UNIVERSIDAD NACIONAL DE TRUJILLO

Facultad de Ingeniería
Programa de Ingeniería Mecatrónica

LABORATORIO N° 8

"ARCHIVOS - MÓDULOS - LIBRERÍAS DIVERSAS"

DESARROLLO DE GUIA DE LABORATORIO

PROGRAMACIÓN I

ESTUDIANTE(S) :
SÁNCHEZ ROJAS JHONATAN ARTEMIO
VALDIVIEZO JIMÉNEZ VÍCTOR JAVIER
VIGO VILLAR CRISTHIAN AARON

DOCENTE :
ASTO RODRIGUEZ EMERSON MAXIMO

CICLO :
2022 I

Trujillo, Perú
2022

INDICE

RESUMEN

El presente informe de laboratorio supone un análisis específico del desarrollo de 4 actividades propuestas sobre programación. Las cuales piden la elaboración de códigos en Python para el desarrollo de cierto programa que se ejecute de la forma especificada en la actividad. La realización de estas actividades surge en torno a la práctica de un curso de Python, por lo que, en este caso, supone el uso de herramientas como los archivos y módulos. Y, de las ya conocidas formas básicas para la iteración, condicionales, excepciones, entre otros.

Finalmente, se pudo llegar a los correctos resultados pedidos en cada ejercicio, concluyendo en lo fácil que es la instalación de nuevos módulos/librerías para facilitar la creación de muchos más códigos, orientados para una infinidad de temas.

DESARROLLO DEL LABORATORIO

1.1. Desarrollo de la experiencia

a) Ejercicio 1

Explicación en el video:

https://drive.google.com/file/d/1VSRB81_g_gl4hI_eGF7di7EpgO7W7Y0H/view?usp=sharing

b) Ejercicio 2

Valdiviezo Jimenez, Victor Javier

- **Escribir un programa que consulte si uno quiere saber sobre algún tema en específico, y luego responda sobre lo consultado usando información de Wikipedia. El programa debe responder usando voz.**

✓ Empezamos importando los módulos que se van a necesitar para la realización del programa, dándonos cuenta, en el ítem, que serán necesarios los módulos de síntesis de voz, reconocimiento de audio y Wikipedia. Además, los inicializamos.

```
import pyttsx3
import speech_recognition
import wikipedia

engine = pyttsx3.init()
recognizer = speech_recognition.Recognizer()
```

✓ Definimos la función "sintesis(text)" que hará decir al programa el valor de "text" que se evalué en la función.

```
def sintesis(text):
 engine.say(text)
 engine.runAndWait()
```

✓ Definimos la función "reconocimiento ( )" que guardara la búsqueda que el usuario hará por voz en el valor "mensaje" y finalmente retornara este mensaje.

```
def reconocimiento():
    try:
        with speech_recognition.Microphone() as source:
            print("Escuchando... ")
            audio = recognizer.listen(source)
            mensaje = recognizer.recognize_google(audio,language="es")
            print(mensaje)
    except:
       pass
    return mensaje
```

- ✓ Finalmente, definimos la función “buscar_Wikipedia ( )” que pasara el valor de “mensaje” hacia el buscador de Wikipedia y arrojara el resultado mediante voz, usando la función “sintesis ( )”. Adicionalmente, agregamos un try except dentro de la función, para evitar la paralización del programa en caso no se reconozca la búsqueda.

```
def buscar_wikipedia():
    try:
        sintesis("Hola, soy tu asistente de wikipedia, di lo que estas
buscando")
        mensaje = reconocimiento()
        busqueda = mensaje
        wikipedia.set_lang("es")
        respuesta = wikipedia.summary(busqueda, sentences = 1)
        sintesis(respuesta)
    except:
        sintesis("¡No se encontraron resultados para tu busqueda!")

buscar_wikipedia()
```

c) Ejercicio 3

Valdiviezo Jimenez, Victor Javier

- **Escribir un programa que lea un archivo csv e imprima la información de forma agradable en la consola.**

✓ Para esto descargamos el módulo de pandas, que nos permite leer el archivo en forma de una tabla con encabezados.

```
(prograunt) C:\Users\SOPORTE\Documents\CICLO III - UNT\PROGRAMACION I\VISUAL ESTUDIO>pip install pandas
Collecting pandas
  Downloading pandas-1.4.3-cp310-cp310-win_amd64.whl (10.5 MB)
     |████████████████████████████████| 10.5 MB 3.3 MB/s
Collecting python-dateutil>=2.8.1
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
     |████████████████████████████████| 247 kB 6.8 MB/s
Collecting pytz>=2020.1
```

✓ Seleccionamos el archivo csv con el que se va a trabajar. En este caso, el archivo elegido es "Iris.csv", compuesto por 151 filas y 6 columnas. Significando la primera fila los encabezados de las columnas.

Iris.csv

```
1   Id,SepalLengthCm,SepalWidthCm,PetalLengthCm,PetalWidthCm,Species
2   1,5.1,3.5,1.4,0.2,Iris-setosa
3   2,4.9,3.0,1.4,0.2,Iris-setosa
4   3,4.7,3.2,1.3,0.2,Iris-setosa
5   4,4.6,3.1,1.5,0.2,Iris-setosa
6   5,5.0,3.6,1.4,0.2,Iris-setosa
7   6,5.4,3.9,1.7,0.4,Iris-setosa
```

...

```
143   142,6.9,3.1,5.1,2.3,Iris-virginica
144   143,5.8,2.7,5.1,1.9,Iris-virginica
145   144,6.8,3.2,5.9,2.3,Iris-virginica
146   145,6.7,3.3,5.7,2.5,Iris-virginica
147   146,6.7,3.0,5.2,2.3,Iris-virginica
148   147,6.3,2.5,5.0,1.9,Iris-virginica
149   148,6.5,3.0,5.2,2.0,Iris-virginica
150   149,6.2,3.4,5.4,2.3,Iris-virginica
151   150,5.9,3.0,5.1,1.8,Iris-virginica
```

✓ Empezamos con la escritura del codigo, importando el archivo y el modulo pandas.

```
import csv
import pandas as pd
```

✓ Inicializamos el modulo panda con la funcion “read_csv” y agregamos el comando “index_col” para cambiar el idenficador propio de la dataframe a la columna “Id”.

```
df = pd.read_csv("Iris.csv", index_col = "Id")
```

✓ Finalmente imprimimos la funcion.

```
print(df)
```

✓ Cabe resaltar que en casos donde las filas son bastantes, la dataframe se imprimira con las 5 primeras filas y las 5 ultimas. Esto es suficiente para el objetivo del item, que nos pide que la lectura sea agradable. Sin embargo, existen funciones para poder ver mas filas como es el “head(n)” o el “tail(n)”, la primera hace un llamado a las n primeras filas, mientras que la segunda a las n ultimas filas.

```
print(df.head(10))
print(df.tail(10))
```

d) Ejercicio 4

Sánchez Rojas, Jhonatan Artemio

- **Escribir un programa en el que el usuario pueda establecer una hora y minuto del día, y sea alertado mediante una notificación o sonido 5 minutos antes.**

Básicamente esto puede interpretarse como un temporizador.

✓ Para esto primero descargamos la librería win10toast, para instalarlo en nuestro entorno virtual solamente debemos colocar lo siguiente en nuestra consola: pip install win10toast.

✓ También se importará el módulo time y el módulo ToastNotifier (este pertenece a la librería win10toast).

```
import time
from win10toast import ToastNotifier
```

✓ Se creará la función que nos permitirá establecer las horas, minutos y segundos para que nos salga la notificación:

```
def conftiempo():
    hora = int(input("Horas :"))
    minutos = int(input("Minutos :"))
    segundos = int(input("Segundos :"))
    tiempo_total = segundos+((minutos*60)-300)+(hora*3600)
    return tiempo_total
```

Se puede apreciar en el código que todo está en base a segundos, además a los minutos se le resta 300 segundos, dado que, se desea recibir la notificación 5 minutos antes.

✓ Se creará una función en el cual se usará el ToastNotifier, el cual nos permitirá mostrar la notificación en Windows.

```
def notif():
    notificacion = ToastNotifier()
    notificacion.show_toast("Reunión virtual en 5 minutos")
    return 0
```

El método .show_toast nos permite colocar de manera escrita lo que queremos que aparezca en la notificación. En este caso, se colocó: “Reunión virtual en 5 minutos”.

✓ Se define otra función para lo que es el intervalo de tiempo, para lo cual se hizo uso del while y el módulo time. El método .sleep nos permite suspender la ejecución del código en el tiempo que se establece con anterioridad.

```
def tiempoinicio(intervalo_tiempo):
    while True:
        time.sleep(intervalo_tiempo)
        notif()
```

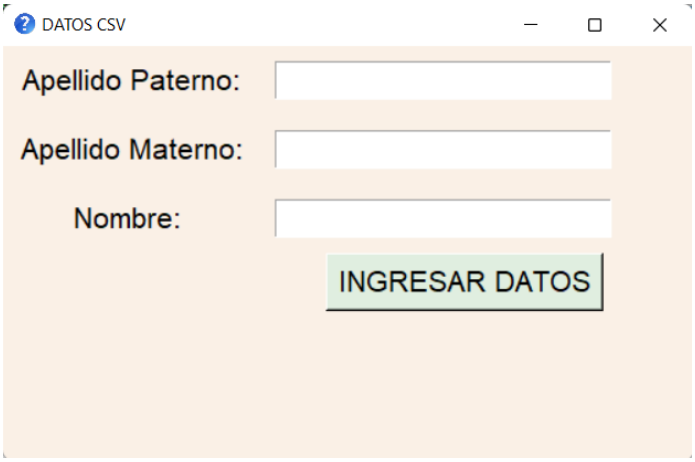
✓ Por último, se usará if para establecer la siguiente condición:

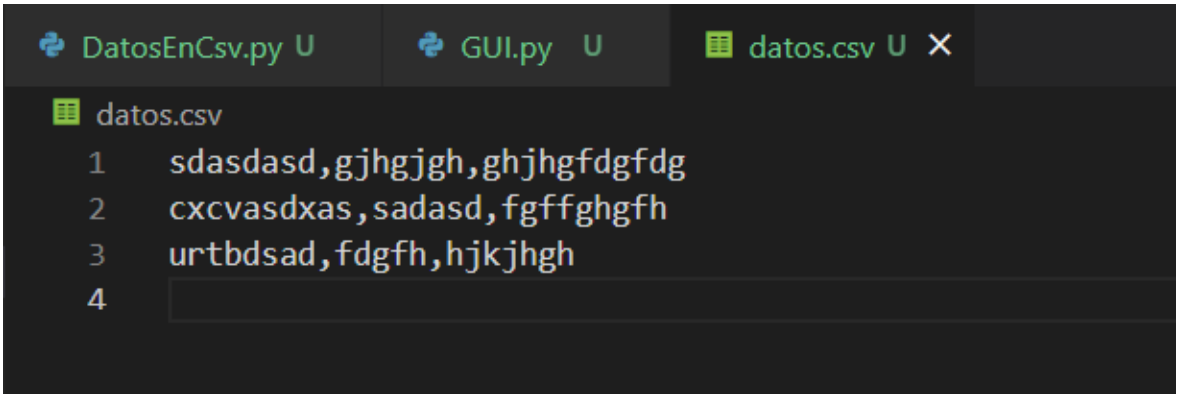
```
if __name__ == '__main__':
    intervalo_tiempo = conftiempo()
    tiempoinicio(intervalo_tiempo)
```

1.2. Resultados de la experiencia

Link del repositorio: https://github.com/VictorValdiviezo/Laboratorio-8-Grupo1-PI-UNT_2022.git

a) Ejercicio 1





b) Ejercicio 2

Como el programa usa el micrófono para poder ejecutarse, ocurrieron muchos problemas para poder grabar la ejecución de este mismo; puesto que, las grabadoras con las que se intentó usaban también el micrófono para poder grabar lo que se diga y al momento de ejecutar el código, el micrófono de desconectaba. Por consiguiente, para comprobar la correcta ejecución del programa se recomienda copiar el código completo en su lector de Python de preferencia y ejecutarlo. Además, cabe recordar que para la correcta ejecución es necesario la instalación de los módulos: síntesis de voz, reconocimiento de audio y Wikipedia.

```
import pyttsx3
import speech_recognition
import wikipedia
engine = pyttsx3.init()
recognizer = speech_recognition.Recognizer()
def sintesis(text):
 engine.say(text)
 engine.runAndWait()
def reconocimiento():
    try:
        with speech_recognition.Microphone() as source:
            print("Escuchando... ")
            audio = recognizer.listen(source)
            mensaje = recognizer.recognize_google(audio,language="es")
            print(mensaje)
    except:
       pass
    return mensaje
def buscar_wikipedia():
    try:
        sintesis("Hola, soy tu asistente de wikipedia, di lo que estas
buscando")
        mensaje = reconocimiento()
        busqueda = mensaje
        wikipedia.set_lang("es")
        respuesta = wikipedia.summary(busqueda, sentences = 1)
        sintesis(respuesta)
```

```
    except:
        sintesis("¡No se encontraron resultados para tu busqueda!")

buscar_wikipedia()
```

c) Ejercicio 3

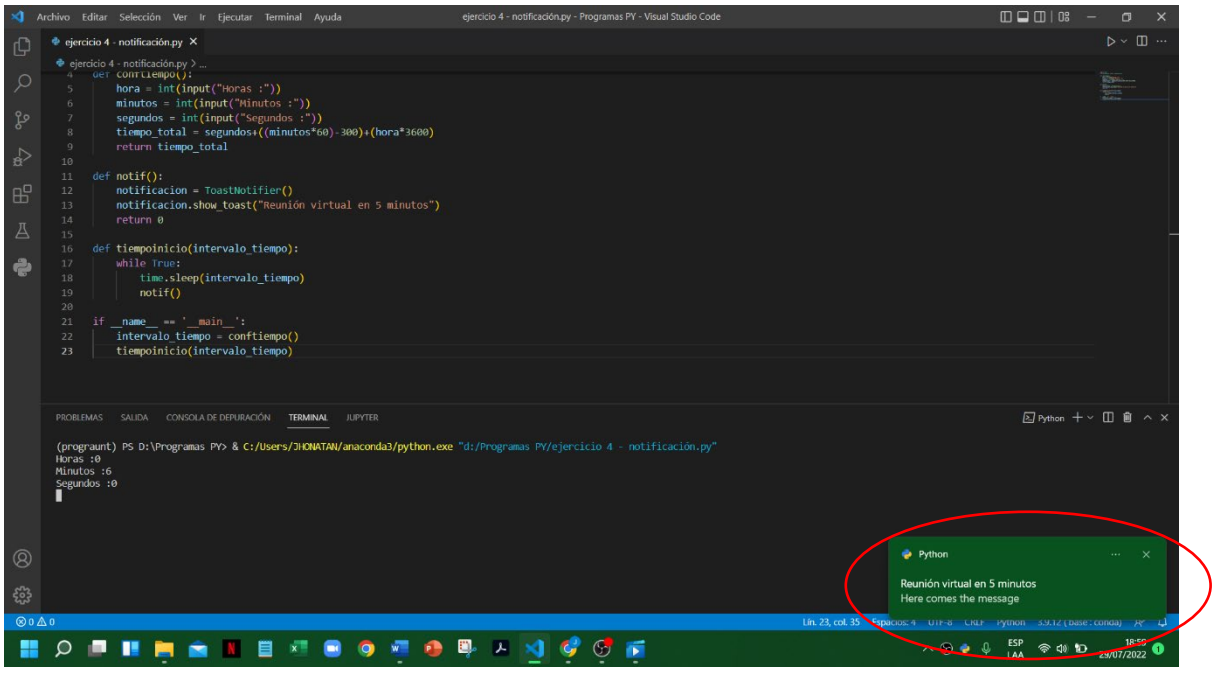
```
(prograunt) C:\Users\SOPORTE\Documents\CICLO III - UNT\PROGRAMACION I\VISUAL ESTUDIO>python lector_archivos_csv.py
     SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm         Species
Id
1              5.1           3.5            1.4           0.2     Iris-setosa
2              4.9           3.0            1.4           0.2     Iris-setosa
3              4.7           3.2            1.3           0.2     Iris-setosa
4              4.6           3.1            1.5           0.2     Iris-setosa
5              5.0           3.6            1.4           0.2     Iris-setosa
..             ...           ...            ...           ...             ...
146            6.7           3.0            5.2           2.3  Iris-virginica
147            6.3           2.5            5.0           1.9  Iris-virginica
148            6.5           3.0            5.2           2.0  Iris-virginica
149            6.2           3.4            5.4           2.3  Iris-virginica
150            5.9           3.0            5.1           1.8  Iris-virginica

[150 rows x 5 columns]
```

d) Ejercicio 4

✓ Primero ejecutamos el código y colocamos la cantidad de horas, minutos y segundos. Para este ejemplo colocamos 6 minutos para que la espera de la notificación no sea muy larga. Al haber colocador 6 minutos, la notificación aparecerá al minuto.

```
(prograunt) PS D:\Programas PY> & C:/Users/JHONATAN/anaconda3/python.exe "d:/Programas PY/Ejercicio 4 - Notificación en Windows.py"
Horas :0
Minutos :6
Segundos :0
```

✓ En la esquina inferior derecha veremos la notificación, la cual dice: “Reunión virtual en 5 minutos”



Link del video de demostración:
https://drive.google.com/file/d/1MnK5HyiFh0Q59_KAyY5KIWHnSP5QTOxg/view?usp=sharing

1.3. Desarrollo de test de comprobación

I. Explique que es el archivo __init__.py

__init__ es una convención en python, la cual siempre se debe crear en cada carpeta de donde queramos importar funciones, datos, o variables que se encuentran en un script diferente al que tenemos, además este se ejecuta implícitamente. __init__.py puede llegar a contener el mismo código Python que contiene cualquier otro módulo.

II. Explique qué significa la w, r, a, t, b, en el manejo de archivos.

w: Abre el archivo para escribir nuevo contenido en reemplazo al anterior.
r: Abre el archivo como modo de lectura.
a: Abre el archivo para escribir nuevo contenido, pero a diferencia del "w", este contenido es adicional al contenido anterior.
t: Guarda un archivo en formato texto con valor predeterminado
b: Guarda archivo en forma de binario.

III. Investigue y explique que son los decoradores en python.

La función básica de los decoradores en Python es alterar el funcionamiento de determinadas piezas del código, tales como una función o una clase, sin la necesidad de alterar su código base. (Guzmán, 2018)
En síntesis, los decoradores añaden nuevos comportamientos en las funciones o clases.

1.4. Recomendaciones

- Utilizar “a” en vez de “w” debido a que “w” sobrescribe los datos en el mismo dato y no genera más en el archivo csv.
- Para trabajar con el micrófono a la hora de la ejecución del asistente virtual, es muy importante que solo exista un micrófono conectado a la computadora, puesto que, al haber más, Python no reconocerá ninguno. En este caso, se tenía conectado el micrófono de audífonos y también el de la cámara, y al ejecutar el programa, ocurría que el programa se mantenía escuchando sin fin la búsqueda ya que no encontraba respuesta nuestra porque no se aceptaba ningún micrófono. Para solucionar esto, desconectamos la cámara (contenía micrófono incluido) y el programa ya reconocía nuestra búsqueda dicha por voz y respondía.
- Llevar un orden a la hora de escribir el código facilita el reconocimiento de cada parte.

1.5. Conclusiones

- La facilidad del lenguaje Python para la instalación de módulos o librerías, supone una gran multiplicación en la variedad de códigos que se pueden crear usando este lenguaje. Por consiguiente, aborda más campos y universaliza el desarrollo de la programación.
- Tener el conocimiento de como funcionan estos módulos y librerías nos abren las puertas para poder crear códigos que posean una mayor complejidad y resultados más eficaces.

REFERENCIAS BIBLIOGRÁFICAS

Guzmán, D. (06 de Agosto de 2018). *Python: ¿Qué es un decorador?* Obtenido de Codingornot: https://codingornot.com/python-que-es-un-decorador

ANEXOS

1.1 Código completo del ejercicio 1:

DatosEnCsv.py:

```
from GUI import GUI
from tkinter import messagebox

try:

    GUI()

except:

    titulo="ERROR"
    mensaje="No se pudo ejecutar"
    messagebox.showerror(titulo,mensaje)
```

GUI.py

```
import tkinter as tk
import csv
import pyttsx3

def GUI():

    engine1 = pyttsx3.init()
    engine1.say("¿Que deseas realizar? ")
    engine1.runAndWait()

    preguntar = tk.Tk()
    preguntar.title("CSV")
    preguntar.geometry("240x250+700+280")
    preguntar.resizable(width=False, height=False)
    preguntar.iconbitmap("pregunta.ico")
    preguntar.config(background="khaki1")

    Label=tk.Label(preguntar,
    text="Elija una opción",
```

```
    background="khaki1",
    font=("Arial", 18, "bold"),
    foreground="black"
    )

    Label.pack(pady=25)

    def entrys_llenar():
        ventana2=tk.Tk()
        ventana2.title("DATOS CSV")
        ventana2.geometry("500x300+500+250")
        ventana2.resizable(width=True, height=True)
        ventana2.iconbitmap("pregunta.ico")
        ventana2.config(background="linen")


        Label2=tk.Label(ventana2,
            text="Apellido Paterno:",
            font=("Arial", 16),
            background="linen",
            foreground="black"
        )
        Label2.grid(column=0, row=0, padx=10, pady=10)

        Label3=tk.Label(ventana2,
            text="Apellido Materno:",
            font=("Arial", 16),
            background="linen",
            foreground="black"
        )
        Label3.grid(column=0, row=1, padx=10, pady=10)

        Label4=tk.Label(ventana2,
            text="Nombre: ",
            font=("Arial", 16),
            background="linen",
            foreground="black"
        )
        Label4.grid(column=0, row=2, padx=10, pady=10)

        Entry2=tk.Entry(ventana2,
```

```
            font=("Arial", 16),
            foreground="black"
        )
        Entry2.grid(column=1, row=0, padx=10, pady=10,columnspan=2)

        Entry3=tk.Entry(ventana2,
            font=("Arial", 16),
            foreground="black"
        )
        Entry3.grid(column=1, row=1, padx=10, pady=10,columnspan=2)

        Entry4=tk.Entry(ventana2,
            font=("Arial", 16),
            foreground="black"
        )
        Entry4.grid(column=1, row=2, padx=10, pady=10,columnspan=2)

        def llenar():
            lista=[]

            a=Entry2.get()
            b=Entry3.get()
            c=Entry4.get()

            lista.append(a)
            lista.append(b)
            lista.append(c)

            writer=csv.writer(open("datos.csv", "a", newline=""))
            writer.writerow(lista)

            lista.clear()
            engine2 = pyttsx3.init()
            engine2.say("¿Desea agregar a otra persona? sino, solo cierre la
ventana y visualice los datos ")
            engine2.runAndWait()


        botoningresar=tk.Button(ventana2,
            text="INGRESAR DATOS",
            font=("Arial", 16),
            background="honeydew2",
```

```
            activebackground="honeydew3",
            command=llenar
        )

        botoningresar.grid(column=2, row=4)

        ventana2.mainloop()


    boton1 = tk.Button(preguntar,
    text = "Registrar mis datos",
    font = ("Times", 14),
    background="orangered2",
    activebackground="orangered4",
    command=entrys_llenar
    )

    boton1.pack(pady=10)

    def consultar():

        reader=csv.reader(open("datos.csv", "r"))
        for row in reader:
            print("Apellido Paterno: {0}, Apellido Materno: {1}, Nombre:
{2}".format(row[0], row[1], row[2]))


    boton2 = tk.Button(preguntar,
    text = "Ver todos los datos",
    font = ("Times", 14),
    background="green2",
    activebackground="green4",
    command=consultar
    )

    boton2.pack(pady=10)


    preguntar.mainloop()
```

1.2 Código completo del ejercicio 2:

```
import pyttsx3
import speech_recognition
import wikipedia

engine = pyttsx3.init()
recognizer = speech_recognition.Recognizer()

def sintesis(text):
 engine.say(text)
 engine.runAndWait()

def reconocimiento():
    try:
        with speech_recognition.Microphone() as source:
            print("Escuchando... ")
            audio = recognizer.listen(source)
            mensaje = recognizer.recognize_google(audio,language="es")
            print(mensaje)
    except:
       pass
    return mensaje

def buscar_wikipedia():
    try:
        sintesis("Hola, soy tu asistente de wikipedia, di lo que estas
buscando")
        mensaje = reconocimiento()
        busqueda = mensaje
        wikipedia.set_lang("es")
        respuesta = wikipedia.summary(busqueda, sentences = 1)
        sintesis(respuesta)
    except:
        sintesis("¡No se encontraron resultados para tu busqueda!")

buscar_wikipedia()
```

1.3 Código completo del ejercicio 3:

```
import csv
import pandas as pd

df = pd.read_csv("Iris.csv", index_col = "Id")

print(df)
```

1.4 Código completo del ejercicio 4:

```
import time
from win10toast import ToastNotifier

def conftiempo():
    hora = int(input("Horas :"))
    minutos = int(input("Minutos :"))
    segundos = int(input("Segundos :"))
    tiempo_total = segundos+((minutos*60)-300)+(hora*3600)
    return tiempo_total

def notif():
    notificacion = ToastNotifier()
    notificacion.show_toast("Reunión virtual en 5 minutos")
    return 0

def tiempoinicio(intervalo_tiempo):
    while True:
        time.sleep(intervalo_tiempo)
        notif()

if __name__ == '__main__':
    intervalo_tiempo = conftiempo()
    tiempoinicio(intervalo_tiempo)
```