```
from tensorflow.keras.layers import SimpleRNN, Dense, Embedding
from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential
import numpy as np
```

```
vocab_size = 5000
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=vocab_size)
```

```
print(x_train[0])
```

```
    [1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941, 4, 173, 36, 256, 5, 25, 100, 43, 838, 112, 50, 670, 2, 9, 35, 48
```

```
word_idx = imdb.get_word_index()
word_idx = {i: word for word, i in word_idx.items()}
print([word_idx[i] for i in x_train[0]])
```

```
    ['the', 'as', 'you', 'with', 'out', 'themselves', 'powerful', 'lets', 'loves', 'their', 'becomes', 'reaching', 'had', 'journalist',
```

```
from tensorflow.keras.preprocessing import sequence

#keeping a fixed length of all reviews to max 400 words
max_words = 400

x_train = sequence.pad_sequences(x_train, maxlen=max_words)
x_test = sequence.pad_sequences(x_test, maxlen=max_words)

x_valid, y_valid = x_train[:64], y_train[:64]
x_train_, y_train_ = x_train[64:], y_train[64:]
```

```
#fixing every word's embedding size to be 32
############### ask maam
embd_len = 32

#creating RNN model
RNN_model = Sequential(name="Simple_RNN")
RNN_model.add(Embedding(vocab_size, embd_len, input_length=max_words))

#In case of a stacked(more than one layer of RNN) use return_sequences=True
RNN_model.add(SimpleRNN(128, activation='tanh', return_sequences=False))
RNN_model.add(Dense(1, activation='sigmoid'))
```

```
RNN_model.compile(loss="binary_crossentropy", optimizer='adam', metrics=['accuracy'])
```

```
history = RNN_model.fit(x_train_, y_train_, batch_size=64, epochs=15, verbose=1, validation_data=(x_valid, y_valid))
```

```
    Epoch 1/15
    390/390 [==============================] - 74s 190ms/step - loss: 0.4182 - accuracy: 0.8094 - val_loss: 0.8333 - val_accuracy: 0.64
    Epoch 2/15
    390/390 [==============================] - 74s 190ms/step - loss: 0.3703 - accuracy: 0.8367 - val_loss: 0.7061 - val_accuracy: 0.73
    Epoch 3/15
    390/390 [==============================] - 73s 187ms/step - loss: 0.3521 - accuracy: 0.8498 - val_loss: 0.6497 - val_accuracy: 0.71
    Epoch 4/15
    390/390 [==============================] - 74s 189ms/step - loss: 0.4451 - accuracy: 0.7857 - val_loss: 0.7463 - val_accuracy: 0.51
    Epoch 5/15
    390/390 [==============================] - 73s 187ms/step - loss: 0.4310 - accuracy: 0.7939 - val_loss: 0.6709 - val_accuracy: 0.64
    Epoch 6/15
    390/390 [==============================] - 72s 184ms/step - loss: 0.4559 - accuracy: 0.7829 - val_loss: 0.7419 - val_accuracy: 0.65
    Epoch 7/15
    390/390 [==============================] - 71s 182ms/step - loss: 0.3903 - accuracy: 0.8258 - val_loss: 0.6545 - val_accuracy: 0.81
    Epoch 8/15
    390/390 [==============================] - 71s 181ms/step - loss: 0.4125 - accuracy: 0.8075 - val_loss: 0.7092 - val_accuracy: 0.73
    Epoch 9/15
    390/390 [==============================] - 71s 181ms/step - loss: 0.4119 - accuracy: 0.8171 - val_loss: 1.0547 - val_accuracy: 0.31
    Epoch 10/15
    390/390 [==============================] - 70s 181ms/step - loss: 0.4934 - accuracy: 0.7440 - val_loss: 0.6885 - val_accuracy: 0.76
    Epoch 11/15
    390/390 [==============================] - 70s 181ms/step - loss: 0.4755 - accuracy: 0.7674 - val_loss: 0.7686 - val_accuracy: 0.59
    Epoch 12/15
    390/390 [==============================] - 71s 181ms/step - loss: 0.4050 - accuracy: 0.8123 - val_loss: 0.7222 - val_accuracy: 0.70
    Epoch 13/15
    390/390 [==============================] - 71s 183ms/step - loss: 0.3689 - accuracy: 0.8386 - val_loss: 0.6560 - val_accuracy: 0.76
    Epoch 14/15
    390/390 [==============================] - 71s 183ms/step - loss: 0.3652 - accuracy: 0.8381 - val_loss: 0.7177 - val_accuracy: 0.68
    Epoch 15/15
    390/390 [==============================] - 71s 183ms/step - loss: 0.3297 - accuracy: 0.8602 - val_loss: 0.7617 - val_accuracy: 0.67
```

```
print()
print("Simple_RNN Score = ", RNN_model.evaluate(x_test, y_test, verbose=0))
```

```
Simple_RNN Score =  [0.6619482040405273, 0.7360799908638]
```