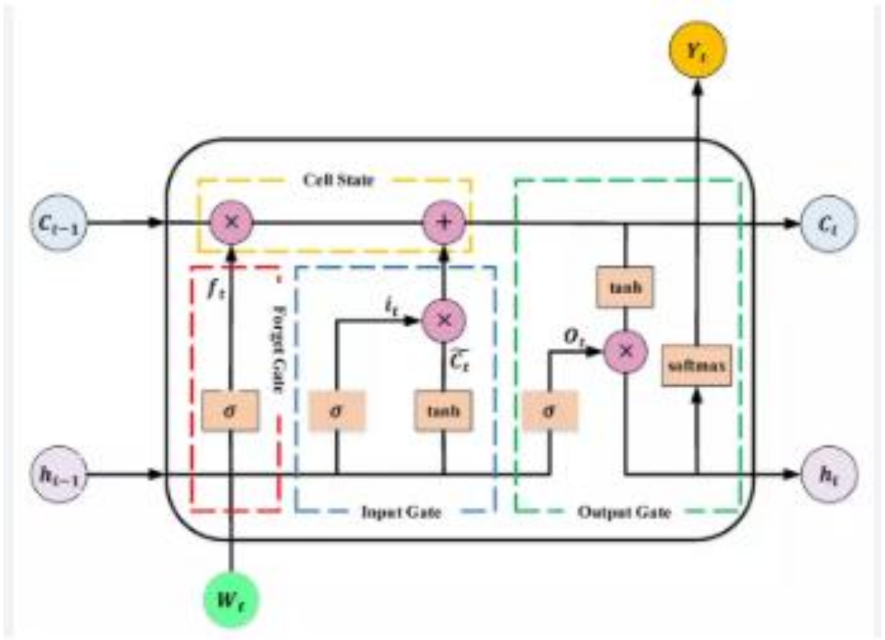
# LAB No. 8

# Implementation of Long Short-Term Memory (LSTM) Network

In this lab, students will learn and implement the **Long Short-Term Memory (LSTM)** network, an advanced type of **Recurrent Neural Network (RNN)** designed to overcome the limitations of basic RNNs. LSTM networks are capable of learning **long-term dependencies** in sequential data through a special memory cell and gating mechanism. Students will apply LSTM models to sequence classification, time-series prediction, and text-based tasks using Python and Keras, and evaluate model performance using appropriate metrics.

## Introduction & Theory (Brief)

**Long Short-Term Memory (LSTM)** is a special kind of RNN that effectively handles the **vanishing gradient problem**. It introduces a **memory cell** that can store information for long periods and three gates that regulate information flow:



## Key Components of LSTM:

- **Forget Gate** – decides what information to discard
- **Input Gate** – decides what new information to store

- **Output Gate** – controls what information to output
- **Cell State** – long-term memory

**Advantages:**

- Learns long-term dependencies
- Suitable for time-series, speech, and text data
- More stable training than basic RNNs

**Applications:**

- Time-series forecasting
- Sentiment analysis
- Speech recognition
- Language translation

**Solved Examples:**

**Example 1: LSTM for Binary Sequence Classification**

Build an LSTM model to predict whether a student **passes or fails** based on performance over multiple time steps.

Solution:

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# Sequential dataset (samples, timesteps, features)
X = np.array([
   [[1], [1], [0]],
   [[1], [1], [1]],
   [[0], [0], [1]],
   [[0], [0], [0]],
   [[1], [0], [1]]
])
```

```
y = np.array([1, 1, 0, 0, 1])

# Build LSTM model
model = Sequential()
model.add(LSTM(8, input_shape=(3, 1)))
model.add(Dense(1, activation='sigmoid'))

# Compile and train
model.compile(optimizer='adam',                              loss='binary_crossentropy',
metrics=['accuracy'])
model.fit(X, y, epochs=100, verbose=0)


# Prediction
prediction = model.predict([[[1], [1], [1]]])
print("Predicted Result (Pass=1, Fail=0):", int(prediction[0][0] > 0.5))
```

The LSTM retains relevant information across time steps using its memory cell and gating mechanism.

**Example 2: LSTM for Time-Series Prediction**

Use an LSTM network to predict the next value in a numerical time-series sequence.

Solution:

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# Time-series input sequences
X = np.array([
    [[1], [2], [3]],
    [[2], [3], [4]],
    [[3], [4], [5]],
    [[4], [5], [6]]
])

y = np.array([4, 5, 6, 7])

# Build LSTM model
model = Sequential()
model.add(LSTM(10, input_shape=(3, 1)))
model.add(Dense(1))

# Compile and train
model.compile(optimizer='adam', loss='mse')
model.fit(X, y, epochs=200, verbose=0)

# Predict next value
prediction = model.predict([[[5], [6], [7]]])
print("Predicted Next Value:", prediction[0][0])
```

LSTM captures temporal dependencies more effectively than basic RNNs for time-series prediction.

**Example 3: LSTM for Text Classification (Sentiment Analysis)**

Build an LSTM model to classify text as **positive** or **negative** sentiment.

Solution:

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

# Text samples
texts = [
    "I love this subject",
    "This lab is excellent",
    "I hate this course",
    "This topic is boring",
    "Very good explanation"
]

labels = [1, 1, 0, 0, 1]

# Tokenization
tokenizer = Tokenizer(num_words=100)
tokenizer.fit_on_texts(texts)

sequences = tokenizer.texts_to_sequences(texts)
padded_sequences = pad_sequences(sequences, maxlen=5)



# Build LSTM model
model = Sequential()
model.add(LSTM(16, input_shape=(5,)))
model.add(Dense(1, activation='sigmoid'))

# Compile and train
model.compile(optimizer='adam',                    loss='binary_crossentropy',
metrics=['accuracy'])
model.fit(padded_sequences, labels, epochs=100, verbose=0)

# Predict sentiment
predictions = model.predict(padded_sequences)
print("Predicted Sentiments:", predictions.round())
```

LSTM processes text sequences while preserving context, leading to improved sentiment classification.

## Comparison: RNN vs LSTM

| Feature | RNN | LSTM |
|---|---|---|
| Long-Term Memory | No | Yes |
| Vanishing Gradient | Yes | No |
| Training Stability | Low | High |

## LAB Assignment No.8

## LSTM code for text predictor

**Question:**

**Using the given LSTM next-word prediction model and tokenizer, write the code to input a sentence, convert it into sequences, pad it, and predict the next word using the trained model.**

**Solution:**

```
faqs = """About the Program
What is the course fee for  Data Science Mentorship Program (DSMP 2023)
The course follows a monthly subscription model where you have to make monthly payments
of Rs 799/month.
What is the total duration of the course?
The total duration of the course is 7 months. So the total course fee becomes 799*7 = Rs
5600(approx.)
What is the syllabus of the mentorship program?
We will be covering the following modules:
Python Fundamentals
Python libraries for Data Science
Data Analysis
SQL for Data Science
Maths for Machine Learning
ML Algorithms
Practical ML
```

MLOPs
Case studies

Will Deep Learning and NLP be a part of this program?
No, NLP and Deep Learning both are not a part of this program's curriculum.
What if I miss a live session? Will I get a recording of the session?
Yes all our sessions are recorded, so even if you miss a session you can go back and watch the recording.
Where can I find the class schedule?
Checkout this google sheet to see month by month time table of the course -
What is the time duration of all the live sessions?
Roughly, all the sessions last 2 hours.
What is the language spoken by the instructor during the sessions?
Hinglish
How will I be informed about the upcoming class?
You will get a mail from our side before every paid session once you become a paid user.
Can I do this course if I am from a non-tech background?
Yes, absolutely.
I am late, can I join the program in the middle?
Absolutely, you can join the program anytime.
If I join/pay in the middle, will I be able to see all the past lectures?
Yes, once you make the payment you will be able to see all the past content in your dashboard.
Where do I have to submit the task?
You don't have to submit the task. We will provide you with the solutions, you have to self evaluate the task yourself.
Will we do case studies in the program?
Yes.
Where can we contact you?
You can mail us at muhammad.hamedoon@tech.uol.edu.pk
Payment/Registration related questions
Where do we have to make our payments? Your YouTube channel or website?
You have to make all your monthly payments on our website.
Unfortunately no, the program follows a monthly subscription model.
What is the validity of monthly subscription? Suppose if I pay on 15th Jan, then do I have to pay again on 1st Feb or 15th Feb
15th Feb. The validity period is 30 days from the day you make the payment. So essentially you can join anytime you don't have to wait for a month to end.
What if I don't like the course after making the payment. What is the refund policy?
You get a 7 days refund period from the day you have made the payment.
I am living outside India and I am not able to make the payment on the website, what should I do?
You have to contact us by sending a mail at muhammad.hamedoon@tech.uol.edu.pk
Post registration queries

Till when can I view the paid videos on the website?

This one is tricky, so read carefully. You can watch the videos till your subscription is valid. Suppose you have purchased subscription on 21st Jan, you will be able to watch all the past paid sessions in the period of 21st Jan to 20th Feb. But after 21st Feb you will have to purchase the subscription again.

But once the course is over and you have paid us Rs 5600(or 7 installments of Rs 799) you will be able to watch the paid sessions till Aug 2024.

Why lifetime validity is not provided?

Because of the low course fee.

Where can I reach out in case of a doubt after the session?

You will have to fill a google form provided in your dashboard and our team will contact you for a 1 on 1 doubt clearance session

If I join the program late, can I still ask past week doubts?

Yes, just select past week doubt in the doubt clearance google form.

I am living outside India and I am not able to make the payment on the website, what should I do?

You have to contact us by sending a mail at muhammad.hamedoon@tech.uol.edu.pk

Certificate and Placement Assistance related queries

What is the criteria to get the certificate?

There are 2 criterias:

You have to pay the entire fee of Rs 5600

You have to attempt all the course assessments.

I am joining late. How can I pay payment of the earlier months?

You will get a link to pay fee of earlier months in your dashboard once you pay for the current month.

I have read that Placement assistance is a part of this program. What comes under Placement assistance?

This is to clarify that Placement assistance does not mean Placement guarantee. So we dont guarantee you any jobs or for that matter even interview calls. So if you are planning to join this course just for placements, I am afraid you will be disappointed. Here is what comes under placement assistance

Portfolio Building sessions

Soft skill sessions

Sessions with industry mentors

Discussion on Job hunting strategies

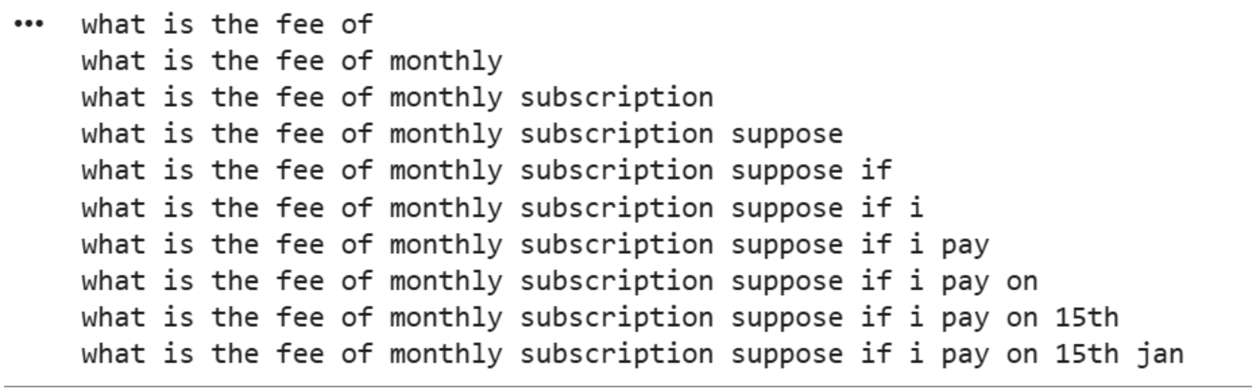
```
"""
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer

# Tokenizer
tokenizer = Tokenizer()
tokenizer.fit_on_texts([faqs])

input_sequences = []
```

```
for sentence in faqs.split('\n'):
    tokenized_sentence = tokenizer.texts_to_sequences([sentence])[0]
    for i in range(1, len(tokenized_sentence)):
        input_sequences.append(tokenized_sentence[:i+1])

max_len = max([len(x) for x in input_sequences])

from tensorflow.keras.preprocessing.sequence import pad_sequences
padded_input_sequences = pad_sequences(input_sequences, maxlen=max_len,
padding='pre')

X = padded_input_sequences[:, :-1]
y = padded_input_sequences[:, -1]

from tensorflow.keras.utils import to_categorical
y = to_categorical(y, num_classes=len(tokenizer.word_index)+1)

# Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense

vocab_size = len(tokenizer.word_index) + 1  # must be dynamic

model = Sequential()
model.add(Embedding(vocab_size, 100, input_length=max_len-1))
model.add(LSTM(150, return_sequences=True))
model.add(LSTM(150))
model.add(Dense(vocab_size, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# IMPORTANT FIX → Build model first
model.build(input_shape=(None, max_len-1))

# Corrected Summary
model.summary()

# Print actual trainable parameters (works in ALL TensorFlow versions)
print("\nTrainable parameters per layer:\n")
for layer in model.layers:
    print(layer.name, ":", layer.count_params())

print("\nTotal Trainable Parameters =", model.count_params())
```

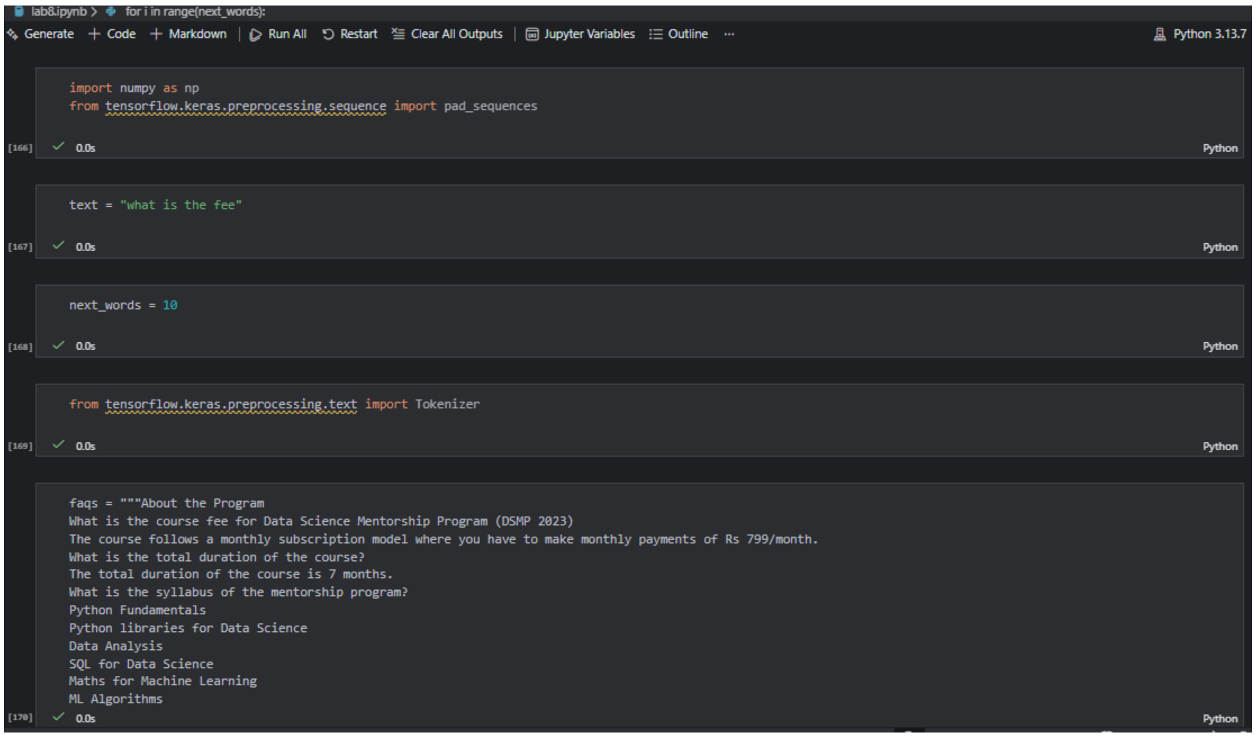
```
model.fit(X,y,epochs=100)
```

```
import numpy as np
import time
from tensorflow.keras.preprocessing.sequence import pad_sequences

text = "what is the fee"

for i in range(10):
    # tokenize
    token_text = tokenizer.texts_to_sequences([text])[0]

    # padding
    padded_token_text = pad_sequences([token_text], maxlen=56, padding='pre')

    # predict
    pos = np.argmax(model.predict(padded_token_text, verbose=0))

    for word, index in tokenizer.word_index.items():
        if index == pos:
            text = text + " " + word
            print(text)
            time.sleep(2)
```

Output :



```
•••  what is the fee of
     what is the fee of monthly
     what is the fee of monthly subscription
     what is the fee of monthly subscription suppose
     what is the fee of monthly subscription suppose if
     what is the fee of monthly subscription suppose if i
     what is the fee of monthly subscription suppose if i pay
     what is the fee of monthly subscription suppose if i pay on
     what is the fee of monthly subscription suppose if i pay on 15th
     what is the fee of monthly subscription suppose if i pay on 15th jan
```



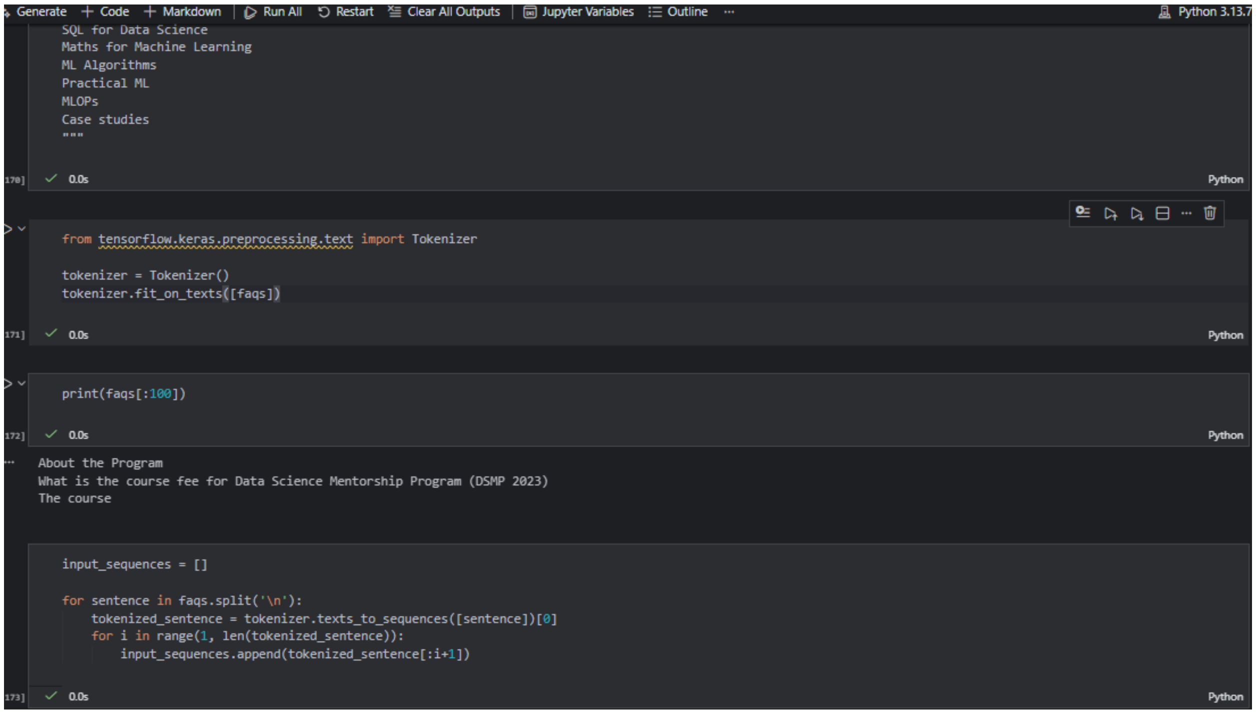
```python
import numpy as np
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

```python
text = "what is the fee"
```

```python
next_words = 10
```

```python
from tensorflow.keras.preprocessing.text import Tokenizer
```

```python
faqs = """About the Program
What is the course fee for Data Science Mentorship Program (DSMP 2023)
The course follows a monthly subscription model where you have to make monthly payments of Rs 799/month.
What is the total duration of the course?
The total duration of the course is 7 months.
What is the syllabus of the mentorship program?
Python Fundamentals
Python libraries for Data Science
Data Analysis
SQL for Data Science
Maths for Machine Learning
ML Algorithms
```



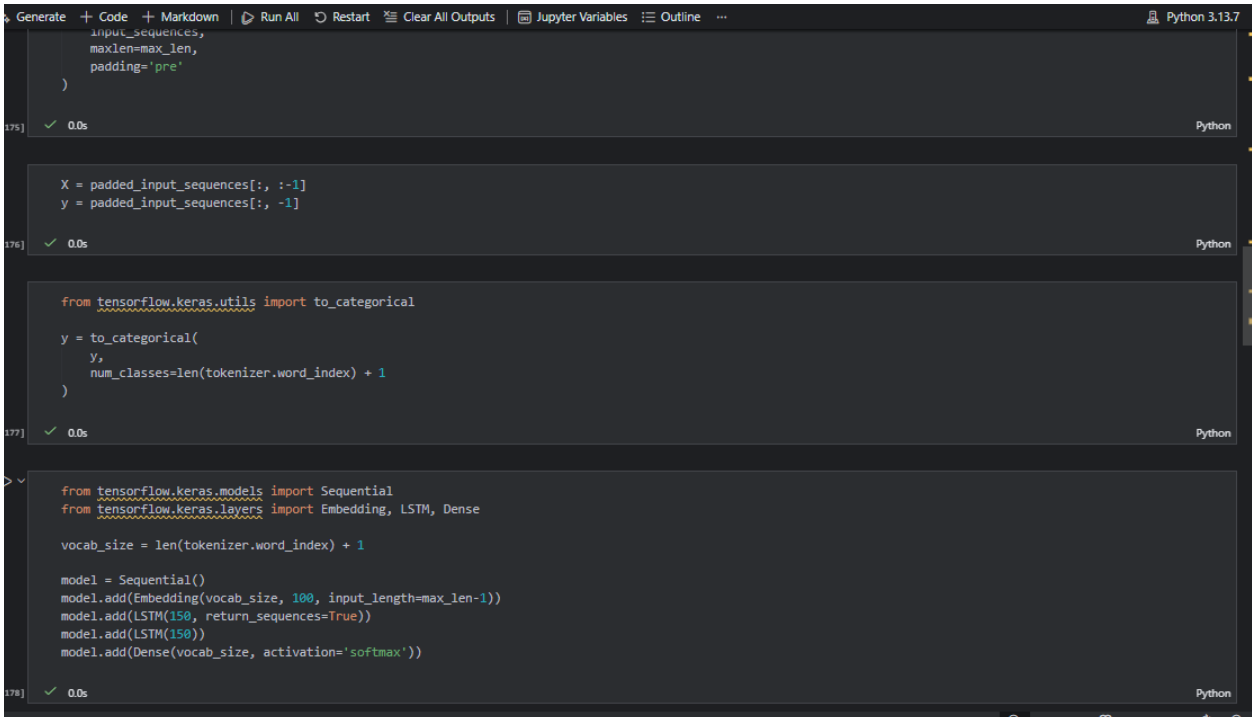
```python
SQL for Data Science
Maths for Machine Learning
ML Algorithms
Practical ML
MLOPs
Case studies
"""
```

```python
from tensorflow.keras.preprocessing.text import Tokenizer

tokenizer = Tokenizer()
tokenizer.fit_on_texts([faqs])
```

```python
print(faqs[:100])
```

```
About the Program
What is the course fee for Data Science Mentorship Program (DSMP 2023)
The course
```

```python
input_sequences = []

for sentence in faqs.split('\n'):
    tokenized_sentence = tokenizer.texts_to_sequences([sentence])[0]
    for i in range(1, len(tokenized_sentence)):
        input_sequences.append(tokenized_sentence[:i+1])
```



```
    input_sequences,
    maxlen=max_len,
    padding='pre'
)
```

[175] 0.0s

```
X = padded_input_sequences[:, :-1]
y = padded_input_sequences[:, -1]
```

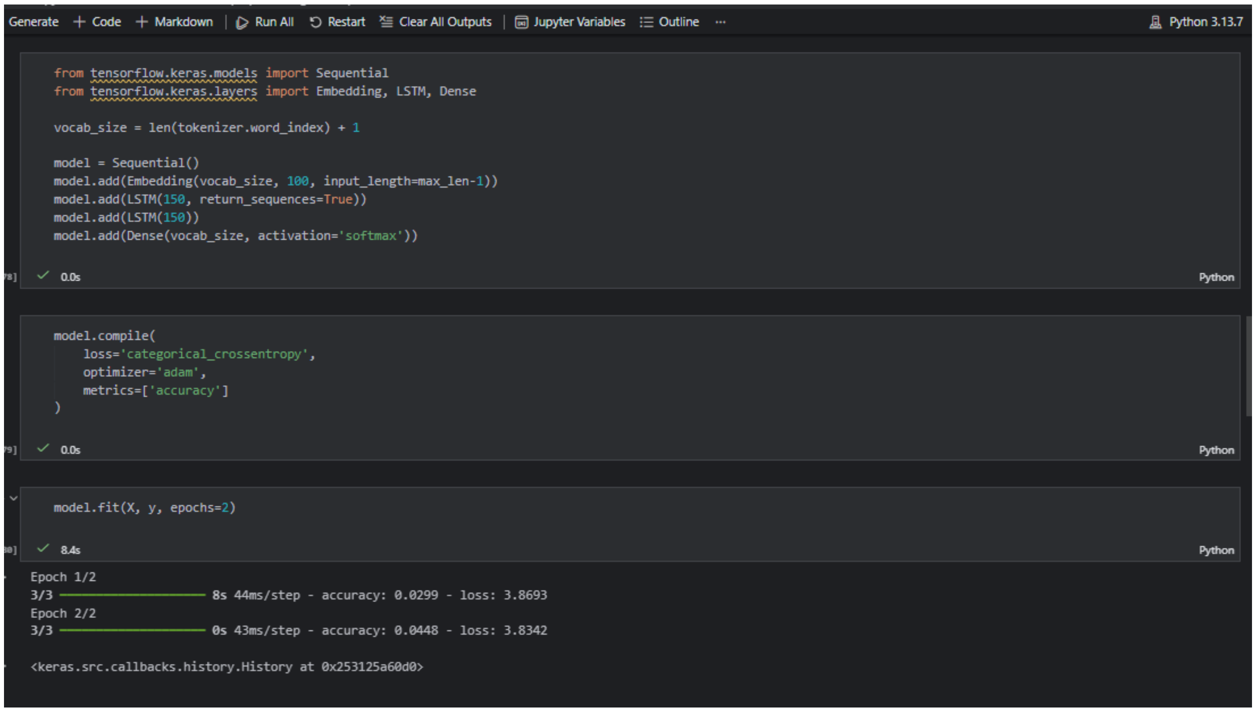
[176] 0.0s

```
from tensorflow.keras.utils import to_categorical

y = to_categorical(
    y,
    num_classes=len(tokenizer.word_index) + 1
)
```

[177] 0.0s

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense

vocab_size = len(tokenizer.word_index) + 1

model = Sequential()
model.add(Embedding(vocab_size, 100, input_length=max_len-1))
model.add(LSTM(150, return_sequences=True))
model.add(LSTM(150))
model.add(Dense(vocab_size, activation='softmax'))
```

[178] 0.0s



```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense

vocab_size = len(tokenizer.word_index) + 1

model = Sequential()
model.add(Embedding(vocab_size, 100, input_length=max_len-1))
model.add(LSTM(150, return_sequences=True))
model.add(LSTM(150))
model.add(Dense(vocab_size, activation='softmax'))
```

0.0s

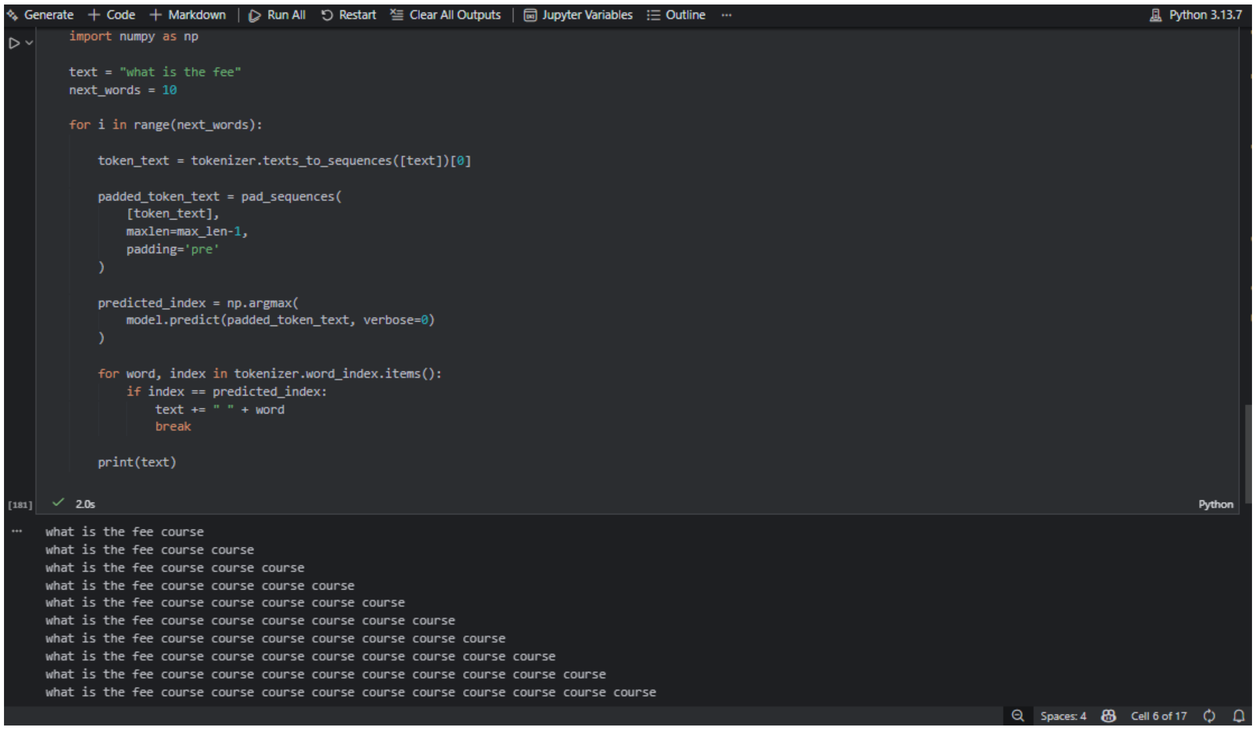
```
model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)
```

0.0s

```
model.fit(X, y, epochs=2)
```

8.4s

```
Epoch 1/2
3/3 ━━━━━━━━━━━━━━━━━━━━ 8s 44ms/step - accuracy: 0.0299 - loss: 3.8693
Epoch 2/2
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step - accuracy: 0.0448 - loss: 3.8342

<keras.src.callbacks.history.History at 0x253125a60d0>
```



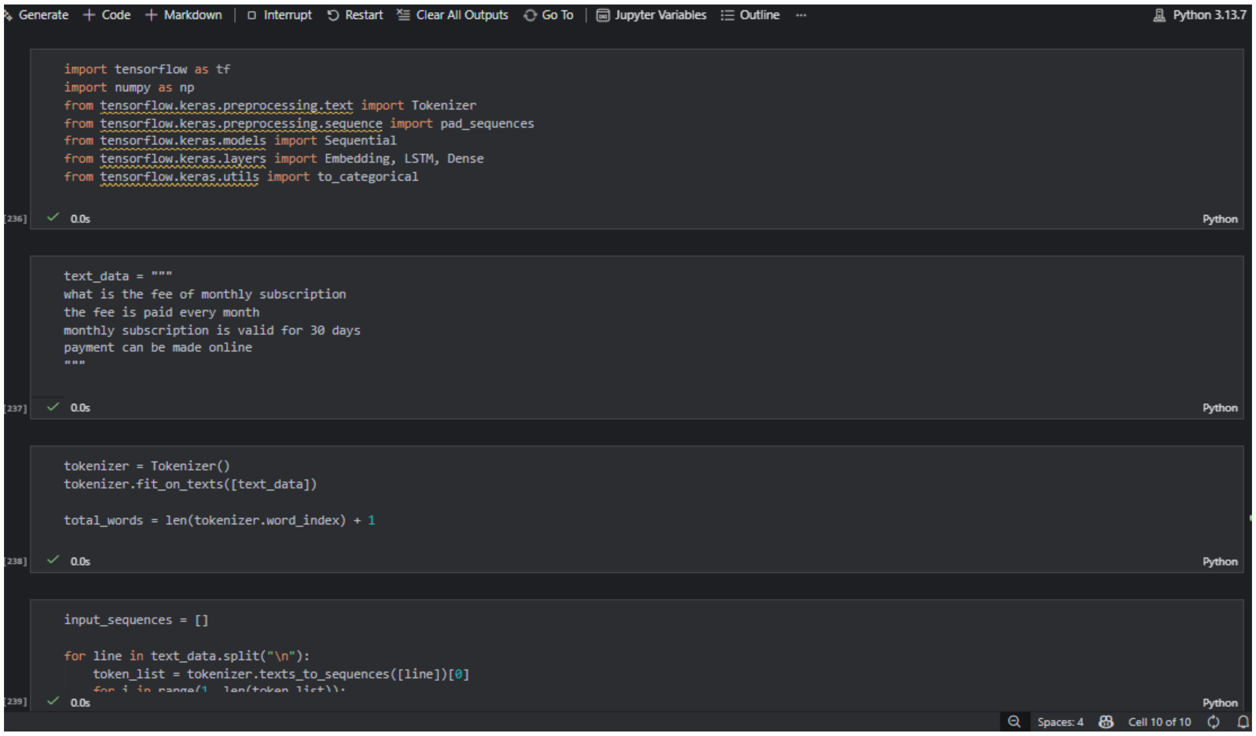
```python
import numpy as np

text = "what is the fee"
next_words = 10

for i in range(next_words):

    token_text = tokenizer.texts_to_sequences([text])[0]

    padded_token_text = pad_sequences(
        [token_text],
        maxlen=max_len-1,
        padding='pre'
    )

    predicted_index = np.argmax(
        model.predict(padded_token_text, verbose=0)
    )

    for word, index in tokenizer.word_index.items():
        if index == predicted_index:
            text += " " + word
            break

    print(text)
```

[181] ✓ 2.0s

```
what is the fee course
what is the fee course course
what is the fee course course course
what is the fee course course course course
what is the fee course course course course course
what is the fee course course course course course course
what is the fee course course course course course course course
what is the fee course course course course course course course course
what is the fee course course course course course course course course course
what is the fee course course course course course course course course course course
```

## Question No. 2

Train an LSTM model on your chosen text dataset and write the code to predict the next word for a user-given input sentence.



```python
import tensorflow as tf
import numpy as np
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.utils import to_categorical
```

[236] ✓ 0.0s

```python
text_data = """
what is the fee of monthly subscription
the fee is paid every month
monthly subscription is valid for 30 days
payment can be made online
"""
```

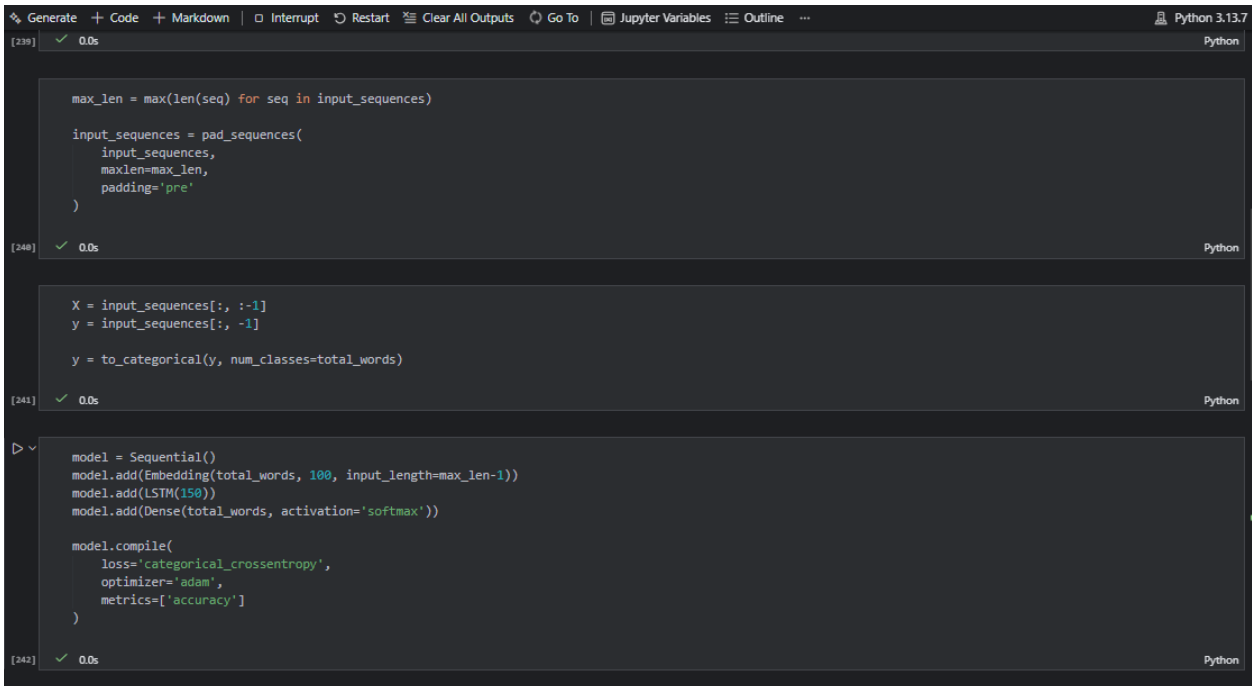
[237] ✓ 0.0s

```python
tokenizer = Tokenizer()
tokenizer.fit_on_texts([text_data])

total_words = len(tokenizer.word_index) + 1
```

[238] ✓ 0.0s

```python
input_sequences = []

for line in text_data.split("\n"):
    token_list = tokenizer.texts_to_sequences([line])[0]
```

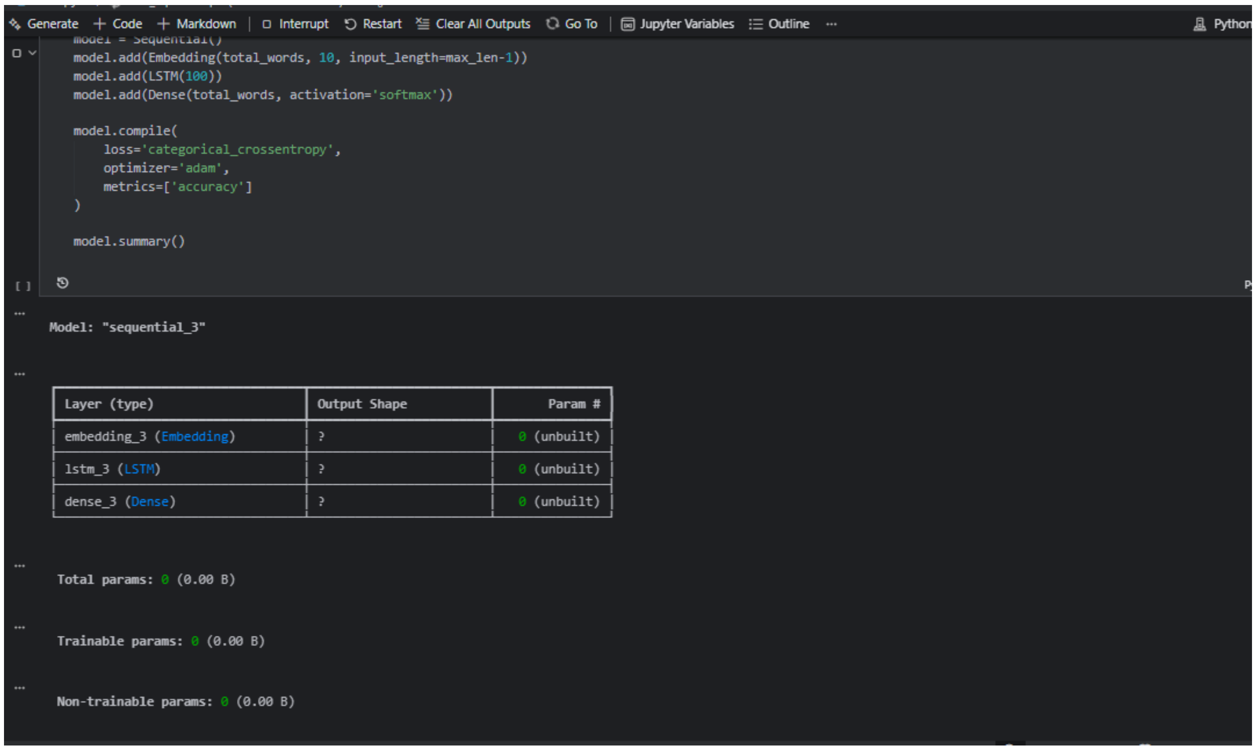
[239] ✓ 0.0s



```python
max_len = max(len(seq) for seq in input_sequences)

input_sequences = pad_sequences(
    input_sequences,
    maxlen=max_len,
    padding='pre'
)
```

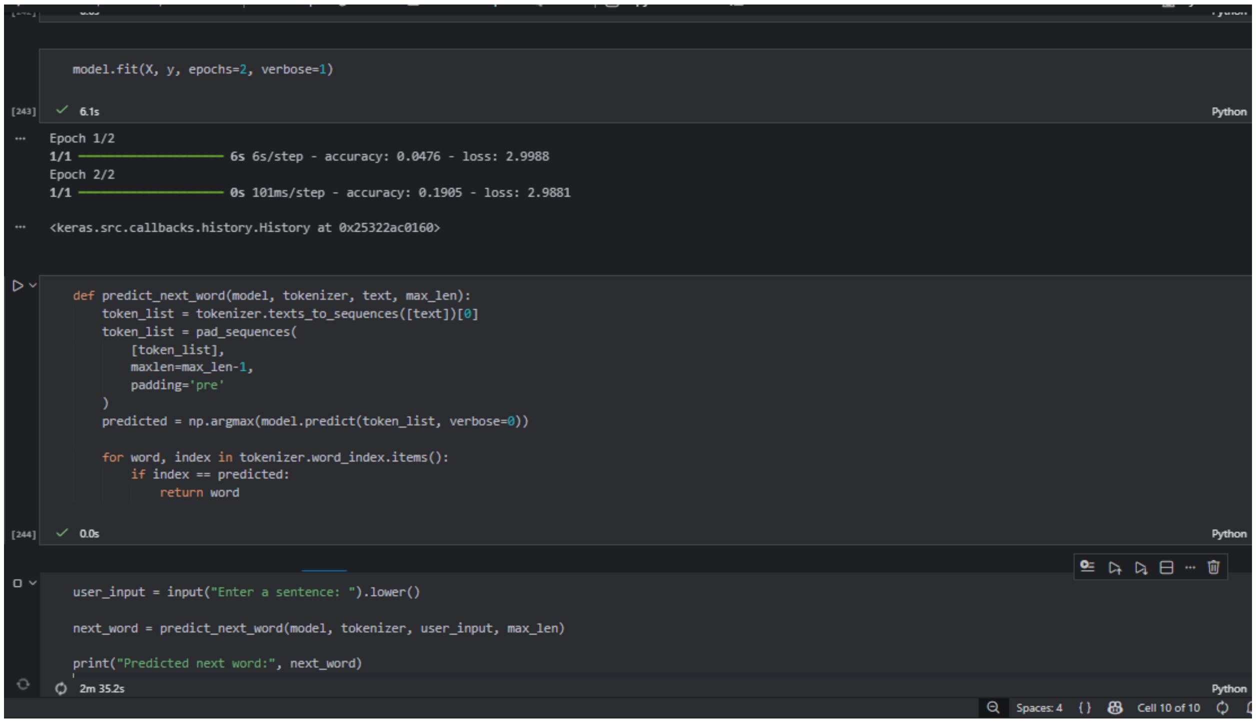
```python
X = input_sequences[:, :-1]
y = input_sequences[:, -1]

y = to_categorical(y, num_classes=total_words)
```

```python
model = Sequential()
model.add(Embedding(total_words, 100, input_length=max_len-1))
model.add(LSTM(150))
model.add(Dense(total_words, activation='softmax'))

model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)
```



```python
model = Sequential()
model.add(Embedding(total_words, 10, input_length=max_len-1))
model.add(LSTM(100))
model.add(Dense(total_words, activation='softmax'))

model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)

model.summary()
```

```
Model: "sequential_3"

┌──────────────────────────────────┬───────────────────────┬───────────────┐
│ Layer (type)                     │ Output Shape          │       Param # │
├──────────────────────────────────┼───────────────────────┼───────────────┤
│ embedding_3 (Embedding)          │ ?                     │   0 (unbuilt) │
├──────────────────────────────────┼───────────────────────┼───────────────┤
│ lstm_3 (LSTM)                    │ ?                     │   0 (unbuilt) │
├──────────────────────────────────┼───────────────────────┼───────────────┤
│ dense_3 (Dense)                  │ ?                     │   0 (unbuilt) │
└──────────────────────────────────┴───────────────────────┴───────────────┘

 Total params: 0 (0.00 B)

 Trainable params: 0 (0.00 B)

 Non-trainable params: 0 (0.00 B)
```



```
model.fit(X, y, epochs=2, verbose=1)
```

[243] ✓ 6.1s Python

```
Epoch 1/2
1/1 ━━━━━━━━━━━━━━━━━━━━ 6s 6s/step - accuracy: 0.0476 - loss: 2.9988
Epoch 2/2
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 101ms/step - accuracy: 0.1905 - loss: 2.9881

<keras.src.callbacks.history.History at 0x25322ac0160>
```

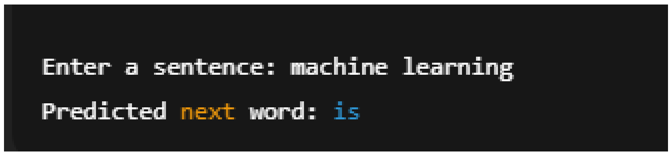
```
def predict_next_word(model, tokenizer, text, max_len):
    token_list = tokenizer.texts_to_sequences([text])[0]
    token_list = pad_sequences(
        [token_list],
        maxlen=max_len-1,
        padding='pre'
    )
    predicted = np.argmax(model.predict(token_list, verbose=0))

    for word, index in tokenizer.word_index.items():
        if index == predicted:
            return word
```

[244] ✓ 0.0s Python

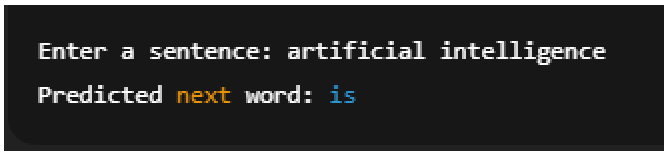
```
user_input = input("Enter a sentence: ").lower()

next_word = predict_next_word(model, tokenizer, user_input, max_len)

print("Predicted next word:", next_word)
```

2m 35.2s Python

Spaces: 4 Cell 10 of 10



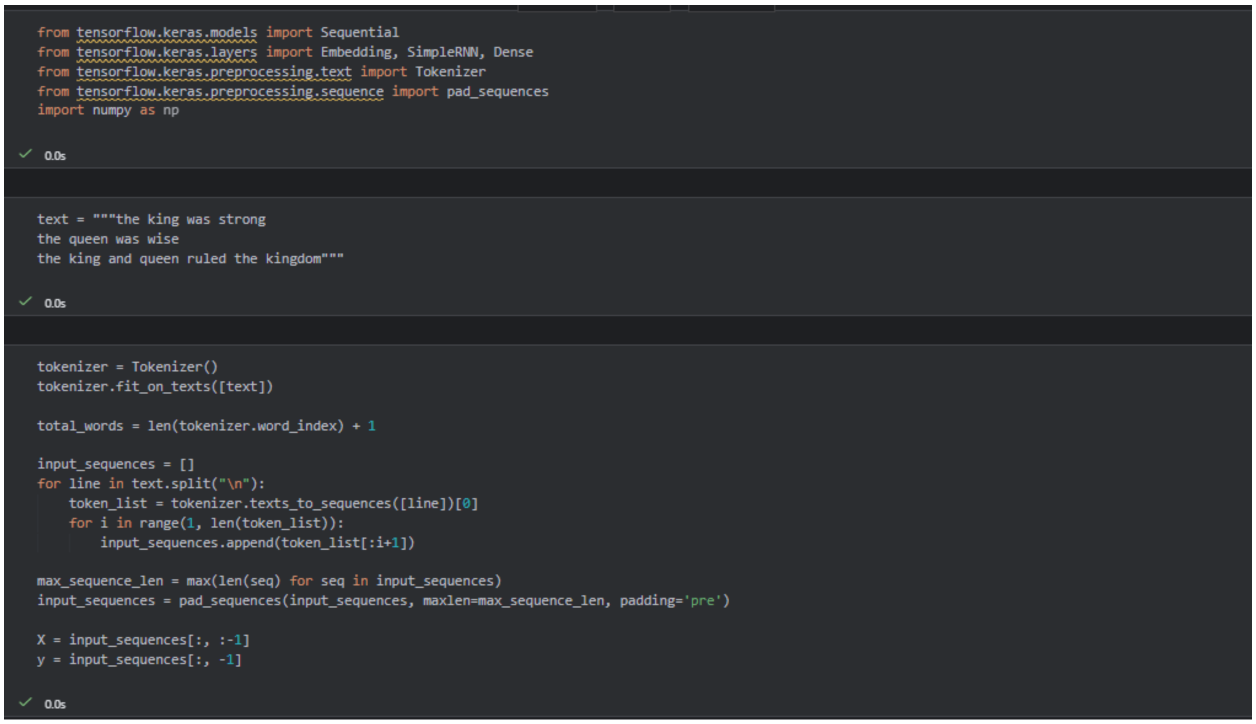
```
Enter a sentence: machine learning
Predicted next word: is
```



```
Enter a sentence: artificial intelligence
Predicted next word: is
```

## Question No. 3

Modify the next-word prediction code so that the model generates 5 new words sequentially instead of just one.



```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import numpy as np
```

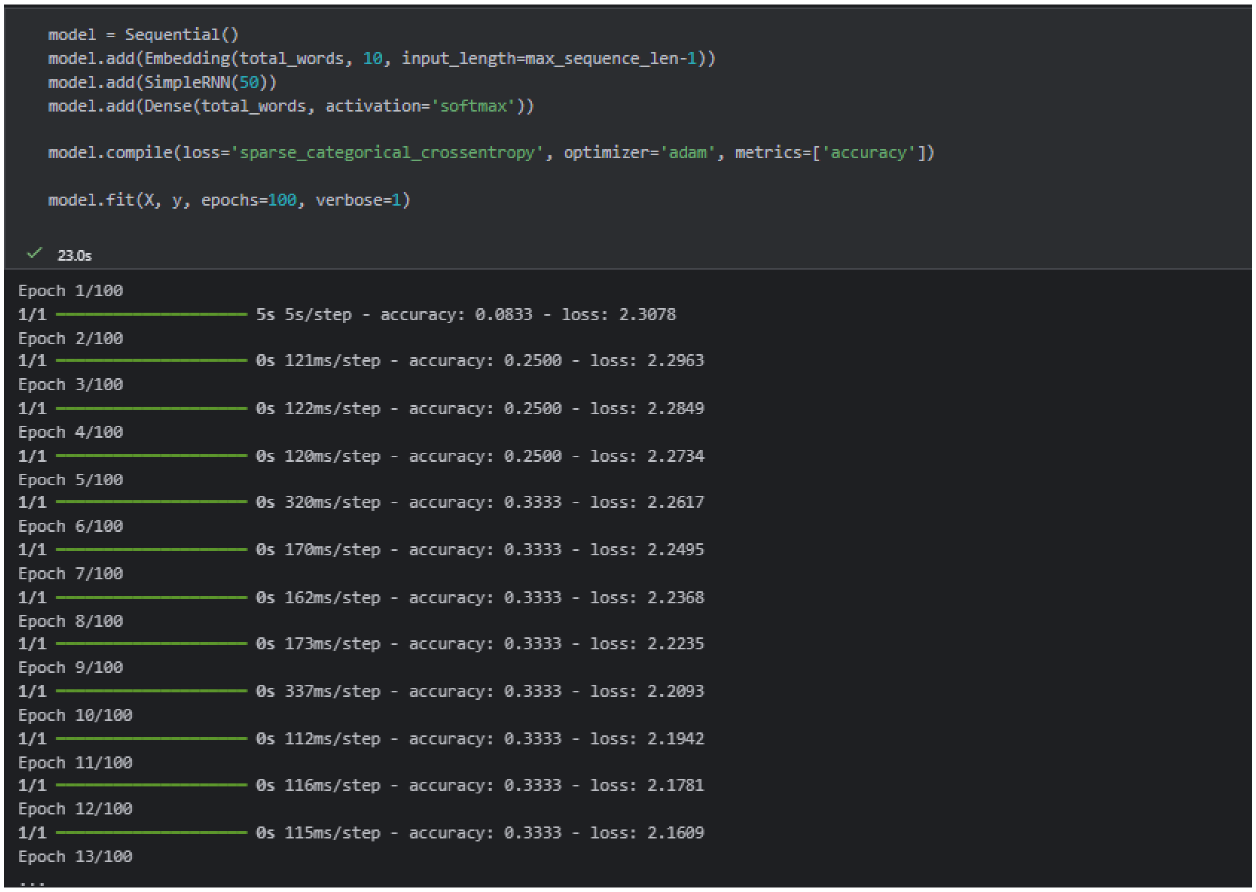
✓ 0.0s

```
text = """the king was strong
the queen was wise
the king and queen ruled the kingdom"""
```

✓ 0.0s

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts([text])

total_words = len(tokenizer.word_index) + 1

input_sequences = []
for line in text.split("\n"):
    token_list = tokenizer.texts_to_sequences([line])[0]
    for i in range(1, len(token_list)):
        input_sequences.append(token_list[:i+1])

max_sequence_len = max(len(seq) for seq in input_sequences)
input_sequences = pad_sequences(input_sequences, maxlen=max_sequence_len, padding='pre')

X = input_sequences[:, :-1]
y = input_sequences[:, -1]
```

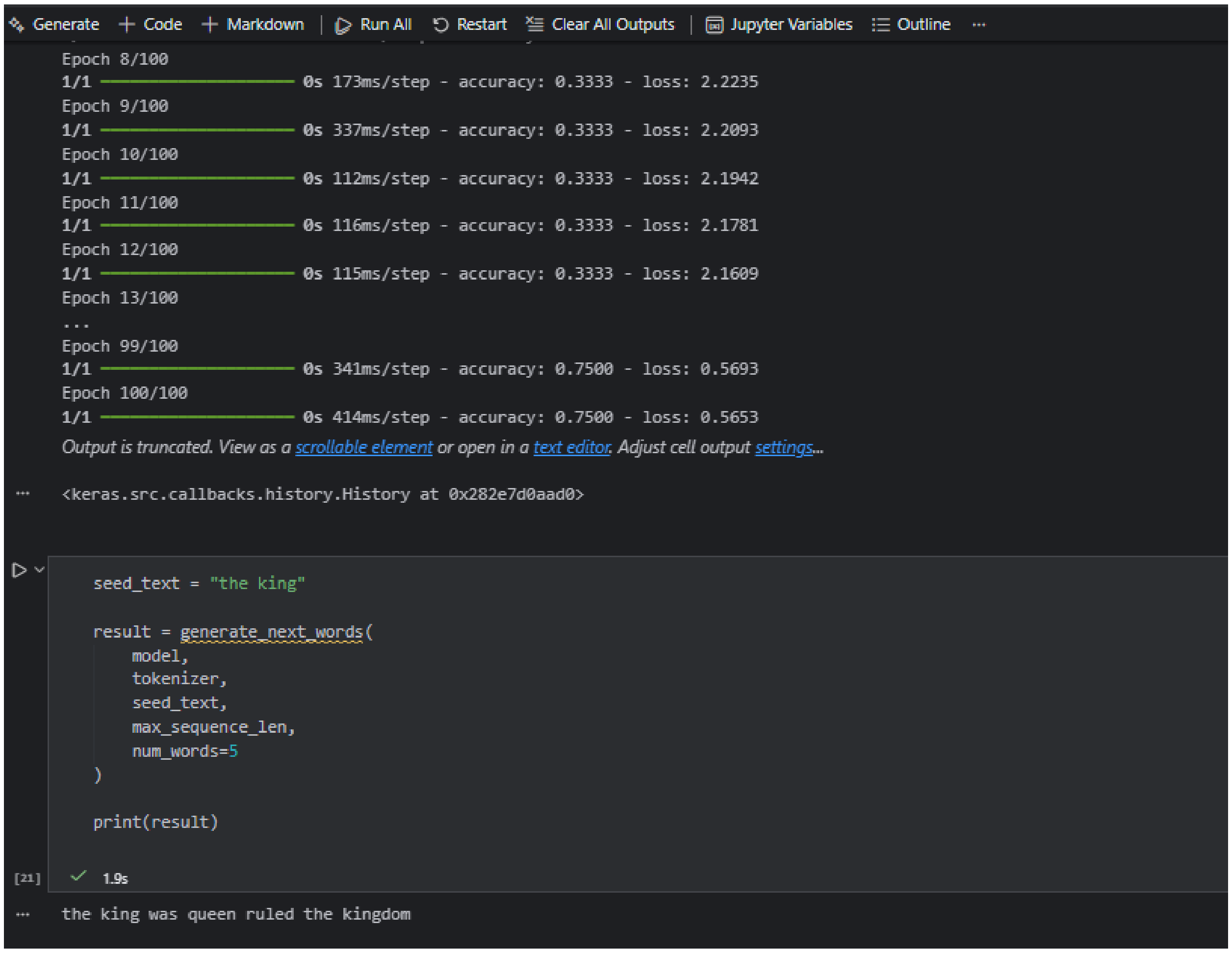
✓ 0.0s



```
model = Sequential()
model.add(Embedding(total_words, 10, input_length=max_sequence_len-1))
model.add(SimpleRNN(50))
model.add(Dense(total_words, activation='softmax'))

model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

model.fit(X, y, epochs=100, verbose=1)
```

✓ 23.0s

```
Epoch 1/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 5s 5s/step - accuracy: 0.0833 - loss: 2.3078
Epoch 2/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 121ms/step - accuracy: 0.2500 - loss: 2.2963
Epoch 3/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 122ms/step - accuracy: 0.2500 - loss: 2.2849
Epoch 4/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 120ms/step - accuracy: 0.2500 - loss: 2.2734
Epoch 5/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 320ms/step - accuracy: 0.3333 - loss: 2.2617
Epoch 6/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 170ms/step - accuracy: 0.3333 - loss: 2.2495
Epoch 7/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 162ms/step - accuracy: 0.3333 - loss: 2.2368
Epoch 8/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 173ms/step - accuracy: 0.3333 - loss: 2.2235
Epoch 9/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 337ms/step - accuracy: 0.3333 - loss: 2.2093
Epoch 10/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 112ms/step - accuracy: 0.3333 - loss: 2.1942
Epoch 11/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 116ms/step - accuracy: 0.3333 - loss: 2.1781
Epoch 12/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 115ms/step - accuracy: 0.3333 - loss: 2.1609
Epoch 13/100
...
```



```
Epoch 8/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 173ms/step - accuracy: 0.3333 - loss: 2.2235
Epoch 9/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 337ms/step - accuracy: 0.3333 - loss: 2.2093
Epoch 10/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 112ms/step - accuracy: 0.3333 - loss: 2.1942
Epoch 11/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 116ms/step - accuracy: 0.3333 - loss: 2.1781
Epoch 12/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 115ms/step - accuracy: 0.3333 - loss: 2.1609
Epoch 13/100
...
Epoch 99/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 341ms/step - accuracy: 0.7500 - loss: 0.5693
Epoch 100/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 414ms/step - accuracy: 0.7500 - loss: 0.5653
```

*Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...*

```
<keras.src.callbacks.history.History at 0x282e7d0aad0>
```

```python
seed_text = "the king"

result = generate_next_words(
    model,
    tokenizer,
    seed_text,
    max_sequence_len,
    num_words=5
)

print(result)
```

[21] ✓ 1.9s

```
the king was queen ruled the kingdom
```