**Programación Avanzada**

**Evaluación Actividad Practica 4-2022**

**Anna Chiauzzi COD 2263737**

**Ingeniería informática -CAU Medellín**

**Docente:**

**DAVID BOHORQUEZ CARO**

**Universidad Santo Tomás**

**Decanatura de División de Educación Abierta y a Distancia**

**Ingeniería informática**

**Centro de Atención Universitario Medellín**

**2022**

**Actividades a desarrollar:**

2. Realice un manual técnico que describa el desarrollo del asistente virtual.  
3. Realice un manual de usuario del asistente virtual.  
4. Sustentar los productos de esta evaluación, a través un video (duración aprox. 5 a 8 min.) explicativo y demostrativo, donde se evidencie el manejo de conceptos y las evidencias.

**Evidencias a entregar:**Enlace del repositorio GitHub que contenga una carpeta llamada “evaluacion\_final” y en su interior:

1. Scripts de Python (.py)
2. Archivo (.pdf) y enlace (público) del documento manual técnico.
3. Archivo (.pdf) y enlace (público) del documento manual de usuario.
4. Enlace (público) del video con la sustentación de productos.



**Desarrollo**

1. Introducción

Este manual es dirigido al equipo técnico que por cómo se pensó el desarrollo de esta solución no ha participado a la codificación. Por no haber creado el producto ni haber participado al proceso, pero siendo ellos responsables del mantenimiento a futuro y del funcionamiento y/o eventuales aplicaciones de mejorías o modificaciones, el objetivo de este manual es explicar el paso a paso técnico que ha llevado a la creación de este asistente virtual. El equipo tiene conocimiento de Python y tiene el conocimiento de cómo funciona este producto y el proceso sin embargo necesita información más detallada por las razones que se mencionaron.

1. Objetivo

Este manual tiene el objetivo de describir las pautas de configuración y la lógica con la que se ha desarrollado esta aplicación de asistente virtual.

* El asistente virtual debe tener un nombre a través del cual se llama y recibe las órdenes.
* El asistente virtual debe reconocer comandos por voz y convertirlos a texto para su posterior procesamiento.
* El asistente virtual debe convertir texto a voz.
* El asistente virtual debe reproducir un video YouTube.
* El asistente virtual debe responder cuándo se le pregunte por la hora actual.
* El asistente virtual debe buscar cualquier información en Wikipedia.
* El asistente virtual debe abrir la página de Google.
* El asistente virtual debe enviar un mensaje de correo electrónico.
* El asistente virtual debe tomar una foto.

1. Requerimientos de hardware y software

Para ejecutar el asistente virtual se sugiere tener un procesador Intel Core2 Dúo de 2 GH y un disco Duro de 100 GB de espacio libre para datos del sistema - Memoria RAM de 4 GB. Sistema operativo Windows 10, versión de Python 3.10. Posible utilizar el softawre Sublime text build 3211.

1. Pruebas

Pruebas unitarias

Las pruebas unitarias son de muy bajo nivel y se realizan cerca de la fuente de la aplicación. Consisten en probar métodos y funciones individuales de las clases, componentes o módulos que usa la aplicación.

🡪Aplicación Juanita: ejecución total

Pruebas de integración

Las pruebas de integración verifican que los distintos módulos o servicios utilizados por tu aplicación funcionan bien en conjunto. Por ejemplo, se puede probar la interacción con la base de datos o asegurarse de que los microservicios funcionan bien en conjunto y según lo esperado.

🡪Aplicación Juanita: ejecución total

Pruebas funcionales

Las pruebas funcionales se centran en los requisitos empresariales de una aplicación. Solo verifican el resultado de una acción y no comprueban los estados intermedios del sistema al realizar dicha acción. A veces, se confunden las pruebas de integración con las funcionales, ya que ambas requieren que varios componentes interactúen entre sí. La diferencia es que una prueba de integración puede simplemente verificar que puedes hacer consultas en la base de datos, mientras que una prueba funcional esperaría obtener un valor específico desde la base de datos, según dicten los requisitos del producto.

🡪Aplicación Juanita: ejecución total

Pruebas integrales

Las pruebas integrales replican el comportamiento de un usuario con el software en un entorno de aplicación completo. Además, verifican que diversos flujos de usuario funcionen según lo previsto, y pueden ser tan sencillos como cargar una página web o iniciar sesión, o mucho más complejos, como la verificación de notificaciones de correo electrónico, pagos en línea, etc. Se recomienda tener algunas pruebas integrales clave y depender más de pruebas de menor nivel (unitarias y de integración) para poder detectar rápidamente nuevos cambios.

🡪Aplicación Juanita: ejecución total

Pruebas de aceptación

Las pruebas de aceptación son pruebas formales que verifican si un sistema satisface los requisitos empresariales. Requieren que se esté ejecutando toda la aplicación durante las pruebas y se centran en replicar las conductas de los usuarios. Sin embargo, también pueden ir más allá y medir el rendimiento del sistema y rechazar cambios si no se han cumplido determinados objetivos.

🡪Aplicación Juanita: ejecución total

Pruebas de rendimiento

Las pruebas de rendimiento evalúan el rendimiento de un sistema con una carga de trabajo determinada. Ayudan a medir la fiabilidad, la velocidad, la escalabilidad y la capacidad de respuesta de una aplicación. Por ejemplo, una prueba de rendimiento puede analizar los tiempos de respuesta al ejecutar un gran número de solicitudes, o cómo se comporta el sistema con una cantidad significativa de datos. Puede determinar si una aplicación cumple con los requisitos de rendimiento, localizar cuellos de botella, medir la estabilidad durante los picos de tráfico y mucho más.

🡪Aplicación Juanita: Medición tiempos de respuesta con las diferentes solicitude

Pruebas de humo

Las pruebas de humo son pruebas básicas que sirven para comprobar el funcionamiento básico de la aplicación. Están concebidas para ejecutarse rápidamente, y su objetivo es ofrecerte la seguridad de que las principales funciones de tu sistema funcionan según lo previsto. Las pruebas de humo pueden resultar útiles justo después de realizar una compilación nueva para decidir si se pueden ejecutar o no pruebas más detalladas, o inmediatamente después de una implementación para asegurarse de que la aplicación funciona correctamente en el entorno que se acaba de implementar

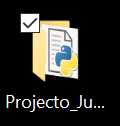
🡪Aplicación Juanita: ejecución total

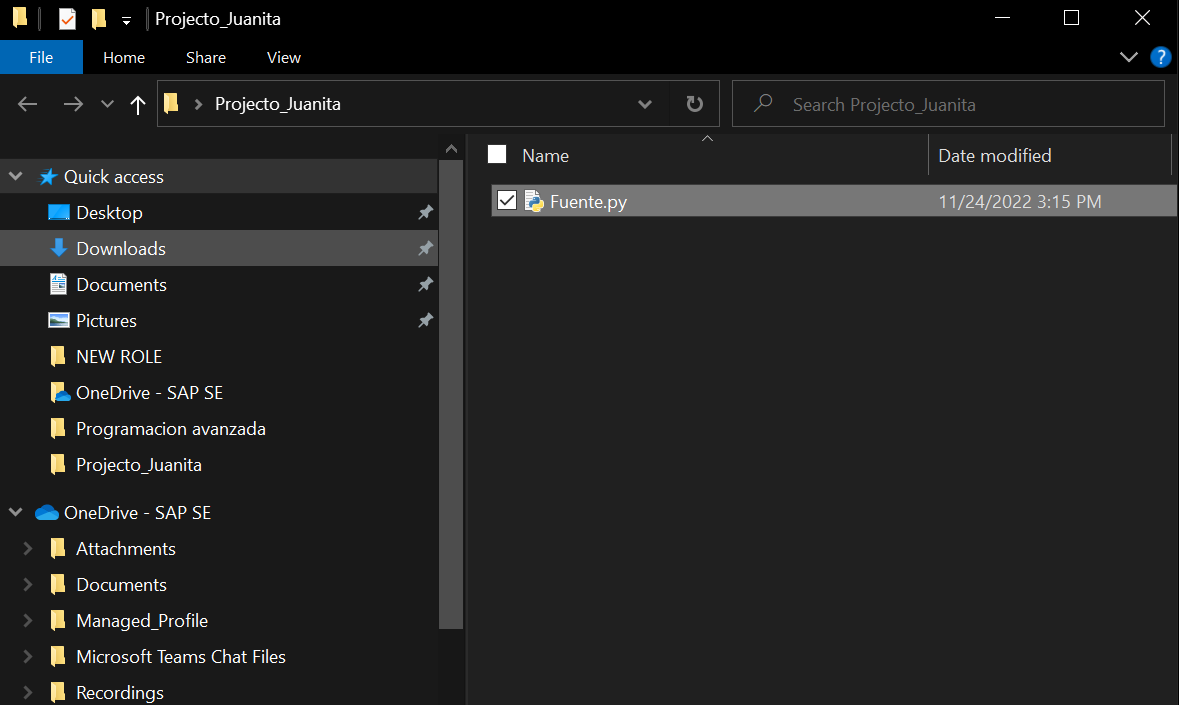
1. Indicaciones.

Este código se desarrolló considerando el PEP 8 – Style Guide for Python Code que se pueden encontrar en forma detallada en el siguiente link <https://peps.python.org/pep-0008/>

PEP 8, a veces escrito como PEP8 o PEP-8, es un documento que brinda pautas y mejores prácticas sobre cómo escribir código Python. Fue escrito en 2001 por Guido van Rossum, Barry Varsovia y Nick Coghlan. El enfoque principal de PEP 8 es mejorar la legibilidad y la consistencia del código de Python.

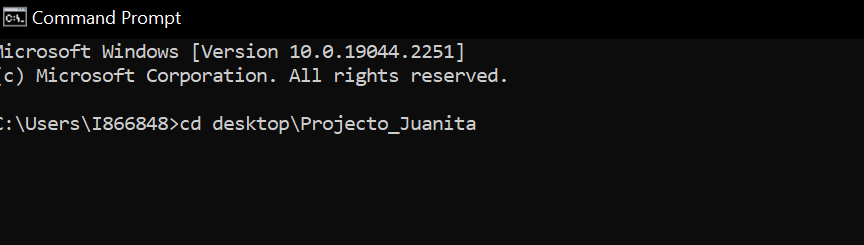
Para la ejecución del programa y verificar su correcto funcionamento es necesario ejecutar desde el menú start la aplicación “Command Prompt”. Por buena práctica se recomenda que el código sorgente en formato .py, que se añade a este manual debe ser guardado en una carpeta en el proprio escritorio con el siguiente nombre “Proyecto\_Juanita”

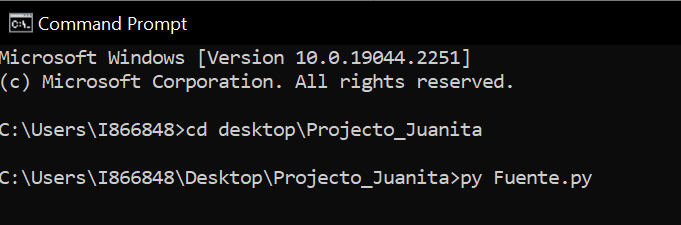


Para poder visualizar el código es posible utilizar la aplicación Sublime Text 3 que permite hacer modificaciones a cualquier parte del código. Los desarrolladores que hacen parte del departamento de TI, interno a la empresa, tendrán roles y perfiles definidos y serán autorizados a hacer cambios, modificaciones mantenimiento a la aplicación. El listado será oficializado y publico para toda la empresa. Se indican a continuación los pasos para la ejecución del código:

1. Ejecutar el Command Prompt del menú start del computer y escribir la siguiente instrucción:



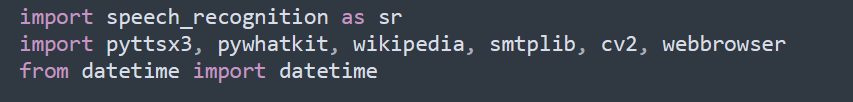
1. Para ejecutar el asistente escribir la siguiente instrucción llamando el código Fuente y el archivo. Py que lo contiene



1. Desarrollo técnico

En esta sección se aclarará el paso a paso ejecutado para el desarrollo del asistente virtual se explicará la función que tiene cada porción de código en definir el alcance que tiene Juanita.

Mencionamos las librerías importadas que se han necesitado para el desarrollo de todas las funciones de Juanita:



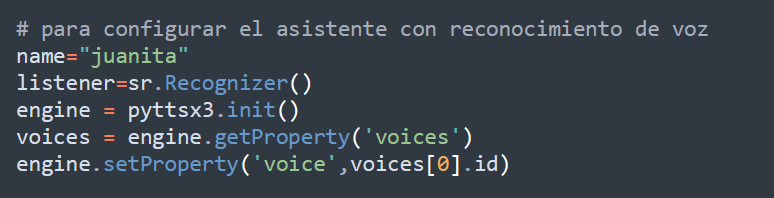
Todas las librerías deben ejecutarse en el Command Prompt para después poder ejecutar el código:

* pip install SpeechRecognition
* pip install pyttsx3
* pip3 install pywhatkit
* $ pip install wikipedia
* pip install secure-smtplib
* pip install opencv-python
* pip install pycopy-webbrowser
* pip install datetime

1. **Speech recognition , pyttsx3, pywhatkit**

Speech recognition es una librería que permite realizar reconocimiento de voz, con soporte para varios motores y APIs, en línea y fuera de línea. Muy importante mencionar la clase Recognizer. El objetivo principal de una instancia de Recognizer es reconocer el habla. Cada instancia viene con una variedad de configuraciones y funciones para reconocer el habla de una fuente de audio. Pyttsx3 es una librería de conversión de texto a voz en Python. A diferencia de las bibliotecas alternativas, funciona sin conexión y es compatible con Python 2 y 3. Sigue el código para configurar el asistente con reconocimiento de voz. Se asigna el nombre al asistente vocal que es como se activaran los comandos. La variable listener que es el comando para que nuestro asistente virtual empiece a reconocer.

La variable engine, en este caso, sirve para inicializar nuestra librería pyttsx3 de conversión de texto a voz en Python. Hemos creado una variable voices en la cual utilizamos un método llamado **get.Property** donde ponemos el mismo parámetro voices. Esto lo hacemos para escoger una voz que sea más agradable en respuesta a la ejecución del comando vocal. El método **setProperty** es para tomar la voz en posición 0 (que corresponde al habla en español)



Siguiente paso es definir los métodos para hablar y escuchar:

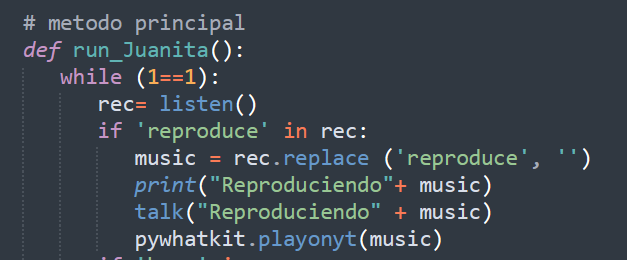
Con la función “talk” permitimos a la computadora que hable. La función “listen” es para que nuestra computadora pueda escuchar, y estamos diciendo que tome el micrófono como fuente de sonido. En la variable “rec” especificamos la forma para que reconozca nuestra voz y la transforme en texto. Esta se obtiene con los servicios de reconocimiento de Google. Con el “if” se permite reconocer el nombre como llamamos nuestro asistente.



Donde especificamos la palabra que tiene que mencionar para reproducir una canción, esta palabra es “reproduce”. Nuestra asistente no dejara el mensaje de lo que hemos preguntado de reproducir, esto se obtiene gracias a la instrucción Print.

La librería pywhatkit es la librería que permite que se pueda reproducir la canción que se elige o cualquier video queremos se reproduzca en Youtube

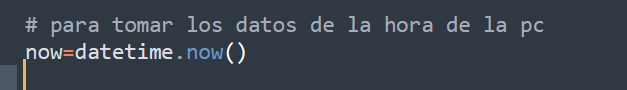
Esta librería es la más popular para la automatización de WhatsApp y YouTube.



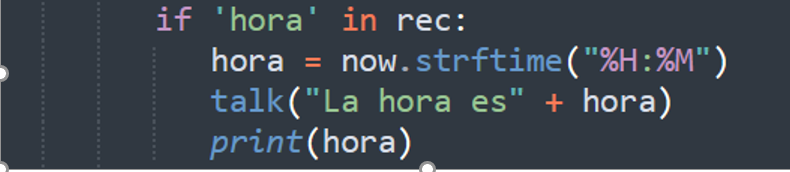
1. **From datetime import datetime**

Otra capacidad que debe tener el asistente virtual es ser capaz de decir la fecha y la hora que tiene la máquina de la persona que lo está ejecutando.

Para obtener este resultado el módulo datetime en Python sirve precisamente para recuperar datos sobre la fecha y la hora.

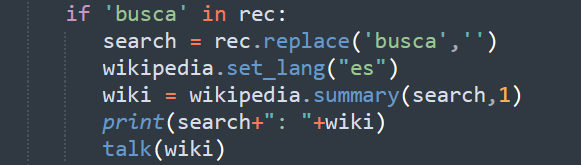


En el método principal se especifica que se debe mencionar la palabra “hora” para que el asistente pueda activar la funcion. También se utiliza el método strftime () que devuelve uno string que representa la fecha y la hora utilizando el objeto datetime.



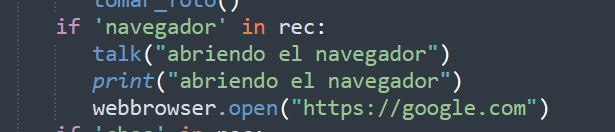
1. **Wikipedia, Webbrowser**

Wikipedia es una libreria de Python que facilita el acceso y el análisis de datos de Wikipedia. Esta librería permite la busqueda en Wikipedia y permite de obtener resúmenes de artículos, datos como enlaces e imágenes de una página y más. Wikipedia envuelve la API de MediaWiki para que pueda usar los datos de Wikipedia. Para que se pueda activar la librería de Wikipedia se debe mencionar al asistente virtual la palabra “busca” y el asistente reportara el texto de la búsqueda hecha y lo ira leyendo.



La librería webbrowser en Python, es un controlador de navegador web Proporciona una interfaz de alto nivel que permite a los usuarios visualizar documentos basados ​​en la Web.

El código que permite que Juanita abra un navegador web es el siguiente:



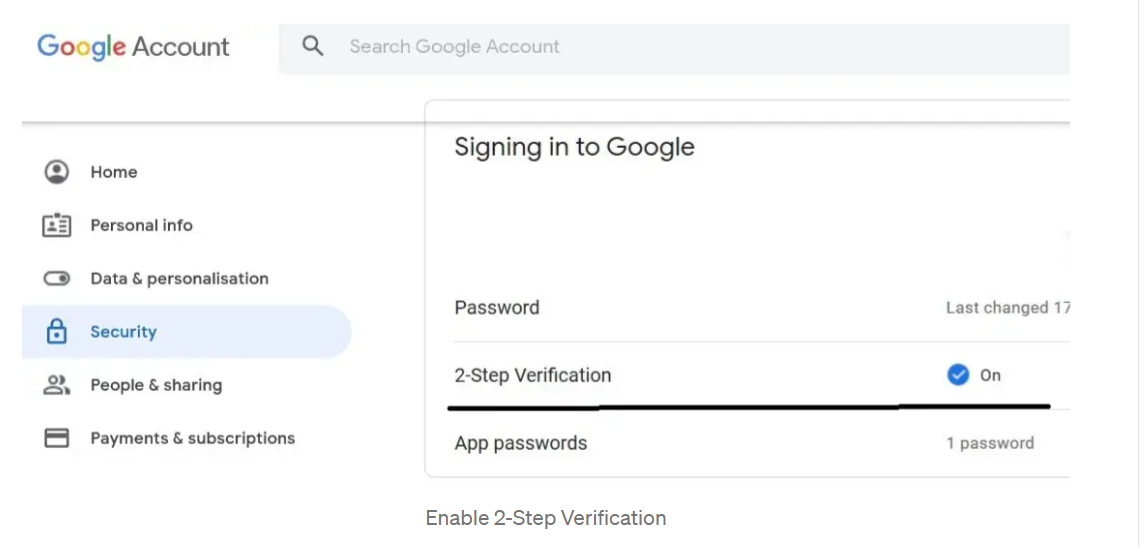
Se debe mencionar la palabra “navegador” para que se abra Google en nuestro ejemplo.

1. **Smtplib, Cv2**

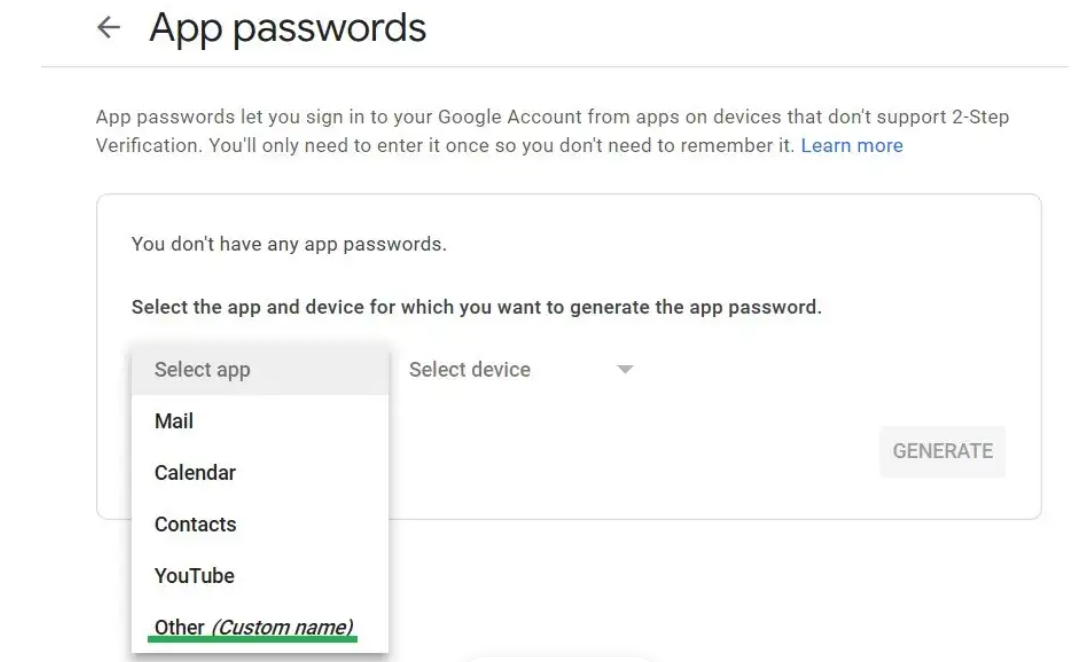
El módulo smtplib define un objeto de sesión de cliente SMTP que se puede usar para enviar correo a cualquier máquina de Internet con un listener daemon SMTP o ESMTP.

Para este punto es muy importante hacer una aclaración. Para usar una cuenta de Gmail para enviar correos electrónicos con una aplicación de terceros, por ejemplo uno Script de Python, en este caso, necesitamos configurar una contraseña de aplicación. Por razones de seguridad, la contraseña normal de Gmail está restringida solo al inicio de sesión web. Y a partir de 2020, incluso si se activa la opción “Less secure app Access” en la cuenta de Google no se podrá iniciar sesión en la cuenta de Gmail con la contraseña normal en un script de Python .

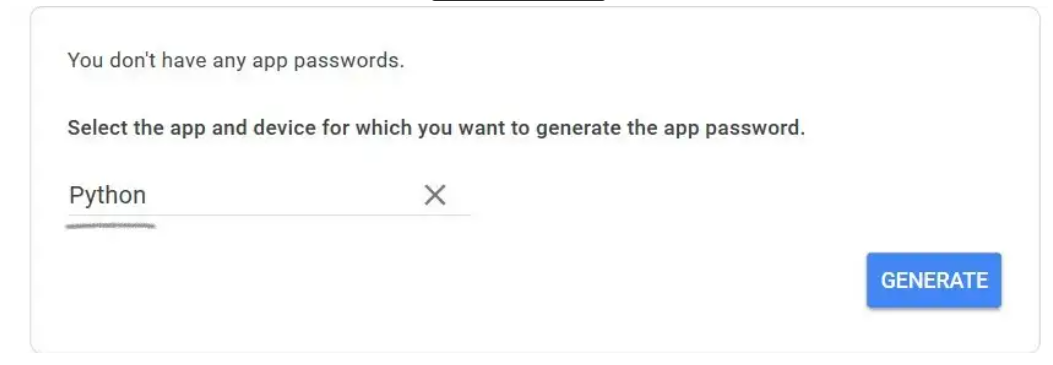
Primero, habilitar la verificación en dos pasos. Simplemente ir a la cuenta de Google > Seguridad > Iniciar sesión en Google, seleccionar 2 steps verifications y seguir las instrucciones:



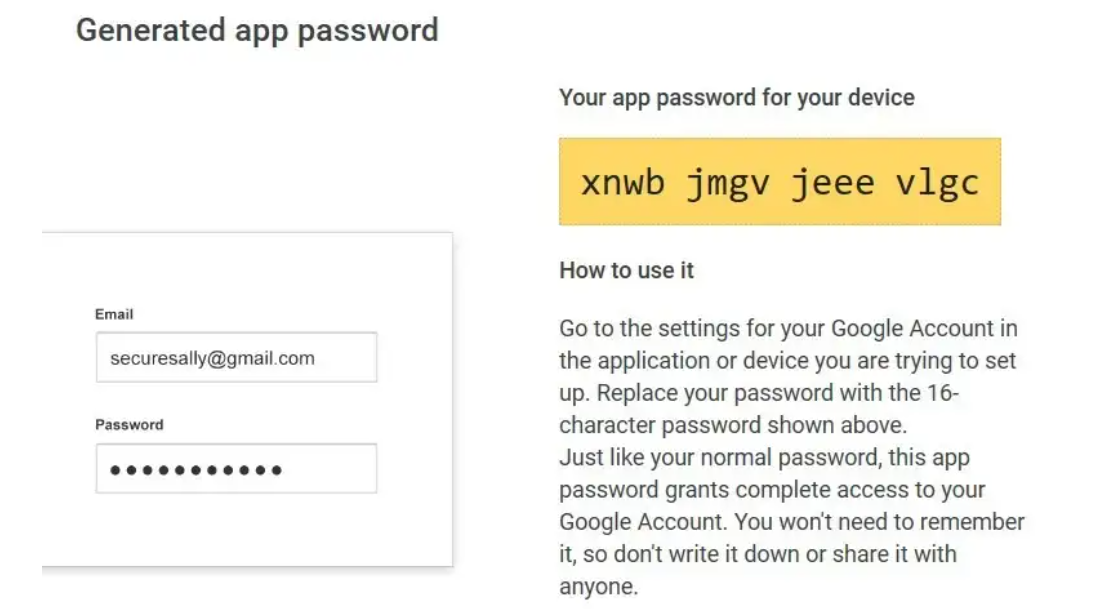
A continuación, crear una contraseña de aplicación. Simplemente seleccionar “App password” en “two steps verification”. Seleccionar “Other” en el menú desplegable “Other applications”



introducir un nombre, por ejemplo, Python y hacer clic en 'GENERAR'. Tener en cuenta que este nombre no tiene ningún vínculo con el script de Python y podría ser cualquier cosa.



Luego se obtendrá una nueva contraseña de la aplicación. Después de esto se debe copiar y guardar la contraseña de 16 caracteres sin espacios, por ejemplo. xnwbjmgvjeeevlgc, para usar en la secuencia de comandos de Python.



El código para que el asistente virtual pueda ejecutar la instrucción de enviar un correo es el siguiente:



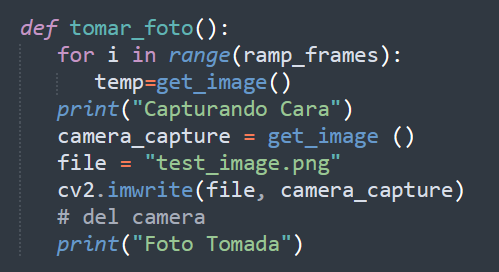
En este método se han definidos los puntos que debe tener el correo, cual debe ser el mensaje el texto contenido en el objecto y parámetros de login especificando el correo que envía el mensaje y cual es el correo que estaría recibiendo. Una vez que el correo se ha logrado enviar exitosamente el asistente nos informa del resultado.

El método general contiene la instrucción para activar el envio que se activa al mencionar la palabra “correo”



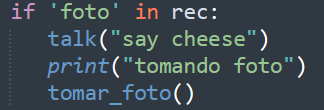
La instrucción de tomar foto necesita OpenCV que es una libreria de Python que se utiliza para resolver problemas de visión por computadora. La visión por computadora incluye comprender y analizar imágenes digitales por computadora y procesar las imágenes o proporcionar datos relevantes después de analizar la imagen. Es importante mencionar que la visión por computadora es un campo de la Inteligencia Artificial enfocado a que las computadoras puedan extraer información a partir de imágenes, ofreciendo soluciones a problemas del mundo real. OpenCV es una librería de código abierto utilizada en el aprendizaje automático y el procesamiento de imágenes. Realiza tareas como reconocer dígitos escritos a mano, rostros humanos y objetos.

El código para que nuestra asistente Juanita pueda tomar una foto es el siguiente:



En este paso se explica la toma del imagen que se reportara en un mensaje “Capturando Cara” y como se guarda la imagen tomada en formato.png. Una vez terminada la toma el asistente nos avisara escribiendo “Foto tomada”

En el método general se especifica lo siguiente:

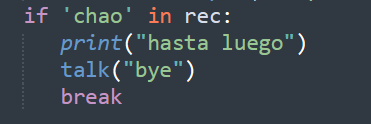


“Foto” es la palabra que activa la asistente para la toma. La asistente nos dirá “Say cheese” y nos dejara un mensaje diciendo que esta tomando la foto.

1. Cierre del asistente

Juanita se puede cerrar en cualquier momento con un mensaje de saludo “chao” y ella saludara la persona que la esta ejecutando con un mensaje vocal y escrito.

El código para ejecutar esta instrucción de despedida es lo siguiente:



Cerrando este manual las recomendaciones son que cualquier cambio al código fuente de Juanita sean para optimaciones (ejemplo de la voz, de definición de imágenes etc…) Buena práctica es acompañar cualquier cambio utilizando el PEP 8 – Style Guide for Python Code para que se respecte una coherencia del lenguaje y que cada cambia quede documentado en manuales técnicos para el Departamanto de TI y que los usuarios puedan ser actualizados mediante la generacions de cambios en los manuales de Usuarios. Cada cambio debe paras por fases de pruebas:

1. Conclusiones y recomendaciones

Cerrando este manual las recomendaciones son que cualquier cambio al código fuente de Juanita sean para optimaciones y mejoras (ejemplo calidad de la voz, definición de imágenes etc…) Buena práctica es acompañar cualquier cambio utilizando el PEP 8 – Style Guide for Python Code para que se respecte una coherencia del lenguaje y que cada cambia quede documentado en manuales técnicos para el Departamanto de TI y que los usuarios puedan ser actualizados mediante la generación de cambios en los manuales de Usuarios. Cada cambio debe pasar por las fases de pruebas mencionados en el punto 4 de este manual considerando las que aplican para el caso específico.