PYTHON II

Daniel Garcia Brun

DAM 1 13/01/2025

Contenido

Ejercicio1

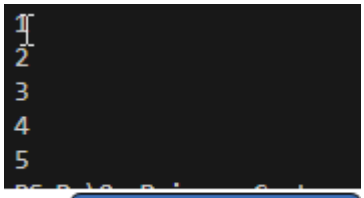
Introducción a los Bucles for

Código:

```
Iteramos sobre una lista de números
for numero in [1, 2, 3, 4, 5]:
# Imprime el número actual en la secuencia
    print(numero)
```

Daniel Garcia Brun

Resultado:



Daniel Garcia Brun

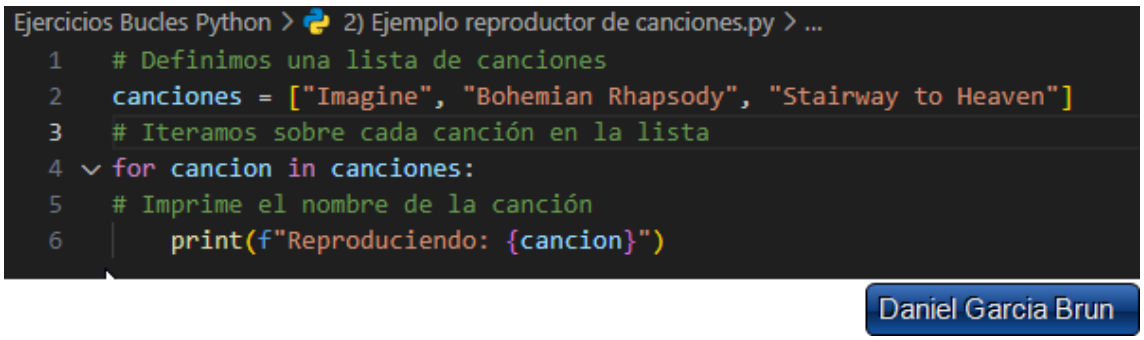
Explicación:

Como se puede ver, primero he creado una lista dentro del bucle for. Una vez creada, le pido que imprima la lista. Al imprimirla, se muestran todos los datos de la lista. Al no especificar limitaciones, se imprime toda la lista.

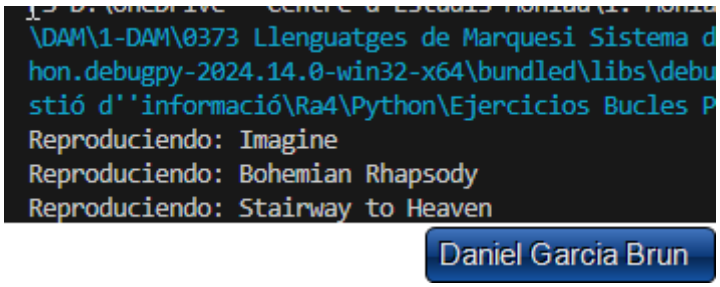
Ejercicio 2

Ejemplo: Reproducción de canciones

Código:



Resultado:



Explicación:

Esta vez he creado una lista sin utilizar bucles. Una vez creada, abro un bucle en el cual le indico que imprima con formato y añada "Reproduciendo: Canción". Al hacer esto, antes de cada elemento de la lista "canción", añade un "Reproduciendo" y continúa así hasta que se quede sin elementos en la lista.

Ejercicio 3

Uso de for con Diccionarios

Código:

```python
# Definimos un diccionario con información sobre bandas
bandas = {
    "The Beatles": {"género": "Rock", "año": 1960},
    "Daft Punk": {"género": "Electrónica", "año": 1993},
    "Queen": {"género": "Rock", "año": 1970}
}

# Iteramos sobre cada clave (nombre de la banda) y valor (detalles)
for banda, detalles in bandas.items():
    # Mostramos la información de la banda
    print(f"{banda}: Género - {detalles['género']}, Año - {detalles['año']}")

```

Daniel Garcia Brun

Resultado:

```
The Beatles: Género - Rock, Año - 1960
Daft Punk: Género - Electrónica, Año - 1993
Queen: Género - Rock, Año - 1970
```

Daniel Garcia Brun

Explicación:

Ahora hemos creado un diccionario en el que almacenamos bandas, genero de música y el año, con un bucle for le decimos que queremos coger toda la info del diccionario y que lo printe en un orden especifico añadiendo texto de por medio

Ejercicio 4

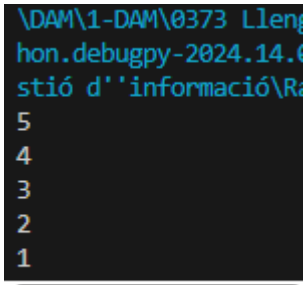
Introducción a los Bucles while

Código:

```
# Inicializamos el contador
contador = 5
# El bucle continúa mientras la condición sea verdadera
while contador > 0:
# Imprime el valor actual del contador
    print(contador)
    # Decrementa el contador en 1 (paso necesario para evitar un bucle infinito)
    contador -=1
```

Daniel Garcia Brun

Resultado:



Daniel Garcia Brun

Explicación:

Este, a pesar de parecer similar al primero, tiene un cambio: va imprimiendo el número del contador. Primero lo establecemos para que empiece en 5 y, una vez hecho esto, le vamos restando uno al contador con contador -= 1. Al llegar a 0, se detiene y no imprime debido a la condición establecida en el while.

Ejercicio 5

Ejemplo Musical: Creación de un Metrónomo

Código:

```python
# Inicializamos el número de tiempos
tiempos = 4
# Mientras haya tiempos restantes
while tiempos > 0:
# Imprimimos el tic correspondiente
    print(f"Tic {tiempos}")
# Reducimos el contador en 1
    tiempos -= 1
print("¡Fin del compás!")
```

Daniel Garcia Brun

Resultado:

```
Tic 4
Tic 3
Tic 2
Tic 1
¡Fin del compás!
```

Daniel Garcia Brun

Explicación:

Aquí comienza como el anterior, pero le añadimos que cada vez que imprima, preceda el texto con "Tic". Una vez que el contador se agote, el bucle se terminará y se imprimirá "Fin del compás

Ejercicio 6

Ejemplo de Juegos: Juego de Adivinanzas

Código:

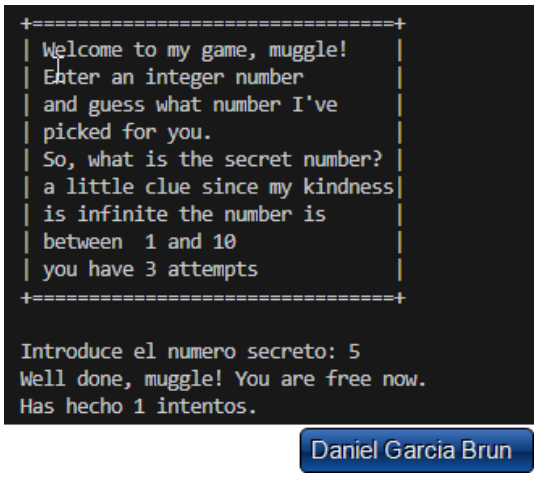
```python
import random

secret_number = random.randint(1, 10)
intentos = 0

print(
"""
+================================+
| Welcome to my game, muggle!    |
| Enter an integer number        |
| and guess what number I've     |
| picked for you.                |
| So, what is the secret number? |
| a little clue since my kindness|
| is infinite the number is      |
| between  1 and 10              |
| you have 3 attempts            |
+================================+
""")
```

Daniel Garcia Brun

```python
while intentos < 3:
    numb = int(input("Introduce el numero secreto: "))
    intentos += 1

    if numb == secret_number:
        print("Well done, muggle! You are free now.")
        print("Has hecho", intentos, "intentos.")
        break
    else:
        print("Ha ha! You're stuck in my loop!.")

if numb != secret_number:
    print("Has superado tu numero de intentos.")
    print("Ha ha! You lose the game.")
    print("Has hecho", intentos, "intentos.")
    print("El numero era:", secret_number)
```

Daniel Garcia Brun

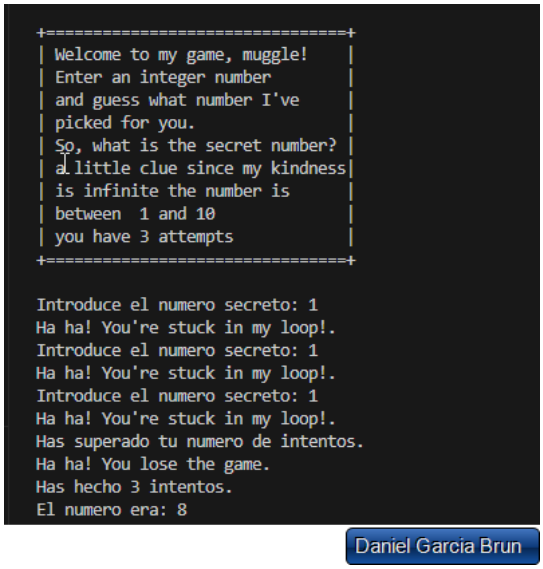
Resultado adivinando numero



Explicación:

Como se puede ver, primero imprime la explicación. Si aciertas el número, te saca y te imprime que eres libre junto con la cantidad de intentos. Esa cantidad se almacena en una variable para controlar que no te excedas. El programa comprueba que aciertes y te saque gracias a numb == secret_number:. Con esto verifica que el número aleatorio coincida y ejecuta un break para salir

Resultado Fallando



Explicación:

En este caso si superas el numero de intentos te echa y te dice el numero, esto es posible gracias al numb != secret_number: si pasa esto como se puede ver te saca del loop y te dice que has pasado el numero de intentos, se rie debido a que has perdido, te dice el numero de intentos echo y por ultimo te dice el numero secreto

Ejercicio 7

Ejercicios - Bucles for y while

Código:

```python
# Lista de notas de la escala cromática
notas_cromaticas = ["C", "C#", "D", "D#", "E", "F", "F#", "G", "G#", "A", "A#", "B"]

# Patrones para escalas mayores y menores
patrones = {
    "Escala Mayor": [2, 2, 1, 2, 2, 2, 1],
    "Escala Menor Natural": [2, 1, 2, 2, 1, 2, 2]
}

# Función para generar la escala
def generar_escala(nota_inicial, tipo_escala):
    if nota_inicial not in notas_cromaticas or tipo_escala not in patrones:
        return "Entrada no válida."

    indice_inicial = notas_cromaticas.index(nota_inicial)
    patron = patrones[tipo_escala]

    return [notas_cromaticas[(indice_inicial + sum(patron[:i])) % len(notas_cromaticas)] for i in range(len

# Solicitar entrada del usuario y generar la escala
nota_inicial = input("Introduce la nota inicial (por ejemplo, C): ")
tipo_escala = input("Introduce el tipo de escala (Escala Mayor o Escala Menor Natural): ")

# Mostrar la escala generada
print("Notas de la escala generada:", generar_escala(nota_inicial, tipo_escala))
```

Daniel Garcia Brun

Resultado Escala Mayor:

```
Introduce la nota inicial (por ejemplo, C): C#
Introduce el tipo de escala (Escala Mayor o Escala Menor Natural): Escala Mayor
Notas de la escala generada: ['C#', 'D#', 'F', 'F#', 'G#', 'A#', 'C', 'C#']
```

Daniel Garcia Brun

Escala Menor Natural:

```
Introduce la nota inicial (por ejemplo, C): C#
Introduce el tipo de escala (Escala Mayor o Escala Menor Natural): Escala Menor Natural
Notas de la escala generada: ['C#', 'D#', 'E', 'F#', 'G#', 'A', 'B', 'C#']
PS D:\OneDrive - Centre d'Estudis Monlau\1. Monlau\FP\Cursos\DAM\1-DAM\0373 Llenguatges de Marquesi Sistema de Gestió d'informació\Ra4\Python>
```

Daniel Garcia Brun

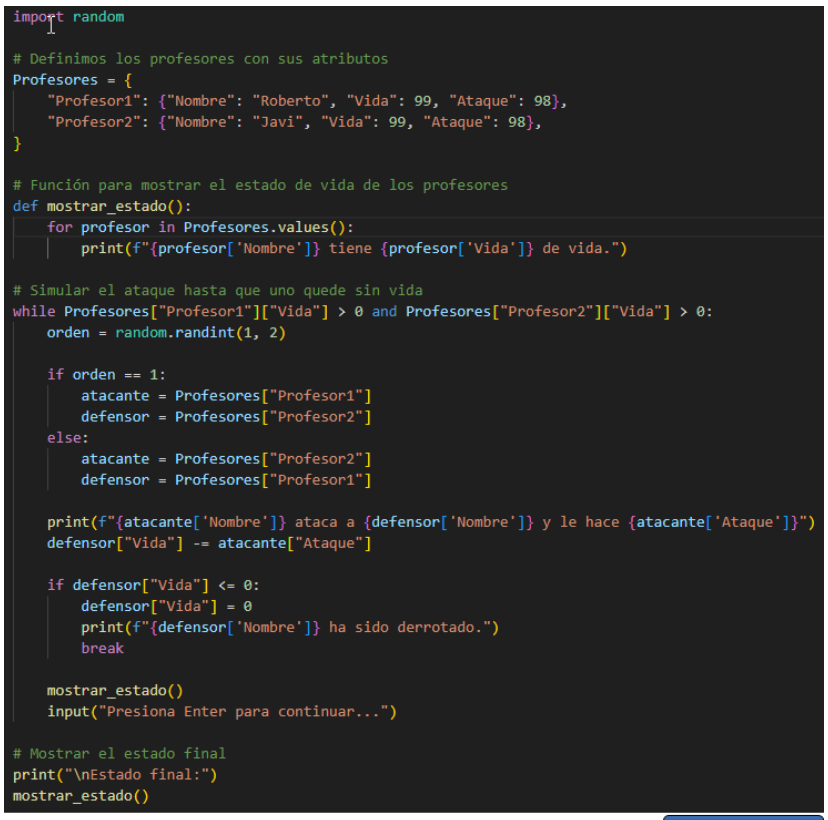
Explicación:

El programa recorre la lista notas_cromaticas. Utiliza un diccionario patrones que contiene los intervalos en semitonos para escalas mayores y menores. Gracias al operador % aplicado a la longitud de la lista (len), el programa puede recorrer la lista de notas de forma circular. Las notas se almacenan en una nueva lista llamada escala_generada, la cual se imprime al finalizar el recorrido, mostrando así las notas de la escala generada según las elecciones del usuario.

Ejercicio 8

Simulador de Batalla de Videojuegos

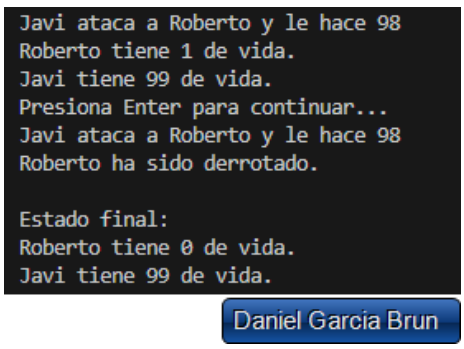
Código:



```python
import random

# Definimos los profesores con sus atributos
Profesores = {
    "Profesor1": {"Nombre": "Roberto", "Vida": 99, "Ataque": 98},
    "Profesor2": {"Nombre": "Javi", "Vida": 99, "Ataque": 98},
}

# Función para mostrar el estado de vida de los profesores
def mostrar_estado():
    for profesor in Profesores.values():
        print(f"{profesor['Nombre']} tiene {profesor['Vida']} de vida.")

# Simular el ataque hasta que uno quede sin vida
while Profesores["Profesor1"]["Vida"] > 0 and Profesores["Profesor2"]["Vida"] > 0:
    orden = random.randint(1, 2)

    if orden == 1:
        atacante = Profesores["Profesor1"]
        defensor = Profesores["Profesor2"]
    else:
        atacante = Profesores["Profesor2"]
        defensor = Profesores["Profesor1"]

    print(f"{atacante['Nombre']} ataca a {defensor['Nombre']} y le hace {atacante['Ataque']}")
    defensor["Vida"] -= atacante["Ataque"]

    if defensor["Vida"] <= 0:
        defensor["Vida"] = 0
        print(f"{defensor['Nombre']} ha sido derrotado.")
        break

    mostrar_estado()
    input("Presiona Enter para continuar...")

# Mostrar el estado final
print("\nEstado final:")
mostrar_estado()
```

Daniel Garcia Brun

Resultado:



```
Javi ataca a Roberto y le hace 98
Roberto tiene 1 de vida.
Javi tiene 99 de vida.
Presiona Enter para continuar...
Javi ataca a Roberto y le hace 98
Roberto ha sido derrotado.

Estado final:
Roberto tiene 0 de vida.
Javi tiene 99 de vida.
```

Daniel Garcia Brun

Explicación:

Como se puede ver, primero he almacenado en listas los atributos y nombres de los combatientes. Luego, he definido una función para que, nada más aparecer, te diga los datos de cada uno. Luego, con un random, elijo quién empezará el combate. Una vez hecho esto, ambos atacan. Si alguno de los dos pierde toda la vida, se sale del bucle e imprime el estado final y la vida con la que ha acabado cada uno.

Ejercicio 9

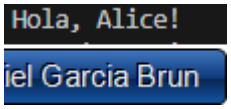
Introducción a las Funciones

Código:

```python
# Definición de una función llamada "saludo" que recibe un parámetro "nombre"
def saludo(nombre):
    return f"Hola, {nombre}!"
# Llamada a la función
print(saludo("Alice")) # Imprime: Hola, Alice!
```

Daniel Garcia Brun

Resultado:



Explicación:

Como se puede ver, definimos una función llamada saludo y le pasamos el parámetro (nombre) porque eso se especificará más adelante. Luego llamamos a la función saludo y reemplazamos nombre con lo que queramos, en este caso un string que es el nombre "Alice".

Ejercicio 10

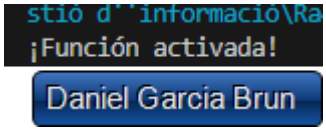
Introducción a las Funciones

Código:

```python
# Paso 1: Definimos la función
def mi_funcion():
# Acción de la función
    print("¡Función activada!")
# Paso 2: Llamamos a la función para ejecutarla
mi_funcion() # Imprime: ¡Función activada!
```

Daniel Garcia Brun

Resultado:



Daniel Garcia Brun

Explicación:

Como se puede ver, definimos una función llamada mi_funcion. Luego, le decimos que mi_funcion es un print que imprime "Función activada". Por último, al ejecutar mi_funcion, aparece el texto "Función activada".

Ejercicio 11

Parámetros y Argumentos

Código:

```python
# Definimos una función que suma dos números
def suma(a, b):
    resultado = a + b
    return resultado # Devolvemos el resultado de la suma
# Llamamos a la función pasando dos argumentos
print(suma(5, 3)) # Imprime: 8
```

Daniel Garcia Brun

¡

Resultado:

```
8
```

Brun

Explicación:

Como se ve definimos la función suma, en el cual le decimos que dentro ira a, b, luego le decimosque resultado es igual a a+b, y le decimos que nos devuelva resultado, por ultimo le decimos que printee la función suma y los valores en este caso 5,3

Ejercicio 12

Ejemplo Musical: Función para Tocar Notas

Código:

```
# Función que simula tocar una nota musical
def tocar_nota(nota):
    print(f"Tocando la nota {nota}")
# Llamamos a la función con diferentes notas
tocar_nota("C") # Imprime: Tocando la nota C
tocar_nota("G") # Imprime: Tocando la nota G
```

Daniel Garcia Brun

Resultado:

```
Tocando la nota C
Tocando la nota G
```

Daniel Garcia Brun

Explicación:

Como se ve, definimos la función tocar_nota. Dentro, añadimos un print que dice "Tocando la nota" seguido de un espacio vacío que representa la variable nota. Al llamarla y pasar el argumento "C", estamos asignando el valor "C" a nota, por lo que al ejecutar el print, se muestra "Tocando la nota C".

Ejercicio 13

Ejemplo de Juegos: Función para Verificar Puntaje

Código:

Ejercicios Bucles Python > 13) Ejemplo de Juegos Función para Verificar Puntaje.py > ...

```python
# Función que verifica si el puntaje supera un límite
def verificar_puntaje(puntaje):
    if puntaje > 100: # Si el puntaje es mayor a 100
        return "¡Nivel superado!"
    else:
        return "Sigue intentando."
# Llamadas a la función con diferentes puntajes
print(verificar_puntaje(150)) # Imprime: ¡Nivel superado!
print(verificar_puntaje(50)) # Imprime: Sigue intentando.
```

Daniel Garcia Brun

Resultado:

```
¡Nivel superado!
Sigue intentando.
```

Daniel Garcia Brun

Explicación:

Como se ve, definimos la función verificar_puntaje. Le decimos que si el puntaje es mayor que 100, indique que has superado el nivel. Si el puntaje es 100 o menor, dirá que no has superado el nivel. Luego, pedimos que se ejecute la función con 150 puntos y con 50 puntos.

Ejercicio 14

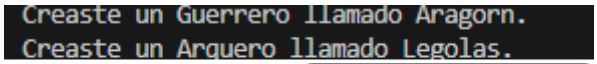
Valores Predeterminados en Parámetros

Código:

```python
# Definimos una función con un valor predeterminado para "clase"
def crear_personaje(nombre, clase="Guerrero"):
    print(f"Creaste un {clase} llamado {nombre}.")
# Llamadas a la función
crear_personaje("Aragorn") # Usa el valor predeterminado: Guerrero
crear_personaje("Legolas", "Arquero") # Especifica el valor: Arquero
```

Daniel Garcia Brun

Resultado:



Daniel Garcia Brun

Explicación:

Como se ve, definimos la función para crear un personaje, en la cual especificamos nombre (que está vacío) y clase (predeterminada como "guerrero"). Además, hacemos que imprima "Creaste un", seguido de la variable de clase, y "llamado", seguido de la variable nombre. Luego llamamos a la función: en el primer caso, solo proporcionamos el nombre, y en el segundo, especificamos tanto el nombre como la clase.

Ejercicio 15

Calculadora Musical

Código:

```python
import random

escala_cromatica = ["C", "C#", "D", "D#", "E", "F", "F#", "G", "G#", "A", "A#", "B"]
nota_random  = random.choice(escala_cromatica)


jugador =input("Escribe una nota de la escala cromatica ")
while jugador != nota_random:
    if jugador < nota_random:
        print("La nota en la que he pensado es mayor a la que has escrito ")


        jugador = input("Escribe una nota de la escala cromatica ")

    elif jugador > nota_random:
        print("La nota en la que estoy pensando es menor a la que has escrito ")

        jugador = input("Escribe una nota de la escala cromatica ")

if jugador == nota_random:

    print ("Felicidades has hacertado la nota")
```

Daniel Garcia Brun

Resultado:

```
Escribe una nota de la escala cromatica C
La nota en la que he pensado es mayor a la que has escrito
Escribe una nota de la escala cromatica E
La nota en la que he pensado es mayor a la que has escrito
Escribe una nota de la escala cromatica G
La nota en la que estoy pensando es menor a la que has escrito
Escribe una nota de la escala cromatica F#
Felicidades has hacertado la nota
```

Daniel Garcia Brun

Explicación:

Como se ve, primero importamos random e creamos una lista con las notas, iniciamos un while cada vez que el jugador falle la nota sin limite de intentos, si la nota es mas alta t lo dice y lo mismo si es mas baja, en caso de que el jugador acierte te avisa gracias al if jugador == nota_random

Ejercicio 16

inventario de videojuegos

Código:

```python
Inventario = {

}

def añadir_objeto(diccionario):
    tipo = input("Introduce el tipo de objeto (por ejemplo, Espadas, Escudo): ")
    nombre = input("Introduce el nombre del objeto: ")
    atributo = input("Introduce el atributo del objeto (por ejemplo, Ataque, Defensa, Curacion): ")
    valor = input("Introduce el valor del atributo: ")

    if tipo in diccionario:
        diccionario[tipo]["Nombre"] = nombre
        diccionario[tipo][atributo] = valor
    else:
        diccionario[tipo] = {"Nombre": nombre, atributo: valor}

    return diccionario

def menu():
    print("Bienvenido, si quisieses salir en cualquier momento pulsa Q")
    print("Para añadir más objetos pulsa A")

    while True:
        opcion = input("¿Qué quieres hacer? ").upper()

        if opcion == "Q":
            print(Inventario)
            break
        elif opcion == "A":
            añadir_objeto(Inventario)
            print(Inventario)
        else:
            print("Opción no válida, intenta de nuevo.")

# Ejecutar el menú
menu()
```

Daniel García Brun

Resultado:

```
Bienvenido, si quisieses salir en cualquier momento pulsa Q
Para añadir más objetos pulsa A
¿Qué quieres hacer? A
Introduce el tipo de objeto (por ejemplo, Espadas, Escudo): Espada
Introduce el nombre del objeto: Excalibur
Introduce el atributo del objeto (por ejemplo, Ataque, Defensa, Curacion): Ataque
Introduce el valor del atributo: 10000
{'Espada': {'Nombre': 'Excalibur', 'Ataque': '10000'}}
¿Qué quieres hacer? A
Introduce el tipo de objeto (por ejemplo, Espadas, Escudo): Escudo
Introduce el nombre del objeto: Tapa de Cacerola
Introduce el atributo del objeto (por ejemplo, Ataque, Defensa, Curacion): Defensa
Introduce el valor del atributo: 1
{'Espada': {'Nombre': 'Excalibur', 'Ataque': '10000'}, 'Escudo': {'Nombre': 'Tapa de Cacerola', 'Defensa': '1'}}
¿Qué quieres hacer? A
Introduce el tipo de objeto (por ejemplo, Espadas, Escudo): Sanacion
Introduce el nombre del objeto: Curitas
Introduce el atributo del objeto (por ejemplo, Ataque, Defensa, Curacion): Curacion
Introduce el valor del atributo: 5
{'Espada': {'Nombre': 'Excalibur', 'Ataque': '10000'}, 'Escudo': {'Nombre': 'Tapa de Cacerola', 'Defensa': '1'}, 'Sanacion': {'Nombre': 'Curitas', 'Curacion': '5'}}
¿Qué quieres hacer? A
Introduce el tipo de objeto (por ejemplo, Espadas, Escudo): Hechizo
Introduce el nombre del objeto: Sana sana colita de rana
Introduce el atributo del objeto (por ejemplo, Ataque, Defensa, Curacion): Curacion
Introduce el valor del atributo: 1000000000000
{'Espada': {'Nombre': 'Excalibur', 'Ataque': '10000'}, 'Escudo': {'Nombre': 'Tapa de Cacerola', 'Defensa': '1'}, 'Sanacion': {'Nombre': 'Curitas', 'Curacion': '5'}, 'Hechizo': {'Nombre': 'Sana sana colita de rana', 'Curacion': '1000000000000'}}
¿Qué quieres hacer? q
{'Espada': {'Nombre': 'Excalibur', 'Ataque': '10000'}, 'Escudo': {'Nombre': 'Tapa de Cacerola', 'Defensa': '1'}, 'Sanacion': {'Nombre': 'Curitas', 'Curacion': '5'}, 'Hechizo': {'Nombre': 'Sana sana colita de rana', 'Curacion': '1000000000000'}}
PS D:\OneDrive - Centre d'Estudis Monlau\1. Monlau\FP\Cursos\DAM\1-DAM\0373 Llenguatges de Marquesi Sistema de Gestió d'informació\Ra4\Python> 
```

Daniel García Brun

Explicación:

Primero, creamos un diccionario vacío. Luego, definimos una función que toma cuatro parámetros: tipo, nombre, atributo y valor. Esta función añade estos elementos al diccionario en el orden en que se proporcionan. Después, configuramos un menú de inicio que da la bienvenida al usuario y explica cómo añadir objetos. Si el usuario escribe "A", se ejecuta la función definida anteriormente. En main, solo llamamos a la función del menú, ya que la primera función está incluida en ella.

Ejercicio 17

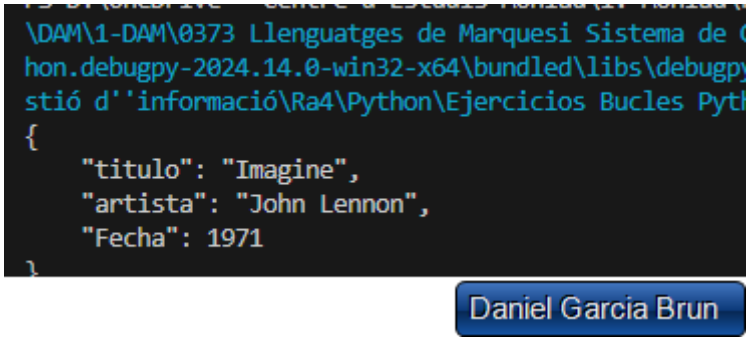
Trabajando con JSON usando el módulo json

Código

```python
import json # Importamos el módulo JSON
# Creamos un diccionario que representa una canción
cancion = {
"titulo": "Imagine",
"artista": "John Lennon",
"Fecha": 1971
}
# Convertimos el diccionario a formato JSON
json_cancion = json.dumps(cancion, indent=4) # indent=4 mejora la legibilidad
print(json_cancion) # Imprime la cadena JSON formateada
```

Daniel Garcia Brun

Resultado:



Daniel Garcia Brun

Explicación:

Primero, importamos la biblioteca json para poder formatear el texto como un objeto JSON. Luego, creamos un diccionario con los datos deseados, como la canción, el artista y el año. A continuación, utilizamos la función json.dumps para formatear el diccionario y almacenarlo en la variable json_cancion. Finalmente, imprimimos la variable json_cancion para ver el contenido en formato JSON.

Ejercicio 18

Guardar Datos JSON en un Archivo

Código

```python
import json
import os

# Creamos una lista de canciones
biblioteca = [
    {"titulo": "Imagine", "artista": "John Lennon", "Fecha": 1971},
    {"titulo": "Bohemian Rhapsody", "artista": "Queen", "Fecha": 1975}
]

# Obtenemos el directorio actual
directorio_actual = os.path.dirname(os.path.realpath(__file__))
ruta_archivo = os.path.join(directorio_actual, "biblioteca.json")

# Guardamos los datos en un archivo JSON en el directorio actual
with open(ruta_archivo, "w") as archivo:
    json.dump(biblioteca, archivo, indent=4)

print(f"Biblioteca musical guardada en '{ruta_archivo}'")
```

Daniel Garcia Brun

Resultado:

```
stió d''informació\Ra4\Python\Ejercicios Bucles Python\18) G
Biblioteca musical guardada en 'biblioteca.json'
```

Daniel Garcia Brun

{} biblioteca.json

Daniel Garcia Brun

Explicación:

Como se ve creamos un diccionario, luego con la función open le decimos donde y con que nombre guardarlo, por último, le decimos el mensaje que debe dar y de esta forma creamos un JSON ejecutando un Python, aparte he añadido el import os para decirle que me lo cree en mí misma carpeta debido a que si no me lo creaba en un directorio superior

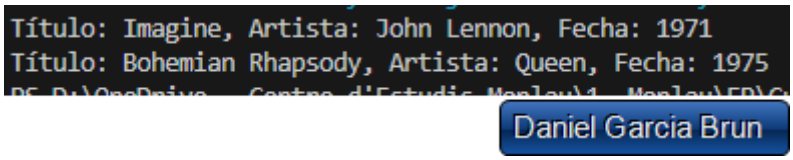
Ejercicio 19

Código

```python
import json
import os

# Obtenemos el directorio actual
directorio_actual = os.path.dirname(os.path.abspath(__file__))
ruta_archivo = os.path.join(directorio_actual, "biblioteca.json")

# Leemos el archivo JSON
with open(ruta_archivo, "r") as archivo:
    datos = json.load(archivo)  # Cargamos los datos en formato Python

# Mostramos los datos cargados
for cancion in datos:
    print(f"Título: {cancion['titulo']}, Artista: {cancion['artista']}, Fecha: {cancion['Fecha']}")

```

Daniel Garcia Brun

Resultado:



Explicación:

Como vemos este ejercicio complementa al anterior buscando el archivo creado y te imprime la información, en este caso hemos vuelto a usar el import os para que pueda ver el json desde la carpeta en la que estoy

Ejercicio 20

Código

```python
import json

# Función para cargar la biblioteca desde un archivo JSON
def cargar_biblioteca():
    try:
        with open("biblioteca.json", "r") as archivo:
            return json.load(archivo)
    except FileNotFoundError:
        return []

# Función para guardar la biblioteca en un archivo JSON
def guardar_biblioteca(biblioteca):
    with open("biblioteca.json", "w") as archivo:
        json.dump(biblioteca, archivo, indent=4)

# Función para agregar una canción
def agregar_cancion(biblioteca):
    titulo = input("Título de la canción: ")
    artista = input("Artista: ")
    year = input("Year: ")
    biblioteca.append({"titulo": titulo, "artista": artista, "year": year})
    print("Canción añadida correctamente.")
```

Daniel García Brun

```python
def main():
    biblioteca = cargar_biblioteca()
    while True:
        print("\n1. Añadir canción\n2. Mostrar biblioteca\n3. Salir")
        opcion = input("Selecciona una opción: ")

        if opcion == "1":
            agregar_cancion(biblioteca)
            guardar_biblioteca(biblioteca)
        elif opcion == "2":
            for cancion in biblioteca:
                # Manejar la falta de la clave 'year'
                year = cancion.get('year', 'Desconocido')
                print(f"{cancion['titulo']} - {cancion['artista']} ({year})")
        elif opcion == "3":
            print("¡Hasta la próxima!")
            break
        else:
            print("Opción no válida.")

if __name__ == "__main__":
    main()
```

Daniel García Brun

Resultado:

```
1. Añadir canción
2. Mostrar biblioteca
3. Salir
Selecciona una opción: 1
Título de la canción: A
Artista: a
Year: 2000
Canción añadida correctamente

1. Añadir canción
2. Mostrar biblioteca
3. Salir
Selecciona una opción: 2
Selecciona una opción: 2
A - a (2000)

1. Añadir canción
2. Mostrar biblioteca
3. Salir
Selecciona una opción: 3
¡Hasta la próxima!
```

Daniel Garcia Brun

Explicación:

Como vemos este ejercicio complementa al anterior buscando el archivo creado y te imprime la información, en este caso hemos vuelto a usar el import os para que pueda ver el json desde la carpeta en la que estoy

Ejercicio 21

Código

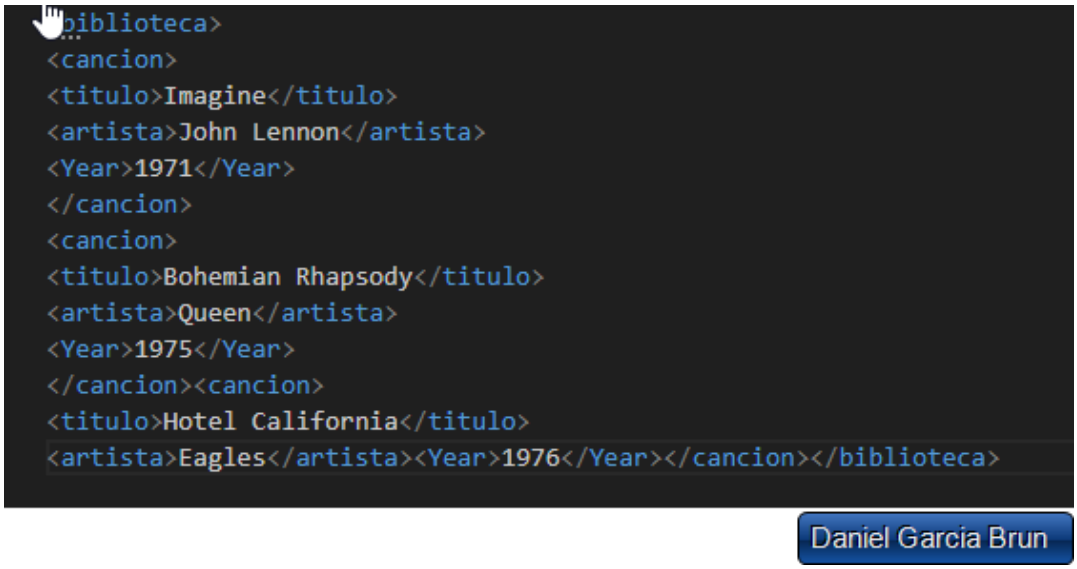
```python
import xml.etree.ElementTree as ET

# Creamos el elemento raíz
biblioteca = ET.Element("biblioteca")

# Añadimos canciones como subelementos
cancion1 = ET.SubElement(biblioteca, "cancion")
ET.SubElement(cancion1, "titulo").text = "Imagine"
ET.SubElement(cancion1, "artista").text = "John Lennon"
ET.SubElement(cancion1, "Year").text = "1971"

cancion2 = ET.SubElement(biblioteca, "cancion")
ET.SubElement(cancion2, "titulo").text = "Bohemian Rhapsody"
ET.SubElement(cancion2, "artista").text = "Queen"
ET.SubElement(cancion2, "Year").text = "1975"

cancion3 = ET.SubElement(biblioteca, "cancion")
ET.SubElement(cancion3, "titulo").text = "Hotel California"
ET.SubElement(cancion3, "artista").text = "Eagles"
ET.SubElement(cancion3, "Year").text = "1976"

# Convertimos el árbol XML en una cadena y lo guardamos en un archivo
arbol = ET.ElementTree(biblioteca)
arbol.write("biblioteca.xml")
print("Archivo XML creado: biblioteca.xml")

# Mostrar el contenido del archivo XML
ET.dump(biblioteca)
```

Daniel Garcia Brun

Resultado:

```
XML.py
Archivo XML creado: biblioteca.xml
<biblioteca><cancion><titulo>Imagine</titulo><artista>John Lennon</artista><Year>1971</Year>
ulo><artista>Queen</artista><Year>1975</Year></cancion><cancion><titulo>Hotel California</ti
cancion></biblioteca>
```

Daniel Garcia Brun



Explicación:

Como podemos ver esto lo que hace es crear el archivo biblioteca.xml y añade todas las canciones o artistas que le añadas esto es gracias al ET element el unico problema que no he sabido resolver es el hecho que los crea en uni línea en vez de crear la estructura xml

Ejercicio 22

Código

22)Leer un Archivo XML.py > ...

```python
import xml.etree.ElementTree as ET

# Cargamos el archivo XML
arbol = ET.parse("biblioteca.xml")
raiz = arbol.getroot()

# Mostramos la información de cada canción
for cancion in raiz.findall("cancion"):
    titulo = cancion.find("titulo").text
    artista = cancion.find("artista").text
    año = cancion.find("Year").text
    print(f"Título: {titulo}, Artista: {artista}, Año: {año}")

```

Daniel Garcia Brun

Resultado:

```
Título: Imagine, Artista: John Lennon, Año: 1971
Título: Bohemian Rhapsody, Artista: Queen, Año: 1975
Título: Hotel California, Artista: Eagles, Año: 1976
```

Daniel Garcia Brun

Explicación:

Como podemos ver esto lo que hace es coger el xml creado en ele ejercicio anterior y mostrarte por pantalla los datos formateados esto es gracias al bucle for que añade las canciones de esra forma, como antes nos volvemos a ayudar del import xml.etree.element tree

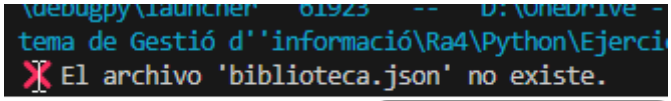
Ejercicio 23

Código

```python
import json

def verificar_json(ruta_archivo):
    try:
        # Abrimos y cargamos el archivo JSON
        with open(ruta_archivo, "r") as archivo:
            datos = json.load(archivo)
            print(f"✅ El archivo '{ruta_archivo}' está bien estructurado (JSON).")
    except json.JSONDecodeError as e:
        print(f"❌ Error en el archivo JSON '{ruta_archivo}': {e}")
    except FileNotFoundError:
        print(f"❌ El archivo '{ruta_archivo}' no existe.")
    except Exception as e:
        print(f"❌ Ocurrió un error inesperado: {e}")

# Ejemplo de uso
verificar_json("biblioteca.json")
```

Daniel Garcia Brun

Resultado 1:

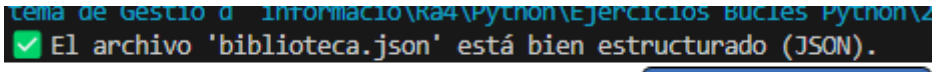


Daniel Garcia Brun

Explicación 1:

Como podemos ver este programa comprueba si tenemos un archivo llamado en este caso biblioteca.json, si no aparece nos dará el error que se muestra esto lo logramos gracias a las excepciones, el programa le decimos el archivo siempre existe y debajo le ponemos excepto si....

Resultado 2



Daniel Garcia Brun

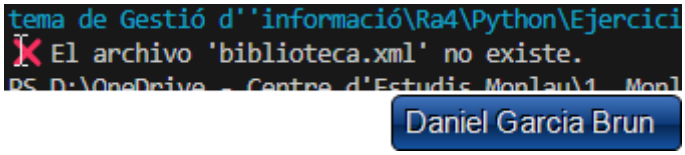
Explicación 2:

Como podemos ver aquí es cuando si existe un biblioteca.json, como no entra en ninguna excepción se ejecuta sin problema

Ejercicio 24

Código

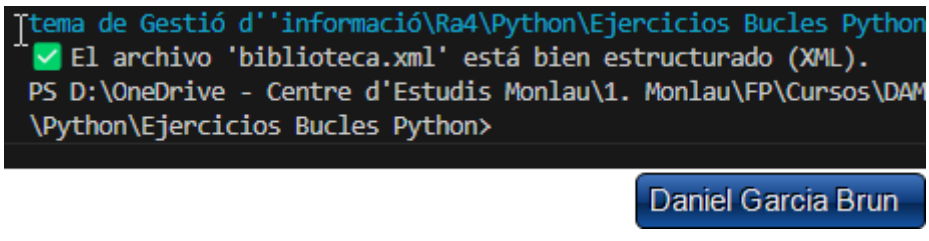
Resultado 1:



Explicación 1:

Como podemos ver este programa comprueba si tenemos un archivo llamado en este caso biblioteca.xml, si no aparece nos dará el error que se muestra esto lo logramos gracias a las excepciones, el programa le decimos el archivo siempre existe y debajo le ponemos excepto si....

Resultado 2



Explicación 2:

Como podemos ver aquí es cuando si existe un biblioteca.xml, como no entra en ninguna excepción se ejecuta sin problema