these are the packages that are required for the project

In [2]:
```
import os
import itertools
import shutil
import matplotlib.pyplot as plt
import cv2
import numpy as np
import imutils
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing.image import ImageDataGenerator
RANDOM_SEED = 123
#
from keras.applications.vgg16 import VGG16
from keras.models import Model, Sequential
from keras import layers
from keras.optimizers import Adam, RMSprop
from keras.callbacks import EarlyStopping
from sklearn.metrics import accuracy_score, confusion_matrix
IMG_SIZE = (224,224)
```

In [5]:

```
Collecting tensorflow
  Downloading tensorflow-2.4.1-cp38-cp38-win_amd64.whl (370.7 MB)
Requirement already satisfied: h5py~=2.10.0 in c:\users\venka\anaconda3\lib\s
ite-packages (from tensorflow) (2.10.0)
Collecting wheel~=0.35
  Using cached wheel-0.36.2-py2.py3-none-any.whl (35 kB)
Collecting google-pasta~=0.2
  Using cached google_pasta-0.2.0-py3-none-any.whl (57 kB)
Collecting grpcio~=1.32.0
  Downloading grpcio-1.32.0-cp38-cp38-win_amd64.whl (2.6 MB)
Collecting protobuf>=3.9.2
  Using cached protobuf-3.15.8-py2.py3-none-any.whl (173 kB)
Collecting opt-einsum~=3.3.0
  Using cached opt_einsum-3.3.0-py3-none-any.whl (65 kB)
Collecting tensorboard~=2.4
  Using cached tensorboard-2.5.0-py3-none-any.whl (6.0 MB)
Collecting astunparse~=1.6.3
  Using cached astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Collecting keras-preprocessing~=1.1.2
  Using cached Keras_Preprocessing-1.1.2-py2.py3-none-any.whl (42 kB)
Collecting termcolor~=1.1.0
  Using cached termcolor-1.1.0.tar.gz (3.9 kB)
Collecting gast==0.3.3
  Downloading gast-0.3.3-py2.py3-none-any.whl (9.7 kB)
Requirement already satisfied: typing-extensions~=3.7.4 in c:\users\venka\ana
conda3\lib\site-packages (from tensorflow) (3.7.4.2)
Collecting numpy~=1.19.2
  Downloading numpy-1.19.5-cp38-cp38-win_amd64.whl (13.3 MB)
Collecting tensorflow-estimator<2.5.0,>=2.4.0
  Downloading tensorflow_estimator-2.4.0-py2.py3-none-any.whl (462 kB)
Collecting wrapt~=1.12.1
  Using cached wrapt-1.12.1.tar.gz (27 kB)
Collecting flatbuffers~=1.12.0
  Using cached flatbuffers-1.12-py2.py3-none-any.whl (15 kB)
Requirement already satisfied: six~=1.15.0 in c:\users\venka\anaconda3\lib\si
te-packages (from tensorflow) (1.15.0)
Collecting absl-py~=0.10
  Using cached absl_py-0.12.0-py3-none-any.whl (129 kB)
Collecting tensorboard-plugin-wit>=1.6.0
  Using cached tensorboard_plugin_wit-1.8.0-py3-none-any.whl (781 kB)
Collecting google-auth<2,>=1.6.3
  Using cached google_auth-1.30.0-py2.py3-none-any.whl (146 kB)
Requirement already satisfied: setuptools>=41.0.0 in c:\users\venka\anaconda3
\lib\site-packages (from tensorboard~=2.4->tensorflow) (49.2.0.post20200714)
Collecting tensorboard-data-server<0.7.0,>=0.6.0
  Using cached tensorboard_data_server-0.6.0-py3-none-any.whl (2.3 kB)
Requirement already satisfied: werkzeug>=0.11.15 in c:\users\venka\anaconda3
\lib\site-packages (from tensorboard~=2.4->tensorflow) (1.0.1)
Collecting google-auth-oauthlib<0.5,>=0.4.1
  Using cached google_auth_oauthlib-0.4.4-py2.py3-none-any.whl (18 kB)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\venka\anaconda
3\lib\site-packages (from tensorboard~=2.4->tensorflow) (2.24.0)
Collecting markdown>=2.6.8
  Using cached Markdown-3.3.4-py3-none-any.whl (97 kB)
Collecting pyasn1-modules>=0.2.1
  Using cached pyasn1_modules-0.2.8-py2.py3-none-any.whl (155 kB)
Collecting cachetools<5.0,>=2.0.0
```

```
  Using cached cachetools-4.2.2-py3-none-any.whl (11 kB)
Collecting rsa<5,>=3.1.4
  Using cached rsa-4.7.2-py3-none-any.whl (34 kB)
Collecting requests-oauthlib>=0.7.0
  Using cached requests_oauthlib-1.3.0-py2.py3-none-any.whl (23 kB)
Collecting pyasn1<0.5.0,>=0.4.6
  Using cached pyasn1-0.4.8-py2.py3-none-any.whl (77 kB)
Requirement already satisfied: idna<3,>=2.5 in c:\users\venka\anaconda3\lib\s
ite-packages (from requests<3,>=2.21.0->tensorboard~=2.4->tensorflow) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in c:\users\venka\anaconda3
\lib\site-packages (from requests<3,>=2.21.0->tensorboard~=2.4->tensorflow)
(3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\venka\anaconda3
\lib\site-packages (from requests<3,>=2.21.0->tensorboard~=2.4->tensorflow)
(2020.6.20)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
c:\users\venka\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorb
oard~=2.4->tensorflow) (1.25.9)
Collecting oauthlib>=3.0.0
  Using cached oauthlib-3.1.0-py2.py3-none-any.whl (147 kB)
Building wheels for collected packages: termcolor, wrapt
  Building wheel for termcolor (setup.py): started
  Building wheel for termcolor (setup.py): finished with status 'done'
  Created wheel for termcolor: filename=termcolor-1.1.0-py3-none-any.whl size
=4835 sha256=d19b830c517c9f991557ba034fb8dd307965ff89f7c9c657c310c1b2fedcb59d
  Stored in directory: c:\users\venka\appdata\local\pip\cache\wheels\a0\16\9c
\5473df82468f958445479c59e784896fa24f4a5fc024b0f501
  Building wheel for wrapt (setup.py): started
  Building wheel for wrapt (setup.py): finished with status 'done'
  Created wheel for wrapt: filename=wrapt-1.12.1-py3-none-any.whl size=19558
sha256=58a6f91b3daf1b276c4f369161df7fc9da39434b813f635b4c94a7c72afc4a45
  Stored in directory: c:\users\venka\appdata\local\pip\cache\wheels\5f\fd\9e
\b6cf5890494cb8ef0b5eaff72e5d55a70fb56316007d6dfe73
Successfully built termcolor wrapt
Installing collected packages: pyasn1, rsa, pyasn1-modules, oauthlib, cacheto
ols, requests-oauthlib, google-auth, wheel, tensorboard-plugin-wit, tensorboa
rd-data-server, protobuf, numpy, markdown, grpcio, google-auth-oauthlib, absl
-py, wrapt, termcolor, tensorflow-estimator, tensorboard, opt-einsum, keras-p
reprocessing, google-pasta, gast, flatbuffers, astunparse, tensorflow
  Attempting uninstall: wheel
    Found existing installation: wheel 0.34.2
    Uninstalling wheel-0.34.2:
      Successfully uninstalled wheel-0.34.2
  Attempting uninstall: numpy
    Found existing installation: numpy 1.18.5
    Uninstalling numpy-1.18.5:
      Successfully uninstalled numpy-1.18.5

ERROR: Could not install packages due to an OSError: [WinError 5] Access is d
enied: 'C:\\Users\\venka\\anaconda3\\Lib\\site-packages\\~umpy\\core\\_multia
rray_tests.cp38-win_amd64.pyd'
Consider using the `--user` option or check the permissions.
```

In [3]:
```
!mkdir TRAIN TEST VAL TRAIN\YES  TRAIN\NO TEST\YES TEST\NO VAL\YES VAL\NO
```

```
A subdirectory or file TRAIN already exists.
Error occurred while processing: TRAIN.
A subdirectory or file TEST already exists.
Error occurred while processing: TEST.
A subdirectory or file VAL already exists.
Error occurred while processing: VAL.
A subdirectory or file TRAIN\YES already exists.
Error occurred while processing: TRAIN\YES.
A subdirectory or file TRAIN\NO already exists.
Error occurred while processing: TRAIN\NO.
A subdirectory or file TEST\YES already exists.
Error occurred while processing: TEST\YES.
A subdirectory or file TEST\NO already exists.
Error occurred while processing: TEST\NO.
A subdirectory or file VAL\YES already exists.
Error occurred while processing: VAL\YES.
A subdirectory or file VAL\NO already exists.
Error occurred while processing: VAL\NO.
```

In [4]:
```
IMG_PATH = 'brain_tumor_dataset/'
# split the data by train/val/test
for CLASS in os.listdir(IMG_PATH):
#     print(CLASS)
#     if not CLASS.startswith('.'):
    print(CLASS)
    IMG_NUM = len(os.listdir(IMG_PATH + CLASS))
#     print(IMG_NUM)
    for (n, FILE_NAME) in enumerate(os.listdir(IMG_PATH + CLASS)):
#         print(n,FILE_NAME)
        img = IMG_PATH + CLASS + '/' + FILE_NAME
        if n < 5:
            shutil.copy(img, 'TEST/' + CLASS.upper() + '/' + FILE_NAME)
        elif n < 0.8*IMG_NUM:
            shutil.copy(img, 'TRAIN/'+ CLASS.upper() + '/' + FILE_NAME)
        else:
            shutil.copy(img, 'VAL/'+ CLASS.upper() + '/' + FILE_NAME)
```

```
no
yes
```

In [5]:
```
def load_data(dir_path):
    X = []
    y = []
    i = 0
    labels = dict()
    for path in os.listdir(dir_path):
        if not path.startswith('.'):
            labels[i] = path
            for file in os.listdir(dir_path + path):
                if not file.startswith('.'):
                    img = cv2.imread(dir_path + path + '/' + file)
                    X.append(img)
                    y.append(i)
            i += 1
    print(y)
    print(labels)
    X = np.array(X)
    y = np.array(y)
    print(y)
    print(f'{len(X)} images loaded from {dir_path} directory.')
    return X, y, labels
```

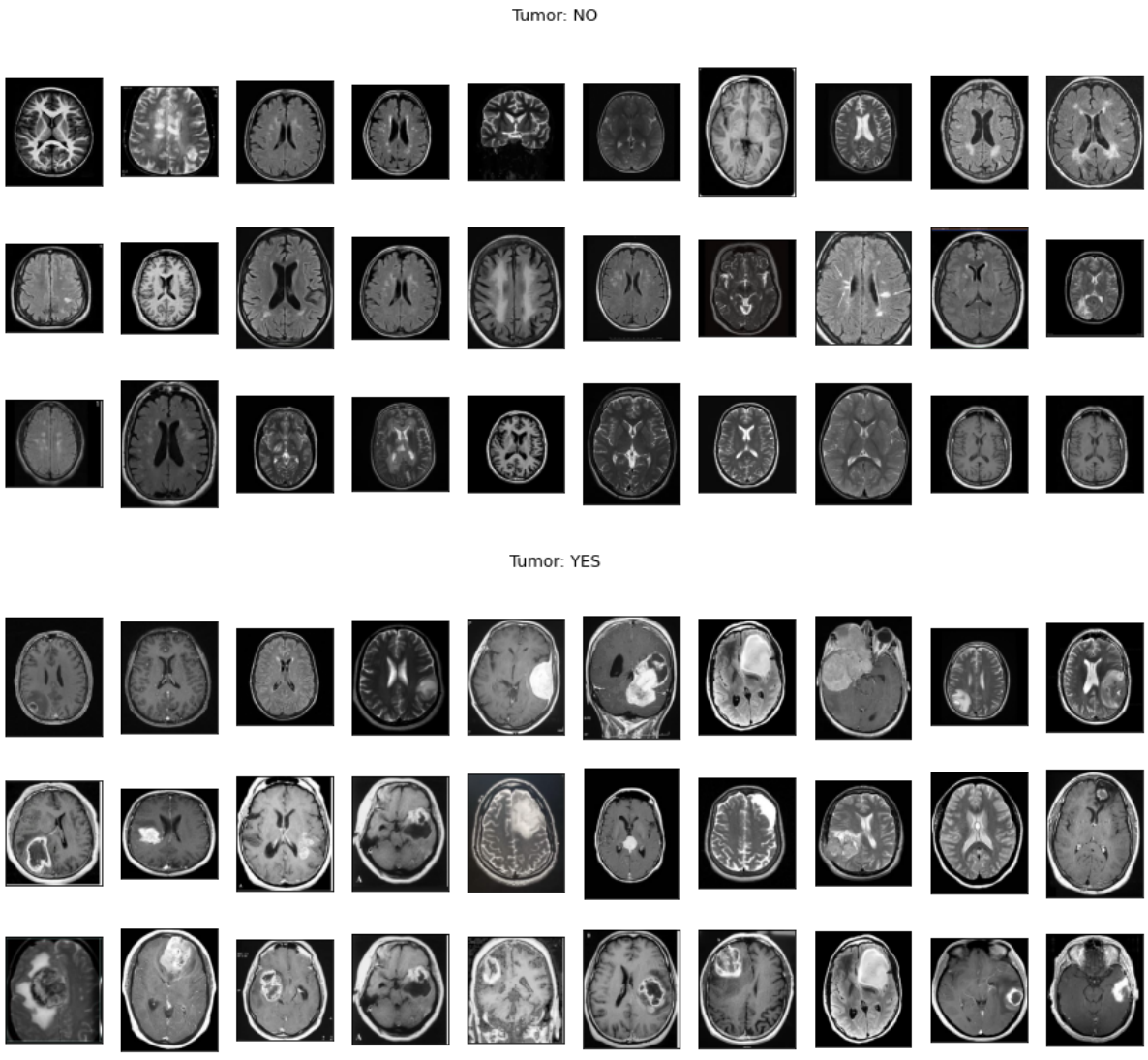
In [6]:

```
TRAIN_DIR = 'TRAIN/'
TEST_DIR = 'TEST/'
VAL_DIR = 'VAL/'
# use predefined function to load the image data into workspace
X_train, y_train, labels = load_data(TRAIN_DIR)
X_test, y_test, _ = load_data(TEST_DIR)
X_val, y_val, _ = load_data(VAL_DIR)
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
{0: 'NO', 1: 'YES'}
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
202 images loaded from TRAIN/ directory.
[0, 0, 0, 0, 0, 1, 1, 1, 1, 1]
{0: 'NO', 1: 'YES'}
[0 0 0 0 0 1 1 1 1 1]
10 images loaded from TEST/ directory.

<ipython-input-5-e15a6012faab>:17: VisibleDeprecationWarning: Creating an nda
rray from ragged nested sequences (which is a list-or-tuple of lists-or-tuple
s-or ndarrays with different lengths or shapes) is deprecated. If you meant t
o do this, you must specify 'dtype=object' when creating the ndarray
  X = np.array(X)

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
{0: 'NO', 1: 'YES'}
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
51 images loaded from VAL/ directory.
```

In [8]:
```
def plot_samples(X, y, labels_dict, n=50):
    """
    Creates a gridplot for desired number of images (n) from the specified set
    """
    for index in range(len(labels_dict)):
        imgs = X[np.argwhere(y == index)][:n]
        j = 10
        i = int(n/j)

        plt.figure(figsize=(15,6))
        c = 1
        for img in imgs:
            plt.subplot(i,j,c)
            plt.imshow(img[0])

            plt.xticks([])
            plt.yticks([])
            c += 1
        plt.suptitle('Tumor: {}'.format(labels_dict[index]))
        plt.show()
```

In [9]:
```
plot_samples(X_train, y_train, labels, 30)
```

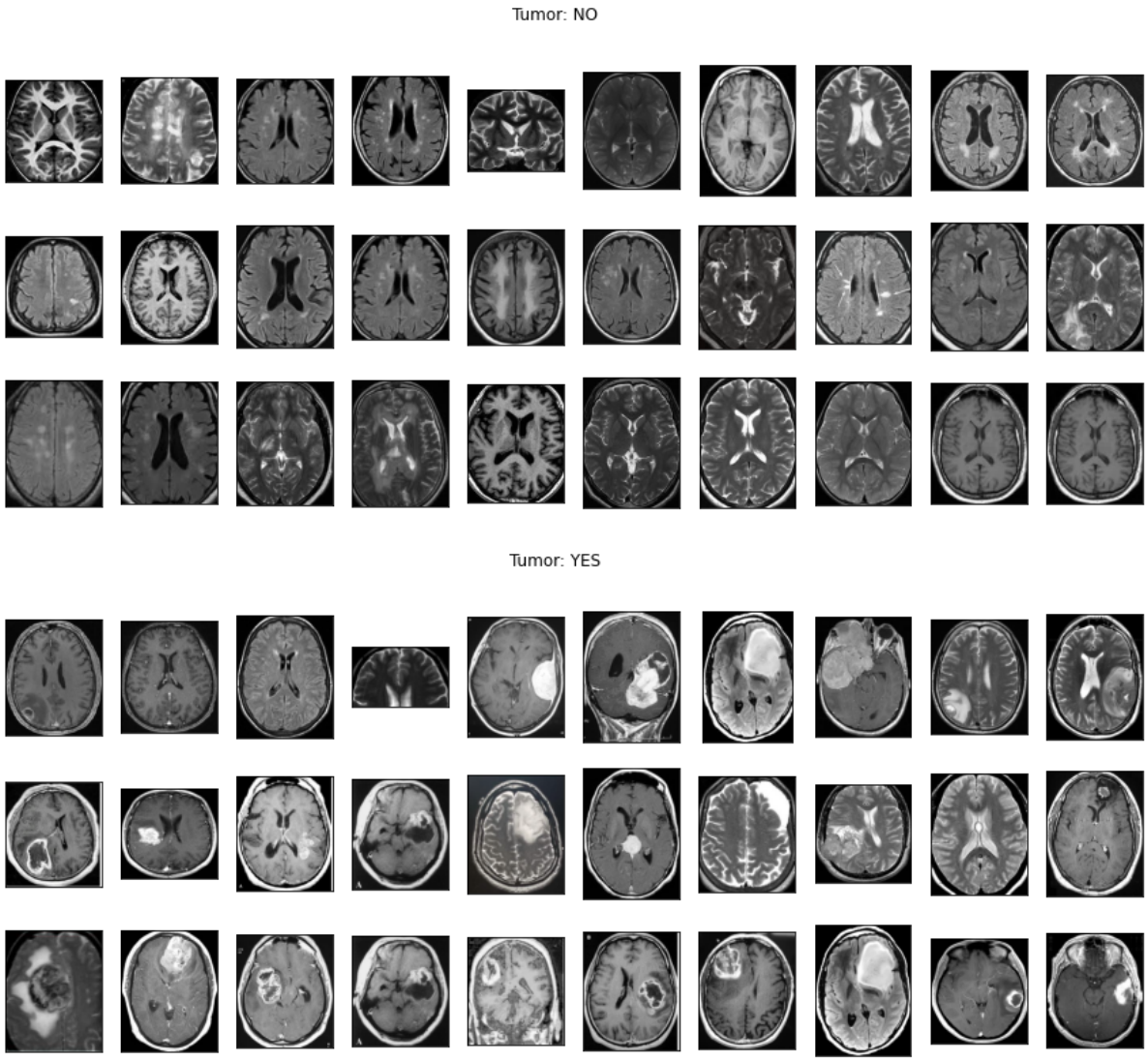


In [10]:
```
def crop_imgs(set_name, add_pixels_value=0):
    """
    Finds the extreme points on the image and crops the rectangular out of the
m
    """
    set_new = []
    for img in set_name:
#         cvtcolor for changing to gray images
# gaussian blur to make the surface smooth
        gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
        gray = cv2.GaussianBlur(gray, (5, 5), 0)

#remove the noises by thresholding.......which seperates regions......
#erode which makes partial '0' to full
# dilate which makes patial '1' to full
        thresh = cv2.threshold(gray, 45, 255, cv2.THRESH_BINARY)[1]
        thresh = cv2.erode(thresh, None, iterations=2)
        thresh = cv2.dilate(thresh, None, iterations=2)

        # find contours in thresholded image, then grab the largest one
        cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_AP
PROX_SIMPLE)
        cnts = imutils.grab_contours(cnts)
        c = max(cnts, key=cv2.contourArea)

        # find the extreme points
        extLeft = tuple(c[c[:, :, 0].argmin()][0])
        extRight = tuple(c[c[:, :, 0].argmax()][0])
        extTop = tuple(c[c[:, :, 1].argmin()][0])
        extBot = tuple(c[c[:, :, 1].argmax()][0])

        ADD_PIXELS = add_pixels_value
        new_img = img[extTop[1]-ADD_PIXELS:extBot[1]+ADD_PIXELS, extLeft[0]-AD
D_PIXELS:extRight[0]+ADD_PIXELS].copy()
        set_new.append(new_img)

    return np.array(set_new)
```

In [11]:
```
# apply this for each set
X_train_crop = crop_imgs(set_name=X_train)
X_val_crop = crop_imgs(set_name=X_val)
X_test_crop = crop_imgs(set_name=X_test)
```

```
<ipython-input-10-dd9deb84f1f8>:34: VisibleDeprecationWarning: Creating an nd
array from ragged nested sequences (which is a list-or-tuple of lists-or-tupl
es-or ndarrays with different lengths or shapes) is deprecated. If you meant
to do this, you must specify 'dtype=object' when creating the ndarray
  return np.array(set_new)
```

In [12]:
```
plot_samples(X_train_crop, y_train, labels, 30)
```



In [13]:
```
def save_new_images(x_set, y_set, folder_name):
    i = 0
    for (img, imclass) in zip(x_set, y_set):
        if imclass == 0:
            cv2.imwrite(folder_name+'NO/'+str(i)+'.jpg', img)
        else:
            cv2.imwrite(folder_name+'YES/'+str(i)+'.jpg', img)
        i += 1
```

In [14]:
```
# saving new images to the folder
!mkdir TRAIN_CROP TEST_CROP VAL_CROP TRAIN_CROP\YES TRAIN_CROP\NO TEST_CROP\YE
S TEST_CROP\NO VAL_CROP\YES VAL_CROP\NO

save_new_images(X_train_crop, y_train, folder_name='TRAIN_CROP/')
save_new_images(X_val_crop, y_val, folder_name='VAL_CROP/')
save_new_images(X_test_crop, y_test, folder_name='TEST_CROP/')
```

```
A subdirectory or file TRAIN_CROP already exists.
Error occurred while processing: TRAIN_CROP.
A subdirectory or file TEST_CROP already exists.
Error occurred while processing: TEST_CROP.
A subdirectory or file VAL_CROP already exists.
Error occurred while processing: VAL_CROP.
A subdirectory or file TRAIN_CROP\YES already exists.
Error occurred while processing: TRAIN_CROP\YES.
A subdirectory or file TRAIN_CROP\NO already exists.
Error occurred while processing: TRAIN_CROP\NO.
A subdirectory or file TEST_CROP\YES already exists.
Error occurred while processing: TEST_CROP\YES.
A subdirectory or file TEST_CROP\NO already exists.
Error occurred while processing: TEST_CROP\NO.
A subdirectory or file VAL_CROP\YES already exists.
Error occurred while processing: VAL_CROP\YES.
A subdirectory or file VAL_CROP\NO already exists.
Error occurred while processing: VAL_CROP\NO.
```

In [15]:
```
def preprocess_imgs(set_name, img_size):
    """
    Resize and apply VGG-15 preprocessing
    """
    set_new = []
    for img in set_name:
        img = cv2.resize(
            img,
            dsize=img_size,
            interpolation=cv2.INTER_CUBIC
        )
# we use preprocess_input inorder to set the images to train the model  in ker
as
        set_new.append(preprocess_input(img))
    return np.array(set_new)
```

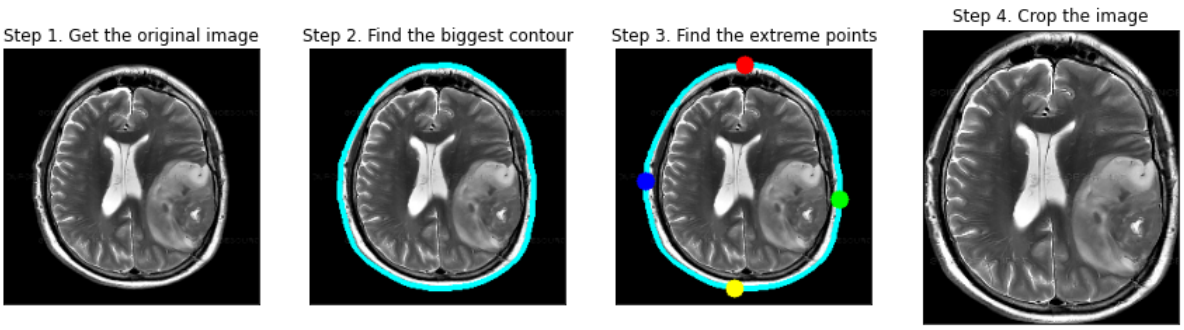
In [16]:
```
X_train_prep = preprocess_imgs(set_name=X_train_crop, img_size=IMG_SIZE)
X_test_prep = preprocess_imgs(set_name=X_test_crop, img_size=IMG_SIZE)
X_val_prep = preprocess_imgs(set_name=X_val_crop, img_size=IMG_SIZE)
```

In [17]:
```
TRAIN_DIR = 'TRAIN_CROP/'
VAL_DIR = 'VAL_CROP/'
train_datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    brightness_range=[0.5, 1.5],
    horizontal_flip=True,
    vertical_flip=True,
    preprocessing_function=preprocess_input
)

test_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input
)

print(test_datagen)
train_generator =train_datagen.flow_from_directory(
    TRAIN_DIR,
    color_mode='rgb',
    target_size=IMG_SIZE,
    batch_size=32, #we are augumenting only 32 images from 193 images..to augu
ment all change value to 193
    class_mode='binary',
    seed=RANDOM_SEED
#     , save_to_dir='preview', save_prefix='aug_img', save_format='jpg'
)

validation_generator = test_datagen.flow_from_directory(
    VAL_DIR,
    color_mode='rgb',
    target_size=IMG_SIZE,
    batch_size=16,
    class_mode='binary',
    seed=RANDOM_SEED
)
```

```
<tensorflow.python.keras.preprocessing.image.ImageDataGenerator object at 0x0
000021A0B7C2310>
Found 215 images belonging to 2 classes.
Found 52 images belonging to 2 classes.
```

In [18]:
```
img = cv2.imread('brain_tumor_dataset/yes/Y108.jpg')
img = cv2.resize(
            img,
            dsize=IMG_SIZE,
            interpolation=cv2.INTER_CUBIC
        )
gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
gray = cv2.GaussianBlur(gray, (5, 5), 0)

# threshold the image, then perform a series of erosions +
# dilations to remove any small regions of noise
thresh = cv2.threshold(gray, 45, 255, cv2.THRESH_BINARY)[1]
thresh = cv2.erode(thresh, None, iterations=2)
thresh = cv2.dilate(thresh, None, iterations=2)

# find contours in thresholded image, then grab the largest one
cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIM
PLE)
cnts = imutils.grab_contours(cnts)
c = max(cnts, key=cv2.contourArea)

# find the extreme points
extLeft = tuple(c[c[:, :, 0].argmin()][0])
extRight = tuple(c[c[:, :, 0].argmax()][0])
extTop = tuple(c[c[:, :, 1].argmin()][0])
extBot = tuple(c[c[:, :, 1].argmax()][0])

# add contour on the image
img_cnt = cv2.drawContours(img.copy(), [c], -1, (0, 255, 255), 4)

# add extreme points
img_pnt = cv2.circle(img_cnt.copy(), extLeft, 8, (0, 0, 255),-1)
img_pnt = cv2.circle(img_pnt, extRight, 8, (0, 255, 0), -1)
img_pnt = cv2.circle(img_pnt, extTop, 8, (255, 0, 0), -1)
img_pnt = cv2.circle(img_pnt, extBot, 8, (255, 255, 0), -1)

# crop
ADD_PIXELS = 0
new_img = img[extTop[1]-ADD_PIXELS:extBot[1]+ADD_PIXELS, extLeft[0]-ADD_PIXELS
:extRight[0]+ADD_PIXELS].copy()
```

In [22]:
```
plt.figure(figsize=(15,6))
plt.subplot(141)
plt.imshow(img)
plt.xticks([])
plt.yticks([])
plt.title('Step 1. Get the original image')
plt.subplot(142)
plt.imshow(img_cnt)
plt.xticks([])
plt.yticks([])
plt.title('Step 2. Find the biggest contour')
plt.subplot(143)
plt.imshow(img_pnt)
plt.xticks([])
plt.yticks([])
plt.title('Step 3. Find the extreme points')
plt.subplot(144)
plt.imshow(new_img)
plt.xticks([])
plt.yticks([])
plt.title('Step 4. Crop the image')
plt.show()
```



In [19]:
```
# set the paramters we want to change randomly
demo_datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.05,
    height_shift_range=0.05,
    rescale=1./255,
    shear_range=0.05,
    brightness_range=[0.1, 1.5],
    horizontal_flip=True,
    vertical_flip=True
)
```

In [19]:
```
# os.mkdir('preview')
# x = X_train_crop[0]
# x = x.reshape((1,) + x.shape)

# i = 0
# for batch in demo_datagen.flow(x, batch_size=1, save_to_dir='preview', save_
prefix='aug_img', save_format='jpg'):
#     i += 1
#     if i > 50:
#         break
```

In [20]:
```
# i=0
# for img in train_generator:
#     i+=1
#     if i==2:
#         break
```

In [20]:
```
vgg16_weight_path = 'vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5'
# vgg16_weight_path=None
base_model = VGG16(
    weights=vgg16_weight_path,
    include_top=False,
    input_shape=IMG_SIZE + (3,)
)
```

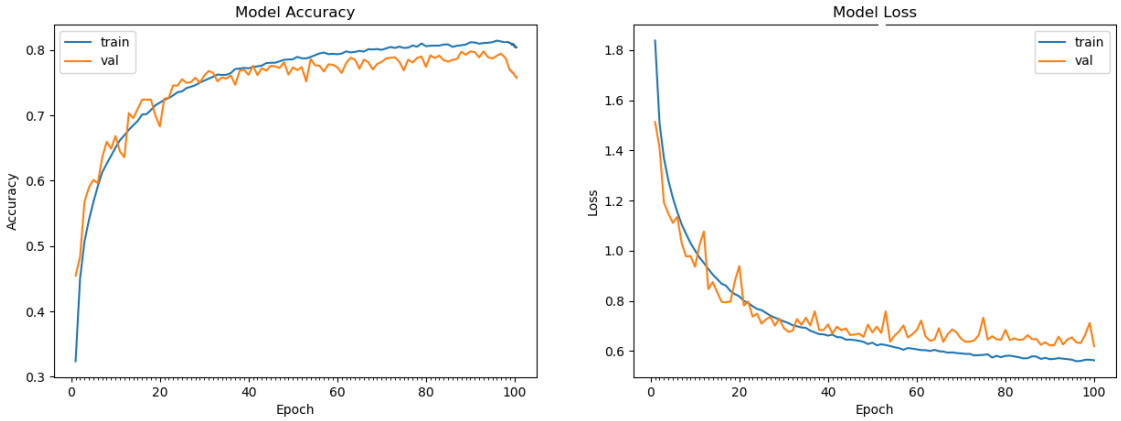
In [21]:
```
NUM_CLASSES = 1

model = Sequential()
model.add(base_model)
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(NUM_CLASSES, activation='sigmoid'))

model.layers[0].trainable = False

model.compile(
    loss='binary_crossentropy',
    optimizer=RMSprop(lr=1e-4),
    metrics=['accuracy']
)

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dropout (Dropout)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 1)                 25089
=================================================================
Total params: 14,739,777
Trainable params: 25,089
Non-trainable params: 14,714,688
_________________________________________________________________
```

In [23]:
```
EPOCHS = 100
es = EarlyStopping(
    monitor='val_accuracy',
    mode='max',
    patience=6
)

history = model.fit(
    train_generator,
    steps_per_epoch=15,
    epochs=EPOCHS,
    validation_data=validation_generator,
    validation_steps=7,
    callbacks=[es]
)
```

```
Epoch 1/30
 7/50 [===>..........................] - ETA: 7:18 - loss: 4.8436 - accuracy:
0.5814WARNING:tensorflow:Your input ran out of data; interrupting training. M
ake sure that your dataset or generator can generate at least `steps_per_epoc
h * epochs` batches (in this case, 1500 batches). You may need to use the rep
eat() function when building your dataset.
WARNING:tensorflow:Your input ran out of data; interrupting training. Make su
re that your dataset or generator can generate at least `steps_per_epoch * ep
ochs` batches (in this case, 25 batches). You may need to use the repeat() fu
nction when building your dataset.
50/50 [==============================] - 85s 1s/step - loss: 4.8436 - accurac
y: 0.5814 - val_loss: 3.6287 - val_accuracy: 0.5000
```

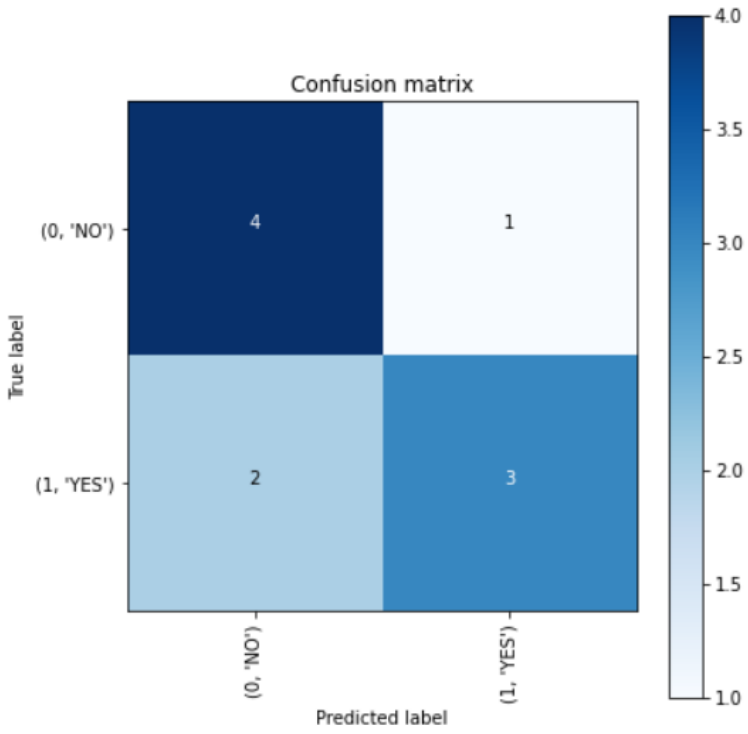
In [24]:
```
# plot model performance
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs_range = range(1, len(history.epoch) + 1)

plt.figure(figsize=(15,5))

plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Train Set')
plt.plot(epochs_range, val_acc, label='Val Set')
plt.legend(loc="best")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Model Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Train Set')
plt.plot(epochs_range, val_loss, label='Val Set')
plt.legend(loc="best")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Model Loss')
plt.tight_layout()
plt.show()
```



In [33]:
```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.figure(figsize = (6,6))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    thresh = cm.max() / 2.
    cm = np.round(cm,2)
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.show()
```

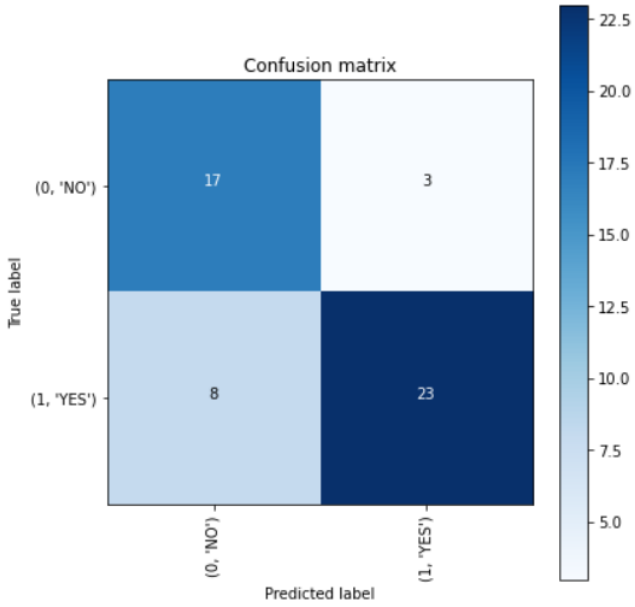
In [34]:
```
# validate on val set
predictions = model.predict(X_val_prep)
predictions = [1 if x>0.5 else 0 for x in predictions]

accuracy = accuracy_score(y_val, predictions)
print('Val Accuracy = %.2f' % accuracy)

confusion_mtx = confusion_matrix(y_val, predictions)
cm = plot_confusion_matrix(confusion_mtx, classes = list(labels.items()), norm
alize=False)
```

```
Val Accuracy = 0.74
```



In [47]:
```
# validate on test set
predictions = model.predict(X_test_prep)
predictions = [1 if x>0.5 else 0 for x in predictions]

accuracy = accuracy_score(y_test, predictions)
print('Test Accuracy = %.2f' % accuracy)

confusion_mtx = confusion_matrix(y_test, predictions)
cm = plot_confusion_matrix(confusion_mtx, classes = list(labels.items()), norm
alize=False)
```

```
Test Accuracy = 0.79
```



In [58]:
```
ind_list = np.argwhere((y_test == predictions) == False)[:,-1]
if ind_list.size == 0:
    print('There are no missclassified images.')
else:
    for i in ind_list:
        plt.figure()
        plt.imshow(X_test_crop[i])
        plt.xticks([])
        plt.yticks([])
        plt.title(f'Actual class: {y_val[i]}\nPredicted class: {predictions[i]
}')
        plt.show()
```

```
There are no missclassified images.
```

In [24]:
```
model.save('brain_tumor_detection.h5')
```

In [59]:
```
model.save('u.h5')
```

In [3]:
```
!pip install opencv-python
!pip install os-win
```

```
Requirement already satisfied: opencv-python in c:\users\venka\anaconda3\lib
\site-packages (4.5.1.48)
Requirement already satisfied: numpy>=1.17.3 in c:\users\venka\anaconda3\lib
\site-packages (from opencv-python) (1.18.5)
Collecting os-win
  Downloading os_win-5.4.0-py3-none-any.whl (273 kB)
Collecting eventlet>=0.22.0
  Downloading eventlet-0.30.2-py2.py3-none-any.whl (224 kB)
Collecting PyMI>=1.0.0; sys_platform == "win32"
  Downloading PyMI-1.0.6-cp38-cp38-win_amd64.whl (317 kB)
Collecting oslo.utils>=4.7.0
  Downloading oslo.utils-4.8.0-py3-none-any.whl (102 kB)
Collecting pbr!=2.1.0,>=2.0.0
  Downloading pbr-5.6.0-py2.py3-none-any.whl (111 kB)
Collecting oslo.config>=6.8.0
  Downloading oslo.config-8.6.0-py3-none-any.whl (128 kB)
Collecting oslo.log>=3.36.0
  Downloading oslo.log-4.4.0-py3-none-any.whl (66 kB)
Collecting wmi>=0.5; sys_platform == "win32"
  Downloading WMI-1.5.1-py2.py3-none-any.whl (28 kB)
Collecting oslo.i18n>=3.15.3
  Downloading oslo.i18n-5.0.1-py3-none-any.whl (42 kB)
Collecting oslo.concurrency>=3.29.0
  Downloading oslo.concurrency-4.4.0-py3-none-any.whl (47 kB)
Requirement already satisfied: greenlet>=0.3 in c:\users\venka\anaconda3\lib
\site-packages (from eventlet>=0.22.0->os-win) (0.4.16)
Collecting dnspython<2.0.0,>=1.15.0
  Downloading dnspython-1.16.0-py2.py3-none-any.whl (188 kB)
Requirement already satisfied: six>=1.10.0 in c:\users\venka\anaconda3\lib\si
te-packages (from eventlet>=0.22.0->os-win) (1.15.0)
Requirement already satisfied: pyparsing>=2.1.0 in c:\users\venka\anaconda3\l
ib\site-packages (from oslo.utils>=4.7.0->os-win) (2.4.7)
Collecting debtcollector>=1.2.0
  Downloading debtcollector-2.2.0-py3-none-any.whl (20 kB)
Collecting iso8601>=0.1.11
  Downloading iso8601-0.1.14-py2.py3-none-any.whl (9.5 kB)
Collecting netaddr>=0.7.18
  Downloading netaddr-0.8.0-py2.py3-none-any.whl (1.9 MB)
Requirement already satisfied: packaging>=20.4 in c:\users\venka\anaconda3\li
b\site-packages (from oslo.utils>=4.7.0->os-win) (20.4)
Requirement already satisfied: pytz>=2013.6 in c:\users\venka\anaconda3\lib\s
ite-packages (from oslo.utils>=4.7.0->os-win) (2020.1)
Collecting netifaces>=0.10.4
  Downloading netifaces-0.10.9.tar.gz (28 kB)
Requirement already satisfied: PyYAML>=5.1 in c:\users\venka\anaconda3\lib\si
te-packages (from oslo.config>=6.8.0->os-win) (5.3.1)
Collecting stevedore>=1.20.0
  Downloading stevedore-3.3.0-py3-none-any.whl (49 kB)
Requirement already satisfied: requests>=2.18.0 in c:\users\venka\anaconda3\l
ib\site-packages (from oslo.config>=6.8.0->os-win) (2.24.0)
Collecting rfc3986>=1.2.0
  Downloading rfc3986-1.4.0-py2.py3-none-any.whl (31 kB)
Requirement already satisfied: python-dateutil>=2.7.0 in c:\users\venka\anaco
nda3\lib\site-packages (from oslo.log>=3.36.0->os-win) (2.8.1)
Collecting oslo.serialization>=2.25.0
  Downloading oslo.serialization-4.1.0-py3-none-any.whl (25 kB)
Collecting oslo.context>=2.20.0
```

```
  Downloading oslo.context-3.2.0-py3-none-any.whl (19 kB)
Requirement already satisfied: pywin32 in c:\users\venka\anaconda3\lib\site-p
ackages (from wmi>=0.5; sys_platform == "win32"->os-win) (227)
Collecting fasteners>=0.7.0
  Downloading fasteners-0.16-py2.py3-none-any.whl (28 kB)
Requirement already satisfied: wrapt>=1.7.0 in c:\users\venka\anaconda3\lib\s
ite-packages (from debtcollector>=1.2.0->oslo.utils>=4.7.0->os-win) (1.11.2)
Requirement already satisfied: idna<3,>=2.5 in c:\users\venka\anaconda3\lib\s
ite-packages (from requests>=2.18.0->oslo.config>=6.8.0->os-win) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in c:\users\venka\anaconda3
\lib\site-packages (from requests>=2.18.0->oslo.config>=6.8.0->os-win) (3.0.
4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
c:\users\venka\anaconda3\lib\site-packages (from requests>=2.18.0->oslo.confi
g>=6.8.0->os-win) (1.25.9)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\venka\anaconda3
\lib\site-packages (from requests>=2.18.0->oslo.config>=6.8.0->os-win) (2020.
6.20)
Requirement already satisfied: msgpack>=0.5.2 in c:\users\venka\anaconda3\lib
\site-packages (from oslo.serialization>=2.25.0->oslo.log>=3.36.0->os-win)
(1.0.0)
Building wheels for collected packages: netifaces
  Building wheel for netifaces (setup.py): started
  Building wheel for netifaces (setup.py): finished with status 'error'
  Running setup.py clean for netifaces
Failed to build netifaces
Installing collected packages: dnspython, eventlet, pbr, PyMI, debtcollector,
iso8601, netaddr, netifaces, oslo.i18n, oslo.utils, stevedore, rfc3986, oslo.
config, oslo.serialization, oslo.context, oslo.log, wmi, fasteners, oslo.conc
urrency, os-win
    Running setup.py install for netifaces: started
    Running setup.py install for netifaces: finished with status 'error'
```

```
  ERROR: Command errored out with exit status 1:
   command: 'C:\Users\venka\anaconda3\python.exe' -u -c 'import sys, setuptoo
ls, tokenize; sys.argv[0] = '"'"'C:\\Users\\venka\\AppData\\Local\\Temp\\pip-
install-z3uyco1q\\netifaces\\setup.py'"'"'; __file__='"'"'C:\\Users\\venka\\A
ppData\\Local\\Temp\\pip-install-z3uyco1q\\netifaces\\setup.py'"'"';f=getattr
(tokenize, '"'"'open'"'"', open)(__file__);code=f.read().replace('"'"'\r
\n'"'"', '"'"'\n'"'"');f.close();exec(compile(code, __file__, '"'"'exe
c'"'"'))' bdist_wheel -d 'C:\Users\venka\AppData\Local\Temp\pip-wheel-zb_ur3a
k'
       cwd: C:\Users\venka\AppData\Local\Temp\pip-install-z3uyco1q\netifaces\
  Complete output (5 lines):
  running bdist_wheel
  running build
  running build_ext
  building 'netifaces' extension
  error: Microsoft Visual C++ 14.0 is required. Get it with "Build Tools for
Visual Studio": https://visualstudio.microsoft.com/downloads/
  ----------------------------------------
  ERROR: Failed building wheel for netifaces
    ERROR: Command errored out with exit status 1:
     command: 'C:\Users\venka\anaconda3\python.exe' -u -c 'import sys, setupt
ools, tokenize; sys.argv[0] = '"'"'C:\\Users\\venka\\AppData\\Local\\Temp\\pi
p-install-z3uyco1q\\netifaces\\setup.py'"'"'; __file__='"'"'C:\\Users\\venka
\\AppData\\Local\\Temp\\pip-install-z3uyco1q\\netifaces\\setup.py'"'"';f=geta
ttr(tokenize, '"'"'open'"'"', open)(__file__);code=f.read().replace('"'"'\r
\n'"'"', '"'"'\n'"'"');f.close();exec(compile(code, __file__, '"'"'exe
c'"'"'))' install --record 'C:\Users\venka\AppData\Local\Temp\pip-record-rnn2
n_xd\install-record.txt' --single-version-externally-managed --compile --inst
all-headers 'C:\Users\venka\anaconda3\Include\netifaces'
         cwd: C:\Users\venka\AppData\Local\Temp\pip-install-z3uyco1q\netiface
s\
    Complete output (5 lines):
    running install
    running build
    running build_ext
    building 'netifaces' extension
    error: Microsoft Visual C++ 14.0 is required. Get it with "Build Tools fo
r Visual Studio": https://visualstudio.microsoft.com/downloads/
    ----------------------------------------
ERROR: Command errored out with exit status 1: 'C:\Users\venka\anaconda3\pyth
on.exe' -u -c 'import sys, setuptools, tokenize; sys.argv[0] = '"'"'C:\\Users
\\venka\\AppData\\Local\\Temp\\pip-install-z3uyco1q\\netifaces\\setup.p
y'"'"'; __file__='"'"'C:\\Users\\venka\\AppData\\Local\\Temp\\pip-install-z3u
yco1q\\netifaces\\setup.py'"'"';f=getattr(tokenize, '"'"'open'"'"', open)(__f
ile__);code=f.read().replace('"'"'\r\n'"'"', '"'"'\n'"'"');f.close();exec(com
pile(code, __file__, '"'"'exec'"'"'))' install --record 'C:\Users\venka\AppDa
ta\Local\Temp\pip-record-rnn2n_xd\install-record.txt' --single-version-extern
ally-managed --compile --install-headers 'C:\Users\venka\anaconda3\Include\ne
tifaces' Check the logs for full command output.
```