Python en el ADN

Tablero de ajedrez

In [1]:
```
1
```

In [2]:
```
1
```

```python
# Tablero de ajedrez.


from PIL import Image, ImageDraw # Importamos parte de la librería PIL para dibujar
w, h = 800, 800 # Inicializamos la altura y ancho del tablero

# ----------------------
img = Image.new("RGB", (w, h)) # Creamos una imagen vacía con la altura y ancho anteriores
dib = ImageDraw.Draw(img) # Creamos una variable que guardará el dibujo del tablero

# ----------------------
for x in range(8):
    for i in range (8):
    # calcula coordenadas, cuadrados de 100 x 100 pixels

        col = i*100
        fila = x *100
        shape = [(col, fila), (col + 100, fila + 100)] # Cramos la cuadrícula aumentando las coordenadas
                                                       # cogiendo los valores del rango

        # Hacemos los condicionales por fila y columna para saber el color a rellenar
        if x%2 != 0 : # Filas impares

            if i%2 != 0 : color = 'white' # Casillas impares
            else : color = 'black' # Casillas pares

        else:

            if i%2 != 0 : color = 'black' # Casillas impares de las filas pares
            else : color = 'white' # Casillas pares de las filas pares


        dib.rectangle(shape, fill = color , outline ="black") # Dibujamos el rectangulo con todos sus cuadrados pintados


img.show() # Mostramos por pantalla el rectángulo obtenido
```

In [3]:
```
1
```

In [4]:
```
1
```

In [5]:
```
1
```

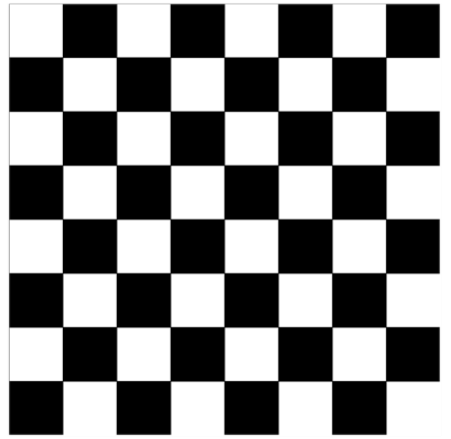
In [6]:
```
1
```

In [7]:
```
1
```

In [8]:
```
1
```

In [9]:
```
1
```

In [10]:
```
1
```



Jugamos?

In [1]:
```
1
```

In [2]:
```
1
```

In [3]:
```
1
```

In [4]:
```
1
```

In [5]:
```
1
```

In [6]:
```
1
```

In [7]:
```
1
```

In [8]:
```
1
```

In [9]:
```
1
```

In [10]:
```
1
```

In [7]:
```
# Piedra, papel, tijera
import requests
import random
from IPython.display import Image, display, clear_output


img_opcions = Image(".\\img\\opcions.jpg", width = 140)
img_pedra = Image(".\\img\\pedra.jpg", width = 140)
img_paper = Image(".\\img\\paper.jpg", width = 140)
img_tisora = Image(".\\img\\tisora.jpg", width = 140)
#------------------------------Mostra les opcions i demana jugada
display(img_opcions)
jdor = input("Escoje y teclea: piedra / papel / tijera ?: ")
#-----------------------Calcula i mostra jugada del ordinador (pc)
pc = random.choice(["piedra", "papel", "tijera"])
print ("Yo juego : ", pc)
if pc == "piedra" :
    display(img_pedra)

elif pc == "papel":
    display(img_paper)
else:
    display(img_tisora)
#-------------------------------------Calcula i mostra resultat

if jdor == pc :
    print("Empatamos!!!")

else:
    if (jdor == "tijera") and (pc == "piedra") or (jdor == "piedra" and pc == "papel")\
    or (jdor == "papel" and pc == "tijera"):
        print("Pierdes :(")
    else:
        print("Tu ganas :)")

```

```
Escoje y teclea: piedra / papel / tijera ?: piedra
Yo juego :  tijera
```

```
Tu ganas :)
```

In [1]: 1

In [2]: 1

In [3]: 1

In [4]: 1

In [5]: 1

In [6]: 1

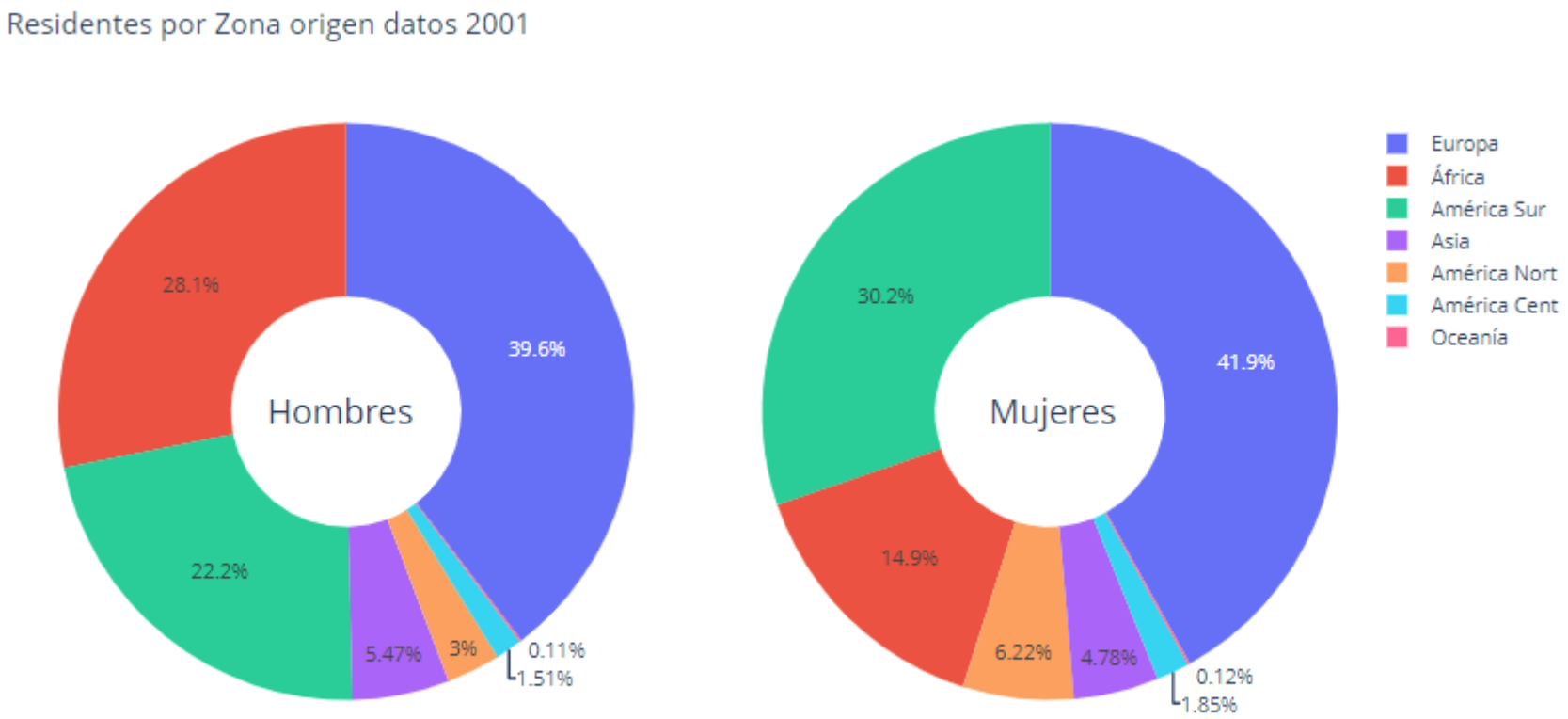
In [7]: 1

In [8]: 1

In [9]: 1

In [10]: 1

A dibujar!!!



Como se hizo

In [1]: 1

In [2]: 1

In [3]: 1

In [4]: 1

In [5]: 1

In [6]: 1

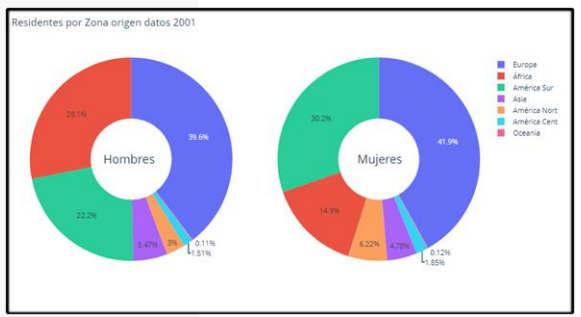
In [7]: 1

In [8]: 1

In [9]: 1

In [10]: 1

```
import plotly.graph_objects as go
import plotly.io as pio
from plotly.subplots import make_subplots
# --------------------------- Preparamos los Datos

# Creamos las listas con los valores
zona = ['Europa', 'África', 'América Nort', 'América Cent', 'América Sur','Asia', 'Oceanía']
hombres = [39.57, 28.11, 3, 1.51, 22.22, 5.47, 0.11]
mujeres = [41.9, 14.89, 6.22, 1.85, 30.23, 4.78, 0.12]


# Le decimos cuantos plots vamos a crear
fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'}, {'type':'domain'}]])

# Creamos el primer subplot con los datos para hombres por zona
fig.add_trace(go.Pie(labels=zona, values=hombres, name="Hombres"),
              1, 1)
# Creamos el segundo subplot con los datos para mujeres por zona
fig.add_trace(go.Pie(labels=zona, values=mujeres, name="Mujeres"),
               1, 2)

# Usamos hole para hacerlo como un donut
fig.update_traces(hole=.4, hoverinfo="label+percent")

# Le añadimos el titulo y anotaciones
fig.update_layout(
    title_text='Residentes por Zona origen datos 2001',
    # Add annotations in the center of the donut pies.
    annotations=[dict(text='Hombres', x=0.16, y=0.5, font_size=20, showarrow=False),
                 dict(text='Mujeres', x=0.83, y=0.5, font_size=20, showarrow=False)])

fig.show() # Imprimimos la gráfica
```



Sobrevivirás?

In [1]: 1

In [2]: 1

In [3]: 1

In [4]: 1

In [5]: 1

In [6]: 1

In [7]: 1

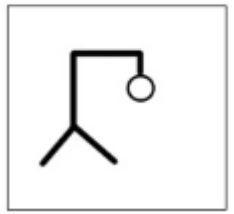
In [8]: 1

In [9]: 1

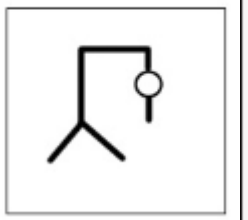
In [10]: 1

```python
import requests
from IPython.display import Image, display, clear_output
import random as r

palabras =["ciervo", "rinoceronte", "jirafa", "bufalo"]
letras_nok=[]
aciertos = 0
fallos = 0
ruta_imagen = "./img/penjat-1.jpg"

print ("Juego del Ahorcado")

# Escoge una palabra de la lista aleatoriamente
palabra = r.choice(palabras)
oculta = "-" * len(palabra)   # patron de guiones
print (oculta)

while palabra != oculta and fallos < 6 :

    letra= input ("letra: ")
    if letra in palabra :
        nueva_oculta = ""
        for i, l in enumerate (palabra):
            if l==letra :
                nueva_oculta += letra
            else:
                nueva_oculta += oculta[i]
        oculta = nueva_oculta
        print(oculta)
    else :
        if letra in letras_nok:
            print ("ya lo has dicho!")
        else:
            letras_nok.append(letra)
            fallos+=1
            rutaf = ruta_imagen.replace("1", str(fallos))
            imagen = Image(rutaf, width = 140)
            display(imagen)

if palabra == oculta :
    print ("Has ganado")
else :
    print ("Has perdido, la palabra era: ", palabra)
```

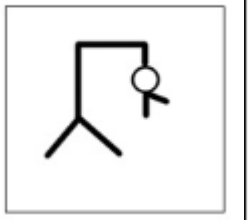
```
Juego del Ahorcado
------
letra: j
```



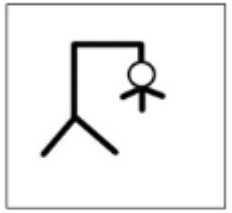
```
letra: h
```



```
letra: r
```



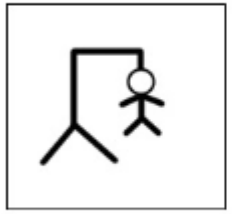
```
letra: i
```



```
letra: a
---a--
letra: l
---al-
```

```
letra: e
```

```
letra: s
```



```
Has perdido, la palabra era:  bufalo
```

Algo de clases

In [1]: 1

In [2]: 1

In [3]: 1

In [4]: 1

In [5]: 1

In [6]: 1

In [7]: 1

In [8]: 1

In [9]: 1

In [10]: 1

```python
class Carrito:

    def __init__(self, lista = []):
        self.lista = lista

    def agregar_al_carro(self, producto):

        if producto not in self.lista:
            self.lista.append(producto)
        else:
            print("El producto ya existe")
    def eliminar_del_carro(self, producto):

        for p in self.lista:
            if producto.referencia == p.referencia:
                self.lista.remove(producto)
                return
        print("No existe el producto")

    def mostrar_carro(self):
        for producto in self.lista:
            print(producto)
    def total_compra(self):
        total = 0
        for p in self.lista:
            total += p.pvp
        print("El total de la compra es: ",total)

carrito = Carrito([l, t])
carrito.agregar_al_carro(b)

carrito.eliminar_del_carro(l)

carrito.mostrar_carro()
carrito.total_compra()
```

Imprimimos el carrito y su total

```
REFERENCIA      H89s
NOMBRE          Tela roja 4mm
PVP             6.5
DESCRIPCIÓN     Tela roja de fieltro
COLOR           Rojo
MATERIAL        Fieltro

REFERENCIA      C43n
NOMBRE          Martillo
PVP             10.25
DESCRIPCIÓN     Martillo con mango de madera
MEDIDA          8
PESO            270

El total de la compra es:  16.75
```

Usando frames con tkinter

In [1]: 1

In [2]: 1

In [3]: 1

In [4]: 1

In [5]: 1

In [6]: 1

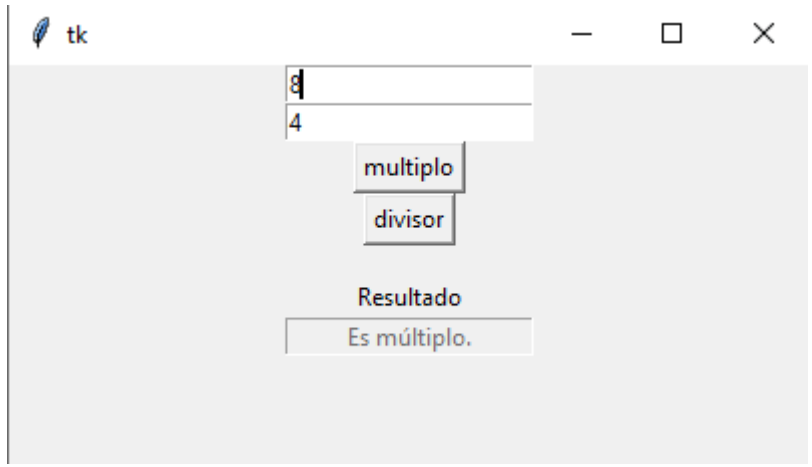
In [7]: 1

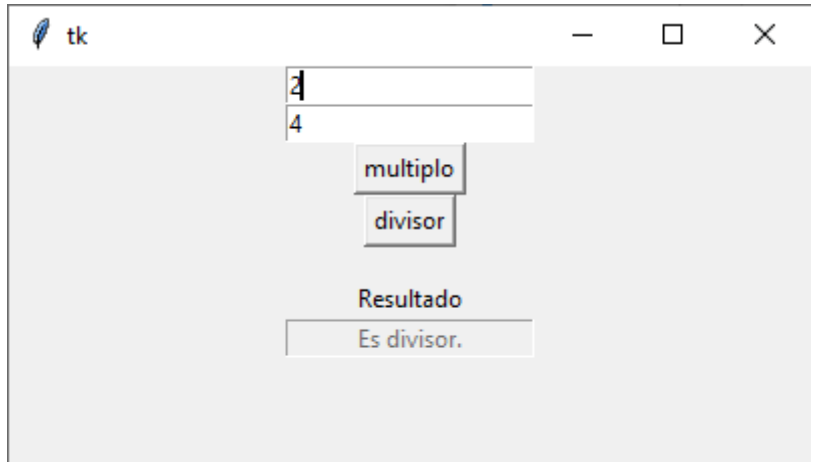
In [8]: 1

In [9]: 1

In [10]: 1

```python
from tkinter import *

def multiplo():
    validar(num1)
    validar(num2)
    if float(num1.get()) % float(num2.get()) == 0:
        r.set("Es múltiplo.")
    else:
        r.set("No es múltiplo.")


def divisor():
    validar(num1)
    validar(num2)
    if float(num2.get()) % float(num1.get()) == 0:
        r.set("Es divisor.")
    else:
        r.set("No es divisor.")

def validar(caja):
    if caja.get() =="" :
        caja.set("rellene el campo")
#------------------------------------------------------------ Proceso principal
window = Tk()
frame = Frame(window)
window.geometry('400x200')
# root = Tk()
num1= StringVar()
num2 = StringVar()
r = StringVar()

Entry(frame, textvariable=num1).grid(row = 0,column = 1)
Entry(frame, textvariable=num2).grid(row = 1,column = 1)

btnRes = Button(frame, text="multiplo", command=multiplo).grid(row = 3,column = 1)
btnRes2 = Button(frame, text="divisor", command=divisor).grid(row = 4,column = 1)

Label(frame, text="\nResultado").grid(row = 5,column = 1)
Entry(frame, justify=CENTER, state=DISABLED, textvariable = r).grid(row = 6, column = 1)

frame.pack()

root.mainloop()
```





Bases de datos

In [1]: 1

In [2]: 1

In [3]: 1

In [4]: 1

In [5]: 1

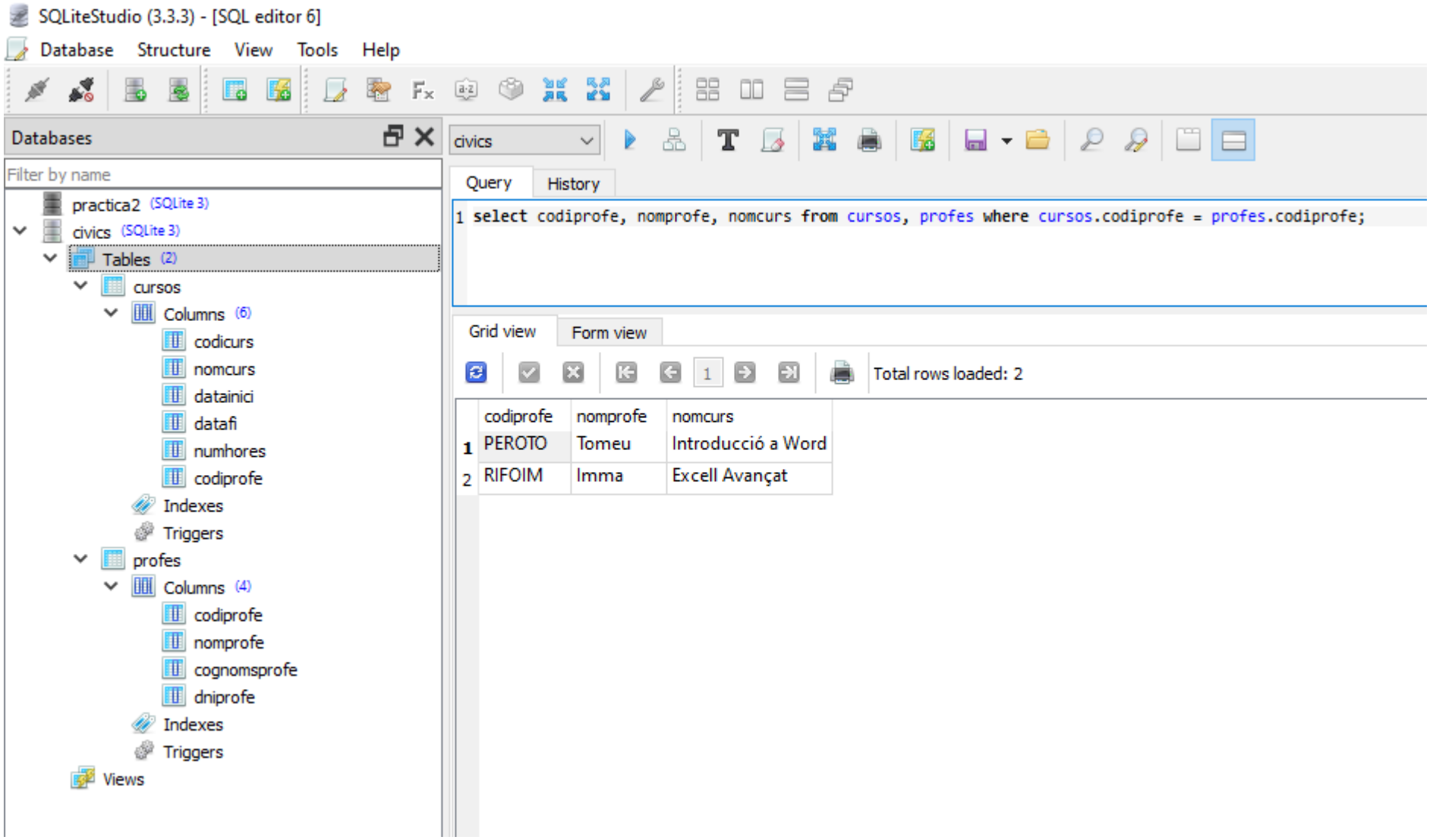
In [6]: 1

In [7]: 1

In [8]: 1

In [9]: 1

In [10]: 1



Usando bases de datos con Python

In [1]:
```
import sqlite3

def conectar_usuarios():
    """ abre la conexion con la base de datos """

    db_name = "./dat/practica2.db"
    conexion = None

    try :
        conexion = sqlite3.connect(db_name)
    except Exception as e:
        print("Error en la base de datos: ", e)
    return (conexion)

def crear_tabla_usuarios():
    """ si no existe, crea la tabla de usuarios """

    conexion = conectar_usuarios()
    cursor = conexion.cursor()

    # Ahora crearemos una tabla de usuarios con nombres, edades y emails
    sql = "CREATE TABLE IF NOT EXISTS usuarios " \
         "(codigo CHAR(6) PRIMARY KEY, nombre VARCHAR(100), activo VARCHAR(1), email VARCHAR(100), depto CHAR(3))"
    cursor.execute(sql)
    conexion.commit()                # Guardar cambios
    conexion.close()

def crear_usuarios(lista_usuarios):
    """ inserta un array de tuplas de usuarios en la base de datos """

    conexion = conectar_usuarios()
    cursor = conexion.cursor()

    # Ahora utilizamos el método executemany() para insertar varios
    cursor.executemany("INSERT INTO usuarios VALUES (?,?,?,?,?)", lista_usuarios)
    conexion.commit()                # Guardar cambios
    conexion.close()

def crear_usuario(usuario):
    """ inserta una tupla de usuario en la base de datos """

    conexion = conectar_usuarios()
    cursor = conexion.cursor()

    sql = "INSERT INTO usuarios(codigo, nombre, activo, email, depto) VALUES (?,?,?,?,?);"
    print(sql, usuario)
    cursor.execute (sql, usuario)
    conexion.commit()                # Guardar cambios
    conexion.close()
```

In [2]: 1

In [3]: 1

In [4]: 1

In [5]: 1

In [6]: 1

In [7]: 1

In [8]: 1

In [9]: 1

In [10]: 1

Funciones con bases de datos

In [1]: 1

In [2]: 1

In [3]: 1

In [4]: 1

In [5]: 1

In [6]: 1

In [7]: 1

In [8]: 1

In [9]: 1

In [10]: 1

In [11]: 1

```python
def eliminar_usuario(codigo):
    """ elimina un usuario """

    conexion = conectar_usuarios()
    cursor = conexion.cursor()

    sql = f"DELETE FROM usuarios WHERE codigo = '{codigo}';"
    cursor.execute (sql)
    conexion.commit()                # Guardar cambios
    conexion.close()

def eliminar_todo():
    """ elimina todos los usuarios """

    conexion = conectar_usuarios()
    cursor = conexion.cursor()

    cursor.execute ("DELETE FROM usuarios")
    conexion.commit()                # Guardar cambios
    conexion.close()

def consultar_usuarios() :
    """ retorna una lista de tuplas de usuarios """

    conexion = conectar_usuarios()
    cursor = conexion.cursor()

    # Recuperamos los registros de la tabla de usuarios
    cursor.execute("SELECT * FROM usuarios")

    # Recorremos todos los registros con fetchall
    # y los volcamos en una lista de usuarios
    personal = cursor.fetchall()
    conexion.close()
    return (personal)

def consultar_usuario(codigo) :
    """ retorna una lista de 1 tupla con el usuario encontrado """

    conexion = conectar_usuarios()
    cursor = conexion.cursor()

    # Recuperamos los registros de la tabla de usuarios
    cursor.execute(f"SELECT * FROM usuarios WHERE codigo = '{codigo}'")

    # Recuperamos todos los registros y los volcamos en una lista de usuarios
    personal = cursor.fetchall()
    conexion.close()
    return (personal)
```

```python

if __name__ == '__main__':
    conectar_usuarios()
    crear_tabla_usuarios()
    eliminar_todo()
#     Crea datos de prueba
    #
    crear_usuario (('0','Pruebas', 'S',  'pruebas@ejemplo.com', 'CES'))
    # Creamos una lista con varios usuarios (codigo, nombre, edad, mail)
    # ========================================================================
    usuarios = [('1','Maria', 'S',  'Maria@ejemplo.com', 'FEM'), \
                ('2','Juan', 'N',  'Juan@ejemplo.com', 'CES'),\
                ('3','Pedro', 'S',  'Pedro@ejemplo.com', 'FBI')]
    #
    # ========================================================================
    crear_usuarios (usuarios)
    eliminar_usuario('0')
    consultar_usuario('0')
    for usuario in consultar_usuarios():
        print(usuario)

```

```
INSERT INTO usuarios(codigo, nombre, activo, email, depto) VALUES (?,?,?,?,?); ('0', 'Pruebas', 'S', 'pruebas@ejemplo.com', 'CE
S')
('1', 'Maria', 'S', 'Maria@ejemplo.com', 'FEM')
('2', 'Juan', 'N', 'Juan@ejemplo.com', 'CES')
('3', 'Pedro', 'S', 'Pedro@ejemplo.com', 'FBI')
```