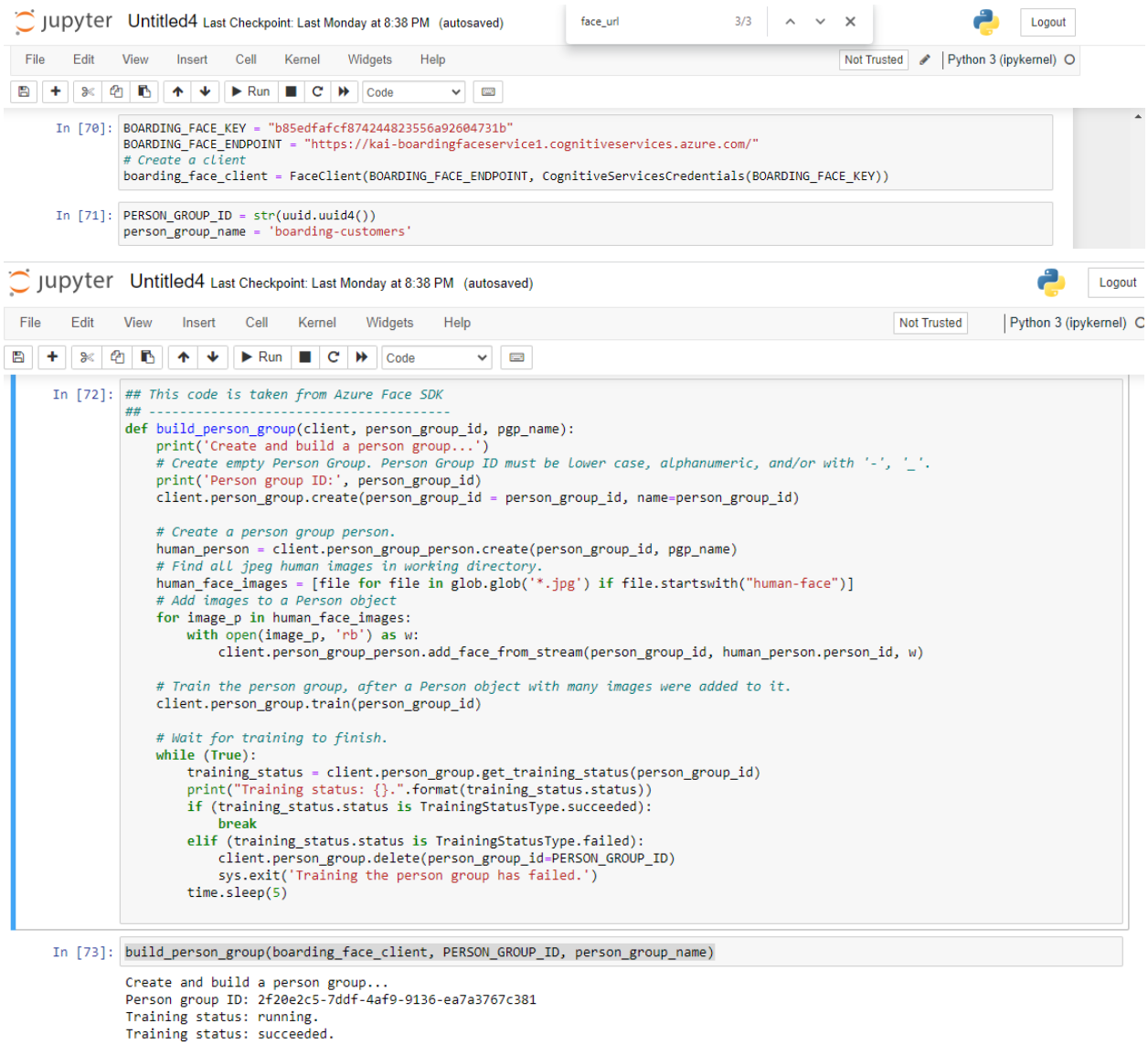
Code snippet:

```
BOARDING_FACE_KEY = "b85edfafcf874244823556a92604731b"
BOARDING_FACE_ENDPOINT =
"https://kai-boardingfaceservice1.cognitiveservices.azure.com/"
# Create a client
boarding_face_client = FaceClient(BOARDING_FACE_ENDPOINT,
CognitiveServicesCredentials(BOARDING_FACE_KEY))
PERSON_GROUP_ID = str(uuid.uuid4())
person_group_name = 'boarding-customers'

## This code is taken from Azure Face SDK
## ---------------------------------------
def build_person_group(client, person_group_id, pgp_name):
    print('Create and build a person group...')
    # Create empty Person Group. Person Group ID must be lower case,
alphanumeric, and/or with '-', '_'.
    print('Person group ID:', person_group_id)
    client.person_group.create(person_group_id = person_group_id,
name=person_group_id)

    # Create a person group person.
    human_person = client.person_group_person.create(person_group_id,
pgp_name)
    # Find all jpeg human images in working directory.
    human_face_images = [file for file in glob.glob('*.jpg') if
file.startswith("human-face")]
    # Add images to a Person object
    for image_p in human_face_images:
        with open(image_p, 'rb') as w:

client.person_group_person.add_face_from_stream(person_group_id,
human_person.person_id, w)

    # Train the person group, after a Person object with many images were
added to it.
    client.person_group.train(person_group_id)

    # Wait for training to finish.
    while (True):
```

```
        training_status =
client.person_group.get_training_status(person_group_id)
        print("Training status: {}.".format(training_status.status))
        if (training_status.status is TrainingStatusType.succeeded):
            break
        elif (training_status.status is TrainingStatusType.failed):
            client.person_group.delete(person_group_id=PERSON_GROUP_ID)
            sys.exit('Training the person group has failed.')
        time.sleep(5)

build_person_group(boarding_face_client, PERSON_GROUP_ID,
person_group_name)

'''
Detect all faces in query image list, then add their face IDs to a new
list.
'''
def detect_faces(client, query_images_list):
    print('Detecting faces in query images list...')

    face_ids = {} # Keep track of the image ID and the related image in a
dictionary
    for image_name in query_images_list:
        image = open(image_name, 'rb') # BufferedReader
        print("Opening image: ", image.name)
        time.sleep(5)

        # Detect the faces in the query images list one at a time, returns
list[DetectedFace]
        faces = client.face.detect_with_stream(image)

        # Add all detected face IDs to a list
        for face in faces:
            print('Face ID', face.face_id, 'found in image',
os.path.splitext(image.name)[0]+'.jpg')
            # Add the ID to a dictionary with image name as a key.
            # This assumes there is only one face per image (since you
can't have duplicate keys)
            face_ids[image.name] = face.face_id
```

```
    return face_ids
test_images = [file for file in glob.glob('*.jpg') if
file.startswith("human-face")]
person_model_ids = detect_faces(boarding_face_client, test_images)
    # Verification example for faces of the same person.
verify_result =
boarding_face_client.face.verify_face_to_face(person_model_ids['human-face
1.jpg'], person_model_ids['human-face4.jpg'])
```

Code screen-shot:



jupyter Untitled4 Last Checkpoint: Last Monday at 8:38 PM (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [76]:
```
'''
Detect all faces in query image list, then add their face IDs to a new list.
'''
def detect_faces(client, query_images_list):
    print('Detecting faces in query images list...')

    face_ids = {} # Keep track of the image ID and the related image in a dictionary
    for image_name in query_images_list:
        image = open(image_name, 'rb') # BufferedReader
        print("Opening image: ", image.name)
        time.sleep(5)

        # Detect the faces in the query images list one at a time, returns list[DetectedFace]
        faces = client.face.detect_with_stream(image)

        # Add all detected face IDs to a list
        for face in faces:
            print('Face ID', face.face_id, 'found in image', os.path.splitext(image.name)[0]+'.jpg')
            # Add the ID to a dictionary with image name as a key.
            # This assumes there is only one face per image (since you can't have duplicate keys)
            face_ids[image.name] = face.face_id

    return face_ids
```

In [77]:
```
test_images = [file for file in glob.glob('*.jpg') if file.startswith("human-face")]
```

In [79]:
```
person_model_ids = detect_faces(boarding_face_client, test_images)
```

```
Detecting faces in query images list...
Opening image:  human-face1.jpg
Face ID 167167eb-f697-4ca9-bbfb-386eee298719 found in image human-face1.jpg
Opening image:  human-face2.jpg
Face ID b32e1a74-0161-432b-b17c-245fea4af7b1 found in image human-face2.jpg
Opening image:  human-face3.jpg
Face ID 8c6af24d-2e6d-433b-ba3e-214f812af14d found in image human-face3.jpg
Opening image:  human-face4.jpg
Face ID 85f7ab15-ad65-4ae4-97dd-53b001abf673 found in image human-face4.jpg
Opening image:  human-face5.jpg
Face ID 445b4d3e-0411-4f5f-bd39-2ebbcc20dc6c found in image human-face5.jpg
Opening image:  human-face6.jpg
Face ID f7638f27-454c-492d-af9b-29651d9cce69 found in image human-face6.jpg
```


jupyter Untitled4 Last Checkpoint: Last Monday at 8:38 PM (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [81]:
```
# Verification example for faces of the same person.
verify_result = boarding_face_client.face.verify_face_to_face(person_model_ids['human-face1.jpg'], person_model_ids['human-face4.
```

In [82]:
```
if verify_result.is_identical:
    print("Faces are of the same (Positive) person, similarity confidence: {}.".format(verify_result.confidence))
else:
    print("Faces are of different (Negative) persons, similarity confidence: {}.".format(verify_result.confidence))
```

```
Faces are of the same (Positive) person, similarity confidence: 0.93741.
```