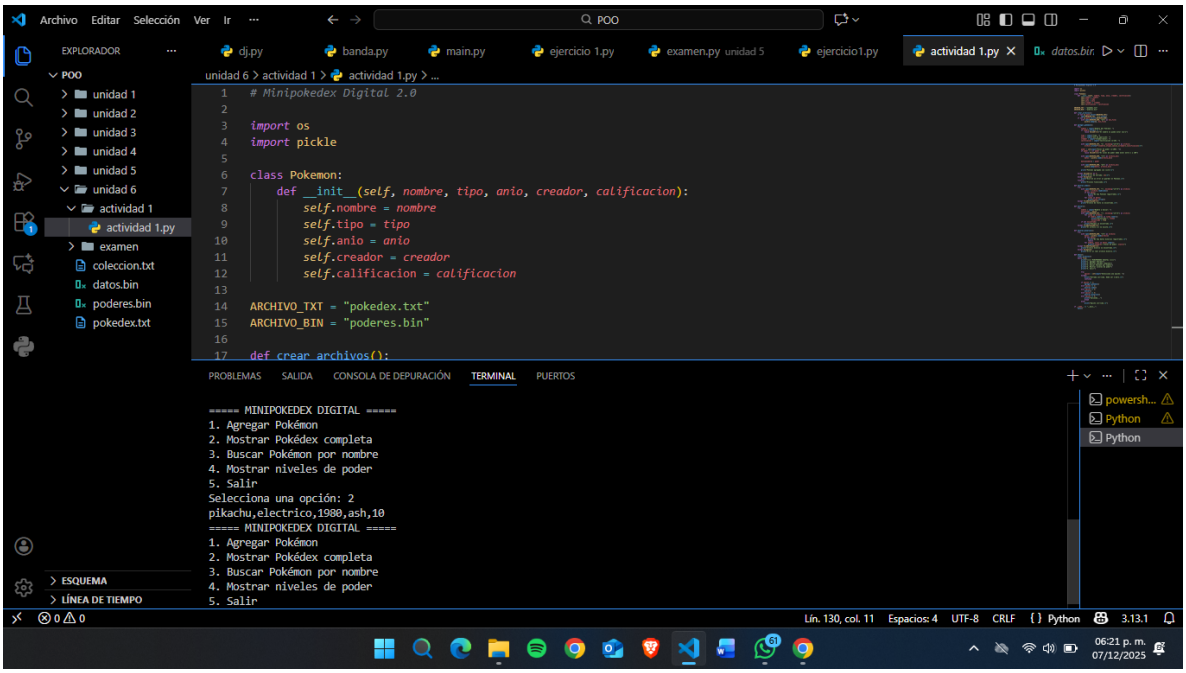

Minipokedex Digital 2.0

```
import os
import pickle

class Pokemon:
    def __init__(self, nombre, tipo, anio, creador, calificacion):
        self.nombre = nombre
        self.tipo = tipo
        self.anio = anio
        self.creador = creador
        self.calificacion = calificacion

ARCHIVO_TXT = "pokedex.txt"
ARCHIVO_BIN = "poderes.bin"
```

```
def crear_archivos():
    if not os.path.exists(ARCHIVO_TXT):
        open(ARCHIVO_TXT, "w").close()
    if not os.path.exists(ARCHIVO_BIN):
        with open(ARCHIVO_BIN, "wb") as bin_file:
            pickle.dump({}, bin_file)

def agregar_pokemon():
    try:
        nombre = input("Nombre del Pokémon: ")
        if nombre.strip() == "":
            raise ValueError("El nombre no puede estar vacío")

        tipo = input("Tipo: ")
        anio = input("Año de aparición: ")
        creador = input("Creador/Autor: ")
        calificacion = input("Calificación (1-10): ")

        with open(ARCHIVO_TXT, "a", encoding="utf-8") as archivo:
            archivo.write(f"{nombre},{tipo},{anio},{creador},{calificacion}\n")

        poder = int(input("Nivel de poder (1-100): "))
        if poder < 1 or poder > 100:
            raise ValueError("El nivel de poder debe estar entre 1 y 100")

        with open(ARCHIVO_BIN, "rb") as archivo_bin:
```

```
            datos = pickle.load(archivo_bin)

        datos[nombre] = poder

        with open(ARCHIVO_BIN, "wb") as archivo_bin:
            pickle.dump(datos, archivo_bin)

        print("Pokémon agregado con éxito!\n")

    except ValueError as e:
        print(f"Error de entrada: {e}\n")
    except Exception:
        print("Ocurrió un error al guardar el Pokémon.\n")
    finally:
        print("Proceso finalizado.\n")

def mostrar_todo():
    try:
        with open(ARCHIVO_TXT, "r", encoding="utf-8") as archivo:
            datos = archivo.readlines()
            if not datos:
                print("No hay Pokémon registrados.\n")
                return
            for linea in datos:
                print(linea.strip())
    except FileNotFoundError:
```

```
        print("Archivo de texto no encontrado.\n")


def buscar():
    try:
        nombre = input("Nombre a buscar: ")
        encontrado = False
        with open(ARCHIVO_TXT, "r", encoding="utf-8") as archivo:
            for linea in archivo:
                if nombre.lower() in linea.lower():
                    print("Encontrado:", linea)
                    encontrado = True
        if not encontrado:
            print("Pokémon no encontrado.\n")
    except FileNotFoundError:
        print("El archivo aún no existe.\n")


def mostrar_binarios():
    try:
        with open(ARCHIVO_BIN, "rb") as archivo:
            datos = pickle.load(archivo)
            if not datos:
                print("No hay datos binarios registrados.\n")
                return
            for nombre, valor in datos.items():
                print(f"{nombre} → Nivel de poder: {valor}")
    except FileNotFoundError:
```

```
        print("Archivo binario no encontrado.\n")
    except Exception:
        print("Error al leer archivo binario.\n")


def menu():
    crear_archivos()
    while True:
        print("===== MINIPOKEDEX DIGITAL =====")
        print("1. Agregar Pokémon")
        print("2. Mostrar Pokédex completa")
        print("3. Buscar Pokémon por nombre")
        print("4. Mostrar niveles de poder")
        print("5. Salir")

        try:
            opcion = int(input("Selecciona una opción: "))
        except:
            print("Entrada inválida. Debe ser número.\n")
            continue

        if opcion == 1:
            agregar_pokemon()
        elif opcion == 2:
            mostrar_todo()
        elif opcion == 3:
            buscar()
```

```
        elif opcion == 4:
            mostrar_binarios()
        elif opcion == 5:
            print("Saliendo...")
            break
        else:
            print("Opción inválida.\n")

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

https://github.com/yael-byte/poo