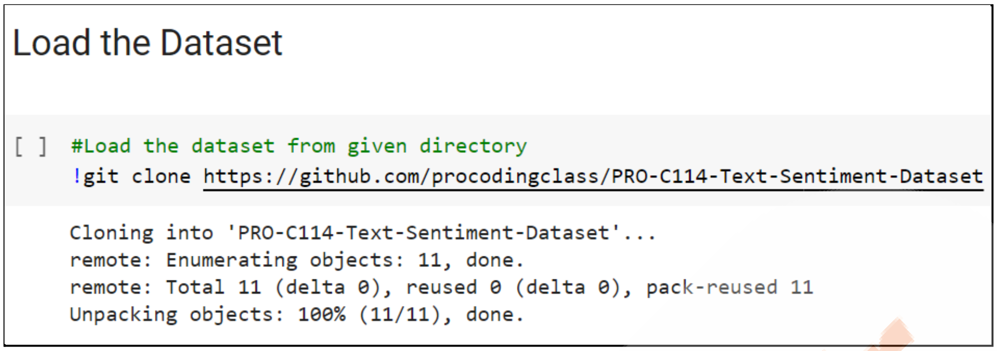
Let’s start with importing the dataset into our code.

Note: Open Teacher activity 2 for the boilerplate code and run all the cells.

Load text dataset:

1. Open Google Colaboratory *Teacher activity 2*
2. Clone the data into the Google Colab Notebook using the following command:

```
!git clone <GitHub Repository URL>
```



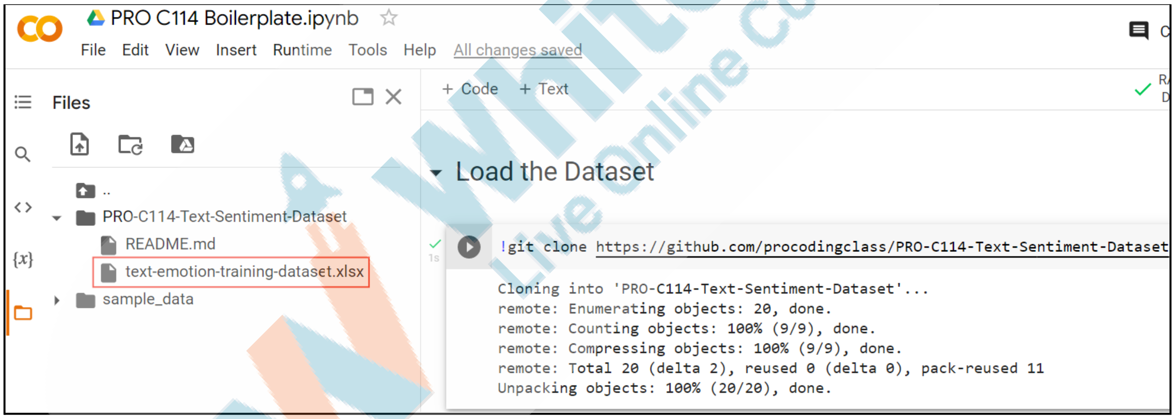
Load the Dataset

```
[ ]  #Load the dataset from given directory
     !git clone https://github.com/procodingclass/PRO-C114-Text-Sentiment-Dataset

     Cloning into 'PRO-C114-Text-Sentiment-Dataset'...
     remote: Enumerating objects: 11, done.
     remote: Total 11 (delta 0), reused 0 (delta 0), pack-reused 11
     Unpacking objects: 100% (11/11), done.
```

***Note**: The data will be cloned into the current Google Colab Notebook while you are working.*

Once the data is cloned:

1. Open the "**Table of contents**" icon on the left side.
2. Open the "**File**" icon on the left side, to see the Text sentiment dataset in xlsx format.



Load the Dataset

```
!git clone https://github.com/procodingclass/PRO-C114-Text-Sentiment-Dataset

Cloning into 'PRO-C114-Text-Sentiment-Dataset'...
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 20 (delta 2), reused 0 (delta 0), pack-reused 11
Unpacking objects: 100% (20/20), done.
```

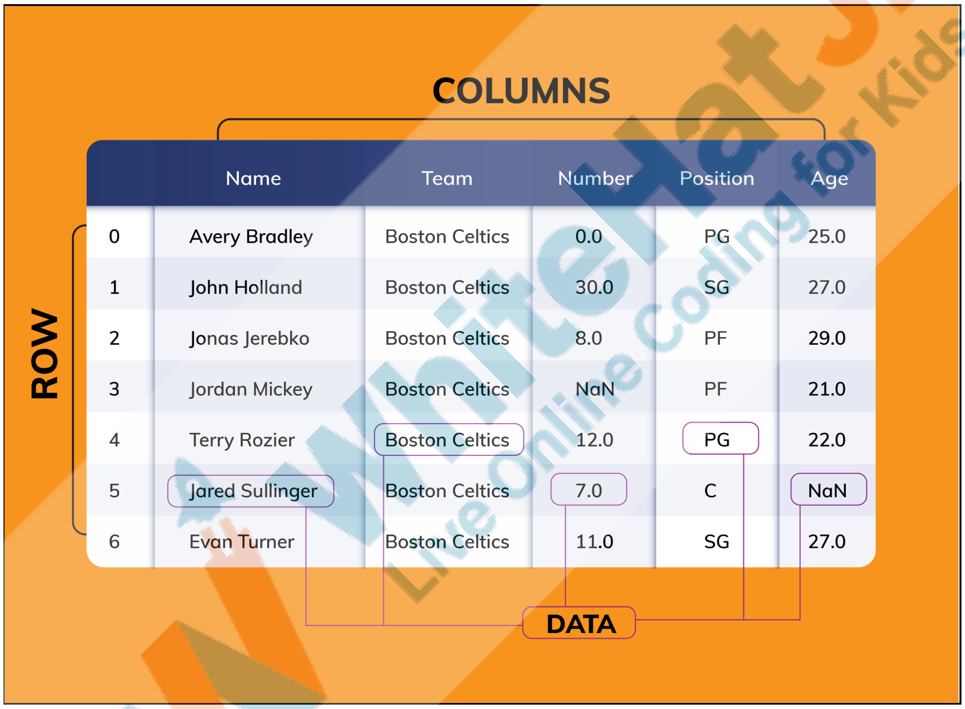
The next step is to represent the dataset. As mentioned before, we'll be using **Pandas** to visualize the text dataset.

Teacher Activity 3:
Reference: Pandas library

Pandas:

Pandas is an open-source library built on top of Numpy and Matplotlib. It provides high-performance, easy-to-use data structures and data analysis tools.

A **DataFrame** is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns same as our testing dataset. Pandas DataFrame consists of three principal components, the **data**, **rows**, and **columns**.

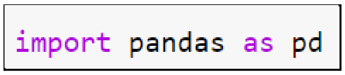


	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

Pandas DataFrame will be created by loading the datasets from existing MS Excel files, CSV files or SQL Database. Pandas DataFrame can also be created from the lists, dictionaries etc.

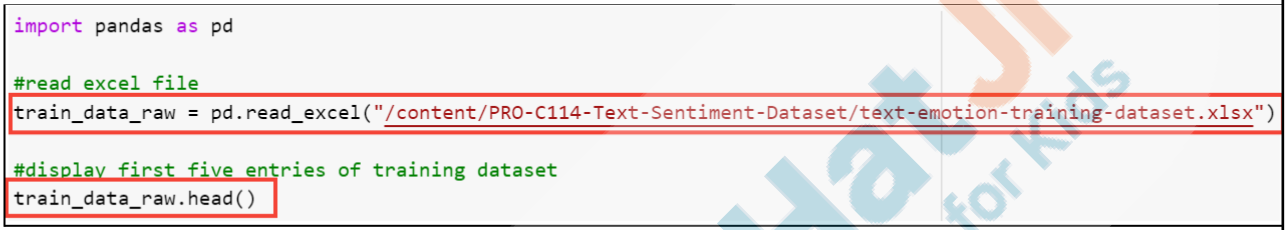
Let’s view our training data first:

1. Import **pandas** as pd



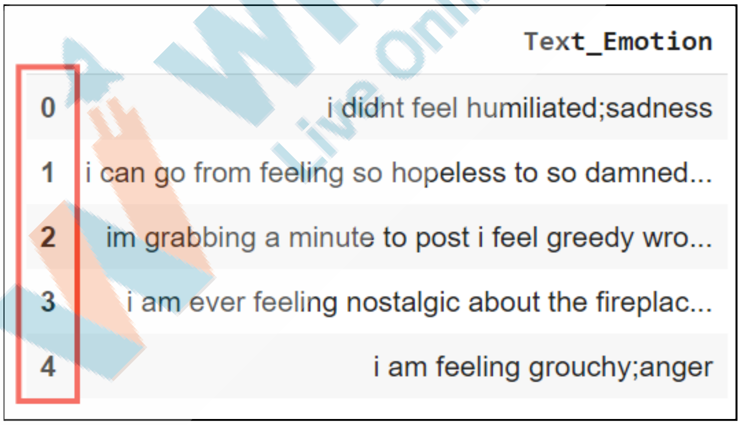
```
import pandas as pd
```

2. Initialize a DataFrame **train_data_raw** for reading the dataset present in the excel file.
3. The **pd.read_excel()** method is used to read the excel file. Provide the path of the training dataset in this method.
4. The **head(n)** method is used to display the first n entries(rows) of the dataset. By default **head()** method displays the first five entries.



```
import pandas as pd

#read excel file
train_data_raw = pd.read_excel("/content/PRO-C114-Text-Sentiment-Dataset/text-emotion-training-dataset.xlsx")

#display first five entries of training dataset
train_data_raw.head()
```

As you can see in the output, by default, an **index** column is added with the number of entries/rows by the **head()** method. The first row of the dataset is considered as the **heading** of the DataFrame.



	Text_Emotion
0	i didnt feel humiliated;sadness
1	i can go from feeling so hopeless to so damned...
2	im grabbing a minute to post i feel greedy wro...
3	i am ever feeling nostalgic about the fireplac...
4	i am feeling grouchy;anger

Our next step will be to segregate the text and emotion.

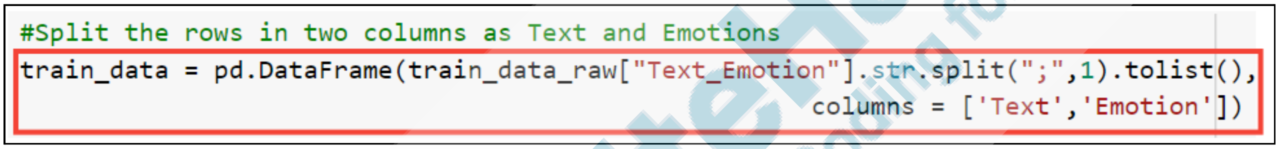
1. Define a new DataFrame **train_data.**
2. Parts of text that are currently separated by a semicolon in the dataset, can be separated by the **str.split(separator, maxsplit)** function.

Separator: The delimiter string. The default separator is any whitespace character such as space, \t, \n, etc.

Maxsplit: (optional) Defines the maximum number of split operations that can be done. Thus, the list can contain at most maxsplit + 1 elements. The default maxsplit is -1 that means unlimited splits.

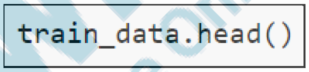
The **str.split()** function returns a list with string elements. These are listed in different columns.

columns: Defines the header of the column.



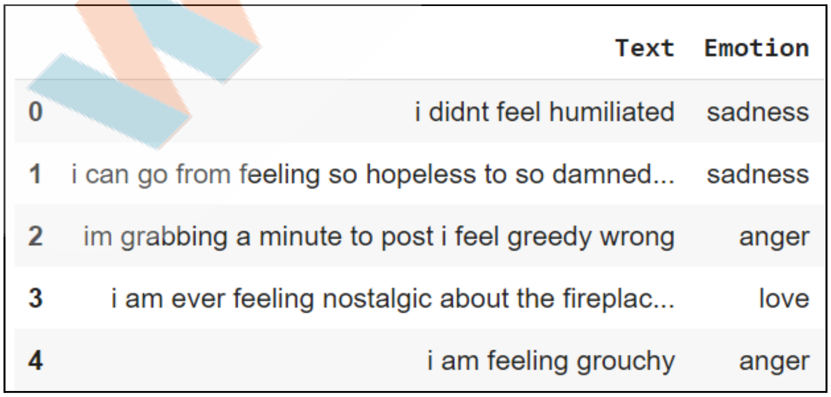
```
#Split the rows in two columns as Text and Emotions
train_data = pd.DataFrame(train_data_raw["Text_Emotion"].str.split(";",1).tolist(),
                                          columns = ['Text','Emotion'])
```

3. After separating the sentiments from text, use **the head()** function for displaying **the DataFrame.**



```
train_data.head()
```

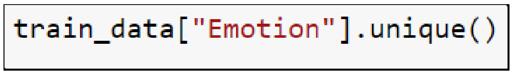
It will give you the first five entries with **Text** and **Emotion** as column header.



	Text	Emotion
0	i didnt feel humiliated	sadness
1	i can go from feeling so hopeless to so damned...	sadness
2	im grabbing a minute to post i feel greedy wrong	anger
3	i am ever feeling nostalgic about the fireplac...	love
4	i am feeling grouchy	anger

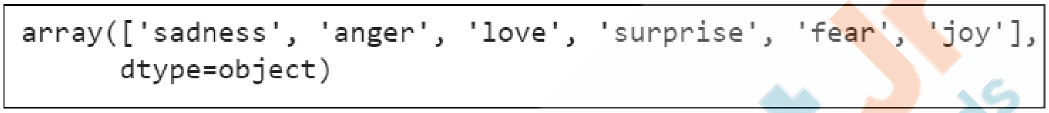
As you can see different emotions are assigned to the text.

We will find out unique emotions which are listed here. Using the **unique()** method on the column ‘Emotion’ will return an array of unique emotions.



```
train_data["Emotion"].unique()
```

The **unique()** method will return a list of unique elements in the column emotion.

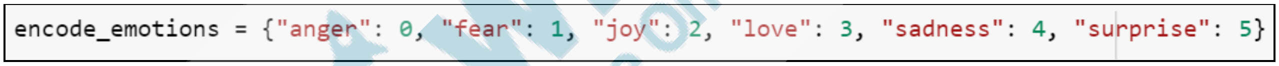


```
array(['sadness', 'anger', 'love', 'surprise', 'fear', 'joy'],
      dtype=object)
```

When we had created **classes** in our image classification project, what was the next step?

YES! Giving labels to these classes.

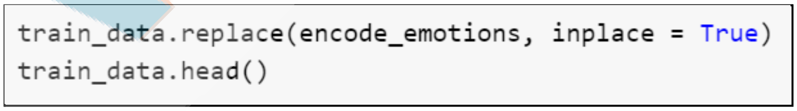
So, the next step is to assign labels to these emotions. We will create a dictionary where the **emotions** are **keys** and the **labels** are **values**.



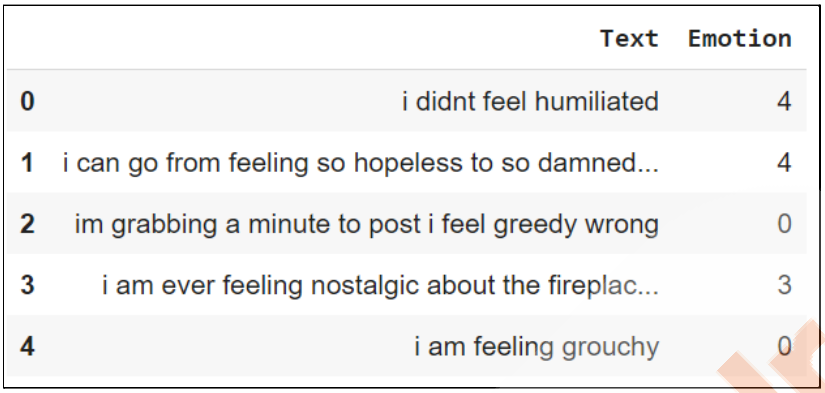
```
encode_emotions = {"anger": 0, "fear": 1, "joy": 2, "love": 3, "sadness": 4, "surprise": 5}
```

Let's replace the emotion with the respective labels. To serve this purpose we will use the **replace()** method.

Once this is done, let’s use the **head()** method to display the DataFrame.



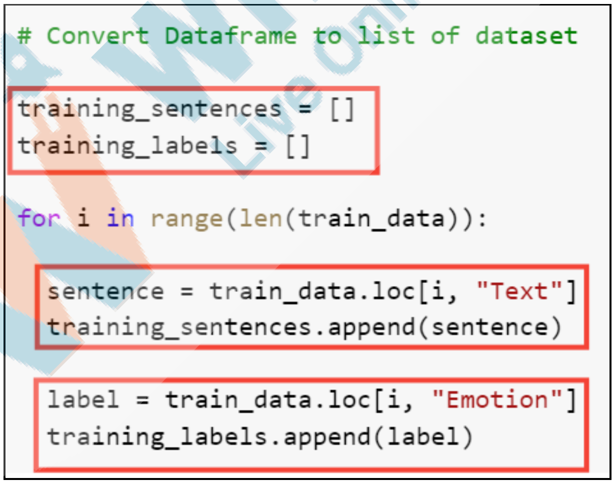
```
train_data.replace(encode_emotions, inplace = True)
train_data.head()
```



	Text	Emotion
0	i didnt feel humiliated	4
1	i can go from feeling so hopeless to so damned...	4
2	im grabbing a minute to post i feel greedy wrong	0
3	i am ever feeling nostalgic about the fireplac...	3
4	i am feeling grouchy	0

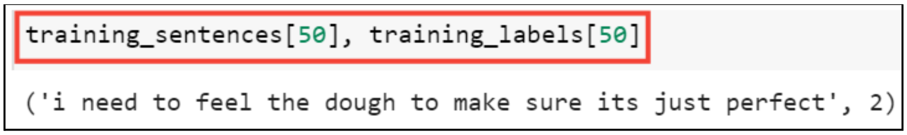
Machine learning model takes numbers as input. This means that we will need to convert the text into numerical value. Steps to prepare the dataset for conversion:

1. Two lists are created to store sentences and labels.
2. Using a loop we can append the data in the respective lists.
3. The **loc()** method is used for accessing the DataFrame elements using the **index number** and **column header**.



```
# Convert Dataframe to list of dataset

training_sentences = []
training_labels = []

for i in range(len(train_data)):

  sentence = train_data.loc[i, "Text"]
  training_sentences.append(sentence)

  label = train_data.loc[i, "Emotion"]
  training_labels.append(label)
```

Let’s check the text and label at a random location.



```
training_sentences[50], training_labels[50]
```

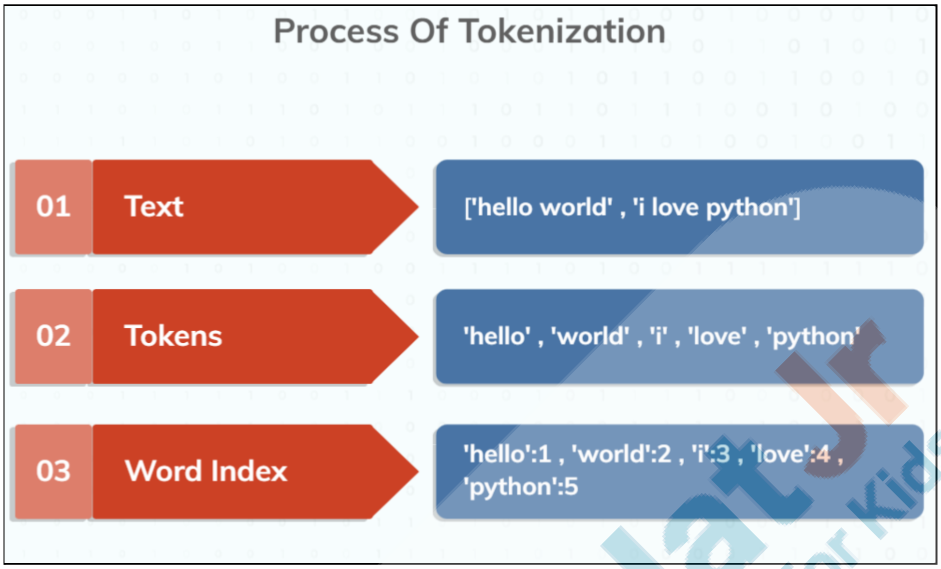
```
('i need to feel the dough to make sure its just perfect', 2)
```

The act of converting text into numbers is known as **Tokenization.** The **Tokenizer** class of **Keras** is used for encoding text input into integer sequences.

Padding is important to make all the sentences contain the same number of words. Zero is used for padding the tokenized sequence to make text contain the same number of tokens.

Steps carried out for tokenization:

1. The process starts with creating tokens for each and every sentence by breaking them into words.
2. These tokens are assigned a numerical value according to their frequency of occurrence by creating a **word_index** dictionary.
3. According to the word_index, sequences are created for each and every text.
4. Padding is done by adding zero to these sequences.



Process Of Tokenization

01 Text → ['hello world' , 'i love python']

02 Tokens → 'hello' , 'world' , 'i' , 'love' , 'python'

03 Word Index → 'hello':1 , 'world':2 , 'i':3 , 'love':4 , 'python':5

Do you remember we had images in the form of pixel arrays?

Similarly for converting the text into a number sequence we will use the **Tokenizer** class of **Keras**.

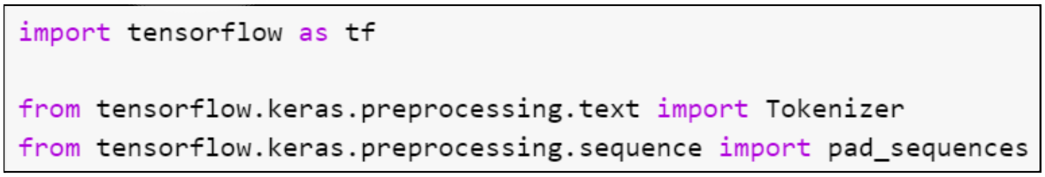
***Note**: Help the student to recollect the **ImageDataGenerator** Class(Keras) and images as pixel arrays.*

Steps to encode text into numbers:

1. Import **Tokenizer** and **pad_sequence** classes from **tensorflow.keras**.

Teacher activity 4:
Reference: Tokenizer for preprocessing text

ESR: Yes.



```
import tensorflow as tf

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

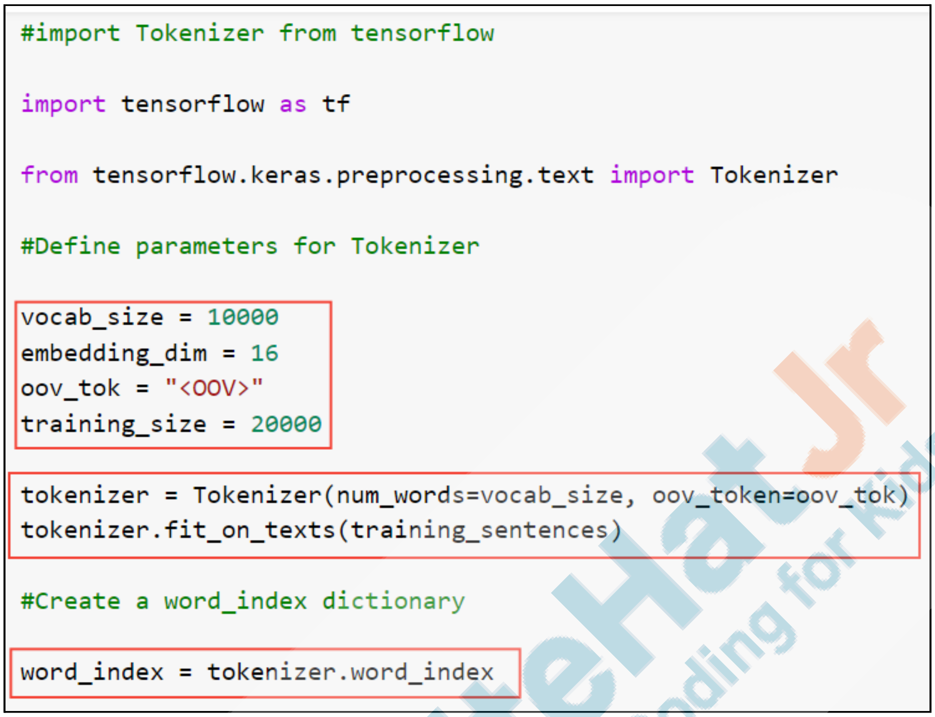
2. **Tokenizer** takes two parameters to tokenize the text:

    a. **Num_words:** the maximum number of words allowed , based on word frequency. Define a variable **vocab_size=1000.**

    b. **oov_token:** It is used to replace the out-of-vocabulary words defined in the **Tokenizer** class during text_to_sequence calls. Define **oov_tok= '<OOV>'** to replace the out-of-vocabulary words with **<OOV>**.
    After tokenization, the words are added into a dictionary called **word_index** using the **fit_on_text()** method. This method creates the vocabulary index based on word frequency.
    For example, if you have "The cat sat on the mat." It will create a dictionary as:

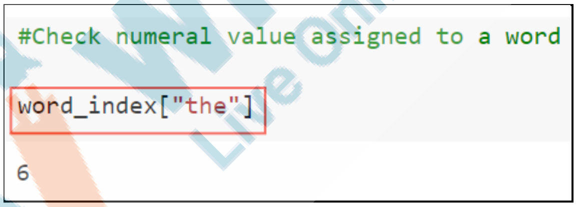
    word_index["the"] = 1; word_index["cat"] = 2.
    So, every word gets a unique integer value.
    0 is reserved for padding. So lower integer means more frequent words

Define a variable **word_index** to save the dictionary using the **tokenizer.word_index()** method.



```
#import Tokenizer from tensorflow

import tensorflow as tf

from tensorflow.keras.preprocessing.text import Tokenizer

#Define parameters for Tokenizer

vocab_size = 10000
embedding_dim = 16
oov_tok = "<OOV>"
training_size = 20000

tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(training_sentences)

#Create a word_index dictionary

word_index = tokenizer.word_index
```

Check any random word for the index given to it.



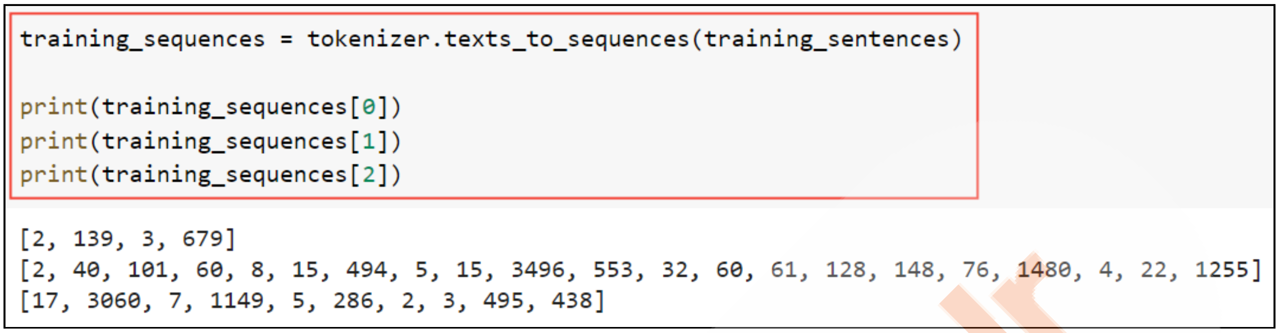
```
#Check numeral value assigned to a word

word_index["the"]

6
```

After assigning numerical values to text, let’s create sequences for sentences.

texts_to_sequences():

It transforms each text or sentence in **training data** into a sequence of integers. So it basically takes each word in the text and replaces it with its corresponding integer value from the **word_index** dictionary.

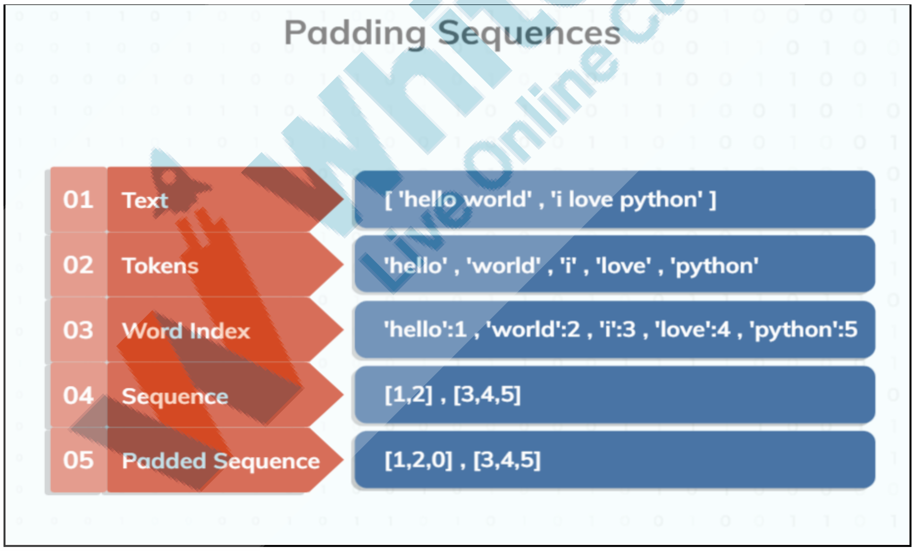
So the sequence is based on text or sentence and the numbers are assigned by the **word_index** dictionary



```
training_sequences = tokenizer.texts_to_sequences(training_sentences)

print(training_sequences[0])
print(training_sequences[1])
print(training_sequences[2])
```

```
[2, 139, 3, 679]
[2, 40, 101, 60, 8, 15, 494, 5, 15, 3496, 553, 32, 60, 61, 128, 148, 76, 1480, 4, 22, 1255]
[17, 3060, 7, 1149, 5, 286, 2, 3, 495, 438]
```

The next step is to pad these sequences. Use the **pad_sequences()** method for padding the sequence.It is used to ensure that all sequences in a list have the same length.
By default, this is done by padding **0** at the beginning or at the end of each sequence until each sequence has the same length as the longest sequence.



Padding Sequences

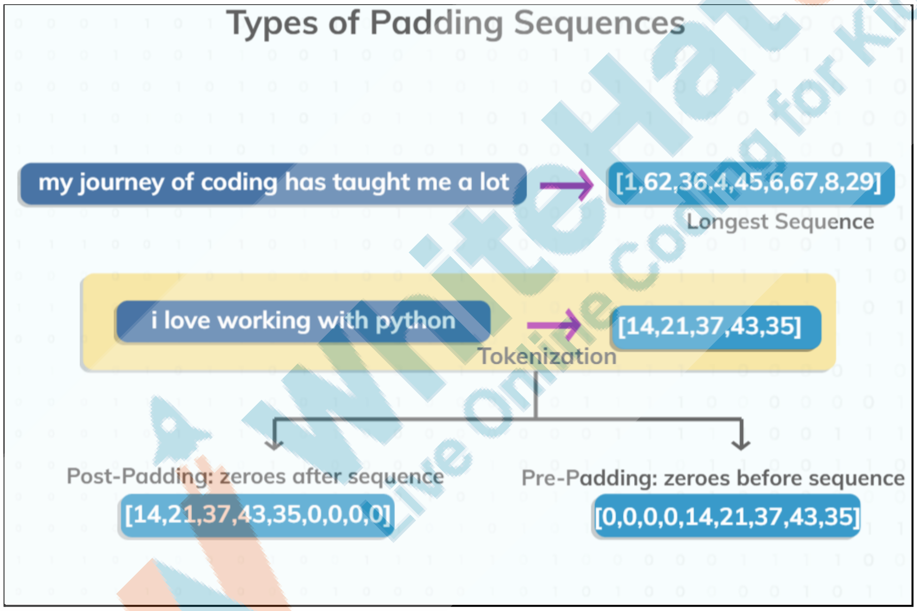
01 Text ['hello world' , 'i love python']
02 Tokens 'hello' , 'world' , 'i' , 'love' , 'python'
03 Word Index 'hello':1 , 'world':2 , 'i':3 , 'love':4 , 'python':5
04 Sequence [1,2] , [3,4,5]
05 Padded Sequence [1,2,0] , [3,4,5]

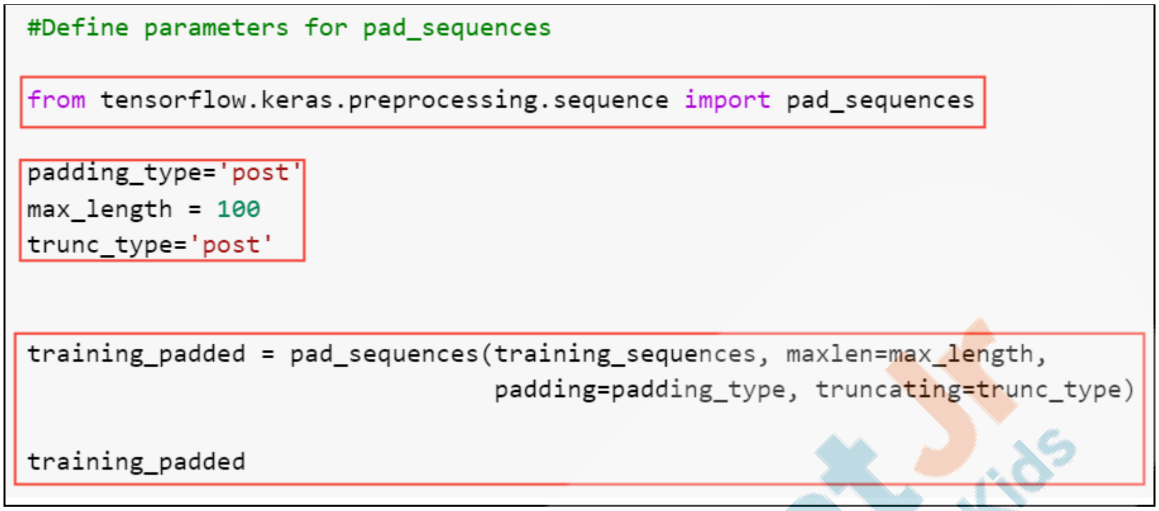
The method takes the following arguments along with the **training_sequences**:

1. **maxlen:** Maximum length of all sequences. If not provided, sequences will be padded to the length of the longest individual sequence.

2. **padding:** It takes values as 'pre' or 'post' (optional, defaults to 'pre'), pad either before or after each sequence.

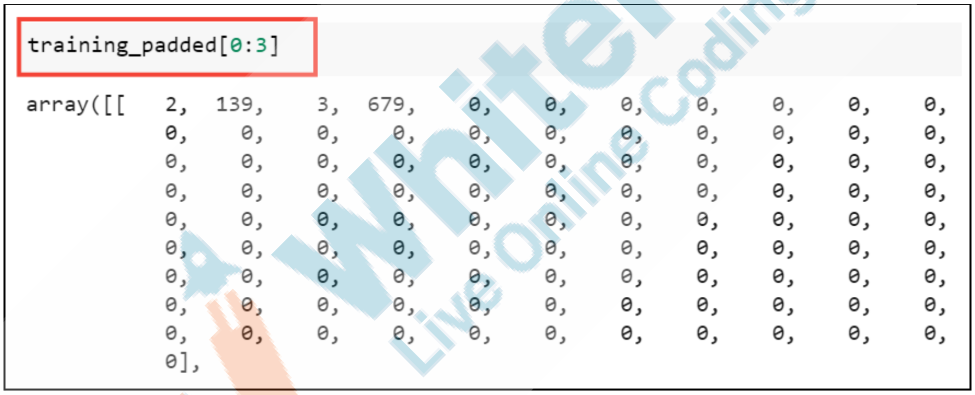
3. **truncating:** remove values from sequences larger than maxlen, either at the beginning or at the end of the sequences.


Types of Padding Sequences
my journey of coding has taught me a lot
[1,62,36,4,45,6,67,8,29]
Longest Sequence
i love working with python
[14,21,37,43,35]
Tokenization
Post-Padding: zeroes after sequence
[14,21,37,43,35,0,0,0,0]
Pre-Padding: zeroes before sequence
[0,0,0,0,14,21,37,43,35]



```
#Define parameters for pad_sequences

from tensorflow.keras.preprocessing.sequence import pad_sequences

padding_type='post'
max_length = 100
trunc_type='post'


training_padded = pad_sequences(training_sequences, maxlen=max_length,
                                padding=padding_type, truncating=trunc_type)

training_padded
```

The output after padding is given in the form of a 2D array.



```
training_padded[0:3]
```

```
array([[   2,  139,    3,  679,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0],
```

Ṁ
Based on the above activity, you can see that we were able to convert each sentence into a fixed-length array of numbers.

Now, It’s your turn to perform the same operation.

Teacher Stops Screen Share

So now it’s your turn.
Please share your screen with me.

Teacher Starts Slideshow Slide 23 to 24 <**Note**: Only Applicable for Classes with VA> Refer to speaker notes and follow the instructions on each slide.	
We have one more class challenge for you. Can you solve it? Let's try. I will guide you through it.	
Teacher Ends Slideshow	
STUDENT-LED ACTIVITY - 20 mins	
- **Ask the student to press the ESC key to come back to the panel.** - **Guide the student to start Screen Share.** - **The teacher gets into Fullscreen.**	
ACTIVITY - **Text Data Preprocessing** - **Display the text data using Pandas DataFrame** - **Perform Tokenization and Padding on the sequences**	
Teacher Action	**Student Action**
Guide the student to download the boilerplate code *Student Activity 1.* ***Note 1**: The student will perform the same activities as done by the teacher.* ***Note 2**: Please refer to Teacher Activities to follow through the steps and guide the student to complete the activities.*	
Guide the kid to perform the activities. 1. Load the dataset using the link. 2. Use pandas to display the data in the tabular	

format. (DataFrame)

3. Use the Tokenizer class to tokenize and pad the data.
4. Display the tokenized and padded sequences

You did amazing work today!

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 5 mins

Teacher Starts Slideshow
Slide 25 to 30
Refer to speaker notes and follow the instructions on each slide.

Activity details

Following are the WRAP-UP session deliverables:

- Appreciate the student.
- Revise the current class activities.
- Discuss the quizzes.

WRAP-UP QUIZ
Click on In-Class Quiz

Continue WRAP-UP Session
Slide 31 to 36
<**Note**: Only Applicable for Classes with VA>

Activity Details
Following are the session deliverables:

- Explain the facts and trivia
- Next class challenge will be finding out the layers required for creating a text analysis model
- Project for the day is to preprocess text data for customer review analysis

FEEDBACK

- **Appreciate the student for his/her efforts in the class.**
- **Ask the student to make notes for the reflection journal along with the code they wrote in today's class.**

Teacher Action	Student Action
You get Hats off for your excellent work! In the next class, we will learn about creating a model to analyze the sentiments of different sentences.	*Make sure you have given at least 2 Hats Off during the class for:*

PROJECT OVERVIEW DISCUSSION

Refer the document below in Activity Links Sections

Teacher Clicks ✖ End Class

ACTIVITY LINKS		
Activity Name	**Description**	**Link**
Teacher Activity 1	Dataset Link	https://github.com/procodingclass/PRO-C114-Text-Sentiment-Dataset
Teacher Activity 2	Boilerplate Code	https://colab.research.google.com/drive/1ERqOGOZAxwUWebc9v3qht0R-Idr6We0h?usp=sharing
Teacher Activity 3	Pandas Library	https://pandas.pydata.org/docs/
Teacher Activity 4	Tokenizer for preprocessing text	https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer
Teacher Activity 5	Teacher Reference code	https://colab.research.google.com/drive/1-XnOC4oGfKqDT6HGdA1guosGMIO6QWkT?usp=sharing
Teacher Reference1	Project	https://s3-whjr-curriculum-uploads.whjr.online/ac009b73-c729-4593-9e24-34eeb80acd61.pdf
Teacher Reference 2	Project Solution	https://colab.research.google.com/drive/1cp6yFYctA6mEj6HPJhf2WG04mCl21Urf?usp=sharing
Teacher Reference 3	Visual-Aid	https://s3-whjr-curriculum-uploads.whjr.online/93135e03-5900-4a3b-b517-27471710e925.html
Teacher Reference 4	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/757d2872-01aa-42f8-951f-c4fe7c4ef538.pdf
Student Activity 1	Boilerplate Code	https://colab.research.google.com/drive/1ERqOGOZAxwUWebc9v3qht0R-Idr6We0h?usp=sharing