UNIVERSIDAD SANTO TOMÁS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA
EDUCACIÓN ABIERTA Y A DISTANCIA
VIGILADA MINEDUCACIÓN - SNIES 1704

ACREDITACIÓN INSTITUCIONAL DE ALTA CALIDAD MULTICAMPUS
Res. MEN No. 01456 del 29 de enero de 2016
Vigencia por seis años

Programación Avanzada

Evaluación Final 2022 -2

Nombre del estudiante:

Peter Alberto Salamanca Alfonso

Código: 2257834

Docente:

David Bohórquez

Universidad Santo Tomás

Decanatura de División de Educación Abierta y a Distancia

Ingeniería en Informática

Centro de Atención Universitario Facatativá

Madrid

2022

UNIVERSIDAD SANTO TOMÁS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA
EDUCACIÓN ABIERTA Y A DISTANCIA
VIGILADA MINEDUCACIÓN - SNIES 1704

ACREDITACIÓN INSTITUCIONAL DE ALTA CALIDAD MULTICAMPUS
Res. MEN No. 01456 del 29 de enero de 2016
Vigencia por seis años

Manual Técnico - Asistente virtual "Alexa"

Peter Alberto Salamanca Alfonso

Universidad Santo Tomás

Decanatura de División de Educación Abierta y a Distancia

Ingeniería en Informática

Centro de Atención Universitario Facatativá

Tabla de contenido

1. Introducción

En la actualidad la tecnología ha estado evolucionado constantemente, dando origen a un concepto de “asistente virtual”, el cual representa un software que permite o ayuda al usuario automatizando y realizando procesos o tareas con la menor interacción entre el hombre y la máquina. De acuerdo a lo anterior, en el presente trabajo, se pretende documentar el manual técnico para la creación de nuestro asistente virtual “Alexa”, el cual cuenta con unas funciones básicas, por lo tanto es el código fuente base y no esta exento de optimizaciones.

2. Objetivos

2.1 Objetivo general

Identificar las librerías y módulos que permitan la creación de nuestro Asistente Virtual

2.2 Objetivos específicos

Determinar las acciones realizadas por el software.

Investigar la información relacionada a las librear utilizadas.

Documentar la información obtenida, dando respuesta a los requerimientos propuestos.

3. Contenido

"Alexa" es un asistente virtual el cual reconoce comandos por voz y los transforma a texto, realizando acciones predeterminadas. A continuación, se dará a conocer como programar tu propia "Alexa":

Para este ejemplo se configurarán las siguientes acciones que deberá realizar nuestra asistente virtual:

1. Debe tener un nombre a través del cual se llama y recibe las órdenes.
2. Debe reconocer comandos por voz y convertirlos a texto para su posterior procesamiento.
3. Debe convertir texto a voz.
4. Debe reproducir un vídeo en YouTube.
5. Debe responder cuándo se le pregunte por la hora actual.
6. Debe buscar cualquier información en Wikipedia.
7. Debe abrir la página de Google.
8. Debe enviar un mensaje de correo electrónico.
9. Debe tomar una foto

- **Procedimiento**
- Software a instalar:

  Python 3.8.6

  Se debe ingresar a la página oficial de Python por medio del enlace:

  https://www.python.org/downloads/

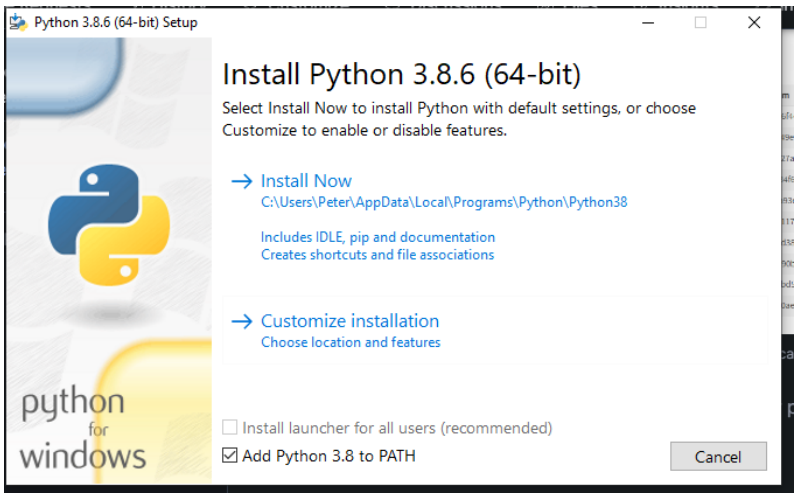
Abrir la opción “Downloads”

Debemos dirigirnos a la versión especifica y descargarla, por medio de la opción “download”

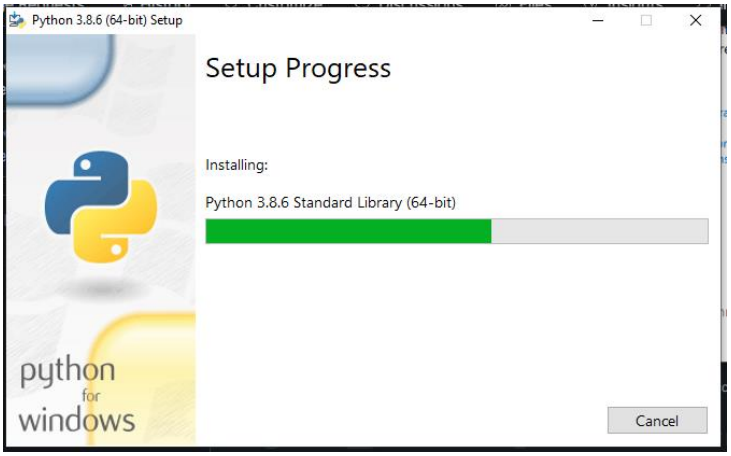
Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		ea132d6f449766623eee886966c7d41f	24377280	SIG
XZ compressed source tarball	Source release		69e73c49eeb1a853cefd26d18c9d069d	18233864	SIG
macOS 64-bit installer	macOS	for OS X 10.9 and later	68170127a953e7f12465c1798f0965b8	30464376	SIG
Windows help file	Windows		4403f334f6c05175cc5edf03f9cde7b4	8531919	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	5f95c5a93e2d8a5b077f406bc4dd96e7	8177848	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	2acba3117582c5177cdd28b91bbe9ac9	28076528	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	c9d599d3880dfbc08f394e4b7526bb9b	1365864	SIG
Windows x86 embeddable zip file	Windows		7b287a90b33c2a9be55fabc24a7febbb	7312114	SIG
Windows x86 executable installer	Windows		02cd63bd5b31e642fc3d5f07b3a4862a	26987416	SIG
Windows x86 web-based installer	Windows		acb0620aea46edc358dee0020078f228	1328200	SIG

Debemos ubicar el apartado de archivos y descargamos el instalador



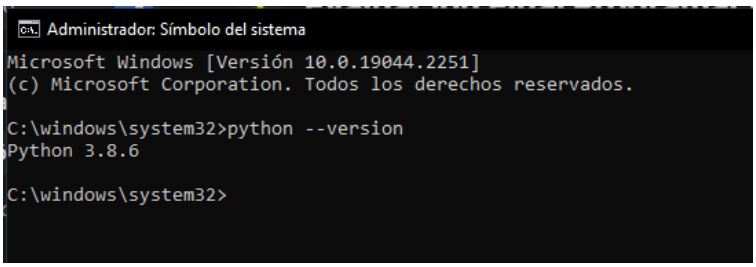
Procedemos a realizar la instalación, abriendo el instalador y seleccionando la opción "Add Python to PATH".



Debemos esperar a que termine de cargar y tendremos instalado Python.

Podemos confirmar la correcta instalación de la versión de python por medio de la consola de comandos o símbolo de sistema:

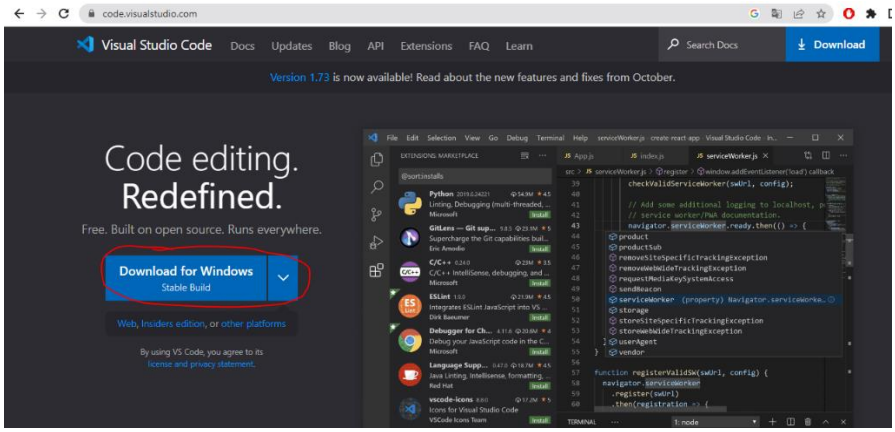
```
//  python --version
```

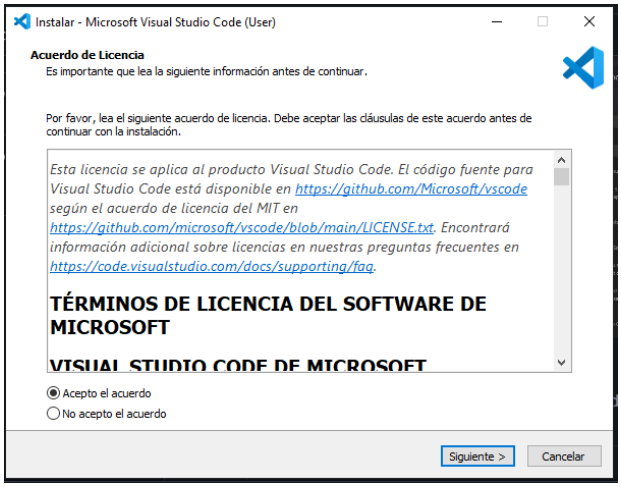


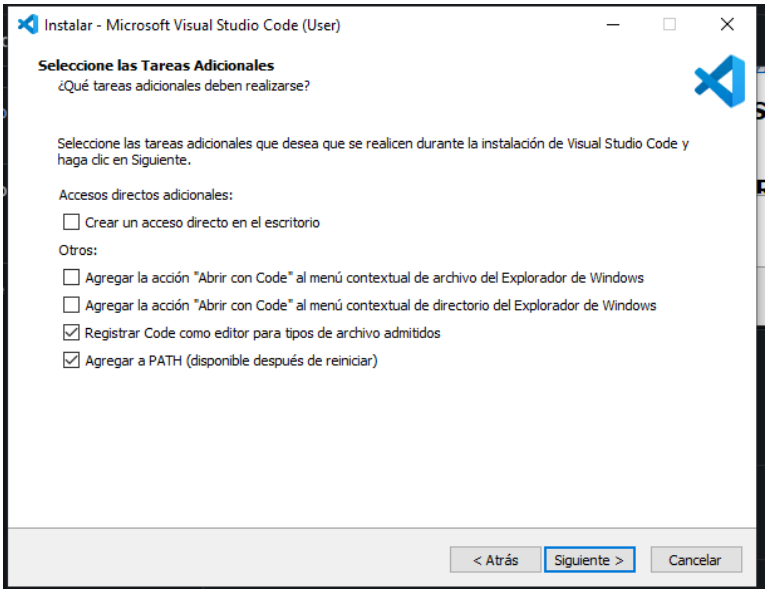
Visual Studio Code

Debemos dirigirnos al siguiente enlace para descargar el archivo:

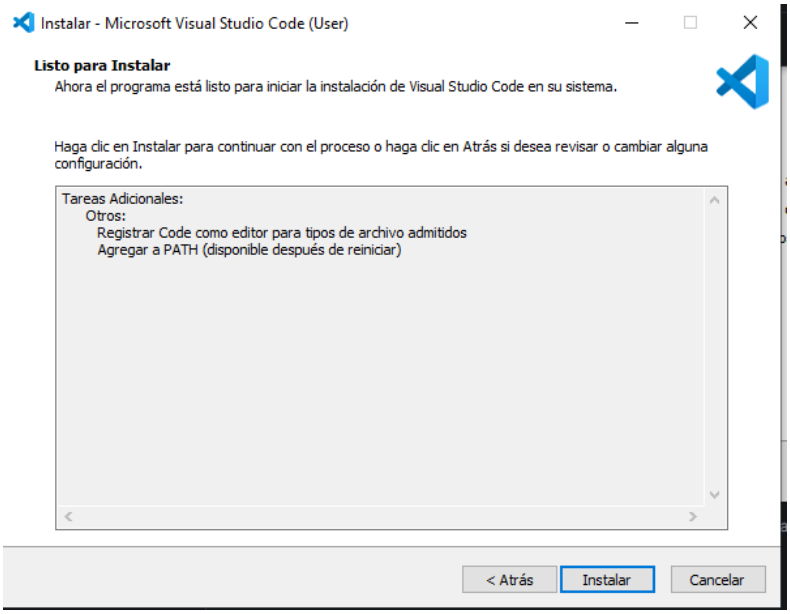
https://code.visualstudio.com/







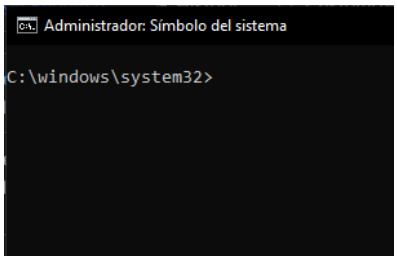
Se puede dejar las opciones predeterminadas o seleccionar las necesarias



Procedemos a instalar, una vez finalizado el proceso nuestro editor de código estará listo.

- **Instalación de librerías externas a utilizar:**

Para configurar nuestro asistente virtual, debemos instalar diferentes módulos y librerías externas, ingresando al modo administrador en la consola de comando o cmd, de esta manera la información instalada será utilizada de manera general para los proyectos que realicemos:



- **pyttsx3:**

  pyttsx es una librería multi-plataforma de texto a voz que es independiente a la plataforma donde se ejecute. La mayor ventaja de usar esta librería de conversión texto-a-voz es que trabaja sin conexión. Para instalar este módulo, escriba el siguiente comando en la terminal:

```
// pip install pyttsx3
```

- **SpeechRecognition:**

  Esto nos permite convertir audio en texto para su procesamiento. Para instalar este módulo, escribe el siguiente comando en la terminal:

```
// pip install SpeechRecognition
```

- **pywhatkit:**

  Esta es una librería de fácil uso que nos ayudará a interactuar con el navegador de forma más sencilla. Para instalar este módulo, ejecuta el siguiente comando en la terminal:

```
// pip install pywhatkit
```

- **wikipedia:**

  Vamos a usarla para buscar una gran variedad de información del sitio web de Wikipedia. Para instalar este módulo, escribe el comando a continuación en la terminal:

```
// pip install wikipedia
```

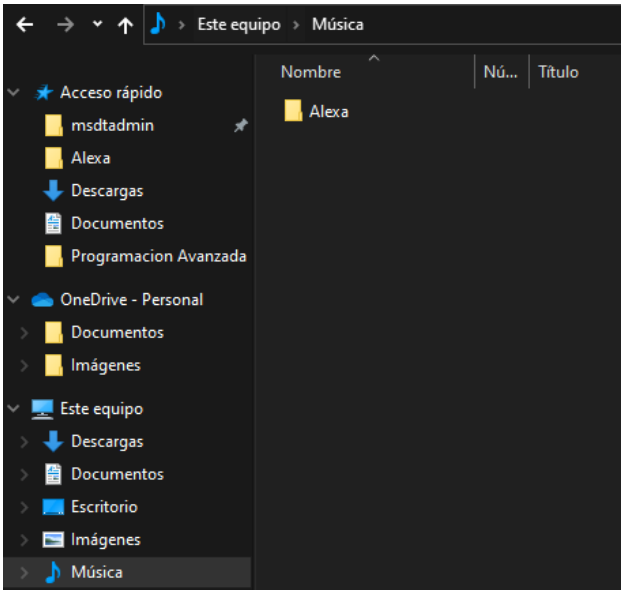
➢ OpenCV

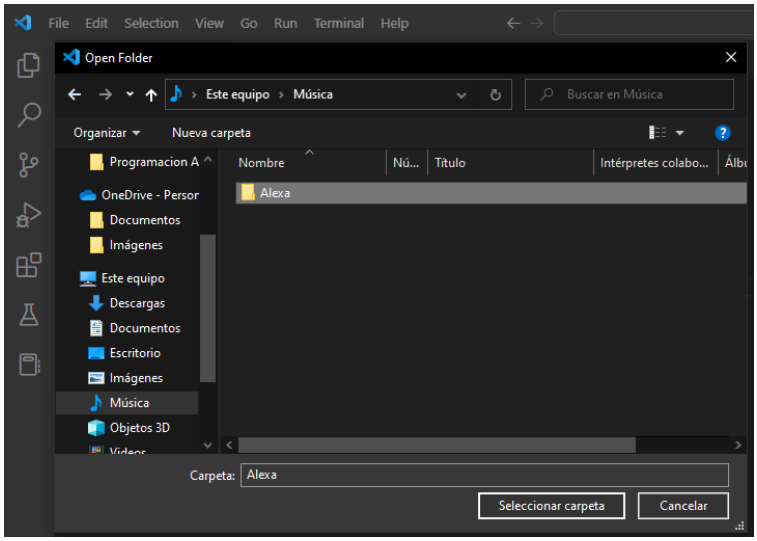
Esta librería nos permitirá realizar la acción para tomar fotos, haciendo uso de la cámara web que tiene nuestro computador portátil o de una cámara externa que le instalemos al ordenador, escribe el comando a continuación en la terminal:

```
// pip install opencv-contrib-python
```

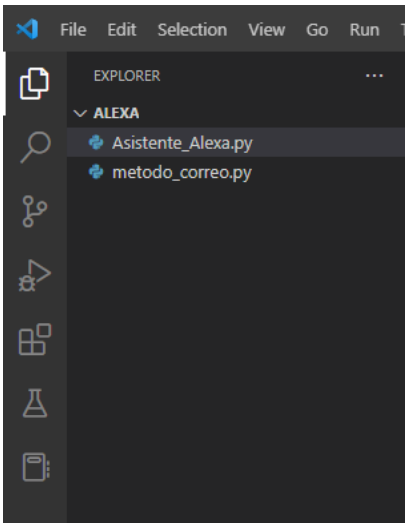
- **Creación del proyecto:**

Procedemos a crear una carpeta en cualquier lugar del explorador de archivos, la cual abriremos posteriormente mediante el editor de código "Visual Studio Code".





Seleccionamos la opción "File" y "Open Folder", debemos ubicar nuestra carpeta y seleccionar carpeta.



Procedemos a crear dos archivos de extensión ".py", el primero en donde estará nuestro código principal **"Asistente_Alexa",** y el segundo en donde crearemos el método de correo **"metodo_correo**".

Abrimos el primer archivo e importamos las librerías que acabamos de instalar:

```
// import speech_recognition as sr
// import pyttsx3, pywhatkit, wikipedia, datetime
// import subprocess as sp
// import cv2
```

La libreria "datetime" nos permitirá manejar los datos de la fecha actual del equipo y la librería "subprocess", nos permitirá realizar algunas acciones posteriormente.

Procedemos a importar de la librería "datetime" los módulos "date", "time" y "datetime", los cuales nos permitirán extraer datos específicos de la hora actual registrada en nuestro ordenador. además de la librería "random" importaremos el módulo "choice", que nos permitirá realizar selecciones aleatorias en un arreglo:

```
// from datetime import date, time, datetime
// from random import choice
```

A continuación, procedemos a crear una variable en donde se almacenará el nombre de nuestro asistente virtual, por el cual se llama y recibe ordenes:

```
// name = "alexa"
```

- **Creemos primero un motor de voz.**

```
engine = pyttsx3.init()

# Configurar velocidad
engine.setProperty('rate', 130)

# Configurar Volumen
engine.setProperty('volume', 2.0)


voices = engine.getProperty("voices")
engine.setProperty("voices", voices[0].id)
```

Analicemos el código anterior. Primero que todo, hemos inicializado un engine usando el módulo pyttsx3.init() en donde se establece la voz de nuestro asistente.

A continuación, vamos a configurar las propiedades de rate y volume del motor de voz usando **setProperty.**

Luego, podemos obtener las voces del motor usando **getProperty. voices** será una lista de voces disponibles en la librería que instalamos.

- **Habilitar la función de habla:**

```
def talk(text):
    ## Usado para decir cualquier texto que le sea entregado
    engine.say(text)
    engine.runAndWait()
```

En talk(), el motor dice oralmente cualquier texto que es entregado mediante el uso de say(). El uso de runAndWait(), se bloquea durante el bucle de eventos y vuelve cuando se borra la cola de comandos.

- **Habilitar la función de saludo:**

Esta función será usada para dar la bienvenida al usuario cada momento que el programa se ejecuta. En concordancia con el horario en tiempo real, saludará con Buenos Días, Buenas Tardes, o Buenas Noches al usuario.

```
def saludo():
    # Saluda al usuario de acuerdo a la hora actual

    hour = datetime.now().hour
    if (hour >= 1) and (hour < 12):
        print(f"Buenos días apreciado usuario")
        talk(f"Buenos días apreciado usuario")

    elif (hour >= 12) and (hour < 18):
        print(f"Buenas tardes apreciado usuario")
        talk(f"Buenas tardes apreciado usuario")
    elif (hour >= 18) and (hour < 24):
        print(f"Buenas noches apreciado usuario")
        talk(f"Buenas noches apreciado usuario")

    print(f"Mi nombre es {name}., como puedo ayudarte?")
    talk(f"Mi nombre es {name}., como puedo ayudarte?")
```

En primer lugar, consideramos la hora actual, en la situación que la hora actual sea las 10:35 AM, la hora será las 10. Si el valor de la hora está entre 1 AM y 12 PM, deseará los *Buenos Días* al usuario. Si el valor está entre 12 AM y 6 PM, le deseará las *Buenas Tardes* al usuario y de la misma manera, si el valor está entre las 6 PM y las 12 PM, le deseará las *Buenas Noches*. Estamos ocupando **talk()** para hablarle al usuario y el comando para **print()** mostrar por pantalla la información que dice.

- **Método para obtener información del usuario**

Usamos esta función para poder tomar instrucciones o comandos por parte del usuario y también, poder reconocer el comando usando el módulo **speech_recognition.**

```
def listen():
    listener = sr.Recognizer()

    try:
        with sr.Microphone() as source:
            talk("Escuchando ....")

            listener.pause_threshold = 1
            print("Escuchando ....")
            listener.adjust_for_ambient_noise(source)
            pc = listener.listen(source)
            rec = listener.recognize_google(pc, language="es")
            rec = rec.lower()
            if name in rec:
                rec = rec.replace(name, "")


    except sr.UnknownValueError:
        print("No te entendí, intenta de nuevo ")
        pass
    return rec
```

Hemos importado el módulo “speech_recognition” como listener. La clase *Recognizer* dentro del módulo speech_recognition nos ayuda a reconocer el audio. El mismo módulo también tiene una clase *Microphone* que nos da acceso al micrófono del dispositivo. Así, con el micrófono como source, escuchamos el audio mediante el uso de listen () en la clase *Recognizer*.

Además, hemos fijado el valor de pause_threshold a 1, para que no compile, aunque hagamos una pausa de 1 segundo mientras hablamos. También, por medio del comando *"listener.adjust_for_ambient_noise(source)"* ajustamos el micrófono al sonido ambiente y reconozca mejor nuestra voz

Luego, usando recognize_google() desde la clase *Recognizer*, tratamos de reconocer el audio. El recognize_google() ejecuta un reconocimiento de voz en el audio que entregado, usando la **API de reconocimiento de voz de Google**.

Hemos fijado el lenguaje a **“es”,** el cual es el español que se habla en la España. Va a dar como resultado la transcripción del audio que no es más que una cadena. La almacenamos en una variable llamada “**pc”.**

Si no hablamos correctamente o no esperamos un segundo para poder comenzar a darle órdenes a "Alexa", nos mostrara un mensaje de que no nos entendió.

Procedemos a crear una lista en donde se almacenará algunas frases para que nuestro asistente diga antes de realizar la acción indicada:

```
opening_text = [
    "Genial, estoy en ello.",
    "Entendido, trabajaré en ello.",
    "Deme un segundo apreciado usuario.",
    "Permitame un momento, estoy en ello",
]
```

- **Funciones configuradas:**

Cada una de las funciones depende de una palabra clave para realizarse, a continuación, se mostrará cada una de las acciones a realizar.

➢ **Reproduce videos en YouTube**

Para reproducir videos en YouTube, usaremos *PyWhatKit, e*l cual ya hemos importado, el cual será reconocido por la palabra clave "reproduce" seguido del nombre del video a buscar.

```
if "reproduce" in rec:
            talk(choice(opening_text))
            music = rec.replace("reproduce", "").strip()
            pywhatkit.playonyt(music)
            talk(f"Reproduciendo {music}.")
```

PyWhatKit tiene un playonyt() que acepta un tema como elemento. Entonces busca dicho tema en YouTube y reproduce el video más apropiado.

➢ **Fecha actual**

Para indicar la fecha actual, utilizaremos la librería de "datetime", utilizando la fecha actual del ordenador, la palabra clave es "fecha actual".

```
elif "fecha actual" in rec:
            talk(choice(opening_text))
            tiempo = datetime.datetime.now()

            talk("La fecha actual es: "+ tiempo.strftime("%d/%m/%Y %H:%M:%S"))
```

Por medio del método "talk" nos indicará la fecha actual en el formato especificado "día, mes, año hora, minutos y segundos"

- **Búsqueda en Wikipedia**

Para buscar en Wikipedia, usaremos el módulo wikipedia que hemos instalado previamente.

```
elif "busca" in rec:
            buscar = rec.replace("busca", "")
            wikipedia.set_lang("es")
            wiki = wikipedia.summary(buscar, 1)
            print(buscar + ": "+ wiki)
            talk(wiki)
```

Dentro del módulo **wikipedia**, tenemos un **summary()** que acepta una variable buscar como elemento. Adicionalmente, podemos indicar el número de oraciones requeridas y el idioma en el cual realizará la búsqueda, y, simplemente mostrar el resultado.

- **Abrir la página de Google**

Para realizar esta acción debemos crear un diccionario en donde almacenaremos los navegadores disponibles y su dirección de enlace:

```
sites = {
    "navegador":"google.com",
}
```

Procedemos a configurar una palabra clave "abre" en donde tomara la siguiente palabra ingresada y la buscara en el diccionario para luego abrir el enlace ya establecido, mediante el subproceso "call".

```
elif "abre" in rec:
            talk(choice(opening_text))
            for site in sites:
                if site in rec:
                    sp.call(f"start chrome.exe {sites[site]}", shell=True)
                    talk(f"Abriendo {site}")
```

- **Enviar un mensaje de correo electrónico**

Para esta función haremos uso del segundo archivo que creamos anteriormente "metodo_correo", en donde importaremos la librería "smtplib" y "ssl", los cuales permitirán asignar un método seguro o esa capa de seguridad para nuestro envió de mensajes. También importaremos el módulo "email.message" de la librería "EmailMessage".

```
import smtplib
import ssl
from email.message import EmailMessage

# Se define los datos del correo de origen, que envia el mensaje
email_sender = 'pruebapython0@gmail.com'
# Enlace de youtube en donde explican como obtener la contraseña
# https://www.youtube.com/watch?v=DDVpKvJXRz8&ab_channel=ThePyCoach
email_password = 'bbbezeogdykfzfqi'


# metodo para enviar un mensaje a un correo receptor
def send_email(email_receiver, subject, body):

    em = EmailMessage()
    em['From'] = email_sender
    em['To'] = email_receiver
    em['Subject'] = subject
    em.set_content(body)

    # Se agrega SSL (capa de seguridad)
    context = ssl.create_default_context()

    # Logeo seguro en el correo y envio del mensaje
    with smtplib.SMTP_SSL('smtp.gmail.com', 465, context=context) as smtp:
        smtp.login(email_sender, email_password)
        smtp.sendmail(email_sender, email_receiver, em.as_string())
```

El correo indicado es uno creado unicamente para este ejemplo, la contraseña se puede obtener mediante el procedimiento del siguiente video:

https://www.youtube.com/watch?v=DDVpKvJXRz8&ab_channel=ThePyCoach

Procedemos a importar este método creado en nuestro archivo principal, mediante el siguiente código:

```
from metodo_correo import send_email
```

Mediante este método, vamos a solicitar por consola al usuario los datos a utilizar para el correo de destino, y por medio de voz se indicará el mensaje y asunto del correo, haciendo uso de la palabra clave "enviar un correo":

```
elif "enviar un correo" in rec:
            talk("A que correo desea enviar el mensaje? Por favor ingreselo en la con
            email_receiver = input("Dirección de correo destino: ")
            talk("Por favor indique el asunto ")
            subject = listen().lower()
            talk("Por favor indique el mensaje a enviar: ")
            body = listen().lower()
            send_email(email_receiver, subject, body)
            talk("He enviado el correo, apreciado usuario.")
```

- **Función para tomar una foto:**

Primero importaremos la librería de OpenCV "cv2", la cual utilizaremos para abrir una ventana en donde se utilizará la cámara web configurada en el ordenador como fuente u origen y mediante la letra "a", tomará una foto y la guardará en la misma carpeta del proyecto con el formato "png" y mediante la letra "q", finalizará la ventana. La palabra clave configurada es "tómame una foto".

```
elif "tómame una foto" in rec:
            talk(choice(opening_text))
            talk("Abriendo Cámara")
            cam = cv2.VideoCapture(0)
            img_counter = 0

            while True:
                ret,frame = cam.read()
                cv2.imshow("video", frame)
                if not ret:
                    break
                if cv2.waitKey(1) & 0xFF == ord('a'):
                    img_name = "imagen_{}.png".format(img_counter)
                    cv2.imwrite(img_name, frame)
                    print("{} written!".format(img_name))
                    img_counter += 1
                if cv2.waitKey(2) & 0xFF == ord('q'):
                    break

            cam.release()
            cv2.destroyAllWindows()
```

- **Función para cerrar el asistente virtual**

Palabra clave "Salir" o "Detener", nos permitirá finalizar el proceso y nos dirá una frase de despedida:

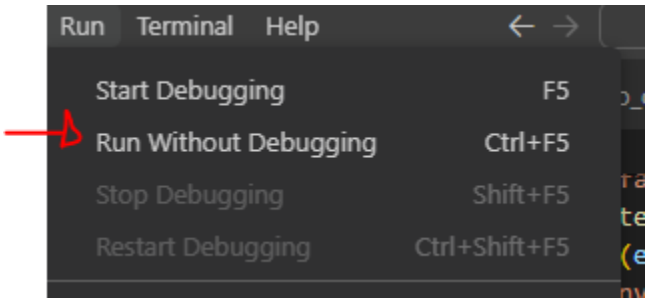
```
elif "salir" or "detener" in rec:
          hora = datetime.datetime.now().hour
          if hora >= 18 and hora < 24:
              talk("Que tenga una buena noche, cuidese!")
          else:
              talk('Que tenga un buen día, vuelva pronto')
      exit()
```

- **Comando principal**

Entonces, dentro del comando principal, la primera cosa que se hace es dar la bienvenida al usuario usando la función **saludar ().**

```
if __name__ == "__main__":
    saludo()
    while True:
        run_alexa()
```

Para poder ejecutar el programa tenemos dos opciones, la primera es ubicar el archivo creado desde la consola de comandos, y la segunda es reproducir el algoritmo mediante el aplicativo de Visual Studio Code:



Comando:

```
python Asistente_Alexa.py
```

Enlace de repositorio en github donde se encuentran los archivos finales de código:

https://github.com/PeterSalamanca/Evaluacion_final.git

Enlace del video explicativo:

https://youtu.be/2rRTaW3R7Pc

Enlace de la pagina en donde se publica este manual:

https://peter-salamanca.gitbook.io/valuacion-final-programacion-avanzada/

4. Conclusiones

Por medio del desarrollo de la presente actividad, puedo deducir que el avance de la tecnología y el aumento de conocimientos en esta área, nos ha permitido como humanidad automatizar procesos de la vida cotidiana, cada vez reduciendo en mayor medida la interacción del usuario y el ordenador, para que, mediante un comando de voz, se realice todo el procedimiento, sin que el usuario ingrese o realice cada uno de los pasos el mismo.

Es interesante como un único símbolo puede generar un conflicto al momento de ejecutar un programa, causando fallas en la compilación del mismo, para evitar estos tipos de error, siempre se puede optimizar un código, reduciendo los riesgos y acelerando el proceso de compilación. En este ejemplo realizado, esta en una fase de prueba, por lo cual aún existe gran espacio para las optimizaciones.

5. Referencias Bibliográficas

Hinojosa Gutiérrez, Á. (2015). Python paso a paso. RA-MA Editorial. https://elibro.net/es/lc/usta/titulos/107213

El Libro De Python. Tomado de: https://ellibrodepython.com/

Chacon, Scott, and Ben Straub. Pro Git, Apress L. P., 2014. ProQuest Ebook Central, https://ebookcentral.proquest.com/lib/bibliotecausta-ebooks/detail.action?docID=6422698.

Sneeringer, Luke. Professional Python, John Wiley & Sons, Incorporated, 2015. ProQuest Ebook Central, https://ebookcentral.proquest.com/lib/bibliotecaustaebooks/detail.action?docID=4187169.