# 人脸检测实践

## 实验说明

上节课中我们讲授了人脸检测功能的输入和输出分别是什么，在这节课上，我们就要亲自动手实践这一过程，学会应用我们提供的人脸检测接口，加深对人脸检测这一功能的理解。

我们将常用的人脸相关的功能都封装到了FaceAPI这个类中，其中就包含以下这个接口：

- detect(img) : 人脸检测功能，返回Rect的列表，每个Rect代表一个被检测出的人脸区域

上面提到了几个结构体，结构体信息如下所示 :

- Point : 属性有x、y，代表点的坐标
- Rect : 属性有p1、p2,代表左上角点和右下角点

在本节课上，我们会用到detect接口，来体会人脸检测的功能。

为了辅助大家形象地感受到图像和人脸框的信息，我们提供了show(img)和showWithRects(img,rects)两个接口， 第一个接口可以可视化输入的img，第二个接口将rects包含的矩形框放在img上后再可视化出来。

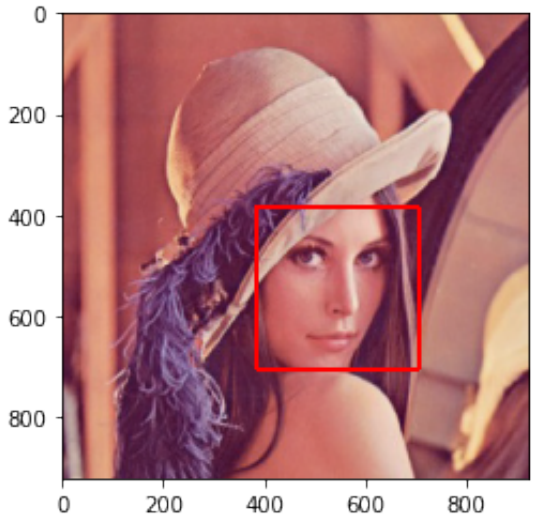
In [2]:
```
from API.FaceAPI_dlib import *
faceAPI = FaceAPI()

img = cv2.imread('data/0.png')
show(img)
rects = faceAPI.detect(img)
print('人脸框: ', rects)
```

```
<Figure size 640x480 with 1 Axes>
```

```
人脸框:  [Rect(384,384,705,705)]
```

接下来将检测到的人脸框可视化到原图上

In [3]:
```
showWithRects(img,rects)
```




# 找出最大的人脸

一般来说，拍照的背景可能有好多不相关的人脸出现，而我们要识别的主体应该是画面中人脸最大的那一个，同学们能否通过循环语句和条件语句来找到当前场景中(面积)最大的人脸并显示出来呢?

In [16]:
```
img = cv2.imread('data/group.png')
rects = faceAPI.detect(img)

area = -1
maxRect = None
for rect in rects:
    area_now = (rect.p2.x-rect.p1.x)*(rect.p2.y-rect.p1.y)
    if area_now > area:
        area = area_now
        maxRect = rect
showWithRects(img,[maxRect])
```

## 中心点坐标

在上面的步骤我们已经能成功应用人脸检测接口，并将其可视化到图片上，同学们可以根据人脸检测接口返回的结果计算出每个人脸框的中心坐标嘛?

## 体验人脸检测

同学们可以自主上传一些图片，在这些图片上进行人脸检测。

## 思考延伸

经过刚才的体验，同学们是否对人脸检测这一功能有了比较直观的认识呢？大家可以思考一下这个功能可以在生活中的哪方面进行应用。