



## PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

### GUIA DE APRENDIZAJE

#### IDENTIFICACIÓN DE LA GUIA DE APRENDIZAJE

- Denominación del Programa de Formación: Análisis y desarrollo de software
- Código del Programa de Formación: 228118
- Nombre del Proyecto: Construcción de software integrador de tecnologías orientadas a servicios.
- Fase del Proyecto: Planeación
- Actividad de Proyecto : Determinar la estructura lógica y tecnológica del sistema.
- Competencias:
- Técnicas:  
220501095 - Diseñar la solución de software de acuerdo con procedimientos y requisitos técnicos.
- Claves:
- 240202501 - Interactuar en lengua inglesa de forma oral y escrita dentro de contextos sociales y laborales según los criterios establecidos por el Marco Europeo de Referencia para las Lenguas.
- Resultados de Aprendizaje Alcanzar:
- 220501095-04 - Verificar los entregables de la fase de diseño del software de acuerdo con lo establecido en el informe de análisis.
- Duración de la Guía: 60 Horas.

## 2. PRESENTACIÓN

El reconocimiento facial es una tecnología y proceso que implica identificar y verificar a individuos basándose en sus rasgos faciales. Es una técnica biométrica que analiza y compara patrones en el rostro de una persona para determinar su identidad. El reconocimiento facial ha ganado una atención significativa y aplicación en varios campos, incluyendo la seguridad, la autenticación, la vigilancia y la interacción humano-computadora. Al finalizar esta guía habrás aprendido cómo funciona generalmente el reconocimiento facial en función de:

**Detección Facial:** El primer paso en el reconocimiento facial es detectar y localizar rostros en una imagen o un flujo de video. Este proceso implica identificar las regiones de una imagen que potencialmente contienen rostros humanos.

**Representación del Rostro:** Las características extraídas se transforman en una representación matemática, a menudo llamada plantilla o embedding facial. Esta representación captura los aspectos distintivos del rostro de manera que se pueda comparar.

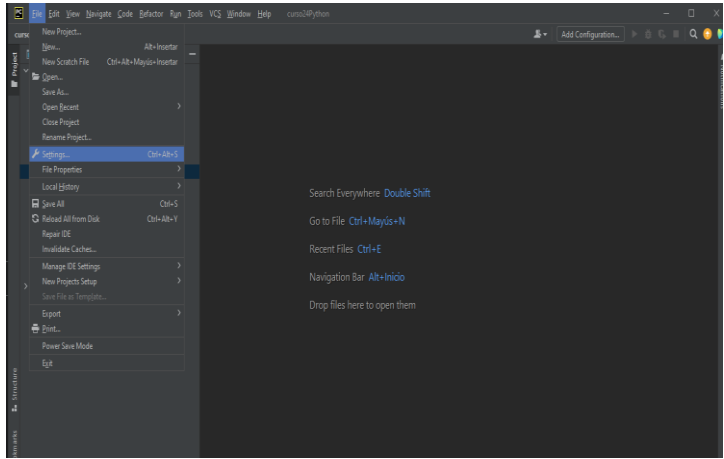
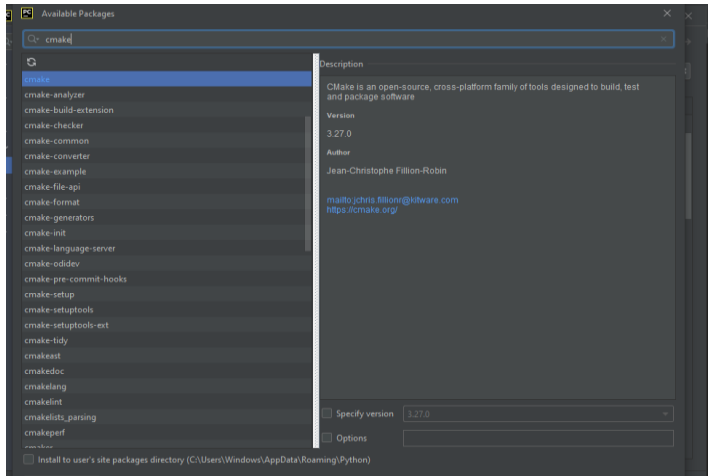
**Comparación y Reconocimiento:** Cuando se necesita identificar o verificar un nuevo rostro, también se extraen sus características y se representan de la misma manera que las plantillas almacenadas. Luego, se realiza una comparación entre la nueva representación facial y las plantillas almacenadas. Si la similitud o distancia calculada entre las representaciones está por debajo de un umbral determinado, se considera una coincidencia y se reconoce al individuo.



Autenticación o Identificación: Dependiendo de la aplicación, el rostro reconocido puede usarse para autenticar (confirmar la identidad de una persona) o identificar (determinar quién es la persona a partir de una base de datos de individuos conocidos). Y lo más importante que lo puedas aplicar en tus proyectos tanto a nivel educativo como a nivel laboral.

### 3. FORMULACIÓN DE LAS ACTIVIDADES DE APRENDIZAJE

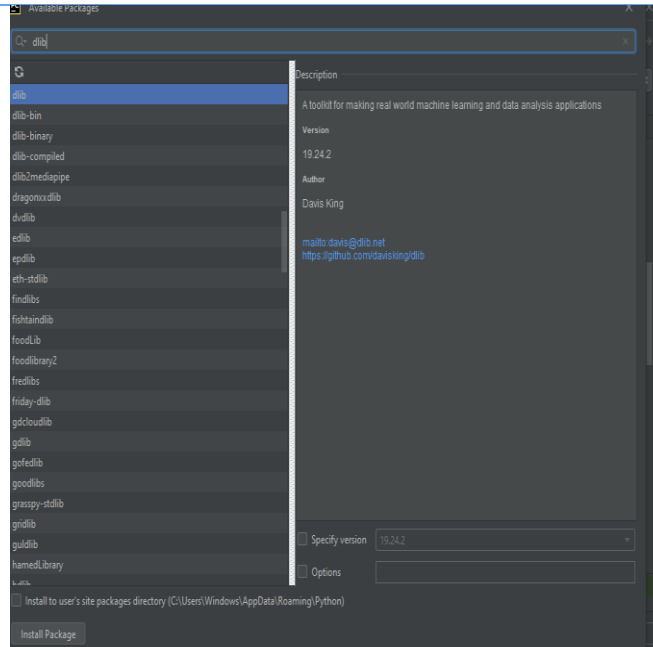
Instalar y configurar las librerías en Pycharm. “Creation of facial recognition checker”.

<p>1. Step one</p>	
<p>2. We install the libraries in pycharm: cmake</p>	



**CMake:** CMake is an open-source, cross-platform build system and configuration management tool. It allows developers to define and manage the build process for their software projects in a platform-independent manner. With CMake, you can create build scripts (CMakeLists.txt files) that describe how to compile, link, and package your code. CMake generates platform-specific build files (such as Makefiles, Visual Studio project files, or Ninja build files) based on the provided CMake configuration, making it easier to maintain and build software across different operating systems and development environments.

### 3. Step install dlib



dlib: is a popular open-source C++ library primarily used for computer vision tasks and machine learning applications. It provides a wide range of tools and algorithms for various tasks such as image processing, object detection, facial recognition, and machine learning model training. dlib is known for its efficiency and high-quality implementations, making it a popular choice among researchers and developers in the computer vision and machine learning fields. In addition to its C++ API, dlib also offers a Python API, which allows developers to use dlib functionalities within Python programming environments. The Python API provides bindings to the underlying C++ library, enabling users to leverage dlib's capabilities using Python syntax.

Some of the key features and functionalities offered by dlib include:

**Face Detection and Recognition:** dlib includes pre-trained models for detecting and recognizing faces in images and videos. Its face recognition capabilities can identify and verify faces based on facial landmarks and features.

**Object Detection:** The library supports object detection using methods like Histogram of Oriented Gradients (HOG) and deep learning-based approaches.

**Image Processing:** dlib provides tools for basic image processing tasks such as resizing, cropping, color manipulation, and geometric transformations.



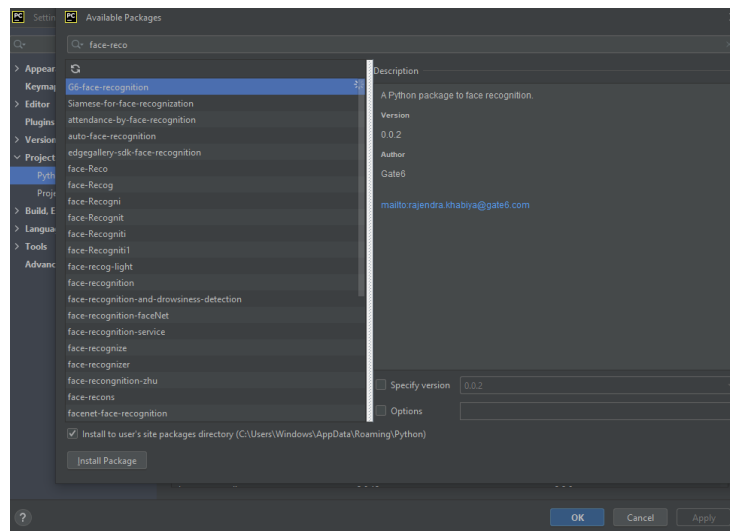
Machine Learning: The library includes various machine learning algorithms, including support vector machines (SVMs), clustering algorithms, and tools for training custom machine learning models.

Shape Analysis: dlib offers tools for shape analysis and manipulation, which can be useful in various applications like medical imaging and computer graphics.

Optimization: The library includes optimization algorithms and tools for solving optimization problems.

Deep Learning: While dlib is not as focused on deep learning as some other libraries like TensorFlow or PyTorch, it does provide some support for neural networks and deep learning-based tasks.

#### 4. Face-recogitio



Face recognition, also known as facial recognition, is a technology and process that involves identifying and verifying individuals based on their facial features. It is a biometric technology that analyzes and compares patterns in a person's face to determine their identity. Face recognition has gained significant attention and application in various fields, including security, authentication, surveillance, and human-computer interaction. Here's how face recognition generally works:

**Face Detection:** The first step in face recognition is to detect and locate faces within an image or a video stream. This process involves identifying the regions of an image that potentially contain human faces. Various algorithms, such as the Histogram of Oriented Gradients (HOG) and deep learning-based convolutional neural networks (CNNs), are used for face detection.

**Feature Extraction:** Once a face is detected, the next step is to extract unique features or characteristics from the face that can be used for identification. These features might include the distances between specific facial landmarks (such as the distance between the eyes, nose, and mouth), the shape of the eyes, nose, and mouth, and other facial attributes.

**Face Representation:** The extracted features are then transformed into a mathematical representation, often referred to as a face template or face embedding. This representation captures the distinguishing aspects of the face in a way that is amenable to comparison.



Comparison and Recognition: When a new face needs to be identified or verified, its features are also extracted and represented in the same manner as the stored templates. A comparison is then made between the new face representation and the stored templates. If the computed similarity or distance between the representations is below a certain threshold, the face is considered a match, and the individual is recognized.

Authentication or Identification: Depending on the application, the recognized face might be used for authentication (confirming the identity of a person) or identification (determining who the person is from a database of known individuals).

Face recognition technology can be applied in various scenarios, including:

Access Control and Security: Unlocking devices, accessing secure areas, and controlling entry based on facial recognition.

Surveillance and Monitoring: Identifying individuals in surveillance footage to enhance security and track suspicious activities.

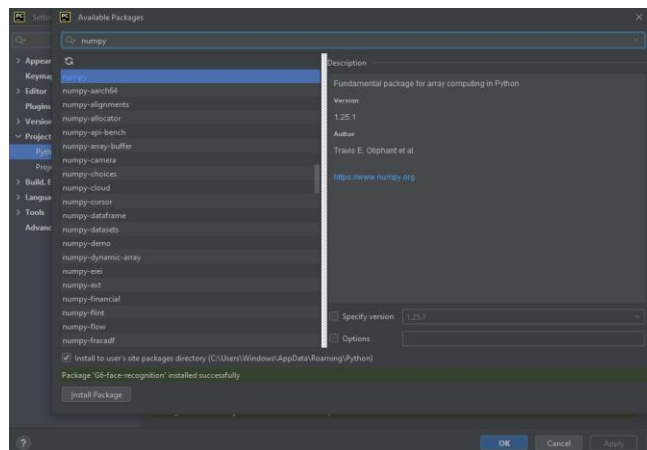
Payment and Transactions: Enabling secure and convenient transactions through facial recognition-based authentication.

Human-Computer Interaction: Creating more natural and intuitive interactions with computers and devices through facial gestures and expressions.

Emotion Analysis: Analyzing facial expressions to determine emotional states and responses.

It's important to note that while face recognition technology offers numerous benefits, it also raises privacy and ethical concerns related to data security, consent, and potential misuse. Regulations and guidelines are evolving to address these concerns and ensure responsible use of facial recognition technology.

## 5. NumPy



NumPy (Numerical Python) is a powerful open-source library for the Python programming language that provides support for large, multi-dimensional arrays and matrices, as well as a wide variety of mathematical functions to operate on these arrays. It is a fundamental package for scientific computing with Python and is widely used in fields such as data analysis, machine learning, image processing, and more.

GFPI-F-135 V01

Key features and components of NumPy include:



Multi-dimensional Arrays: NumPy's main feature is its ndarray (n-dimensional array) object, which allows you to create and manipulate arrays of various dimensions efficiently. These arrays can hold elements of the same data type, enabling vectorized operations.

Array Operations: NumPy provides a wide range of mathematical and logical operations that can be applied element-wise to arrays. This enables you to perform computations without the need for explicit loops, making your code more concise and efficient.

Broadcasting: NumPy's broadcasting rules allow you to perform operations on arrays with different shapes and dimensions, automatically handling the alignment of dimensions as needed.

Mathematical Functions: NumPy includes a comprehensive set of mathematical functions, such as trigonometric, exponential, logarithmic, and statistical functions, which can be applied to arrays directly.

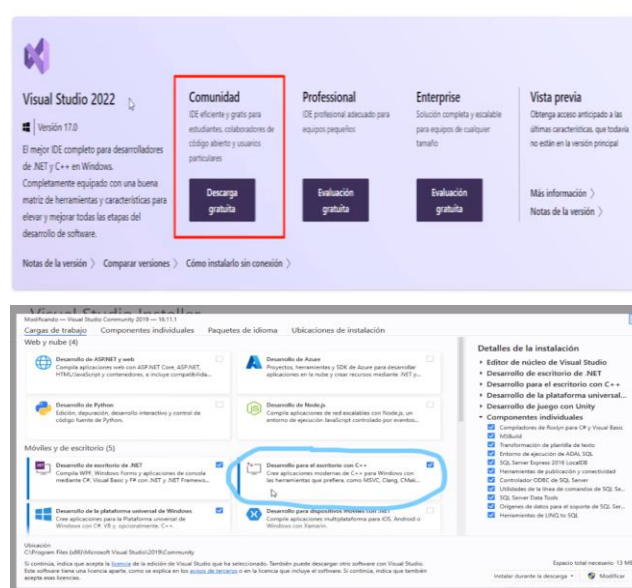
Array Indexing and Slicing: NumPy supports powerful indexing and slicing capabilities that allow you to access and manipulate specific elements or subsets of arrays.

Linear Algebra: NumPy provides functions for performing various linear algebra operations, including matrix multiplication, eigenvale decomposition, singular value decomposition, and more.

Random Number Generation: The numpy.random module offers tools for generating random numbers and random arrays, which are useful for simulations and statistical analysis. Integration with other Libraries: NumPy plays a foundational role in the Python scientific computing ecosystem and is often used in conjunction with other libraries such as SciPy (Scientific Python) for more advanced scientific and engineering computations.

## 6. <https://visualstudio.microsoft.com/es/downloads/>

### 6.1 Community



A developer desktop for C++ on Windows typically refers to a software environment that provides tools and resources for programmers working with the C++ programming language on the Windows operating system. It's designed to facilitate the development,



testing, and debugging of C++ applications and projects. Here are the key components that make up a developer desktop for C++ on Windows:

**Integrated Development Environment (IDE):** An IDE is a software application that provides a comprehensive set of tools for software development. For C++ on Windows, popular choices include Microsoft Visual Studio, Visual Studio Code with appropriate extensions, and JetBrains CLion. These IDEs offer features such as code editing, syntax highlighting, project management, debugging, and more.

**Compiler and Build Tools:** A C++ compiler translates your source code into executable machine code. Microsoft's Visual C++ Compiler (MSVC) is commonly used on Windows. Additionally, you might need build tools like CMake or Microsoft Build Tools to automate the build process and manage dependencies.

**Debugger:** Debugging tools are essential for identifying and resolving issues in your code. IDEs like Visual Studio come with powerful debugging capabilities, allowing you to set breakpoints, inspect variables, step through code, and analyze memory usage.

**Version Control:** Version control systems like Git help you manage your codebase, track changes, collaborate with team members, and maintain different versions of your software. You can use Git directly from the command line or integrate it with your chosen IDE.

**Code Analysis and Profiling:** These tools help you identify performance bottlenecks, memory leaks, and other code-related issues. Visual Studio and other IDEs often provide integrated profiling and analysis tools.

**Documentation and Help Resources:** Access to documentation, tutorials, and online resources is crucial for learning and troubleshooting. Most IDEs offer online documentation and allow you to integrate external resources for quick access.

**Libraries and Frameworks:** C++ developers often rely on libraries and frameworks to streamline development and take advantage of pre-built functionality. For Windows, the Windows API provides a vast range of functions and classes for GUI, file I/O, networking, and more. Additionally, C++ developers might use third-party libraries like Boost, Qt, or SDL.

**Package Management:** Package managers like vcpkg or Conan can help you manage external libraries and their dependencies more efficiently.

**Testing Tools:** Unit testing and integration testing are essential for ensuring the reliability of your code. Frameworks like Google Test and Catch2 are commonly used for C++ testing.

**Coding Standards and Formatting:** Consistent coding standards and formatting are essential for maintainable code. IDEs often include tools or extensions to help enforce coding standards.

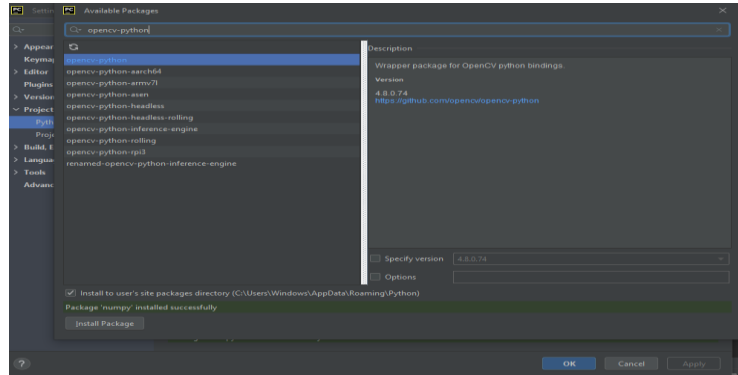
**Documentation Generation:** Generating documentation from your code comments helps keep your project well-documented. Tools like Doxygen or tools integrated into your IDE can automate this process.

**Collaboration and Communication Tools:** For team-based development, communication and collaboration tools like Slack, Microsoft Teams, or other project management software can enhance productivity and facilitate communication. A well-equipped developer desktop for C++ on Windows should

encompass these components to provide a complete and efficient environment for building C++ applications.



## 7. Opencv



OpenCV (Open Source Computer Vision Library) is an open-source computer vision and image processing library that provides a wide range of tools, algorithms, and functions for tasks related to computer vision, image and video analysis, and machine vision applications. It was originally developed by Intel and later maintained by the OpenCV community.

OpenCV is written in C++ and provides interfaces for various programming languages, including Python, C, and Java. It is designed to be highly efficient, cross-platform, and easy to use, making it a popular choice for researchers, developers, and practitioners working in the field of computer vision.

Key features and components of OpenCV include:

**Image and Video Processing:** OpenCV offers functions for reading, writing, and processing images and videos. It includes operations like filtering, edge detection, image transformation, and more.

**Feature Detection and Extraction:** OpenCV provides methods for detecting and extracting key features in images, such as corners, edges, and blobs. These features can be used for object recognition, tracking, and matching.

**Object Detection and Tracking:** OpenCV supports object detection and tracking using techniques like Haar cascades, HOG (Histogram of Oriented Gradients), and deep learning-based methods.

**Camera Calibration and 3D Reconstruction:** OpenCV includes tools for camera calibration, stereo vision, and 3D scene reconstruction from multiple images.

**Machine Learning Integration:** OpenCV can be integrated with machine learning libraries like TensorFlow and PyTorch to build and deploy computer vision models.

**Graphical User Interface (GUI):** OpenCV provides graphical tools for displaying images and videos, creating interactive interfaces, and handling user input.

**Support for Various Platforms:** OpenCV is cross-platform and can be used on Windows, macOS, Linux, and even mobile platforms like Android and iOS.

**Integration with Hardware:** OpenCV can interface with cameras, depth sensors, and other hardware devices for real-time vision applications. **OpenCL and CUDA Support:** OpenCV provides support for parallel processing using technologies like OpenCL and CUDA, which can accelerate certain operations on compatible hardware.

GFPI-F-135 V01





- Cargar imágenes

```
import cv2
import face_recognition as fr

#cargar imagenes
foto_control = fr.load_image_file('Empleados/FotoA.jpg')
foto_prueba = fr.load_image_file('Empleados/FotoB.jpg')

#pasar imagenes rgb
foto_control = cv2.cvtColor(foto_control, cv2.COLOR_BGR2RGB)
foto_prueba = cv2.cvtColor(foto_prueba, cv2.COLOR_BGR2RGB)

#mostrar imagenes
cv2.imshow('Foto control', foto_control)
cv2.imshow('Foto prueba', foto_prueba)

#mantener programa abierto
cv2.waitKey(0)
```

- Mostrar Caras

```
import cv2
import face_recognition as fr
#cargar imagenes
foto_control = fr.load_image_file('Empleados/FotoA.jpg')
foto_prueba = fr.load_image_file('Empleados/FotoB.jpg')

#pasar las imagenes a formato rgb
foto_control = cv2.cvtColor(foto_control, cv2.COLOR_BGR2RGB)
foto_prueba = cv2.cvtColor(foto_prueba, cv2.COLOR_BGR2RGB)

#localizar cara control
lugar_cara_A = fr.face_locations(foto_control)[0]
cara_codificada_A = fr.face_encodings(foto_control)[0]

lugar_cara_B = fr.face_locations(foto_prueba)[0]
cara_codificada_B = fr.face_encodings(foto_prueba)[0]

#mostrar rectangulos
cv2.rectangle(foto_control, (lugar_cara_A[3], lugar_cara_A[0]),
              (lugar_cara_A[1], lugar_cara_A[2]), (0,255,0),2)
cv2.rectangle(foto_prueba, (lugar_cara_B[3], lugar_cara_B[0]),
              (lugar_cara_B[1], lugar_cara_B[2]), (0,255,0),2)
```



```
#print(lugar_cara_A)

#mostrar imagenas
cv2.imshow('Foto control', foto_control)
cv2.imshow('Foto prueba', foto_prueba)

#mantener programa abierto
cv2.waitKey(0)
```

- Comparar Caras

```
import cv2
import face_recognition as fr
#cargar imagenes
foto_control = fr.load_image_file('Empleados/FotoA.jpg')
foto_prueba = fr.load_image_file('Empleados/FotoB.jpg')

#pasar las imagenes a formato rgb
foto_control = cv2.cvtColor(foto_control, cv2.COLOR_BGR2RGB)
foto_prueba = cv2.cvtColor(foto_prueba, cv2.COLOR_BGR2RGB)

#localizar cara control
lugar_cara_A = fr.face_locations(foto_control)[0]
cara_codificada_A = fr.face_encodings(foto_control)[0]

lugar_cara_B = fr.face_locations(foto_prueba)[0]
cara_codificada_B = fr.face_encodings(foto_prueba)[0]

#mostrar rectangulos
cv2.rectangle(foto_control, (lugar_cara_A[3], lugar_cara_A[0]),
              (lugar_cara_A[1], lugar_cara_A[2]), (0,255,0),2)
cv2.rectangle(foto_prueba, (lugar_cara_B[3], lugar_cara_B[0]),
              (lugar_cara_B[1], lugar_cara_B[2]), (0,255,0),2)

#Realizar comparación
resultado = fr.compare_faces([cara_codificada_A], cara_codificada_B)
print(resultado)

#mostrar imagenas
cv2.imshow('Foto control', foto_control)
cv2.imshow('Foto prueba', foto_prueba)

#mantener programa abierto
cv2.waitKey(0)
```



- Medir\_distancias\_caras.py

```
import cv2
import face_recognition as fr
#cargar imagenes
foto_control = fr.load_image_file('Empleados/FotoA.jpg')
foto_prueba = fr.load_image_file('Empleados/FotoB.jpg')

#pasar las imagenes a formato rgb
foto_control = cv2.cvtColor(foto_control, cv2.COLOR_BGR2RGB)
foto_prueba = cv2.cvtColor(foto_prueba, cv2.COLOR_BGR2RGB)

#localizar cara control
lugar_cara_A = fr.face_locations(foto_control)[0]
cara_codificada_A = fr.face_encodings(foto_control)[0]

lugar_cara_B = fr.face_locations(foto_prueba)[0]
cara_codificada_B = fr.face_encodings(foto_prueba)[0]

#mostrar rectangulos
cv2.rectangle(foto_control, (lugar_cara_A[3], lugar_cara_A[0]),
              (lugar_cara_A[1], lugar_cara_A[2]), (0,255,0),2)
cv2.rectangle(foto_prueba, (lugar_cara_B[3], lugar_cara_B[0]),
              (lugar_cara_B[1], lugar_cara_B[2]), (0,255,0),2)

#Realizar comparación
resultado = fr.compare_faces([cara_codificada_A], cara_codificada_B)
print(resultado)

#Medidas de la distancia
distancia = fr.face_distance([cara_codificada_A], cara_codificada_B)
print(distancia)

#mostrar imagenas
cv2.imshow('Foto control', foto_control)
cv2.imshow('Foto prueba', foto_prueba)

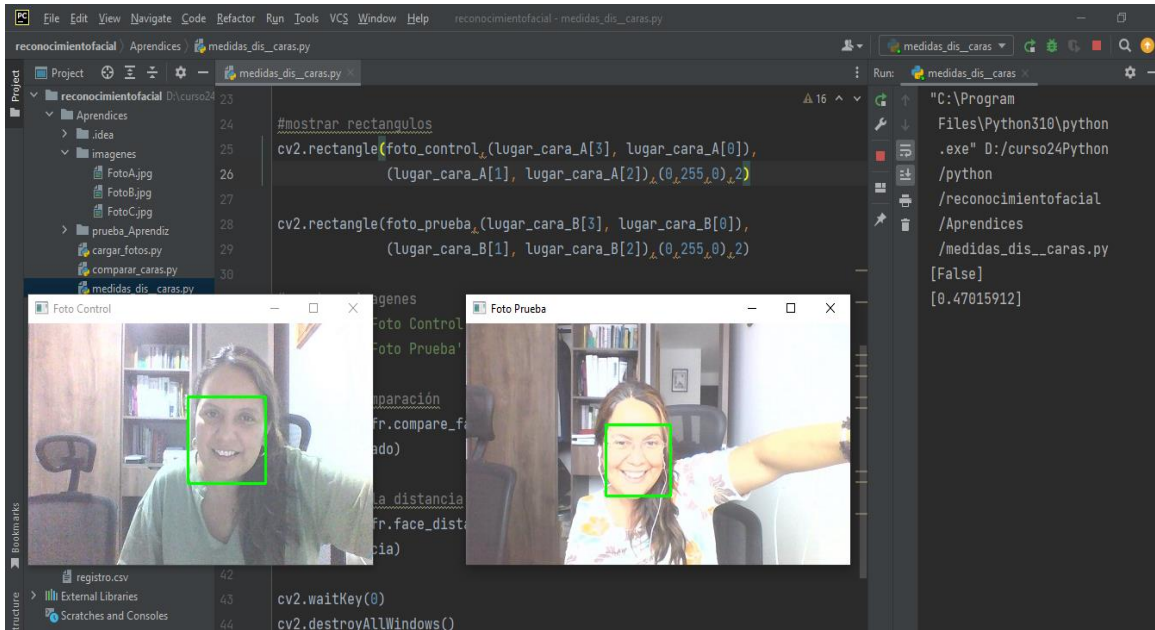
#mantener progama abierto
cv2.waitKey(0)
```



Ahora vamos a jugar un poco con el índice de tolerancia:

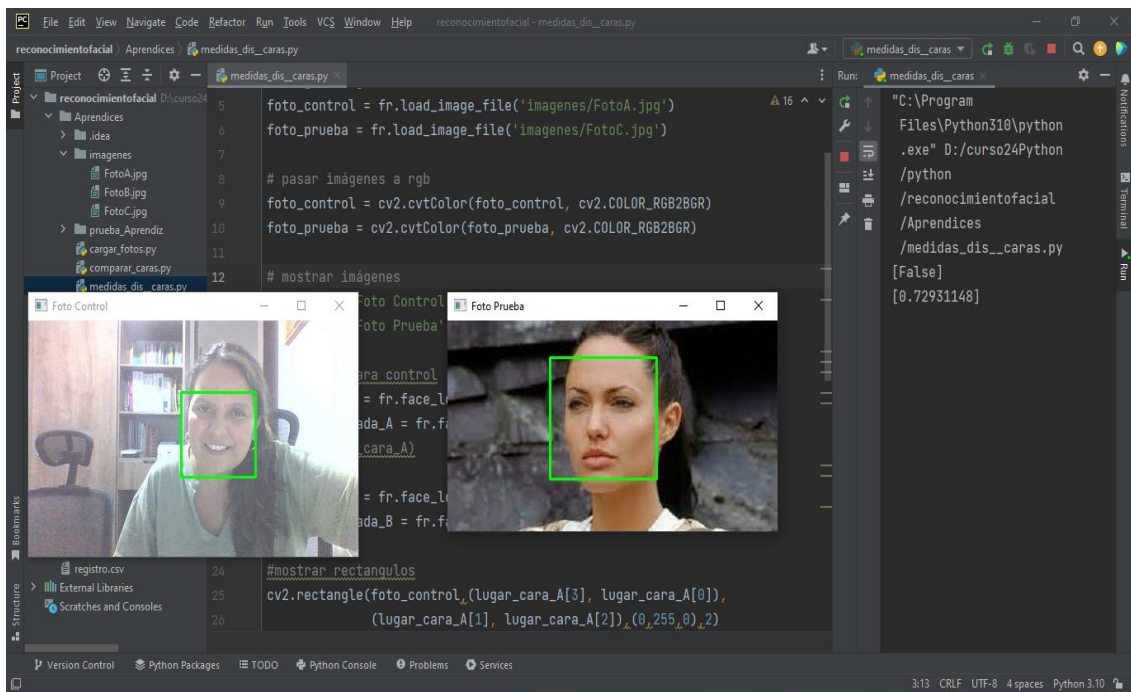
```
#Realizar comparación
resultado = fr.compare_faces([cara_codificada_A], cara_codificada_B, 0.3)
print(resultado)
```

Ejecutamos y observa que...



A pesar de que las dos fotos corresponden a su instructora: Sonia el margen de tolerancia nos dice que es falso. En otras palabras, la sonia de la izquierda no es la misma de la derecha.

Ahora hacemos la prueba con Angelina.



```
#Realizar comparación
resultado = fr.compare_faces([cara_codificada_A], cara_codificada_B,0.9)
print(resultado)
```

The screenshot shows a Python IDE with a project named 'reconocimiento facial'. The code in the editor includes file paths for 'Aprendices', 'imagenes', and 'prueba\_Aprendiz', and uses OpenCV functions like 'cv2.rectangle' and 'cv2.imshow' to display faces with bounding boxes. Two windows, 'Foto Control' and 'Foto Prueba', are open, showing a woman and a woman's face respectively, both with green bounding boxes around their faces. The IDE also shows a file explorer on the left and a terminal on the right.



Nivel de tolerancia es de 0.72931148, para suerte de Angelina. Ahora estamos diciendo que ambas fotos coinciden y por tanto su similitud es verdadera. O muy parecidas que opinan?...Jejeje.

En el contexto del reconocimiento facial en Python, la tolerancia se refiere a un parámetro que controla la sensibilidad con la que se comparan las características faciales extraídas de una imagen con las características almacenadas en una base de datos o modelo. Esta tolerancia determina cuánta diferencia se permite entre las características faciales para considerar que dos caras pertenecen a la misma persona.

En otras palabras, la tolerancia establece un margen de variación aceptable al comparar las características faciales. Si la diferencia entre las características faciales de dos imágenes cae dentro de este margen (tolerancia), se considera que las imágenes representan a la misma persona. Si la diferencia excede la tolerancia, se considera que las imágenes pertenecen a personas diferentes.

Ajustar la tolerancia adecuadamente es esencial para lograr un equilibrio entre la precisión y la adaptabilidad del sistema de reconocimiento facial. Una tolerancia alta permitirá lidiar mejor con cambios menores en la apariencia, como iluminación variable o expresiones faciales, pero podría aumentar la posibilidad de falsos positivos. Por otro lado, una tolerancia baja puede hacer que el sistema sea más preciso en la coincidencia de caras, pero podría aumentar la posibilidad de falsos negativos.

En el reconocimiento facial en Python, la tolerancia generalmente se configura como un valor numérico que se pasa como argumento a las funciones o métodos que realizan comparaciones de caras. La elección del valor de tolerancia depende de las necesidades específicas de la aplicación y del equilibrio deseado entre precisión y flexibilidad.

### Mostrar las distancias

```
import cv2
import face_recognition as fr
#cargar imagenes
foto_control = fr.load_image_file('Empleados/FotoA.jpg')
foto_prueba = fr.load_image_file('Empleados/FotoC.jpg')

#pasar las imagenes a formato rgb
foto_control = cv2.cvtColor(foto_control, cv2.COLOR_BGR2RGB)
foto_prueba = cv2.cvtColor(foto_prueba, cv2.COLOR_BGR2RGB)

#localizar cara control
lugar_cara_A = fr.face_locations(foto_control)[0]
cara_codificada_A = fr.face_encodings(foto_control)[0]

lugar_cara_B = fr.face_locations(foto_prueba)[0]
cara_codificada_B = fr.face_encodings(foto_prueba)[0]

#mostrar rectangulos
cv2.rectangle(foto_control, (lugar_cara_A[3], lugar_cara_A[0]),
              (lugar_cara_A[1], lugar_cara_A[2]), (0,255,0), 2)
cv2.rectangle(foto_prueba, (lugar_cara_B[3], lugar_cara_B[0]),
```



```
(lugar_cara_B[1], lugar_cara_B[2]), (0,255,0),2)

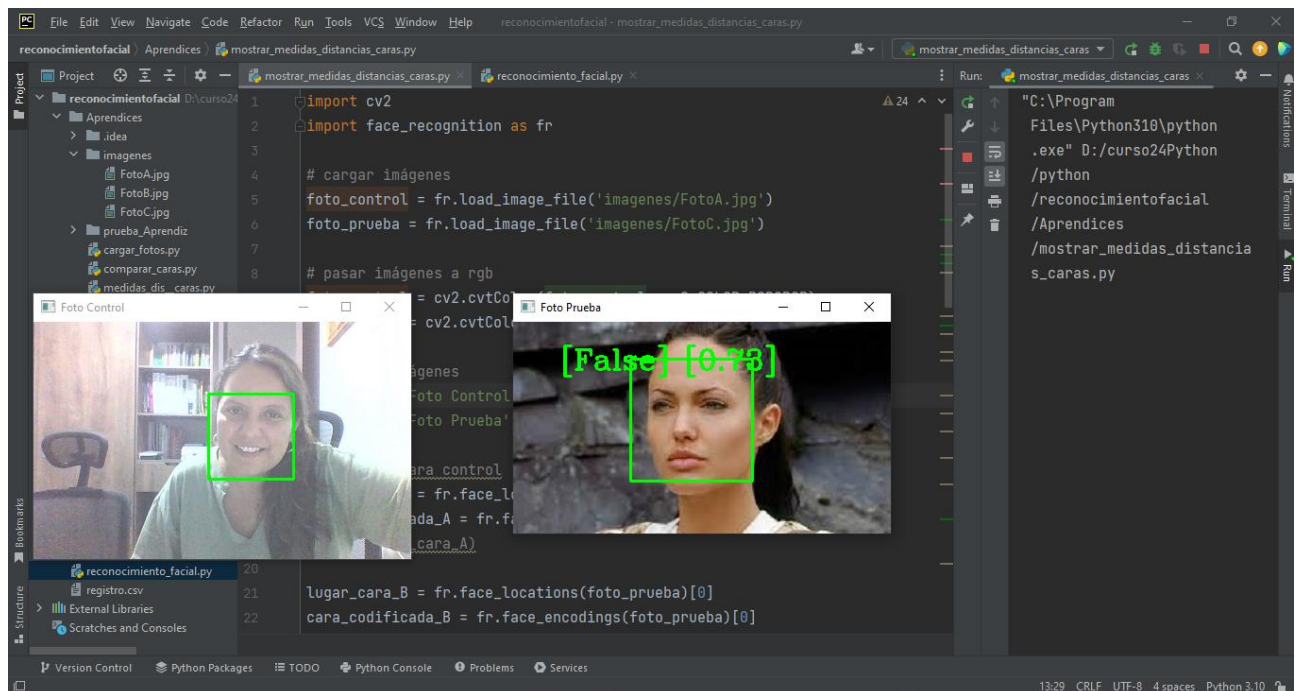
#Realizar comparación
resultado = fr.compare_faces([cara_codificada_A], cara_codificada_B)
print(resultado)

#Medidas de la distancia
distancia = fr.face_distance([cara_codificada_A], cara_codificada_B)
print(distancia)

#Mostrar resultados
cv2.putText(foto_prueba,
            f'{{resultado}} {{distancia.round(2)}}',
            (50,50), cv2.FONT_HERSHEY_COMPLEX, 1, (0,255,0),2)

#mostrar imagenas
cv2.imshow('Foto control', foto_control)
cv2.imshow('Foto prueba', foto_prueba)

#mantener progama abierto
cv2.waitKey(0)
```

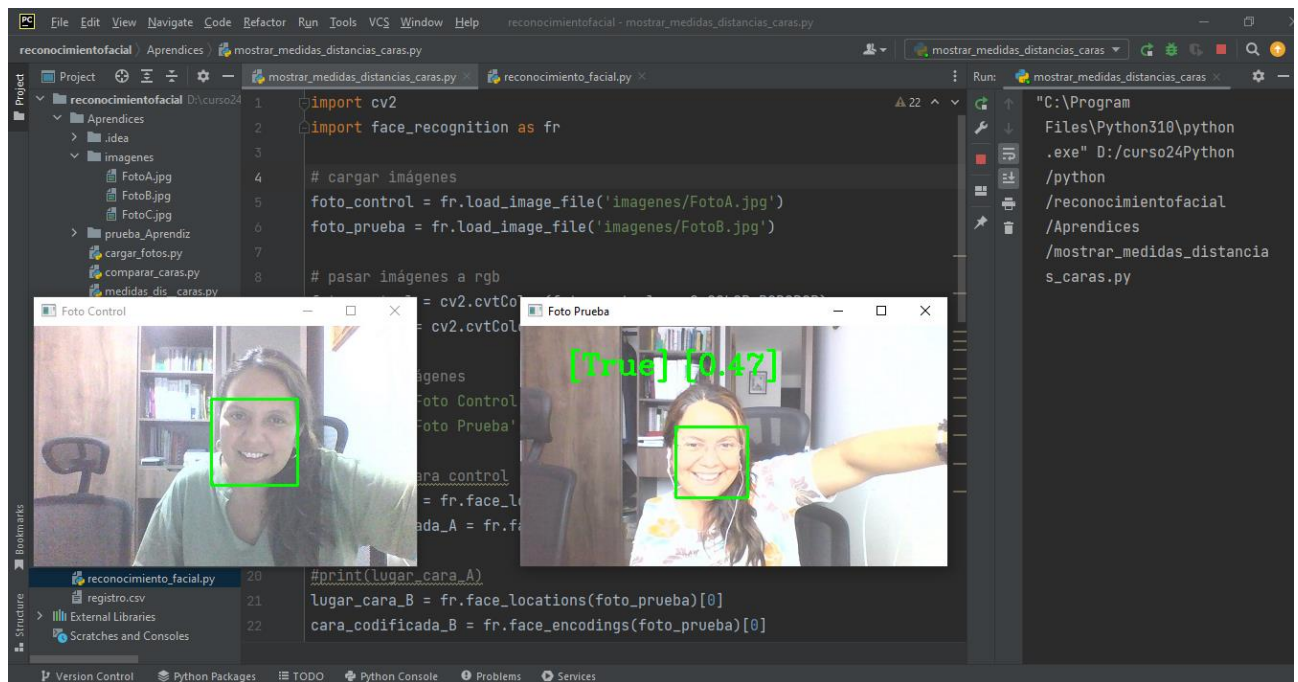


De esta manera ya ahora el algoritmo me muestra de que efectivamente es falsa la foto de Angelina y por lo tanto estamos hablando de dos personas distintas.

GFPI-F-135 V01

Ahora bien si comparamos nuevamente.





Wualá....Estamos hablando de la misma persona. Por tanto nuestro algoritmo funciona.

### Proyecto: Asistencia para empleados con reconocimiento facial.

Crear base de datos con los rostros de los empleados

```
import cv2
import face_recognition as fr
import os

#crear base de datos

ruta = "Empleados"
mis_imagenes = []
nombres_empleados = []
lista_empleados = os.listdir(ruta)
print(lista_empleados)

#imprimir los nombres de los empleados, sin la terminacion .jpg, png.
for nombre in lista_empleados:
```





```
imagen_actual = cv2.imread(f'{ruta}\\{nombre}')
mis_imagenes.append(imagen_actual)
nombres_empleados.append(os.path.splitext(nombre)[0])
print(nombres_empleados)
# codificar imagenes
def codificar(imagenes):
    # crear una lista nueva
    lista_codificada = []

    # pasar todas las imagenes a rgb
    for imagen in imagenes:
        imagen = cv2.cvtColor(imagen, cv2.COLOR_BGR2RGB)

        # codificar
        encodings = fr.face_encodings(imagen)

        if len(encodings) > 0:
            # If there's at least one face detected, use the first encoding
            codificado = encodings[0]
            lista_codificada.append(codificado)
        else:
            # No face detected, you can choose to handle this situation as
needed
            # For example, you can append None to the list to indicate no
encoding
            lista_codificada.append(None)

    # devolver lista codificada
    return lista_codificada

lista_empleados_codificada = codificar(mis_imagenes)

print(len(lista_empleados_codificada))

#mantener ventana abierta
cv2.waitKey(0)
```

Encontrar  
coincidencia  
s

GFPI-F-135 V01

```
import cv2
import face_recognition as fr
```



```
import os

#crear base de datos

ruta = "Empleados"
mis_imagenes = []
nombres_empleados = []
lista_empleados = os.listdir(ruta)
print(lista_empleados)

#imprimir los nombres de los empleados, sin la terminacion .jpg, png.
for nombre in lista_empleados:
    imagen_actual = cv2.imread(f'{ruta}\\{nombre}')
    mis_imagenes.append(imagen_actual)
    nombres_empleados.append(os.path.splitext(nombre)[0])
print(nombres_empleados)
# codificar imagenes
def codificar(imagenes):
    # crear una lista nueva
    lista_codificada = []

    # pasar todas las imagenes a rgb
    for imagen in imagenes:
        imagen = cv2.cvtColor(imagen, cv2.COLOR_BGR2RGB)

        # codificar
        encodings = fr.face_encodings(imagen)

        if len(encodings) > 0:
            # If there's at least one face detected, use the first encoding
            codificado = encodings[0]
            lista_codificada.append(codificado)
        else:
            # No face detected, you can choose to handle this situation as
            # needed
            # For example, you can append None to the list to indicate no
            # encoding
            lista_codificada.append(None)

    # devolver lista codificada
    return lista_codificada

lista_empleados_codificada = codificar(mis_imagenes)

#print(len(lista_empleados_codificada))

#tomar una imagen de camara web
captura = cv2.VideoCapture(0, cv2.CAP_DSHOW)

#leer la imagen de la camara
exito, imagen = captura.read()

if not éxito:
    print("No se ha podido tomar la captura")
```



```
else:
    cara_captura = fr.face_locations(imagen)

    #codificar cara capturada
    cara_capturada_codificada = fr.face_encodings(imagen, cara_captura)

    #buscar coincidencias
    for caracodif, caraubic in zip(cara_capturada_codificada, cara_captura):
        coincidencias = fr.compare_faces(lista_empleados_codificada,
        caracodif)
        distancias = fr.face_distance(lista_empleados_codificada, caracodif)

        print(distancias)

#mantener ventana abierta
cv2.waitKey(0)
```

Al  
ejecutarlo....

The screenshot shows a Python IDE with the following components:

- Project Explorer:** Shows a project named 'reconocimientofacial' with a folder 'Empleados' containing image files (Cosmo Kramer.jpg, Elaine Benes.jpg, Federico Garay.jpg, George Constanza.jpg, Jerry Seinfeld.jpg, Sonia\_0.png) and a folder 'Lib' containing various files.
- Code Editor:** Displays the script 'encontrar\_coincidencias.py' with the following code:

```
#codificar cara capturada
cara_capturada_codificada = fr.face_encodings(imagen, cara_captura)

#buscar coincidencias
for caracodif, caraubic in zip(cara_capturada_codificada, cara_captura):
    coincidencias = fr.compare_faces(lista_empleados_codificada, caracodif)
    distancias = fr.face_distance(lista_empleados_codificada, caracodif)

    print(distancias)

#mantener ventana abierta
cv2.waitKey(0)
```
- Run Console:** Shows the output of the script execution:

```
.exe" D:/curso24Python/python
/reconocimientofacial
/encontrar_coincidencias.py
['Cosmo Kramer.jpg',
'Elaine Benes.jpg',
'Federico Garay.jpg',
'George Constanza.jpg',
'Jerry Seinfeld.jpg',
'Sonia_0.png']
['Cosmo Kramer', 'Elaine Benes', 'Federico Garay', 'George Constanza', 'Jerry Seinfeld', 'Sonia_0']
[0.86352032 0.83755092
0.76027583 0.75539432
0.87385068 0.61861501]
Process finished with
exit code 0
```



Importamos a numpy

```
import cv2
import face_recognition as fr
import os
import numpy
#crear base de datos

ruta = "Empleados"
mis_imagenes = []
nombres_empleados = []
lista_empleados = os.listdir(ruta)
print(lista_empleados)

#imprimir los nombres de los empleados, sin la terminacion .jpg,
png.
for nombre in lista_empleados:
    imagen_actual = cv2.imread(f'{ruta}\{nombre}')
    mis_imagenes.append(imagen_actual)
    nombres_empleados.append(os.path.splitext(nombre)[0])
print(nombres_empleados)
# codificar imagenes
def codificar(imagenes):
    # crear una lista nueva
    lista_codificada = []

    # pasar todas las imagenes a rgb
    for imagen in imagenes:
        imagen = cv2.cvtColor(imagen, cv2.COLOR_BGR2RGB)

        # codificar
        encodings = fr.face_encodings(imagen)

        if len(encodings) > 0:
            # If there's at least one face detected, use the
            first encoding
            codificado = encodings[0]
            lista_codificada.append(codificado)
        else:
            # No face detected, you can choose to handle this
            situation as needed
            # For example, you can append None to the list to
            indicate no encoding
            lista_codificada.append(None)

    # devolver lista codificada
    return lista_codificada

lista_empleados_codificada = codificar(mis_imagenes)

#print(len(lista_empleados_codificada))
```



```
#tomar una imagen de camara web
captura = cv2.VideoCapture(0, cv2.CAP_DSHOW)

#leer la imagen de la camara
exito, imagen = captura.read()

if not exito:
    print("No se ha podido tomar la captura")
else:
    cara_captura = fr.face_locations(imagen)

    #codificar cara capturada
    cara_capturada_codificada = fr.face_encodings(imagen,
    cara_captura)

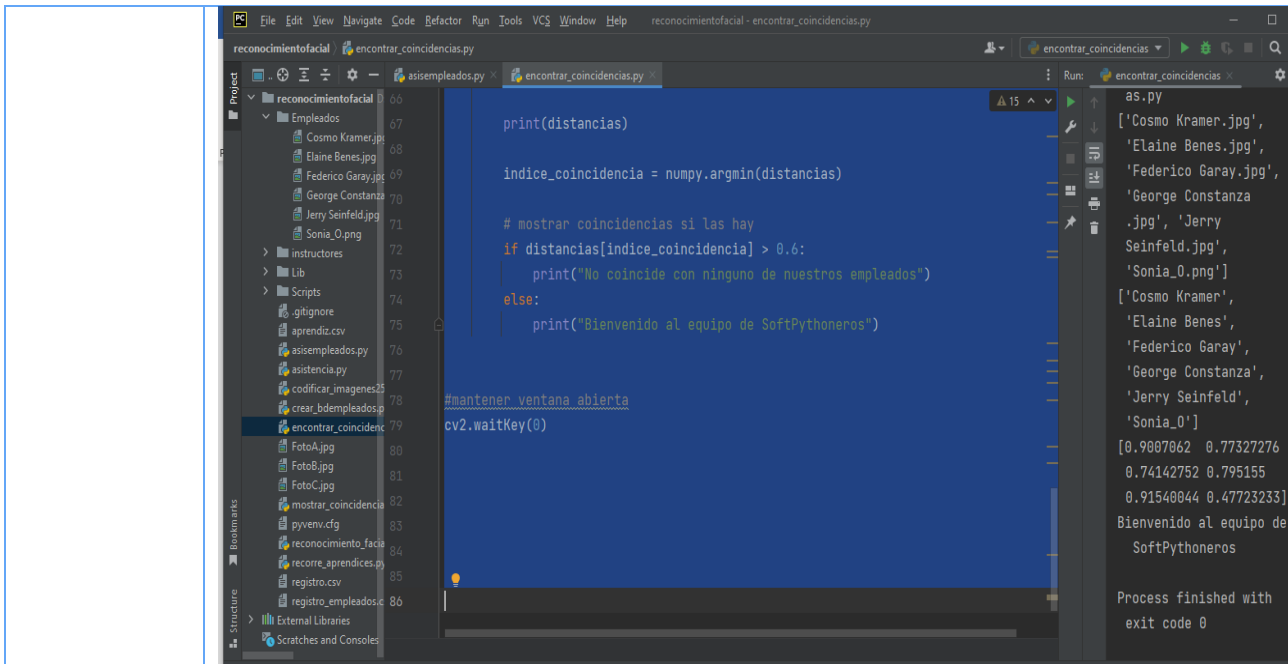
    #buscar coincidencias
    for caracodif, caraubic in zip(cara_capturada_codificada,
    cara_captura):
        coincidencias =
    fr.compare_faces(lista_empleados_codificada, caracodif)
        distancias = fr.face_distance(lista_empleados_codificada,
    caracodif)

        print(distancias)

        indice_coincidencia = numpy.argmin(distancias)

        # mostrar coincidencias si las hay
        if distancias[indice_coincidencia] > 0.6:
            print("No coincide con ninguno de nuestros
    empleados")
        else:
            print("Bienvenido al equipo de SoftPythoneros")

#mantener ventana abierta
cv2.waitKey(0)
```



**Mostrar  
coincidencia  
s**

```
import cv2
import face_recognition as fr
import os
import numpy
#crear base de datos

ruta = "Empleados"
mis_imagenes = []
nombres_empleados = []
lista_empleados = os.listdir(ruta)
print(lista_empleados)

#imprimir los nombres de los empleados, sin la terminacion .jpg, png.
for nombre in lista_empleados:
    imagen_actual = cv2.imread(f'{ruta}\{nombre}')
    mis_imagenes.append(imagen_actual)
    nombres_empleados.append(os.path.splitext(nombre)[0])
print(nombres_empleados)
# codificar imagenes
def codificar(imagenes):
    # crear una lista nueva
```



```
lista_codificada = []

# pasar todas las imagenes a rgb
for imagen in imagenes:
    imagen = cv2.cvtColor(imagen, cv2.COLOR_BGR2RGB)

    # codificar
    encodings = fr.face_encodings(imagen)

    if len(encodings) > 0:
        # If there's at least one face detected, use the first encoding
        codificado = encodings[0]
        lista_codificada.append(codificado)
    else:
        # No face detected, you can choose to handle this situation as
        # needed
        # For example, you can append None to the list to indicate no
        # encoding
        lista_codificada.append(None)

# devolver lista codificada
return lista_codificada

lista_empleados_codificada = codificar(mis_imagenes)

#print(len(lista_empleados_codificada))

#tomar una imagen de camara web
captura = cv2.VideoCapture(0, cv2.CAP_DSHOW)

#leer la imagen de la camara
exito, imagen = captura.read()

if not exito:
    print("No se ha podido tomar la captura")
else:
    cara_captura = fr.face_locations(imagen)

    #codificar cara capturada
    cara_capturada_codificada = fr.face_encodings(imagen, cara_captura)

    #buscar coincidencias
    for caracodif, caraubic in zip(cara_capturada_codificada, cara_captura):
        coincidencias = fr.compare_faces(lista_empleados_codificada,
        caracodif)
        distancias = fr.face_distance(lista_empleados_codificada, caracodif)

        print(distancias)

        indice_coincidencia = numpy.argmin(distancias)

        # mostrar coincidencias si las hay
        if distancias[indice_coincidencia] > 0.6:
            print("No coincide con ninguno de nuestros empleados")
```



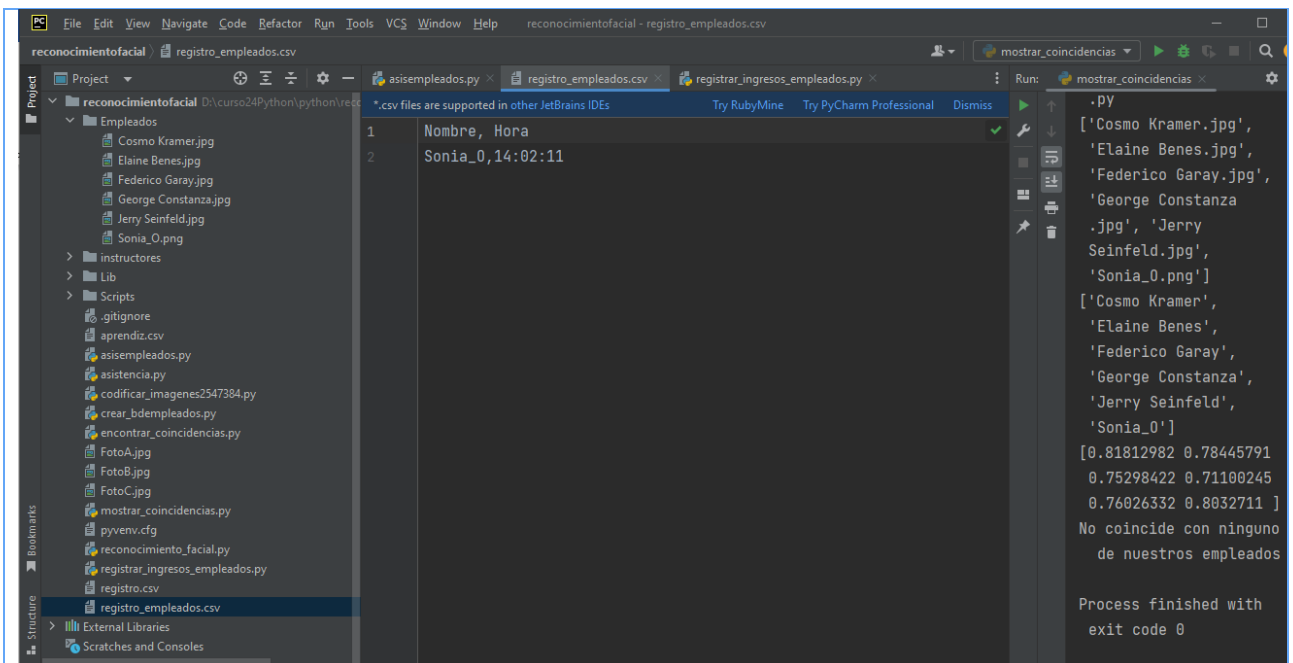
```
else:
    # buscar el nombre del empleado encontrado
    nombre = nombres_empleados[indice_coincidencia]
    y1, x2, y2, x1 = caraubic
    cv2.rectangle(imagen, (x1, y1), (x2, y2), (0, 255, 0), 2)
    cv2.rectangle(imagen, (x1, y2 - 35), (x2, y2), (0, 255, 0),
cv2.FILLED)
    cv2.putText(imagen, nombre, (x1 + 6, y2 - 6),
cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)

    # mostrar la imagen obtenida
    cv2.imshow('Imagen web', imagen)

    # mantener ventana abierta
    cv2.waitKey(0)
```

	Puedes probar con la foto de cualquier compañero de tu grupo de trabajo acercando la foto de tu amigo desde tu celular.
<b>Registrar asistencias</b>	<p>Para esta práctica debemos importar la librería</p> <pre>from datetime import datetime</pre> <p>Creamos un archivo de extensión .csv. Que es el registro que me validara los ingresos al sistema</p>





```
import cv2
import face_recognition as fr
import os
import numpy
#crear base de datos

ruta = "Empleados"
mis_imagenes = []
nombres_empleados = []
lista_empleados = os.listdir(ruta)
print(lista_empleados)

#imprimir los nombres de los empleados, sin la terminacion .jpg,
png.
for nombre in lista_empleados:
    imagen_actual = cv2.imread(F'{ruta}\{nombre}')
    mis_imagenes.append(imagen_actual)
    nombres_empleados.append(os.path.splitext(nombre)[0])
print(nombres_empleados)
# codificar imagenes
def codificar(imagenes):
    # crear una lista nueva
    lista_codificada = []

    # pasar todas las imagenes a rgb
    for imagen in imagenes:
        imagen = cv2.cvtColor(imagen, cv2.COLOR_BGR2RGB)
```



```
# codificar
encodings = fr.face_encodings(imagen)

if len(encodings) > 0:
    # If there's at least one face detected, use the
    first encoding
    codificado = encodings[0]
    lista_codificada.append(codificado)
else:
    # No face detected, you can choose to handle this
    situation as needed
    # For example, you can append None to the list to
    indicate no encoding
    lista_codificada.append(None)

# devolver lista codificada
return lista_codificada

# registrar los ingresos
def registrar_ingresos(persona):
    f = open('registro_empleados.csv', 'r+')
    lista_datos = f.readlines()
    nombres_registro = []
    for linea in lista_datos:
        ingreso = linea.split(',')
        nombres_registro.append(ingreso[0])

    if persona not in nombres_registro:
        ahora = datetime.now()
        string_ahora = ahora.strftime('%H:%M:%S')
        f.writelines(f'\n{persona},{string_ahora}')

lista_empleados_codificada = codificar(mis_imagenes)

#print(len(lista_empleados_codificada))

#tomar una imagen de camara web
captura = cv2.VideoCapture(0, cv2.CAP_DSHOW)

#leer la imagen de la camara
exito, imagen = captura.read()

if not exito:
    print("No se ha podido tomar la captura")
else:
    cara_captura = fr.face_locations(imagen)

    #codificar cara capturada
    cara_capturada_codificada = fr.face_encodings(imagen,
    cara_captura)

    #buscar coincidencias
```



	<pre>for caracodif, caraubic in zip(cara_capturada_codificada, cara_captura):     coincidencias = fr.compare_faces(lista_empleados_codificada, caracodif)     distancias = fr.face_distance(lista_empleados_codificada, caracodif)      print(distancias)      indice_coincidencia = numpy.argmin(distancias)      # mostrar coincidencias si las hay     if distancias[indice_coincidencia] &gt; 0.6:         print("No coincide con ninguno de nuestros empleados")     else:         # buscar el nombre del empleado encontrado         nombre = nombres_empleados[indice_coincidencia]         y1, x2, y2, x1 = caraubic         cv2.rectangle(imagen, (x1, y1), (x2, y2), (0, 255, 0), 2)         cv2.rectangle(imagen, (x1, y2 - 35), (x2, y2), (0, 255, 0), cv2.FILLED)         cv2.putText(imagen, nombre, (x1 + 6, y2 - 6), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)          registrar_ingresos(nombre)          # mostrar la imagen obtenida         cv2.imshow('Imagen web', imagen)          # mantener ventana abierta         cv2.waitKey(0)</pre>
	Ensayo probando con tus amigos de equipo de desarrollo
Reto1.	Realiza el siguiente video tutorial que dejo en el enlace(Actividad Individual). Realiza la guía pasoa paso a modo de manual del usuario explicando tu ejercicio.
	<a href="https://www.youtube.com/watch?v=QbQuNyEgMvY">https://www.youtube.com/watch?v=QbQuNyEgMvY</a> GFPI-F-135 V01



<b>Reto2</b>	Ahora impleméntalo en tu proyecto.(Actividad grupal)
--------------	--

- Ambiente Requerido.  
Virtual espacio asignado. Plan de mejoramiento: **GA4-220501095-AA2-EV04 - Diagrama de clases del proyecto de software**
- Materiales

**Programas instalados: Python 3.8**  
**Pycharm**  
**VisualStudio Code.**

#### 4. ACTIVIDADES DE EVALUACIÓN

Evidencias de Aprendizaje	Criterios de Evaluación	Técnicas e Instrumentos de Evaluación
<b>Evidencias de Conocimiento :</b>	<p>Enlaza una serie de elementos breves, concretos y sencillos para crear un manual técnico del sistema, tomando como referente la aplicación de buenas prácticas en programación corregir errores detectados en el código.</p> <p>Hace uso de vocabulario relacionado con técnicas para el reconocimiento facial, usos, funcionalidades y las aplica en su proyecto o actividad laboral.</p> <p>Entrega el manual técnico de su taller de sus sistema de reconocimiento facial, adjunto al código requerido</p>	<b>IE-GA4-240202501-AA1-EV01</b> Cuestionario
<b>Evidencias de Desempeño</b>		<b>IE-GA4-240202501-AA1-EV02</b> Lista de chequeo .
<b>Evidencias de Producto:</b>		<b>IE-GA4-220501095-AA4- EV04</b> Lista de chequeo



## **5. GLOSARIO DE TÉRMINOS.**

**Reconocimiento Facial:** La tecnología que utiliza algoritmos para identificar y autenticar personas mediante el análisis de características faciales únicas.

**Algoritmo de Reconocimiento Facial:** Un conjunto de instrucciones diseñado para analizar y comparar características faciales y determinar si coinciden con un individuo específico.

**Características Faciales:** Elementos distintivos del rostro de una persona, como la disposición de ojos, nariz, boca y otras características anatómicas.

**Biométrica:** El uso de características biológicas únicas, como las faciales, para la identificación y autenticación de individuos.

**Modelo 3D del Rostro:** Una representación tridimensional de la cara de una persona, que permite un análisis más detallado y preciso.

**Base de Datos de Caras:** Un conjunto de imágenes faciales utilizadas para entrenar y mejorar algoritmos de reconocimiento facial.

**Puntos de Referencia Faciales:** Puntos específicos en el rostro, como las esquinas de los ojos o la nariz, utilizados para medir y comparar características.

**Precisión del Reconocimiento Facial:** La capacidad del sistema para identificar correctamente a una persona sin errores.

**Recuerdo del Reconocimiento Facial:** La habilidad de un sistema para identificar correctamente a una persona después de haberla visto anteriormente.

**Falso Positivo:** Cuando un sistema de reconocimiento facial identifica erróneamente a una persona como una coincidencia válida.

**Falso Negativo:** Cuando un sistema de reconocimiento facial no identifica correctamente a una persona que debería haber sido reconocida.

**Tasa de Aceptación:** La proporción de intentos de reconocimiento facial exitosos en comparación con el total de intentos.

**Tasa de Rechazo:** La proporción de intentos de reconocimiento facial rechazados en comparación con el total de intentos.

**Liveness Detection (Detección de Vida):** La capacidad de un sistema para determinar si la persona presente frente a la cámara es un ser humano real y no una foto o video.

GFPI-F-135 V01

**Privacidad y Seguridad:** Las preocupaciones éticas y legales relacionadas con la recopilación, almacenamiento y uso de datos biométricos de personas.



RGPD (Reglamento General de Protección de Datos): Una regulación de la Unión Europea que establece reglas sobre la protección de datos personales y la privacidad.

Consentimiento Informado: El proceso de obtener el permiso explícito y consciente de un individuo antes de recopilar y procesar sus datos faciales.

Detección de Emociones: La capacidad de un sistema para identificar las emociones de una persona a partir de su expresión facial.

Reconocimiento Facial en Tiempo Real: La capacidad de identificar a una persona en tiempo real, generalmente en video en vivo.

Descriptores Faciales: Valores numéricos que representan características faciales específicas, utilizados para comparar y analizar rostros.

Análisis de Similitud de Rostros: La comparación de características faciales para determinar la semejanza entre dos rostros.

## **6. REFERENTES BIBLIOGRÁFICOS.**

Definición. (2013). Definición de oratoria. <https://definicion.de/oratoria/>

Definición. (2020). Definición de comunicación. <https://definicion.de/comunicacion/>

Bourque, P., Dupuis, R., Abran, A., Moore, J. W., & Tripp, L. (1999). The guide to the software engineering body of knowledge. IEEE software, 16(6), 35-44.

Kupchyk, L., & Litvinchuk, A. (2020) Constructing Personal Learning Environments through ICT-Mediated Foreign

Language Instruction. EasyChair Preprint 4251. [https://easychair.org/publications/preprint\\_download/kxvw](https://easychair.org/publications/preprint_download/kxvw)

Moure, O. (1999). El acento en las palabras de dos sílabas.

<http://www.ompersonal.com.ar/ompronounce/unit11/page1.htm>

RAE. (2020). Argumento. <https://dle.rae.es/>

Systems, V. (2013). Inglés: grado superior. McGraw-Hill <https://elibro-net.bdigital.sena.edu.co/es/ereader/senavirtual/50221?page=1>



## 7. CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha
Autor (es)	Sonia Yamile Ortega Carrillo	Instructora Vocera – Lider ADSO	TICS	16-08-23

## 8. CONTROL DE CAMBIOS (diligenciar únicamente si realiza ajustes a la guía)

	Nombre	Cargo	Dependencia	Fecha	Razón del Cambio
Autor (es)					