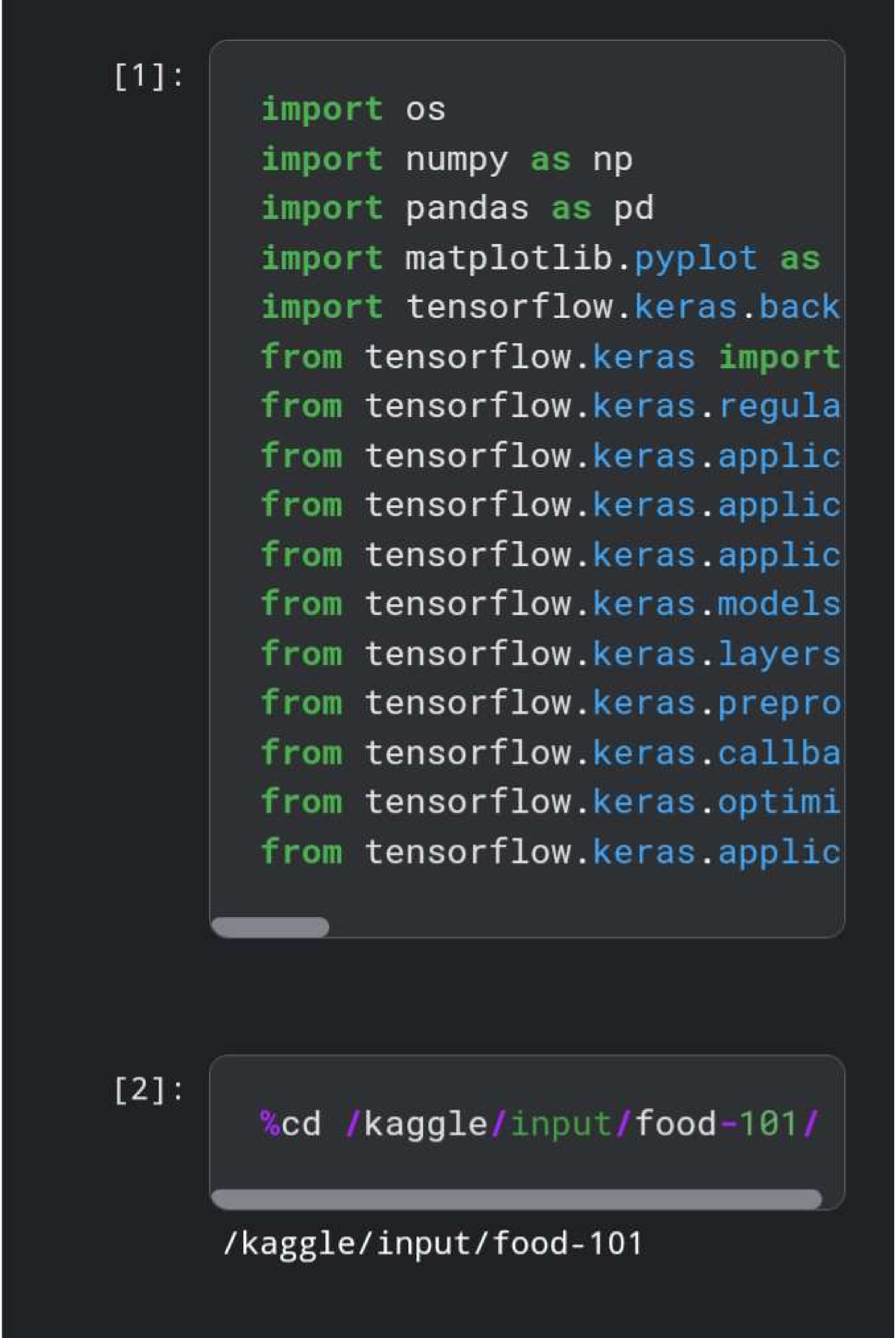


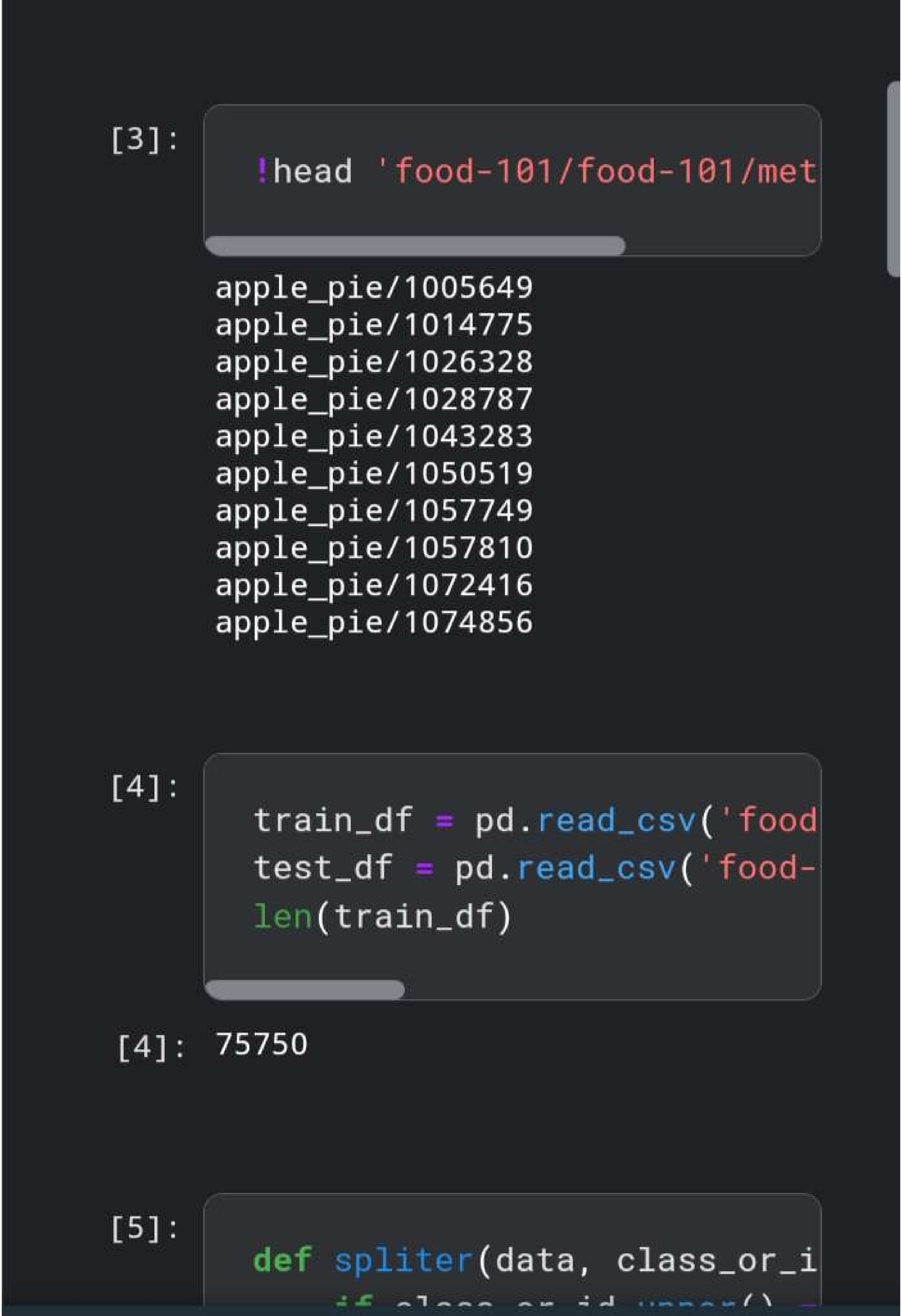
[1]:

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as
import tensorflow.keras.back
from tensorflow.keras import
from tensorflow.keras.regula
from tensorflow.keras.applic
from tensorflow.keras.applic
from tensorflow.keras.applic
from tensorflow.keras.models
from tensorflow.keras.layers
from tensorflow.keras.prepro
from tensorflow.keras.callba
from tensorflow.keras.optimi
from tensorflow.keras.applic
```

[2]:

```
%cd /kaggle/input/food-101/
```

```
/kaggle/input/food-101
```



[3]:

```
!head 'food-101/food-101/met
```

```
apple_pie/1005649
apple_pie/1014775
apple_pie/1026328
apple_pie/1028787
apple_pie/1043283
apple_pie/1050519
apple_pie/1057749
apple_pie/1057810
apple_pie/1072416
apple_pie/1074856
```

[4]:

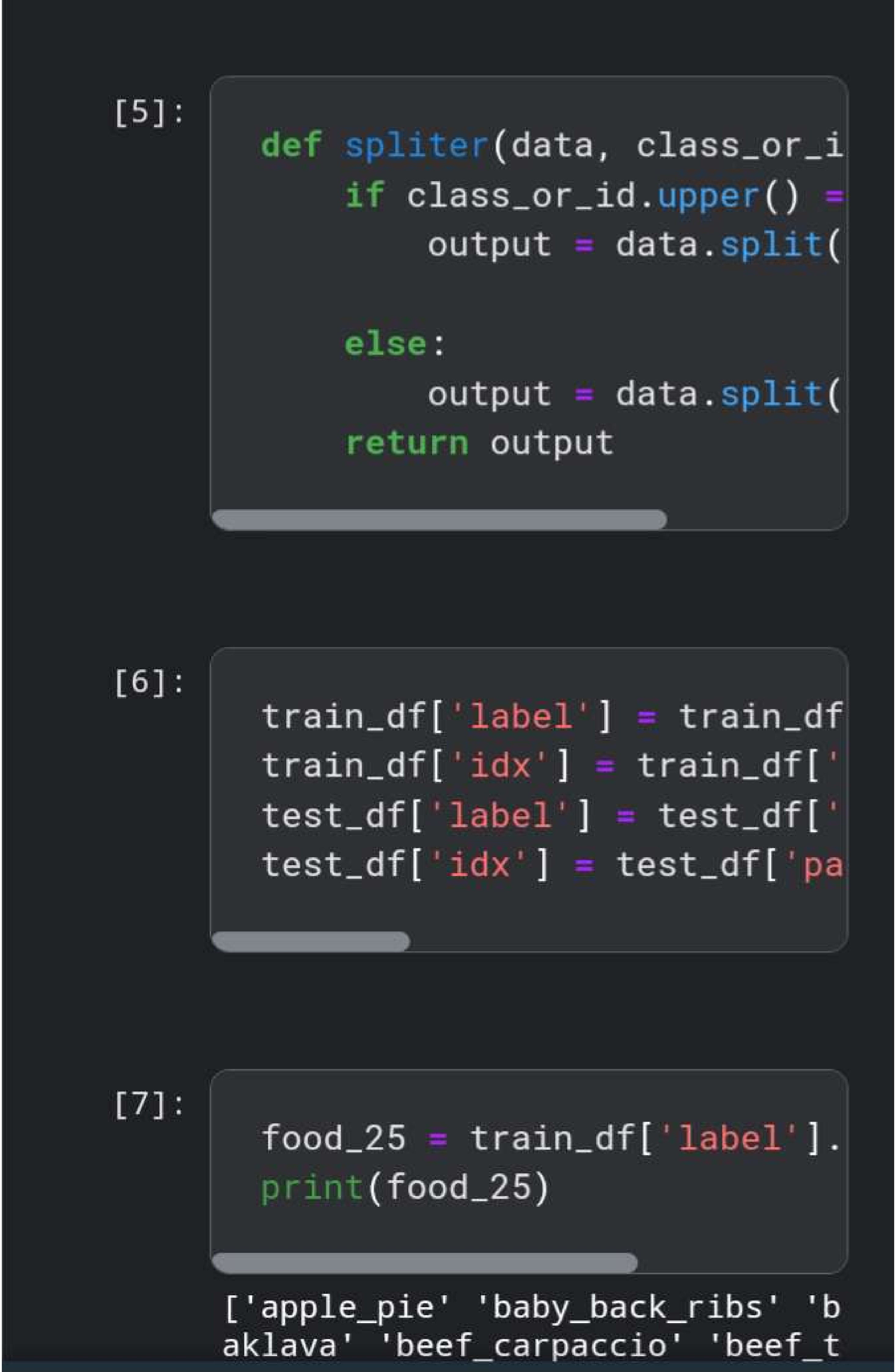
```
train_df = pd.read_csv('food
test_df = pd.read_csv('food-
len(train_df)
```

[4]:

```
75750
```

[5]:

```
def spliter(data, class_or_i
    if class_or_id.upper() =
```



[5]:

```
def spliter(data, class_or_i
    if class_or_id.upper() =
        output = data.split(

    else:
        output = data.split(
    return output
```

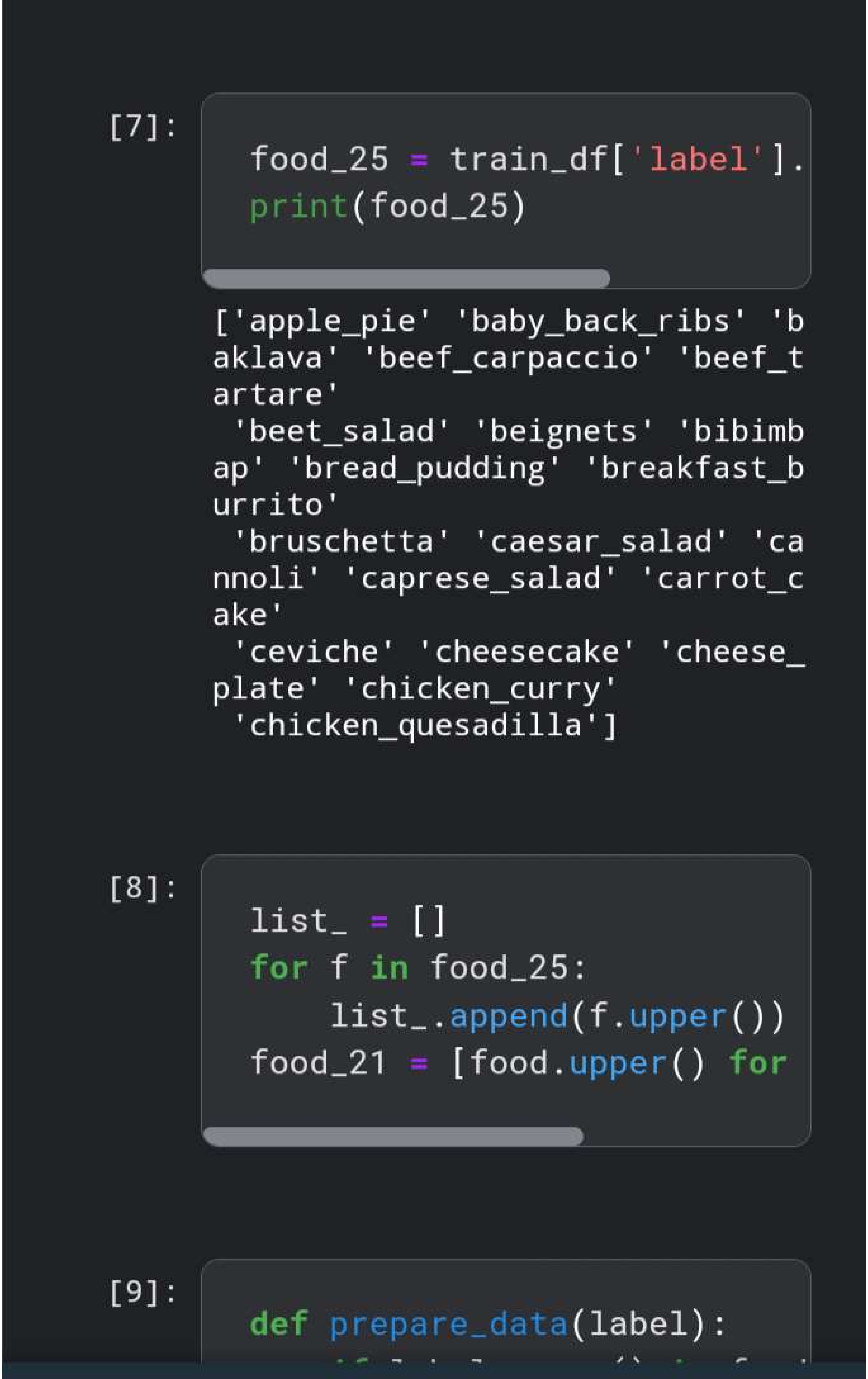
[6]:

```
train_df['label'] = train_df
train_df['idx'] = train_df['
test_df['label'] = test_df['
test_df['idx'] = test_df['pa
```

[7]:

```
food_25 = train_df['label'].
print(food_25)
```

```
['apple_pie' 'baby_back_ribs' 'b
aklava' 'beef_carpaccio' 'beef_t
```



[7]:

```
food_25 = train_df['label'].
print(food_25)
```

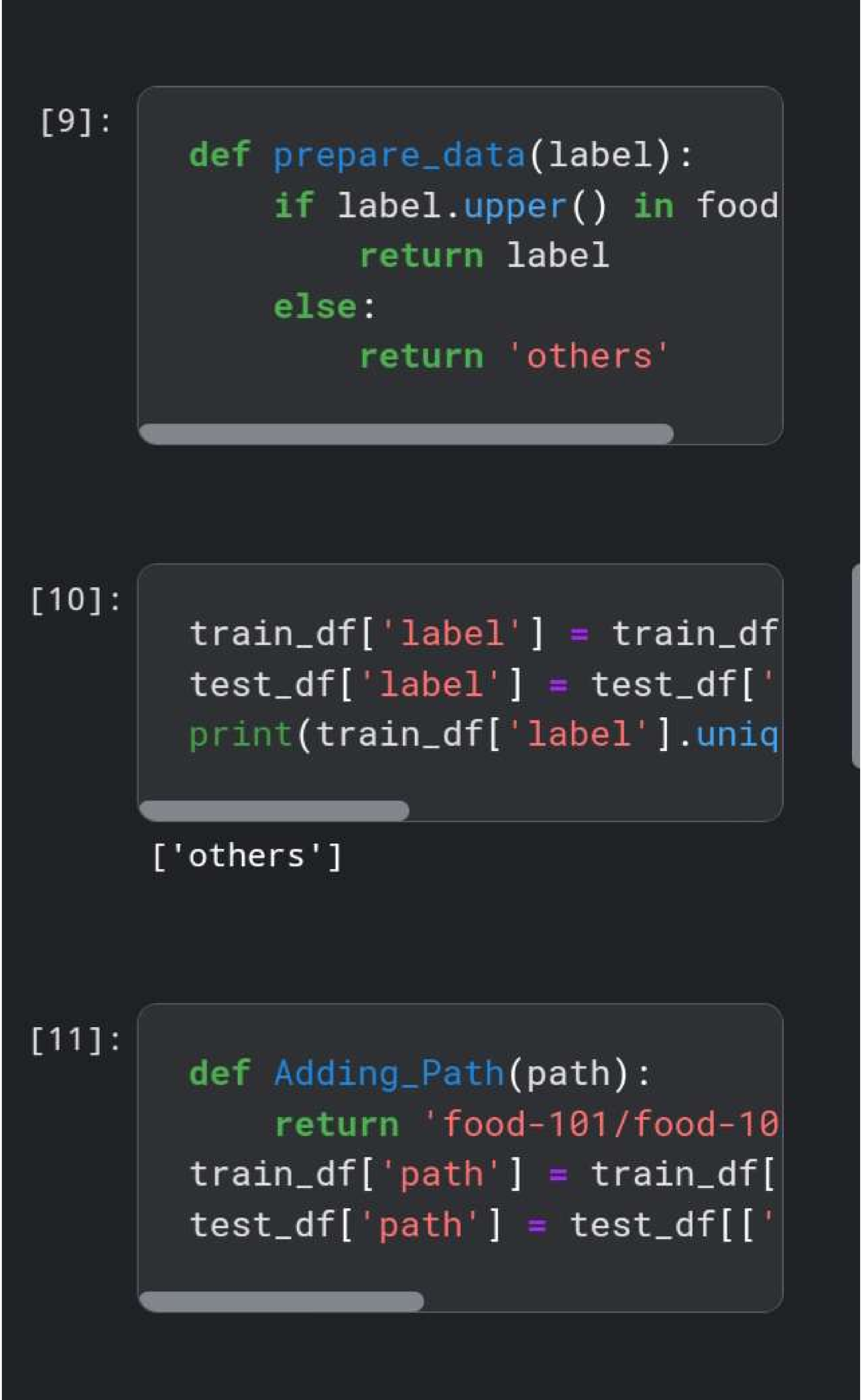
```
['apple_pie' 'baby_back_ribs' 'b
aklava' 'beef_carpaccio' 'beef_t
artare'
 'beet_salad' 'beignets' 'bibimb
ap' 'bread_pudding' 'breakfast_b
urrito'
 'bruschetta' 'caesar_salad' 'ca
nnoli' 'caprese_salad' 'carrot_c
ake'
 'ceviche' 'cheesecake' 'cheese_
plate' 'chicken_curry'
 'chicken_quesadilla']
```

[8]:

```
list_ = []
for f in food_25:
    list_.append(f.upper())
food_21 = [food.upper() for
```

[9]:

```
def prepare_data(label):
```



[9]:

```
def prepare_data(label):
    if label.upper() in food
        return label
    else:
        return 'others'
```

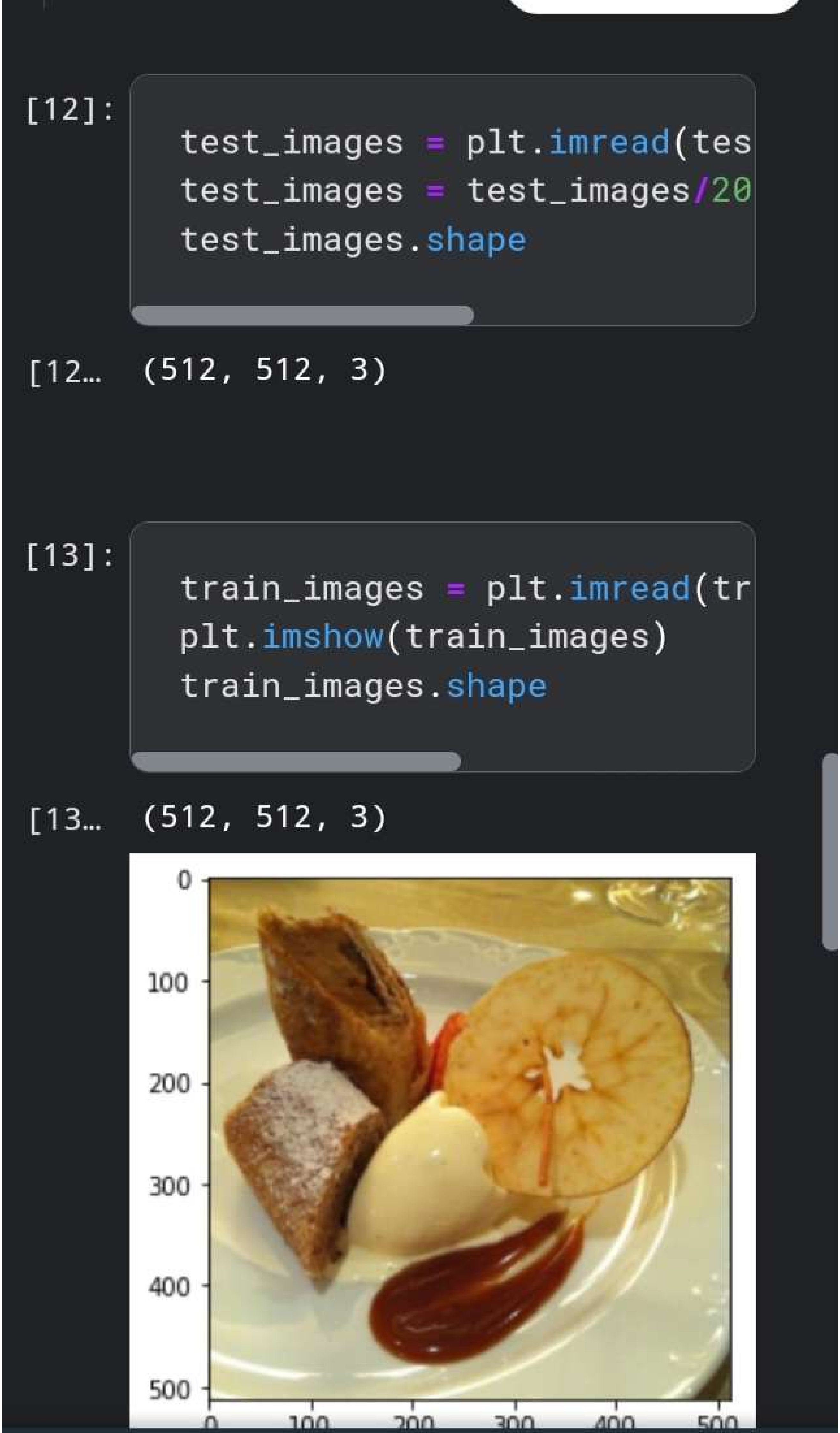
[10]:

```
train_df['label'] = train_df
test_df['label'] = test_df['
print(train_df['label'].uniq
```

```
['others']
```

[11]:

```
def Adding_Path(path):
    return 'food-101/food-10
train_df['path'] = train_df[
test_df['path'] = test_df[['
```



[12]:

```
test_images = plt.imread(tes
test_images = test_images/20
test_images.shape
```

[12…

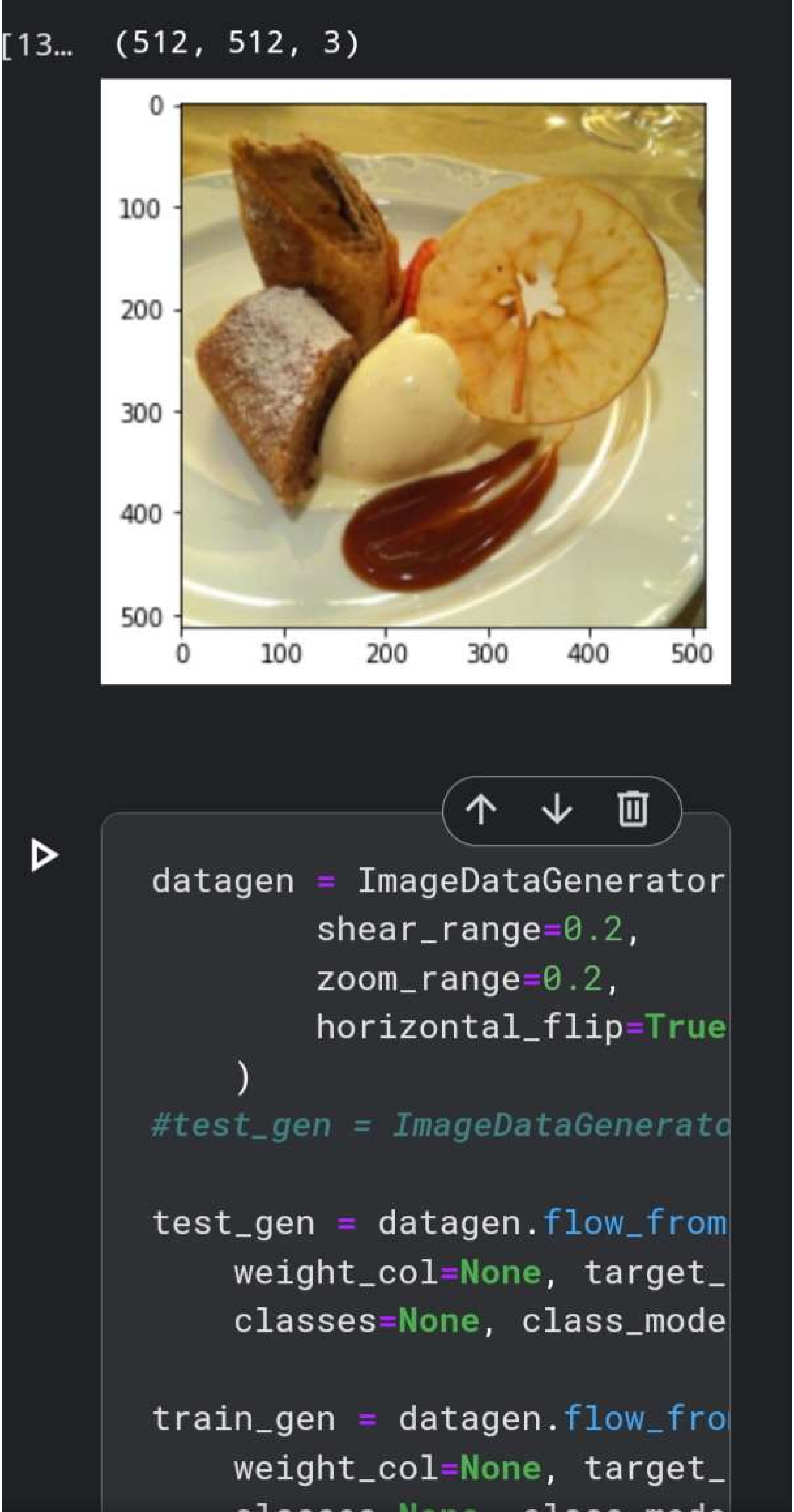
```
(512, 512, 3)
```

[13]:

```
train_images = plt.imread(tr
plt.imshow(train_images)
train_images.shape
```

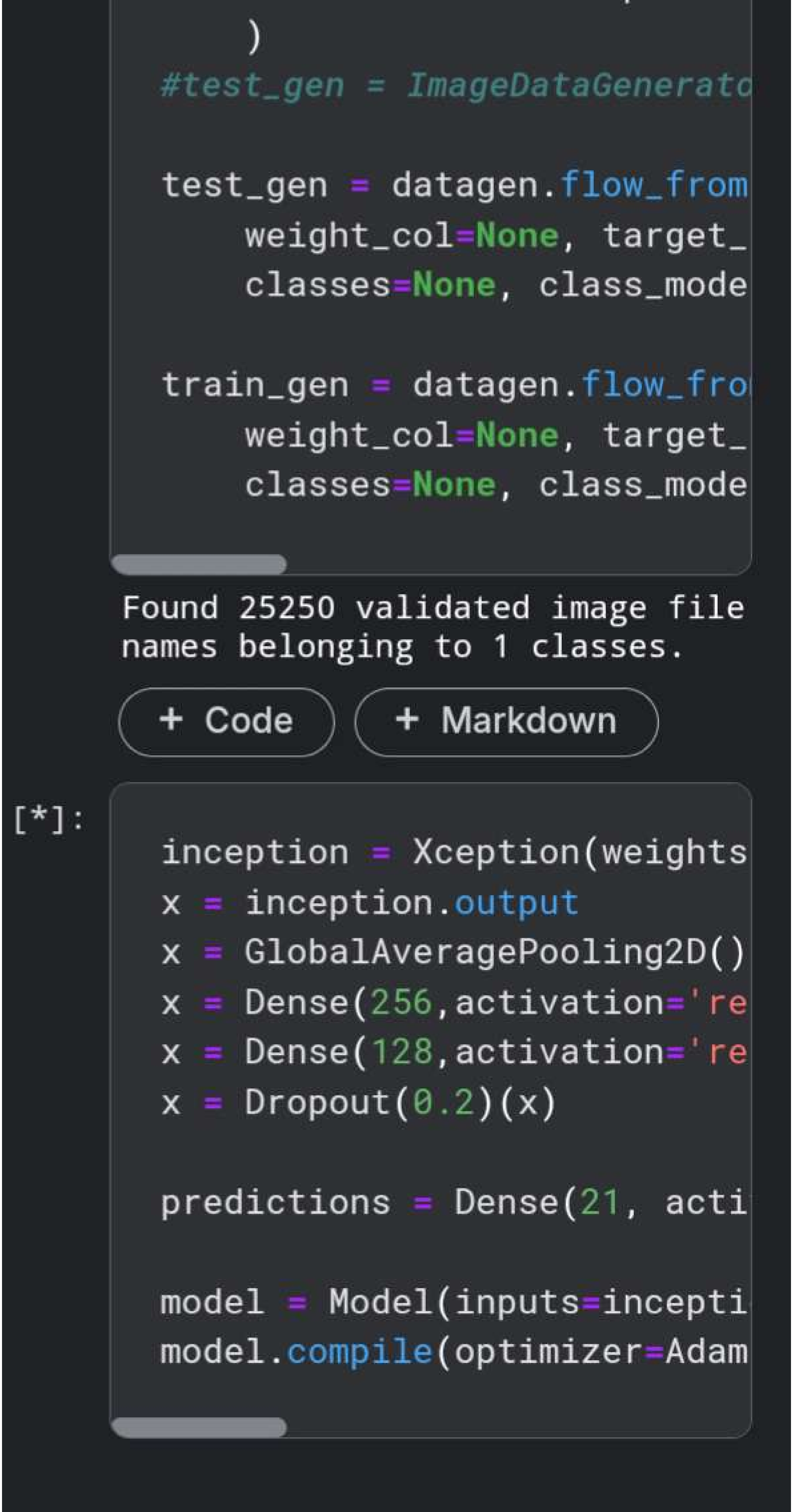
[13…

```
(512, 512, 3)
```



```
(512, 512, 3)
```

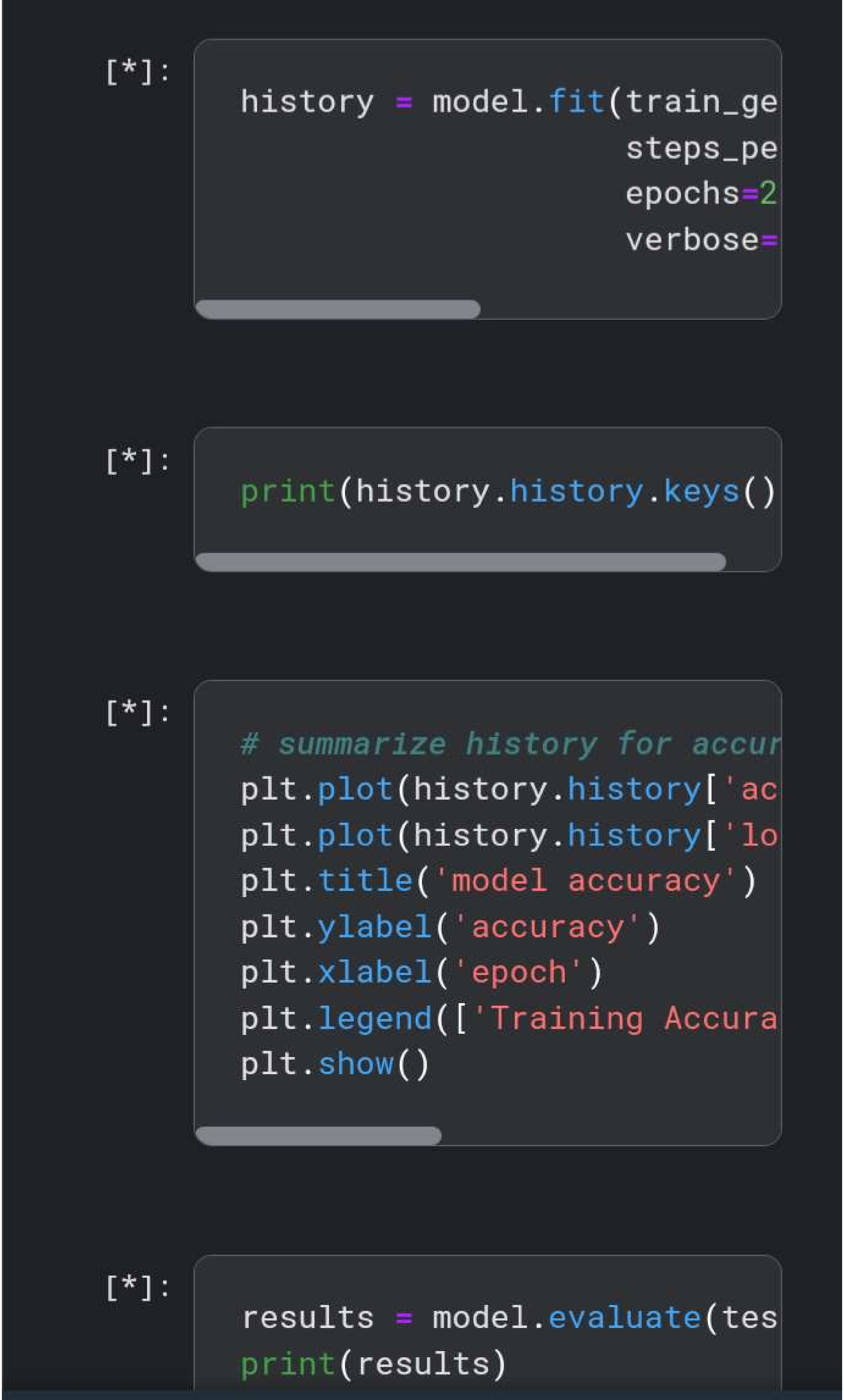
```
datagen = ImageDataGenerator
          shear_range=0.2,
          zoom_range=0.2,
          horizontal_flip=True
     )
#test_gen = ImageDataGenerato

test_gen = datagen.flow_from
    weight_col=None, target_
    classes=None, class_mode

train_gen = datagen.flow_fro
    weight_col=None, target_
    classes=None, class_mode
```



```
    )
#test_gen = ImageDataGenerato

test_gen = datagen.flow_from
    weight_col=None, target_
    classes=None, class_mode

train_gen = datagen.flow_fro
    weight_col=None, target_
    classes=None, class_mode
```

```
Found 25250 validated image file
names belonging to 1 classes.
```

```
inception = Xception(weights
x = inception.output
x = GlobalAveragePooling2D()
x = Dense(256,activation='re
x = Dense(128,activation='re
x = Dropout(0.2)(x)

predictions = Dense(21, acti

model = Model(inputs=incepti
model.compile(optimizer=Adam
```



[*]:

```
history = model.fit(train_ge
                    steps_pe
                    epochs=2
                    verbose=
```

[*]:

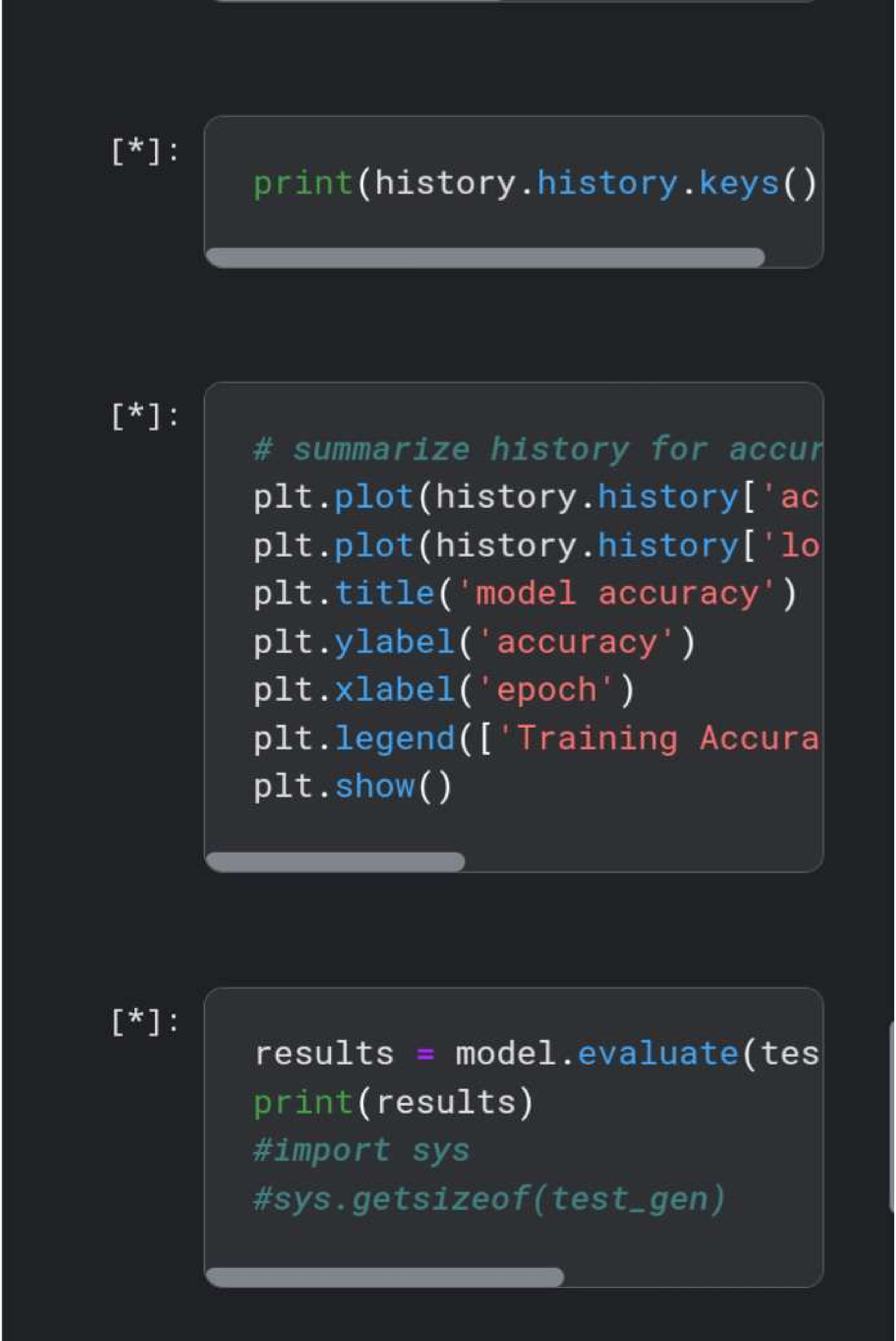
```
print(history.history.keys()
```

[*]:

```
# summarize history for accur
plt.plot(history.history['ac
plt.plot(history.history['lo
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Training Accura
plt.show()
```

[*]:

```
results = model.evaluate(tes
print(results)
```



[*]:

```
print(history.history.keys()
```

[*]:

```
# summarize history for accur
plt.plot(history.history['ac
plt.plot(history.history['lo
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Training Accura
plt.show()
```

[*]:

```
results = model.evaluate(tes
print(results)
#import sys
#sys.getsizeof(test_gen)
```