```
shakespeare_url = "https://homl.info/shakespeare"                                            # webpage for text
filepath = tf.keras.utils.get_file("shakespeare.txt", shakespeare_url)
with open(filepath) as f:
    shakespeare_text = f.read()

print(shakespeare_text[:80])
```

```
First Citizen:
Before we proceed any further, hear me speak.

All:
Speak, speak.
```

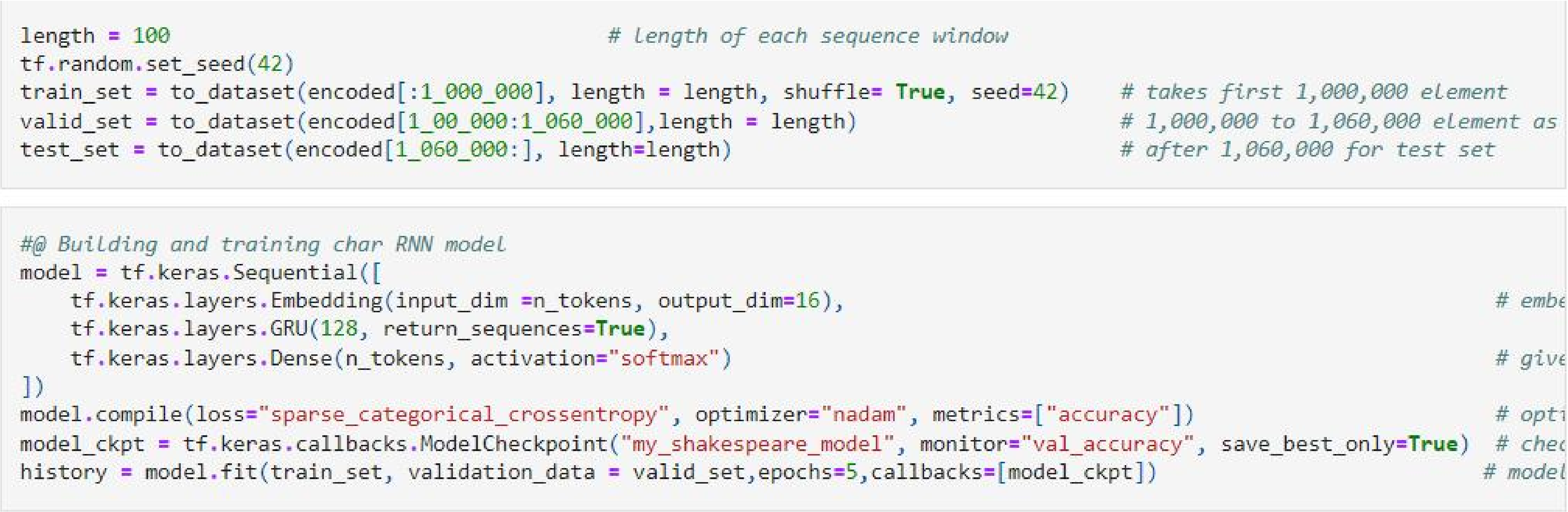
```
#encoding of text
text_vec_layer = tf.keras.layers.TextVectorization(split="character",standardize="lower")
text_vec_layer.adapt([shakespeare_text])
encoded = text_vec_layer([shakespeare_text])[0]
```

```
encoded -= 2                                        # dropping token 0 fro pad and 1 for unkown
n_tokens = text_vec_layer.vocabulary_size() - 2     # subtracting 2 from distinct chars
dataset_size = len(encoded)                         # total number of chars
```

```
dataset_size
```

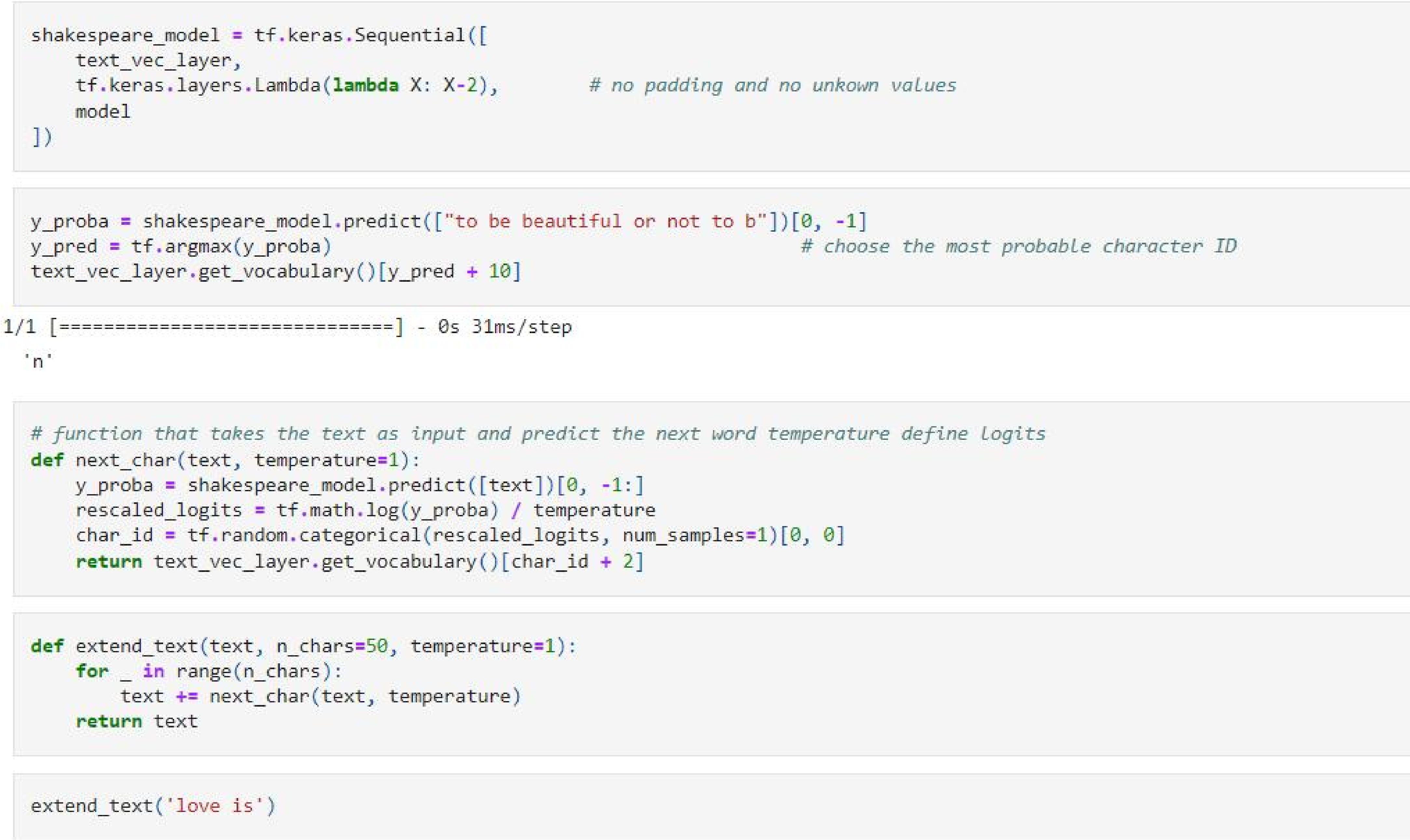
```
1115394
```

```
# function that creat window like 1 window takes "hell" another take "ello" for word hello, if shuffle
def to_dataset(sequence, length, shuffle=False, seed = None, batch_size = 32):
    ds = tf.data.Dataset.from_tensor_slices(sequence)                     # create a tf dataset from the sequence
    ds = ds.window(length+1, shift=1, drop_remainder= True)               # create overlapping window of length
    ds = ds.flat_map(lambda window_ds: window_ds.batch(length + 1))
    if shuffle:                                                           # shuffle the dataset
        ds = ds.shuffle(buffer_size=100_000, seed=seed)
    ds = ds.batch(batch_size)                                             # batches of given size        # map w
    return ds.map(lambda window: (window[:, :-1], window[:, 1:])).prefetch(1)
```



```
length = 100                                   # length of each sequence window
tf.random.set_seed(42)
train_set = to_dataset(encoded[:1_000_000], length = length, shuffle= True, seed=42)    # takes first 1,000,000 element
valid_set = to_dataset(encoded[1_00_000:1_060_000],length = length)                    # 1,000,000 to 1,060,000 element as
test_set = to_dataset(encoded[1_060_000:], length=length)                               # after 1,060,000 for test set
```

```
#@ Building and training char RNN model
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(input_dim =n_tokens, output_dim=16),                                                    # embe
    tf.keras.layers.GRU(128, return_sequences=True),
    tf.keras.layers.Dense(n_tokens, activation="softmax")                                                             # give
])
model.compile(loss="sparse_categorical_crossentropy", optimizer="nadam", metrics=["accuracy"])                        # opti
model_ckpt = tf.keras.callbacks.ModelCheckpoint("my_shakespeare_model", monitor="val_accuracy", save_best_only=True)  # chec
history = model.fit(train_set, validation_data = valid_set,epochs=5,callbacks=[model_ckpt])                          # model
```



```
shakespeare_model = tf.keras.Sequential([
    text_vec_layer,
    tf.keras.layers.Lambda(lambda X: X-2),        # no padding and no unkown values
    model
])
```

```
y_proba = shakespeare_model.predict(["to be beautiful or not to b"])[0, -1]
y_pred = tf.argmax(y_proba)                                            # choose the most probable character ID
text_vec_layer.get_vocabulary()[y_pred + 10]
```

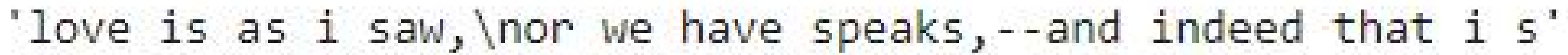
```
1/1 [==============================] - 0s 31ms/step
```

```
'n'
```

```
# function that takes the text as input and predict the next word temperature define logits
def next_char(text, temperature=1):
    y_proba = shakespeare_model.predict([text])[0, -1:]
    rescaled_logits = tf.math.log(y_proba) / temperature
    char_id = tf.random.categorical(rescaled_logits, num_samples=1)[0, 0]
    return text_vec_layer.get_vocabulary()[char_id + 2]
```

```
def extend_text(text, n_chars=50, temperature=1):
    for _ in range(n_chars):
        text += next_char(text, temperature)
    return text
```

```
extend_text('love is')
```



```
'love is as i saw,\nor we have speaks,--and indeed that i s'
```