Import

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

Create character mappings

```
chars = sorted(list(set(text)))
char_to_index = {char: i for i, char in enumerate(chars)}
index_to_char = {i: char for i, char in enumerate(chars)}
```

Generate training data

```
max_len = 20
step = 3
sentences = []
next_chars = []
for i in range(0, len(text) - max_len, step):
    sentences.append(text[i:i + max_len])
    next_chars.append(text[i + max_len])
```

Vectorize input and output

```
X = np.zeros((len(sentences), max_len, len(chars)), dtype=np.float32)
y = np.zeros((len(sentences), len(chars)), dtype=np.float32)
for i, sentence in enumerate(sentences):
    for t, char in enumerate(sentence):
        X[i, t, char_to_index[char]] = 1
    y[i, char_to_index[next_chars[i]]] = 1
```

Build the model

```
model = Sequential([
    LSTM(128, input_shape=(max_len, len(chars))),
    Dense(len(chars), activation='softmax')
])
model.compile(loss='categorical_crossentropy', optimizer='adam')
```

Train the model

```
model.fit(X, y, batch_size=100, epochs=10)
```

```
Epoch 1/10
1/1 [==============================] - 0s 45ms/step - loss: 2.8696
Epoch 2/10
1/1 [==============================] - 0s 44ms/step - loss: 2.8626
Epoch 3/10
1/1 [==============================] - 0s 51ms/step - loss: 2.8557
Epoch 4/10
1/1 [==============================] - 0s 45ms/step - loss: 2.8490
Epoch 5/10
1/1 [==============================] - 0s 46ms/step - loss: 2.8423
Epoch 6/10
1/1 [==============================] - 0s 46ms/step - loss: 2.8354
Epoch 7/10
1/1 [==============================] - 0s 44ms/step - loss: 2.8290
Epoch 8/10
1/1 [==============================] - 0s 46ms/step - loss: 2.8239
Epoch 9/10
1/1 [==============================] - 0s 42ms/step - loss: 2.8188
Epoch 10/10
1/1 [==============================] - 0s 42ms/step - loss: 2.8123
<keras.src.callbacks.History at 0x79de401b5c00>
```

Function to generate film names

```
def generate_film_name(seed=None, temperature=1.0):
    if seed is None:
        start_index = random.randint(0, len(text) - max_len - 1)
        seed = text[start_index:start_index + max_len]
    generated = seed
    for _ in range(40):
        x_pred = np.zeros((1, max_len, len(chars)))
        for t, char in enumerate(seed):
            x_pred[0, t, char_to_index[char]] = 1

        preds = model.predict(x_pred, verbose=0)[0]
        next_index = sample(preds, temperature)
        next_char = index_to_char[next_index]

        generated += next_char
        seed = seed[1:] + next_char
    return generated
```

Helper function to sample an index from a probability array

```
def sample(preds, temperature=1.0):
    preds = np.asarray(preds).astype('float64')
    preds = np.log(preds) / temperature
    exp_preds = np.exp(preds)
    preds = exp_preds / np.sum(exp_preds)
    probas = np.random.multinomial(1, preds, 1)
    return np.argmax(probas)
```

Generate film names

```
for _ in range(10):
    print(generate_film_name() + ",", end=" ")
    print(generate_film_name())
```

```
on Pulp Fiction The thetKktIigenr ltnrnemTKrcghretiKiDrD m o, c Park Avatar Titani
tion Harry Potter IneM FthiDRhh FtredemvhFmihiosIe ivLKkm  r, lp Fiction The Lord
The Shawshank Redempet  r ogt iaFPtetkhirrKA  t rr hhnmi f t, rk Avatar Titanic Th
hawshank Redemption TtmTe gekeyinerD dK vitstgn ToomLTiTtA  , he Godfather Jurassi
tter Indiana Jones B eIi imI ntfhnr  cnr hegmnTaigoo h  kha , vatar Titanic The Ma
o the Future The Dar r   a TinDemnkggohe a oe  Lhtntn ttgmeh,  The Godfather Juras
rk Knight The Lion K  hthLlnhrhhal  TLsTh eFas gnhinfeThs Tl, tion The Lord of the
ture The Dark KnightcggT egdDgeFesiitKgik TiFhTfr meKsTenher, ht The Lion King Fin
mption Harry Potter Athnm KneiT ItLolore r mIt RtReDLthnetee,  Jurassic Park Avata
ngs The Shawshank RedTtge aTo elAiormr DnnaeLegtFieoshKh i o, ght The Lion King Fi
```