Assignment -3

Python Programming

Assignment Date	18-11- 2022
Student Name	ANBARASAN.S
Student Roll Number	922519205011
Maximum Marks	2 Marks

Question-1:

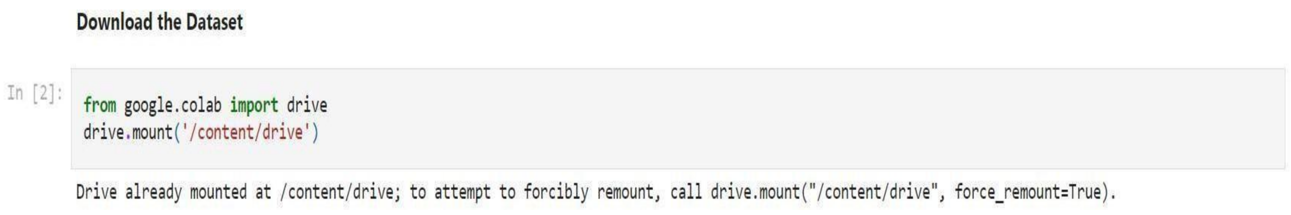
Download the Dataset

Solution:

```
from google.colab

import drivedrive.mount(' /content/drive')
```

#---#

#---#



Download the Dataset

In [2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Question-2:

Image Augmentation

Solution :

Image Augmentation

```
In [3]:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
import cv2
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
import PIL
import random
import cv2
from tensorflow.keras import layers, models
import tensorflow as tf
import pandas as pd
from sklearn.model_selection import train_test_split
import seaborn as sns
import pickle
import zipfile
tf.__version__
```

```
Out[3]: '2.8.2'
```

```
In [4]:
!ls
```

```
drive  sample_data
```

```
In [5]:
try:
    tpu = tf.distribute.cluster_resolver.TPUClusterResolver()
    print('Device:', tpu.master())
    tf.config.experimental_connect_to_cluster(tpu)
    tf.tpu.experimental.initialize_tpu_system(tpu)
    strategy = tf.distribute.experimental.TPUStrategy(tpu)
except:
    strategy = tf.distribute.get_strategy()
print('Number of replicas:', strategy.num_replicas_in_sync)
```

```
Number of replicas: 1
```

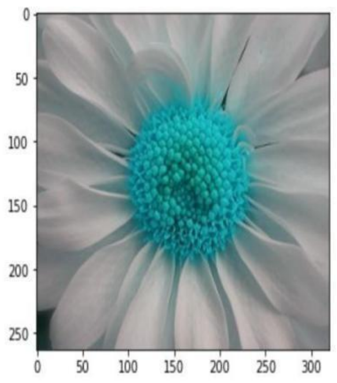
```
In [6]:
AUTOTUNE = tf.data.experimental.AUTOTUNE
batch_size = 32
IMAGE_SIZE = [128, 128]
EPOCHS = 25
```

```
In [7]:
image = cv2.imread(r'/content/drive/MyDrive/Flowers-Dataset/flowers/daisy/100080576_f52e8ee070_n.jpg')
```

```
In [8]:
print(image.shape)
```

```
(263, 320, 3)
```

```
In [9]:
imgplot = plt.imshow(image)
plt.show()
```



In [10]:
```
GCS_PATH = "/content/drive/MyDrive/Flowers-Dataset/flowers"

CLASS_NAMES = np.array([str(tf.strings.split(item, os.path.sep)[-1].numpy())[2:-1]
                        for item in tf.io.gfile.glob(str(GCS_PATH + "*/*"))])
CLASS_NAMES
```

Out[10]:
```
array(['daisy', 'rose', 'dandelion', 'sunflower', 'tulip'], dtype='<U9')
```

In [11]:
```
files_count = []
for i,f in enumerate(CLASS_NAMES):
    folder_path = os.path.join(GCS_PATH, f)
    for path in os.listdir(os.path.join(folder_path)):
        files_count.append(['{}/{}'.format(folder_path,path), f, i])
flowers_df = pd.DataFrame(files_count, columns=['filepath', 'class_name', 'label'])
flowers_df.head()
```

Out[11]:

	filepath	class_name	label
0	/content/drive/MyDrive/Flowers-Dataset/flowers...	daisy	0
1	/content/drive/MyDrive/Flowers-Dataset/flowers...	daisy	0
2	/content/drive/MyDrive/Flowers-Dataset/flowers...	daisy	0
3	/content/drive/MyDrive/Flowers-Dataset/flowers...	daisy	0
4	/content/drive/MyDrive/Flowers-Dataset/flowers...	daisy	0

In [12]:
```
flowers_df.class_name.value_counts()
```

Out[12]:
```
dandelion    1052
tulip         984
rose          784
daisy         764
sunflower     733
Name: class_name, dtype: int64
```

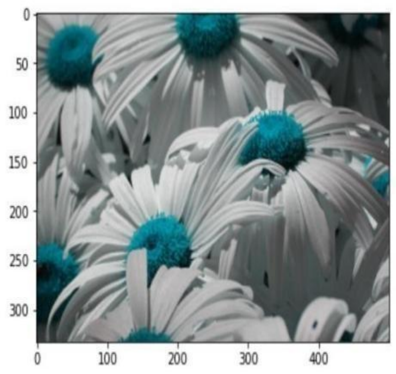
In [13]:
```
quantidade_por_class = 500
flowers_df = pd.concat([flowers_df[flowers_df['class_name']== i][:quantidade_por_class] for i in CLASS_NAMES])
```

In [14]:
```
flowers_df.class_name.value_counts()
```

Out[14]:
```
daisy        500
rose         500
dandelion    500
sunflower    500
tulip        500
```

```
sunflower    500
tulip        500
Name: class_name, dtype: int64
```

In [15]:
```
image = cv2.imread(flowers_df.filepath[100])
imgplot = plt.imshow(image)
plt.show()
```



Create Model

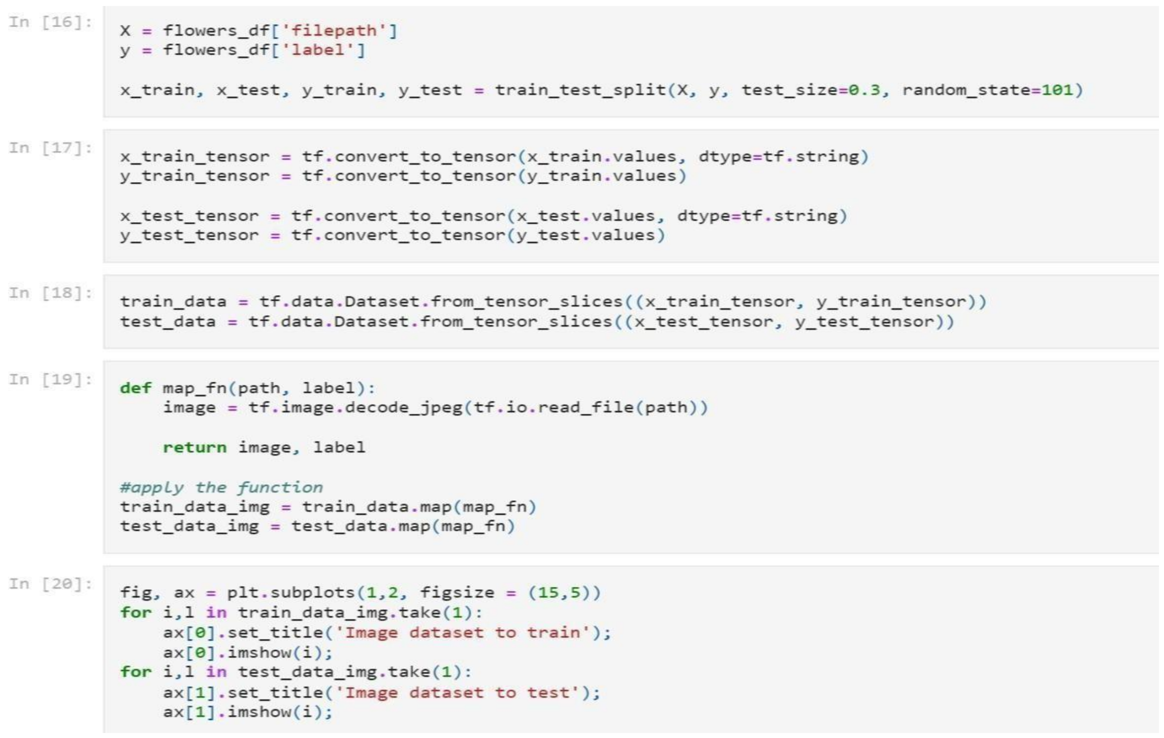
In [16]:
```
X = flowers_df['filepath']
y = flowers_df['label']

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
```

In [17]:

Question-3:

Create Model

Solution:



```
In [16]:  X = flowers_df['filepath']
          y = flowers_df['label']

          x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
```

```
In [17]:  x_train_tensor = tf.convert_to_tensor(x_train.values, dtype=tf.string)
          y_train_tensor = tf.convert_to_tensor(y_train.values)

          x_test_tensor = tf.convert_to_tensor(x_test.values, dtype=tf.string)
          y_test_tensor = tf.convert_to_tensor(y_test.values)
```

```
In [18]:  train_data = tf.data.Dataset.from_tensor_slices((x_train_tensor, y_train_tensor))
          test_data = tf.data.Dataset.from_tensor_slices((x_test_tensor, y_test_tensor))
```

```
In [19]:  def map_fn(path, label):
              image = tf.image.decode_jpeg(tf.io.read_file(path))

              return image, label

          #apply the function
          train_data_img = train_data.map(map_fn)
          test_data_img = test_data.map(map_fn)
```

```
In [20]:  fig, ax = plt.subplots(1,2, figsize = (15,5))
          for i,l in train_data_img.take(1):
              ax[0].set_title('Image dataset to train');
              ax[0].imshow(i);
          for i,l in test_data_img.take(1):
              ax[1].set_title('Image dataset to test');
              ax[1].imshow(i);
```