In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import cv2
```

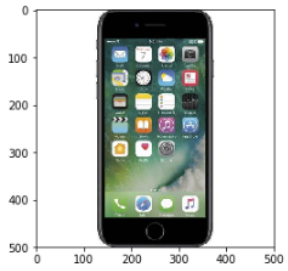
In [2]:
```
from keras.applications import inception_v3
```

In [3]:
```
model = inception_v3.InceptionV3(weights='imagenet')

model.summary()
```

```
Downloading data from https://github.com/fchollet/deep-learning-models/releases/download/v0.5/inception_v3_weights_tf_dim_ordering_tf_kernels.h5
96116736/96112376 [==============================] - 69s 1us/step
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
input_1 (InputLayer)             (None, None, None, 3 0
____________________________________________________________________________________________________
conv2d_1 (Conv2D)                (None, None, None, 3 864         input_1[0][0]
____________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, None, None, 3 96          conv2d_1[0][0]
____________________________________________________________________________________________________
activation_1 (Activation)        (None, None, None, 3 0           batch_normalization_1[0][0]
____________________________________________________________________________________________________
conv2d_2 (Conv2D)                (None, None, None, 3 9216        activation_1[0][0]
____________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, None, None, 3 96          conv2d_2[0][0]
____________________________________________________________________________________________________
activation_2 (Activation)        (None, None, None, 3 0           batch_normalization_2[0][0]
```

In [5]:
```
from PIL import Image
image=Image.open("C:/Users/kavindu/Pictures/iphone.jpg")
plt.imshow(image)
```

Out[5]:
```
<matplotlib.image.AxesImage at 0x1f71ed0e390>
```



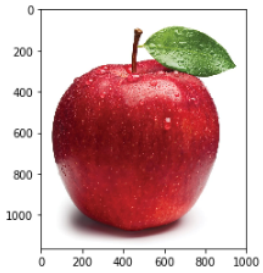
In [7]:
```
from keras import preprocessing

img = preprocessing.image.load_img("C:/Users/kavindu/Pictures/iphone.jpg", target_size=(299, 299))
x = preprocessing.image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = inception_v3.preprocess_input(x)
predictions = model.predict(x)
labels = inception_v3.decode_predictions(predictions, top=3)[0]
```

In [8]:
```
labels
```

Out[8]:
```
[('n03584254', 'iPod', 0.49112463),
 ('n02992529', 'cellular_telephone', 0.40493697),
 ('n03485407', 'hand-held_computer', 0.05292658)]
```

In [12]:
```
image=Image.open("C:/Users/kavindu/Pictures/apple.jpg")
plt.imshow(image)
```

Out[12]:
```
<matplotlib.image.AxesImage at 0x1f7189b7278>
```



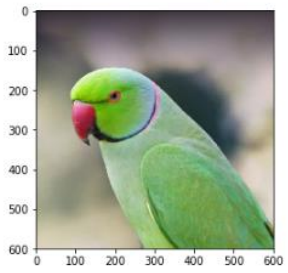
In [9]:
```
img = preprocessing.image.load_img("C:/Users/kavindu/Pictures/apple.jpg", target_size=(299, 299))
y = preprocessing.image.img_to_array(img)
y = np.expand_dims(y, axis=0)
y = inception_v3.preprocess_input(y)
predictions = model.predict(y)
labels = inception_v3.decode_predictions(predictions, top=3)[0]
```

In [10]:
```
labels
```

Out[10]:
```
[('n07742313', 'Granny_Smith', 0.42690778),
 ('n07747607', 'orange', 0.1823984),
 ('n07749582', 'lemon', 0.024204673)]
```

In [13]:
```
image=Image.open("C:/Users/kavindu/Pictures/parrot.jpg")
plt.imshow(image)
```

Out[13]:
```
<matplotlib.image.AxesImage at 0x1f718a00978>
```



In [14]:
```
img = preprocessing.image.load_img("C:/Users/kavindu/Pictures/parrot.jpg", target_size=(299, 299))
p = preprocessing.image.img_to_array(img)
p = np.expand_dims(p, axis=0)
p = inception_v3.preprocess_input(p)
predictions = model.predict(p)
labels = inception_v3.decode_predictions(predictions, top=5)[0]
labels
```

Out[14]:
```
[('n01820546', 'lorikeet', 0.3065806),
 ('n01818515', 'macaw', 0.18955562),
 ('n01843383', 'toucan', 0.029432733),
 ('n01828970', 'bee_eater', 0.021090003),
 ('n04116512', 'rubber_eraser', 0.0050814236)]
```

In [16]:

```
import time
# get the reference to the webcam
camera = cv2.VideoCapture(0)
camera_height = 500

while(True):
    # read a new frame
    _, frame = camera.read()

    # flip the frameq
    frame = cv2.flip(frame, 1)

    # rescaling camera output
    aspect = frame.shape[1] / float(frame.shape[0])
    res = int(aspect * camera_height) # landscape orientation - wide image
    frame = cv2.resize(frame, (res, camera_height))

    # add rectangle
    cv2.rectangle(frame, (300, 75), (650, 425), (240, 100, 0), 2)

    # get ROI(Region of Interest)
    roi = frame[75+2:425-2, 300+2:650-2]

    # parse BRG to RGB
    roi = cv2.cvtColor(roi, cv2.COLOR_BGR2RGB)

    # resize to 224*224
    roi = cv2.resize(roi, (399, 399))
    roi = inception_v3.preprocess_input(roi)

    # predict
    roi2 = np.array([cv2.cvtColor(roi, cv2.COLOR_BGR2RGB)])
```

```
    predictions = model.predict(roi2)

    labels = inception_v3.decode_predictions(predictions, top=3)[0]

    # add text
    label_1 = '{} - {}%'.format(labels[0][1], int(labels[0][2]*100))
    cv2.putText(frame, label_1, (70, 170),
                cv2.FONT_HERSHEY_SIMPLEX, 0.9, (20, 240, 150), 2)

    # add text
    label_2 = '{} - {}%'.format(labels[1][1], int(labels[1][2]*100))
    cv2.putText(frame, label_2, (70, 200),
                cv2.FONT_HERSHEY_SIMPLEX, 0.9, (20, 240, 240), 2)

    # add text
    label_3 = '{} - {}%'.format(labels[2][1], int(labels[2][2]*100))
    cv2.putText(frame, label_3, (70, 230),
                cv2.FONT_HERSHEY_SIMPLEX, 0.9, (20, 20, 240), 2)

    # show the frame
    cv2.imshow("Real Time object detection", frame)

    key = cv2.waitKey(1)

    # quit camera if 'q' key is pressed
    if key & 0xFF == ord("q"):
        break

camera.release()
cv2.destroyAllWindows()
```