

You Are Beautiful

{Python Project}

Team : URBeautiful

Members : Tiffany Yen, Luhan Shen, Yueping Li, Xiaoyue Shen





You Are Beautiful is a project that combines:

- Applicability in Real Life
- Research into deeper python functions

We want to create a simple, beauty camera application that is easy to use

Abstract

Why We Choose This Topic

According to Zhicha Big Data, Beauty & Filter Camera users have exceeded **100 million** people since 2017

Rare to see an unedited picture

Decorated pictures are a MUST now!

People are always taking pictures

- Vacation
- Coffee Dates
- Casual Outing

Our group is all girls

What do we all love?

Picture

Picture

Picture

Introduction to Libraries

Face_recognition

Recognize and
manipulate the
faces

99.38% Accuracy

PIL

Python Imaging
Library = displays,
process, and
archives images

OpenCV

Strong focus in
real-time
application

It's FREE

Face and Facial Feature Recognition

- ✓ Prepare the library

```
import face_recognition
```

- ✓ Load the file into a numpy array

```
image = face_recognition.load_image_file("picture.jpg")
```

- ✓ Get locations and outlines of facial features. Store the values into a dictionary

```
face_landmarks_list = face_recognition.face_landmarks(image)
```

Makeup (Still Image)

- Prepare the Library using PIL modules
- Use for-loop to beautify faces
- Use polygon function and line function to draw the eyebrows, gloss the lips, sparkle the eyes and apply the eyeliners.
- Display and save image



Makeup (Still Image)

CODE

```
In [ ]: from PIL import Image, ImageDraw
for face_landmarks in face_landmarks_list:
    pil_image = Image.fromarray(image)
    d = ImageDraw.Draw(pil_image, 'RGBA')
    d.polygon(face_landmarks['left_eyebrow'], fill=(68, 54, 39, 128))
    d.polygon(face_landmarks['right_eyebrow'], fill=(68, 54, 39, 128))
    d.line(face_landmarks['left_eyebrow'], fill=(68, 54, 39, 150), width=5)
    d.line(face_landmarks['right_eyebrow'], fill=(68, 54, 39, 150), width=5)
    d.polygon(face_landmarks['top_lip'], fill=(150, 0, 0, 128))
    d.polygon(face_landmarks['bottom_lip'], fill=(150, 0, 0, 128))
    d.line(face_landmarks['top_lip'], fill=(150, 0, 0, 64), width=8)
    d.line(face_landmarks['bottom_lip'], fill=(150, 0, 0, 64), width=8)
    d.polygon(face_landmarks['left_eye'], fill=(255, 255, 255, 30))
    d.polygon(face_landmarks['right_eye'], fill=(255, 255, 255, 30))
    d.line(face_landmarks['left_eye'] + [face_landmarks['left_eye'][0]], fill=(0, 0, 0, 110), width=6)
    d.line(face_landmarks['right_eye'] + [face_landmarks['right_eye'][0]], fill=(0, 0, 0, 110), width=6)
pil_image.show()
pil_image.save('pictures/beautiful.jpg')
```

Makeup (examples)



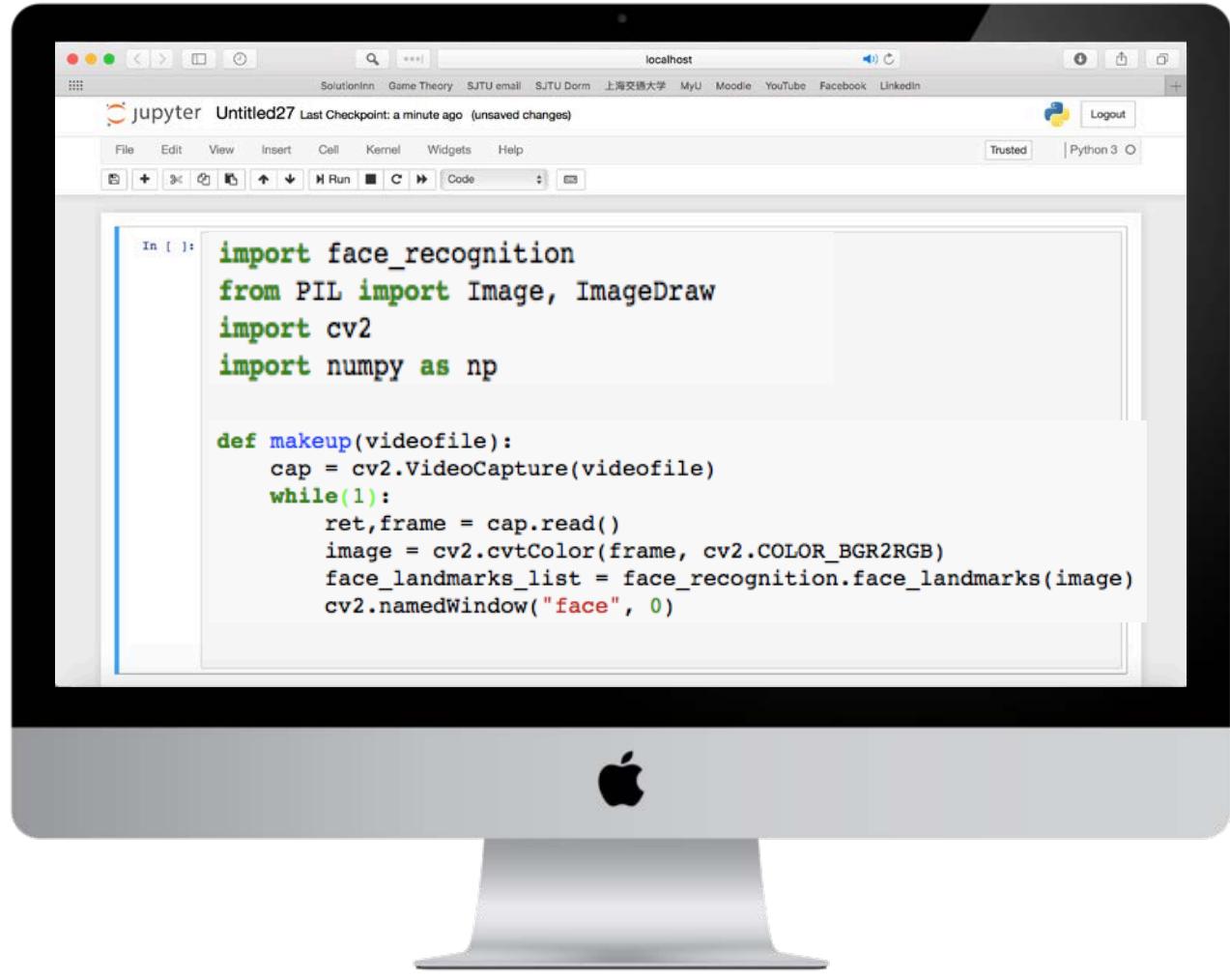
Makeup (Moving Image)

- Prepare the Libraries
- Split and read the video using while-loop
- Apply makeup (same as still image)
- Present the outcome in video format
- Execute the functions defined



Makeup (Moving Image)

CODE



The image shows a silver iMac monitor displaying a Jupyter Notebook interface. The notebook has a title bar "jupyter Untitled27 Last Checkpoint: a minute ago (unsaved changes)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar below the menu bar includes icons for New, Open, Save, Run, Cell, Code, and Help. The code cell contains the following Python code:

```
In [ ]:
import face_recognition
from PIL import Image, ImageDraw
import cv2
import numpy as np

def makeup(videofile):
    cap = cv2.VideoCapture(videofile)
    while(1):
        ret,frame = cap.read()
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        face_landmarks_list = face_recognition.face_landmarks(image)
        cv2.namedWindow("face", 0)
```

Makeup (Moving Image)

CODE

```
pil_image = Image.fromarray(image)
d = ImageDraw.Draw(pil_image, 'RGBA')
for face_landmarks in face_landmarks_list:
    d.polygon(face_landmarks['left_eyebrow'], fill=(68, 54, 39, 180))
    d.polygon(face_landmarks['right_eyebrow'], fill=(68, 54, 39, 180))
    d.line(face_landmarks['left_eyebrow'], fill=(68, 54, 39, 150), width=3)
    d.line(face_landmarks['right_eyebrow'], fill=(68, 54, 39, 150), width=3)

    d.polygon(face_landmarks['top_lip'], fill=(150, 0, 0, 150))
    d.polygon(face_landmarks['bottom_lip'], fill=(150, 0, 0, 150))
    d.line(face_landmarks['top_lip'], fill=(150, 0, 0, 80), width=5)
    d.line(face_landmarks['bottom_lip'], fill=(150, 0, 0, 80), width=5)

    d.polygon(face_landmarks['left_eye'], fill=(255, 255, 255, 10))
    d.polygon(face_landmarks['right_eye'], fill=(255, 255, 255, 10))

    d.line(face_landmarks['left_eye'] + [face_landmarks['left_eye'][0]], fill=(0, 0, 0, 80), width=4)
    d.line(face_landmarks['right_eye'] + [face_landmarks['right_eye'][0]], fill=(0, 0, 0, 80), width=4)

img = cv2.cvtColor(np.asarray(pil_image), cv2.COLOR_RGB2BGR)
cv2.imshow("face", img)
cv2.waitKey(1)
cap.release()
```

Makeup (examples)

```
makeup('video.mp4')
```



Makeup (examples)

```
makeup( 'video.mp4' )
```



Stickers

Stickers will make the images more interesting and cute. Two types of sticker function that we have included are:

- Slogan
- Filter (similar to Snapchat's stickers)



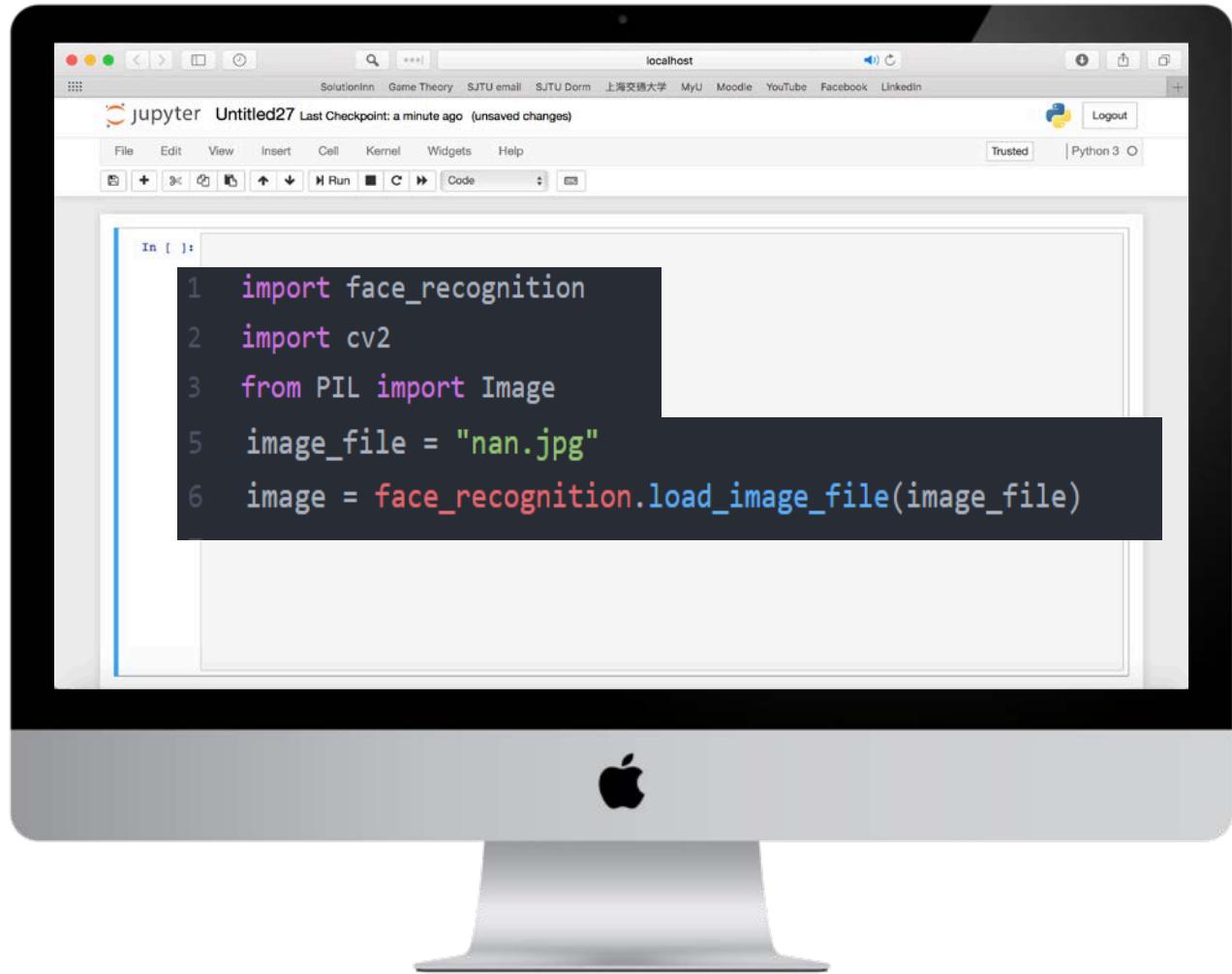
How Slogan Stickers are Made

- Prepare the Library
- Find face locations in the picture
- Read, open , and detect the size of the picture
- Resize the slogan to fit the background image
- Past the stickers onto the picture



1.Prepare the library (face_recognition&PIL)

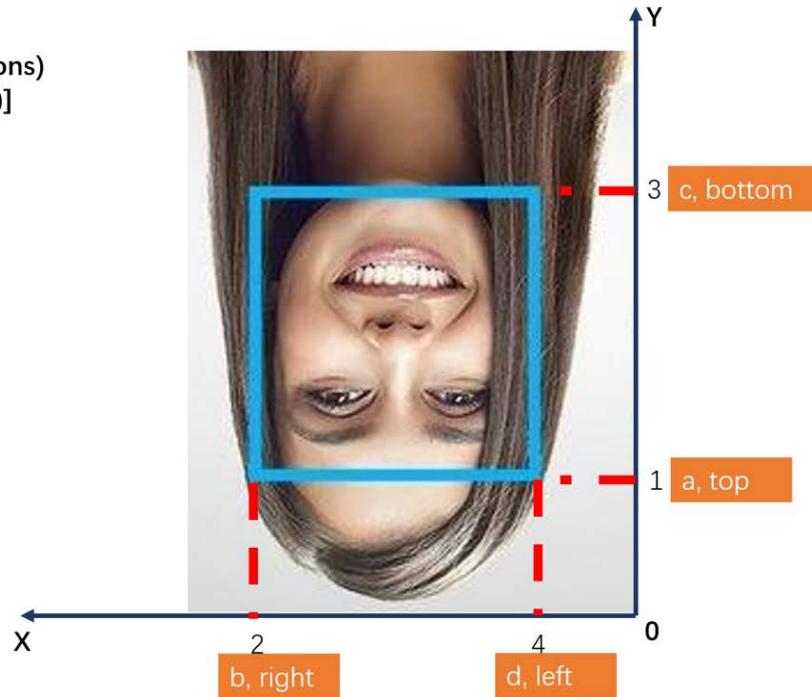
2. open the image with the face



3. Get the location of the face

```
print(face_locations)  
Output: [(1,2,3,4)]
```

top: a=1
right: b=2
bottom: c=3
left: d=4



```
9  face_locations = face_recognition.face_locations(image)  
10 print('face_locations:',face_locations)  
11  
12 a=int(face_locations[0][0])  
13 b=int(face_locations[0][1])  
14 c=int(face_locations[0][2])  
15 d=int(face_locations[0][3])
```

4. Resize the sticker

```
resizedIm= im2.resize(((c-a)//2),(c-a)))
```

```
print(face_locations)  
Output: [(1,2,3,4)]
```

```
top: a=1  
right: b=2  
bottom: c=3  
left: d=4
```

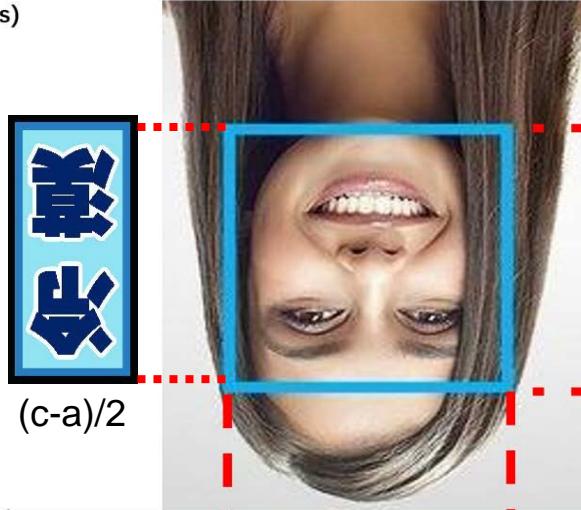
(c-a)

$(c-a)/2$

x



2



d, left

1

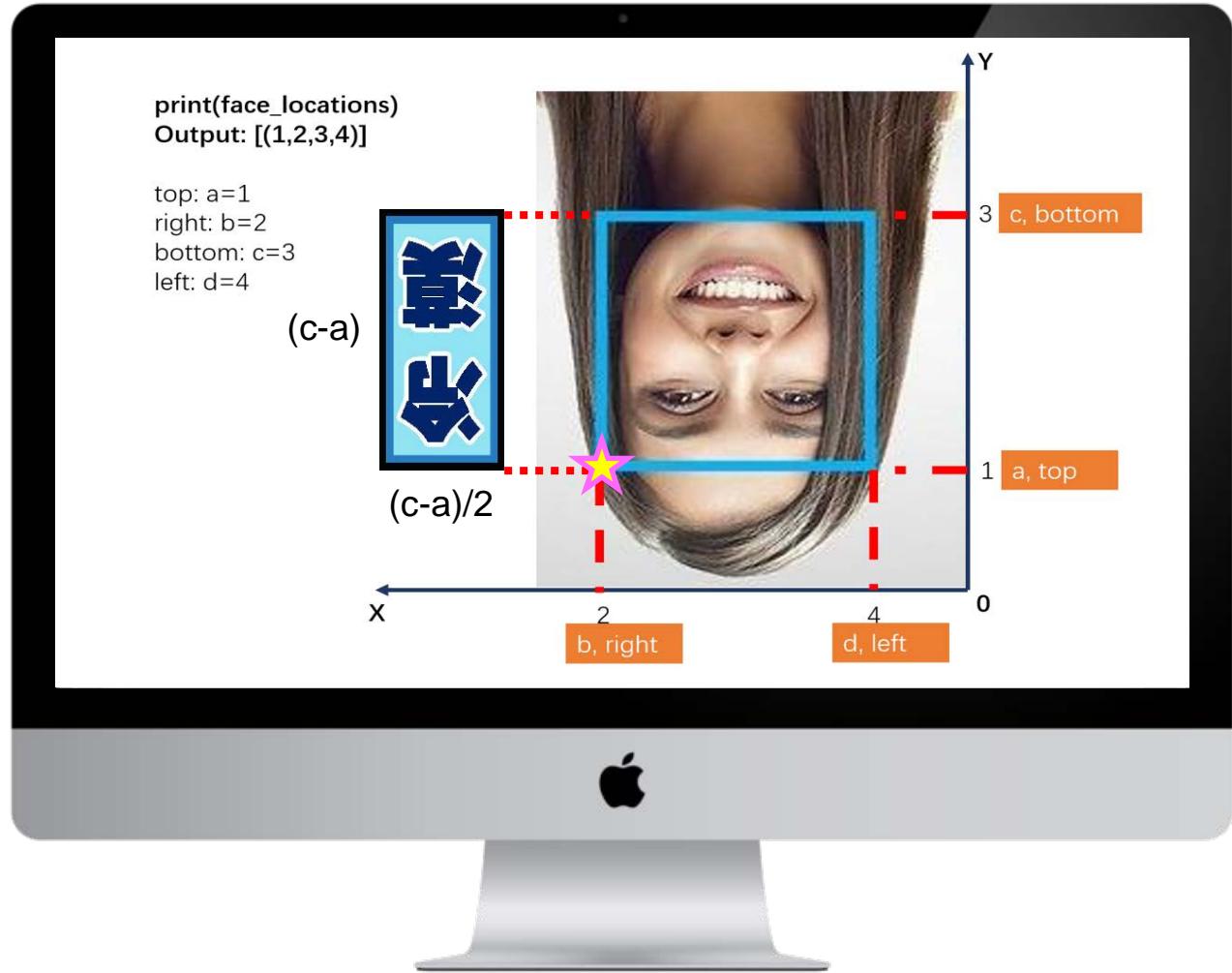
c, bottom

a, top

0



5. Past stickers onto the background and show the result



Finally, we get
the result!



图片来源：拍信 Paixin.com

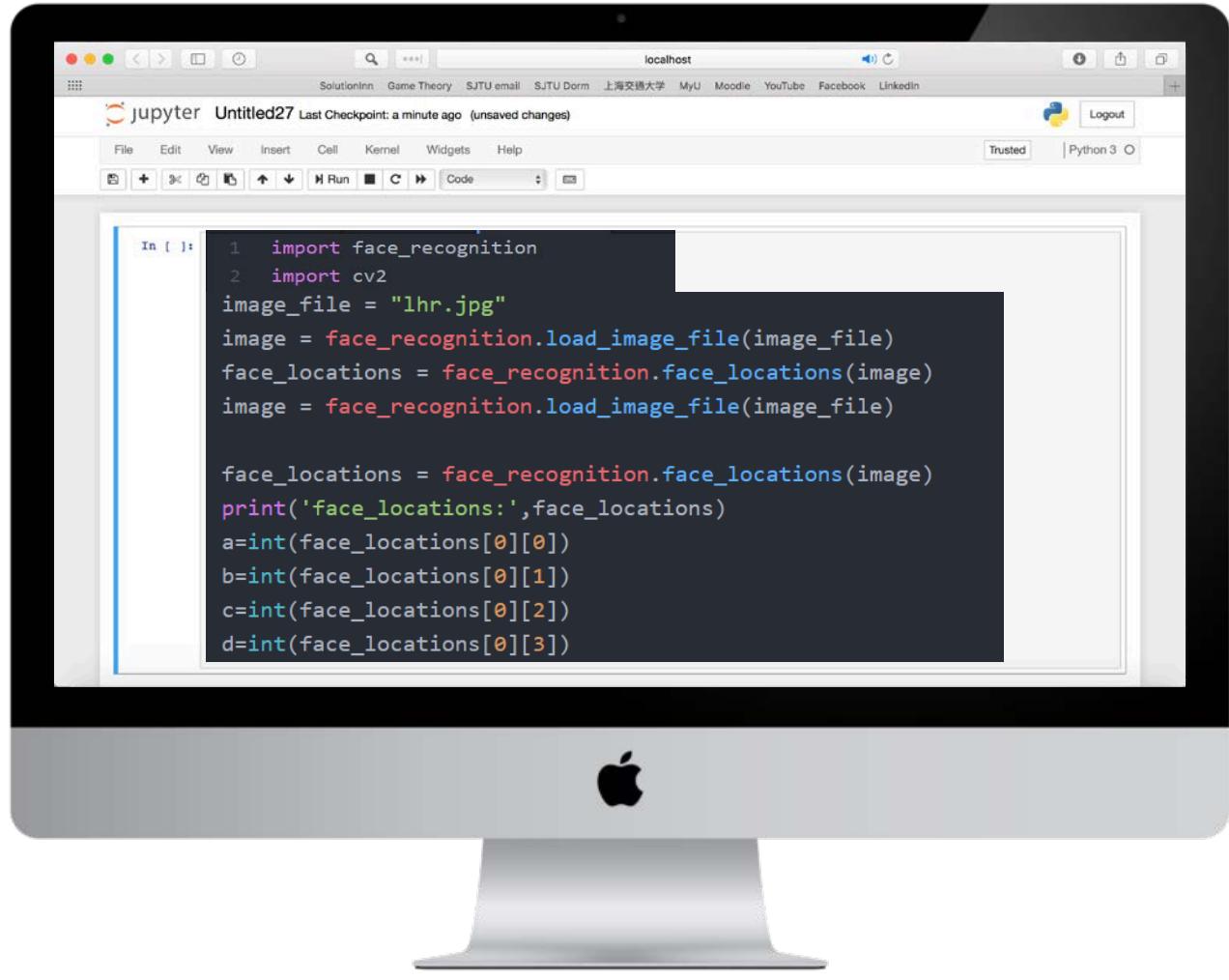
How Filter Stickers are Made

- Prepare the library
- Recognize the face location
- Resize the sticker
- Reformat the sticker and paste it
- Combine the picture and save/show result



1. Prepare the library.

2. Recognize the face location

A photograph of a silver Apple iMac monitor. On the screen is a Jupyter Notebook window titled "Untitled27". The notebook shows Python code for face recognition. The code imports the "face_recognition" and "cv2" libraries, loads an image file named "lhr.jpg", and uses the "face_recognition" module to find face locations. It then prints the locations and extracts four integer values (a, b, c, d) representing the bounding box of the first face.

```
In [ ]:
1 import face_recognition
2 import cv2
image_file = "lhr.jpg"
image = face_recognition.load_image_file(image_file)
face_locations = face_recognition.face_locations(image)
image = face_recognition.load_image_file(image_file)

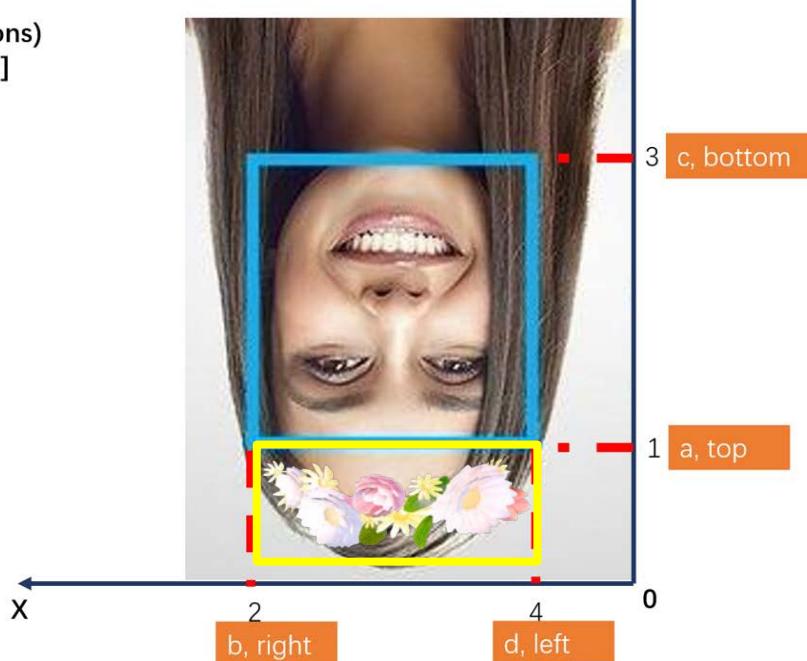
face_locations = face_recognition.face_locations(image)
print('face_locations:',face_locations)
a=int(face_locations[0][0])
b=int(face_locations[0][1])
c=int(face_locations[0][2])
d=int(face_locations[0][3])
```

3. Resize the sticker

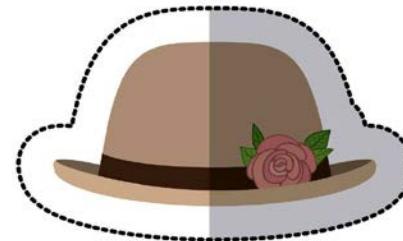
```
im1_path = r'C:\Users\Lucia\Documents\交大\Python\week11-code\f.png'  
im2_path = r'C:\Users\Lucia\Documents\交大\Python\week11-code\lhr2.jpg'  
im1 = Image.open(im1_path)  
im2 = Image.open(im2_path)  
if (abs(b-d)//2)>a:  
    resizedIm = im1.resize((abs(b-d),a )) #防止边界过小  
else:  
    resizedIm= im1.resize((abs(b-d),(abs(b-d)//2))) #让贴纸的宽度和脸的宽度一致, 贴纸的高度为其宽度的1/2  
resizedIm.save(r'C:\Users\Lucia\Documents\交大\Python\week11-code\changeflower.png') #把改变过大小的贴纸保存
```

```
print(face_locations)  
Output: [(1,2,3,4)]
```

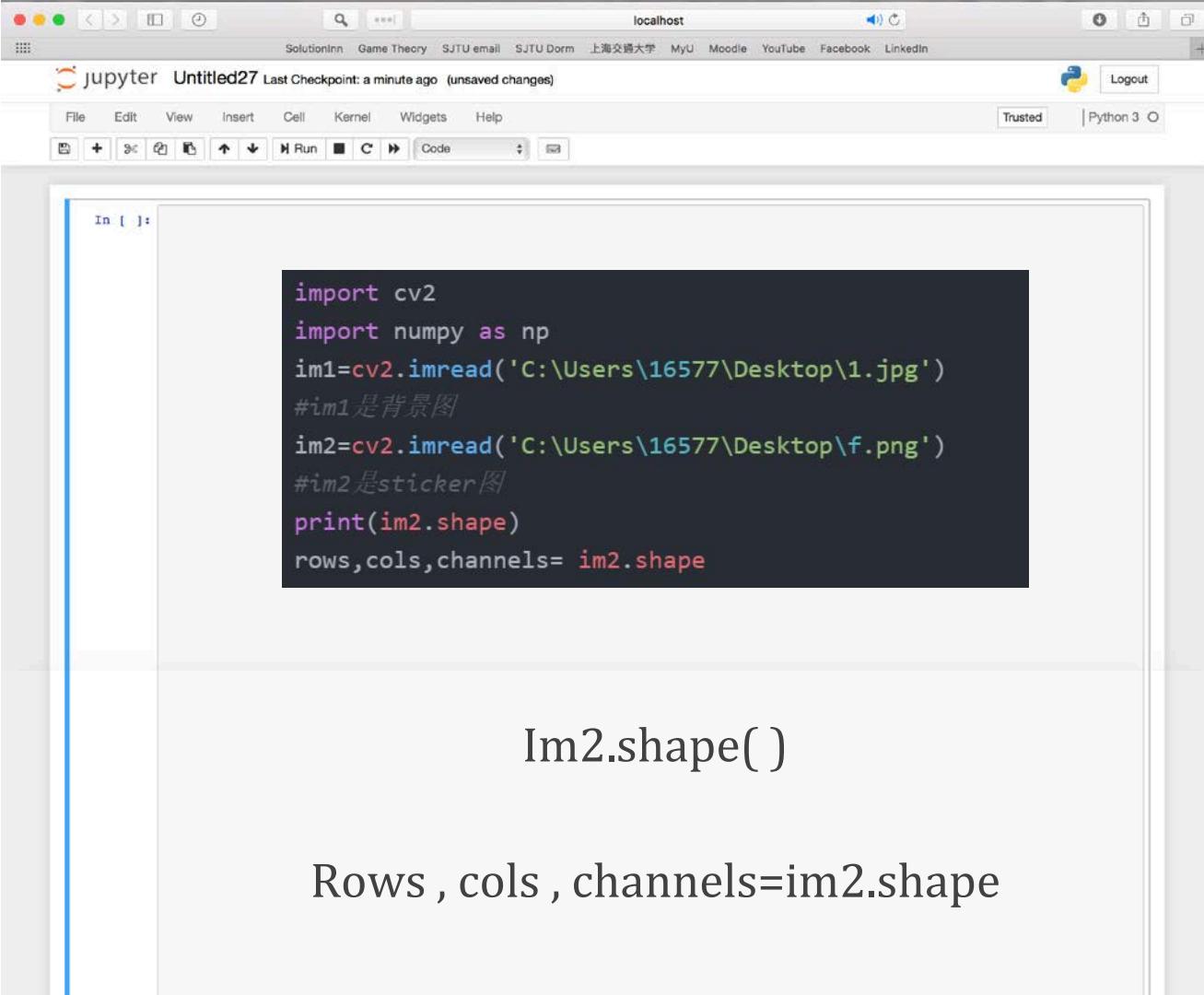
top: a=1
right: b=2
bottom: c=3
left: d=4



4. Sticker Matting and Pasting



Step 1: Preparation



The screenshot shows a Jupyter Notebook interface running on localhost. The title bar indicates it's a Jupyter notebook titled "Untitled27" with a last checkpoint a minute ago. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar below has icons for New, Run, Cell, Code, and Cell Type. The status bar shows "Trusted" and "Python 3". The main area shows a code cell labeled "In []:" containing the following Python code:

```
import cv2
import numpy as np
im1=cv2.imread('C:\Users\16577\Desktop\1.jpg')
#im1是背景图
im2=cv2.imread('C:\Users\16577\Desktop\f.png')
#im2是sticker图
print(im2.shape)
rows,cols,channels= im2.shape
```

Below the code cell, the output is shown as "Im2.shape()".

Rows , cols , channels=im2.shape

Step 1: Preparation

SolutionInn Game Theory SJTU email SJTU Dorm 上海交通大学 MyU Moodle YouTube Facebook LinkedIn

jupyter Untitled27 Last Checkpoint: a minute ago (unsaved changes) Logout Trusted Python 3 O

In []:

```
roi=im1[(a-rows):a,d:(d+cols)]
```



[(a-rows):a , d:(d+cols)]

Step 2: Change Sticker to Gray Scale image:

jupyter Untitled27 Last Checkpoint: a minute ago (unsaved changes)

In []:

```
gray=cv2.cvtColor(im2,cv2.COLOR_BGR2GRAY)
```

灰度图：Gray Scale Image
把白色与黑色之间按对数关系分为若干等级，
称为灰度，灰度分为256阶。

CV2.cvtColor
(im2, cv2.COLOR_BRG2GRAY)



localhost

SolutionInn Game Theory SJTU email SJTU Dorm 上海交通大学 MyU Moodle YouTube Facebook LinkedIn

Logout

Trusted Python 3

Step 3: Fixed threshold binarization



Binarization二值化：
Cv2.THRESH_BINARY

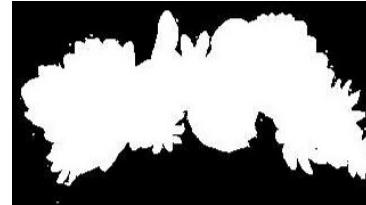
将图像上的像素点的灰度值设置为0或255，
整个图像呈现只有黑和白的视觉效果。

Threshold阈值：
CV2.Threshold (Gray, 50,255)
寻找图像二值化阈值 (50,255) ，
根据阈值将图像分为Foreground (White) ,或者Background (Black) 。

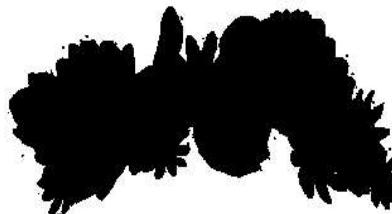
Step 3: Fixed threshold binarization



```
_ , mask=cv2.threshold(gray,50,255,cv2.THRESH_BINARY)  
mask_inv=cv2.bitwise_not(mask)
```

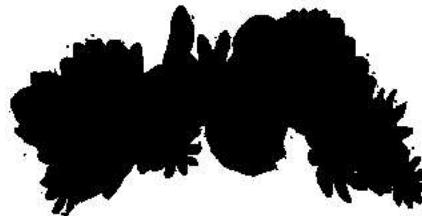


```
mask_inv=cv2.bitwise_not(mask)
```



Step 4: Combine Pictures (1)

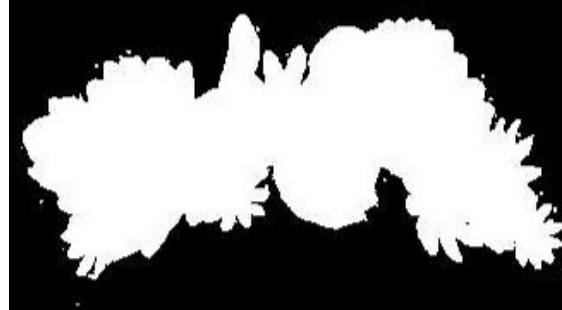
```
img1_bg=cv2.bitwise_and(roi,roi,mask=mask_inv)
```



Cv2.bitwise_and()
Mask=mask_inv

Step 4: Combine Pictures (2)

```
img2_fg=cv2.bitwise_and(im2,im2,mask=mask)
```



Step 4: Combine Pictures (3)



Combined effects



Weaknesses in Our Project

Lack of Accuracy

The position of makeup applied was not always accurate

Sometimes sticker gets placed at awkward position



Perfect Outcome not Guaranteed

Project was created using simpler python modules

Can't compare with Meitu and B612 app

Conclusion



The project a good learning experience to:

- ✓ Apply what we learned in class to real use
- ✓ Cross our comfort zone to learn more
- ✓ Explore image processing power in python
- ✓ Feel proud with our own written program

Thank You