```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES
# TO THE CORRECT LOCATION (/kaggle/input) IN YOUR NOTEBOOK,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.

import os
import sys
from tempfile import NamedTemporaryFile
from urllib.request import urlopen
from urllib.parse import unquote, urlparse
from urllib.error import HTTPError
from zipfile import ZipFile
import tarfile
import shutil

CHUNK_SIZE = 40960
DATA_SOURCE_MAPPING = 'crop-pest-and-disease-detection:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2F4280832%2F7368427%2Fbunc

KAGGLE_INPUT_PATH='/kaggle/input'
KAGGLE_WORKING_PATH='/kaggle/working'
KAGGLE_SYMLINK='kaggle'

!umount /kaggle/input/ 2> /dev/null
shutil.rmtree('/kaggle/input', ignore_errors=True)
os.makedirs(KAGGLE_INPUT_PATH, 0o777, exist_ok=True)
os.makedirs(KAGGLE_WORKING_PATH, 0o777, exist_ok=True)

try:
  os.symlink(KAGGLE_INPUT_PATH, os.path.join("..", 'input'), target_is_directory=True)
except FileExistsError:
  pass
try:
  os.symlink(KAGGLE_WORKING_PATH, os.path.join("..", 'working'), target_is_directory=True)
except FileExistsError:
  pass

for data_source_mapping in DATA_SOURCE_MAPPING.split(','):
    directory, download_url_encoded = data_source_mapping.split(':')
    download_url = unquote(download_url_encoded)
    filename = urlparse(download_url).path
    destination_path = os.path.join(KAGGLE_INPUT_PATH, directory)
    try:
        with urlopen(download_url) as fileres, NamedTemporaryFile() as tfile:
            total_length = fileres.headers['content-length']
            print(f'Downloading {directory}, {total_length} bytes compressed')
            dl = 0
            data = fileres.read(CHUNK_SIZE)
            while len(data) > 0:
                dl += len(data)
                tfile.write(data)
                done = int(50 * dl / int(total_length))
                sys.stdout.write(f"\r[{'=' * done}{' ' * (50-done)}] {dl} bytes downloaded")
                sys.stdout.flush()
                data = fileres.read(CHUNK_SIZE)
            if filename.endswith('.zip'):
              with ZipFile(tfile) as zfile:
                zfile.extractall(destination_path)
            else:
              with tarfile.open(tfile.name) as tarfile:
                tarfile.extractall(destination_path)
            print(f'\nDownloaded and uncompressed: {directory}')
    except HTTPError as e:
        print(f'Failed to load (likely expired) {download_url} to path {destination_path}')
        continue
    except OSError as e:
        print(f'Failed to load {download_url} to path {destination_path}')
        continue

print('Data source import complete.')
```

```
Downloading crop-pest-and-disease-detection, 1337579661 bytes compressed
[==================================================] 1337579661 bytes downloaded
Downloaded and uncompressed: crop-pest-and-disease-detection
Data source import complete.
```

```
import os
from PIL import Image
import numpy as np
import shutil



data_dir=r'/kaggle/input/crop-pest-and-disease-detection'
bad_img_list=[]
total=0
good=0
bad=0
classes=sorted(os.listdir(data_dir))
for klass in classes:
    good_class=0
    bad_class=0
    total_class=0
    msg=f'processing class {klass}'
    print(msg, '\r', end= '')
    classpath=os.path.join(data_dir, klass)
    flist=sorted(os.listdir(classpath))
    for f in flist:
        total +=1
        total_class +=1
        fpath=os.path.join(classpath,f)
        try:
            img= Image.open(fpath)
            array=np.asarray(img)
            good +=1
            good_class +=1
        except:
            bad_img_list.append(fpath)
            bad +=1
            bad_class +=1

    msg=f'class {klass} contains {total_class} files, {good_class} are valid image files and {bad_class} defective image files'
    print (msg)
msg=f'the dataset contains {total} image files, {good} are valid image files and {bad} are defective image files'
print (msg)
if bad>0:
    ans=input('to print a list of defective image files enter P, to not print press Enter')
    if ans == 'P' or ans == 'p':
        for f in bad_img_list:
            print (f)
```

```
 class Cashew anthracnose contains 1729 files, 1729 are valid image files and 0 defective image files
 class Cashew gumosis contains 392 files, 392 are valid image files and 0 defective image files
 class Cashew healthy contains 1368 files, 1368 are valid image files and 0 defective image files
 class Cashew leaf miner contains 1378 files, 1378 are valid image files and 0 defective image files
 class Cashew red rust contains 1682 files, 1682 are valid image files and 0 defective image files
 class Cassava bacterial blight contains 2614 files, 2614 are valid image files and 0 defective image files
 class Cassava brown spot contains 1481 files, 1481 are valid image files and 0 defective image files
 class Cassava green mite contains 1015 files, 1015 are valid image files and 0 defective image files
 class Cassava healthy contains 1193 files, 1193 are valid image files and 0 defective image files
 class Cassava mosaic contains 1205 files, 1205 are valid image files and 0 defective image files
 class Maize fall armyworm contains 285 files, 285 are valid image files and 0 defective image files
 class Maize grasshoper contains 673 files, 673 are valid image files and 0 defective image files
 class Maize healthy contains 208 files, 204 are valid image files and 4 defective image files
 class Maize leaf beetle contains 948 files, 933 are valid image files and 15 defective image files
 class Maize leaf blight contains 1006 files, 990 are valid image files and 16 defective image files
 class Maize leaf spot contains 1259 files, 1239 are valid image files and 20 defective image files
 class Maize streak virus contains 979 files, 965 are valid image files and 14 defective image files
 class Tomato healthy contains 470 files, 466 are valid image files and 4 defective image files
 class Tomato leaf blight contains 1301 files, 1288 are valid image files and 13 defective image files
 class Tomato leaf curl contains 518 files, 511 are valid image files and 7 defective image files
 class Tomato septoria leaf spot contains 2743 files, 2743 are valid image files and 0 defective image files
 class Tomato verticulium wilt contains 773 files, 772 are valid image files and 1 defective image files
 the dataset contains 25220 image files, 25126 are valid image files and 94 are defective image files
 to print a list of defective image files enter P, to not print press Enter10
```

```
# this code creates a corrected dataset with the defective image files removed
# you can use this dataset to create a model.
working_dir=r'/kaggle/working/'
corrected_dir=os.path.join(working_dir, 'corrected dataset') # where the corrected dataset will be stored
copied_count = 0
if os.path.isdir(corrected_dir):
    shutil.rmtree(corrected_dir) # make sure the corrected_dir is empty
os.mkdir(corrected_dir)
for klass in classes:
    classpath=os.path.join(data_dir, klass)
    dest_classpath=os.path.join(corrected_dir, klass)
    os.mkdir(dest_classpath)
    flist= os.listdir(classpath)
    for f in flist:
```

```
            fpath=os.path.join(classpath,f)
            dest_fpath=os.path.join(dest_classpath,f)
            if fpath not in bad_img_list:
                shutil.copy(fpath, dest_fpath)
                copied_count +=1
    msg=f'{copied_count} valid image files were stored in {corrected_dir}'
    print(msg)
```

```
25126 valid image files were stored in /kaggle/working/corrected dataset
```