Sesión 5 — Adaptación de ventanas al diseño del proyecto

Carlos David Martínez Rocha (A01352717)
Adrian Navarro Romo (A00575101)
Diego Adiel Flores Navarro (A00573953)
Andrés Alejandro Sánchez Rábago
Pedro López Casillas (A00575320)

28 de septiembre del 2025

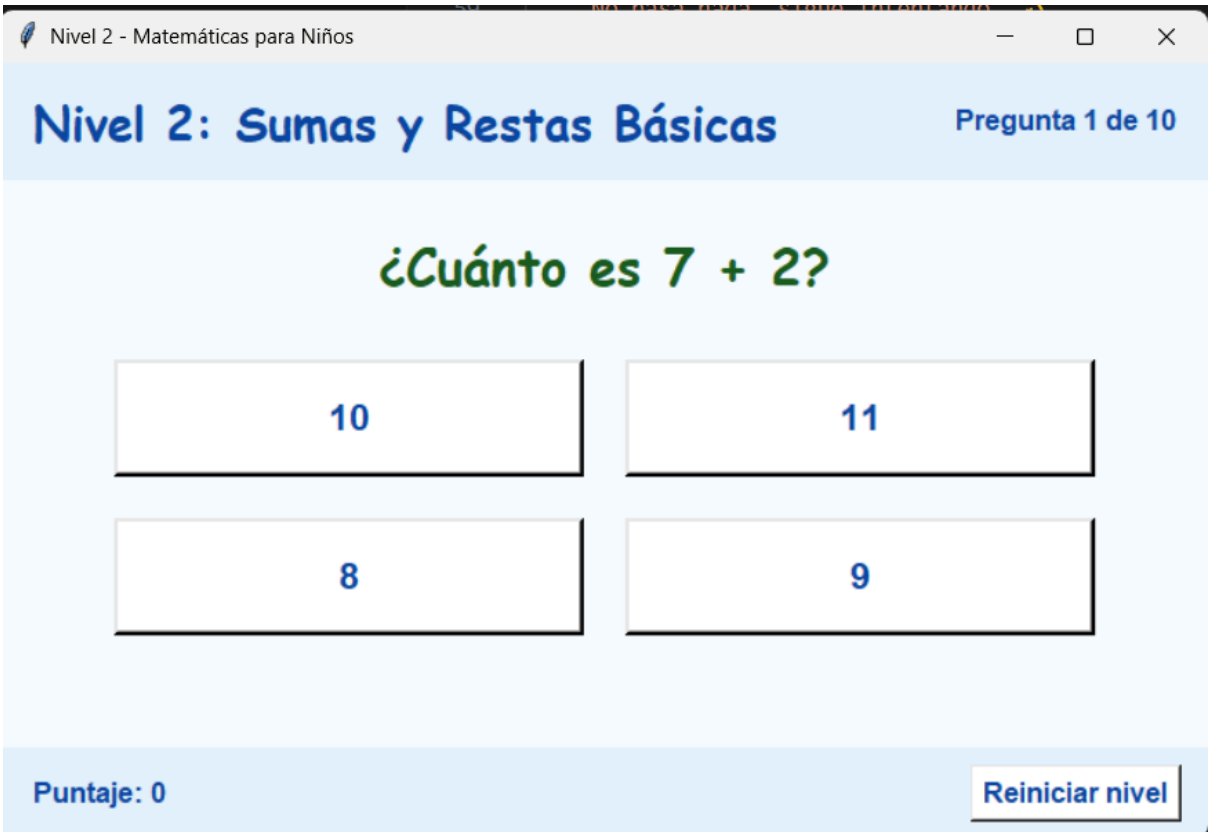
Tecnológico de Monterrey

Departamento de Ciencias

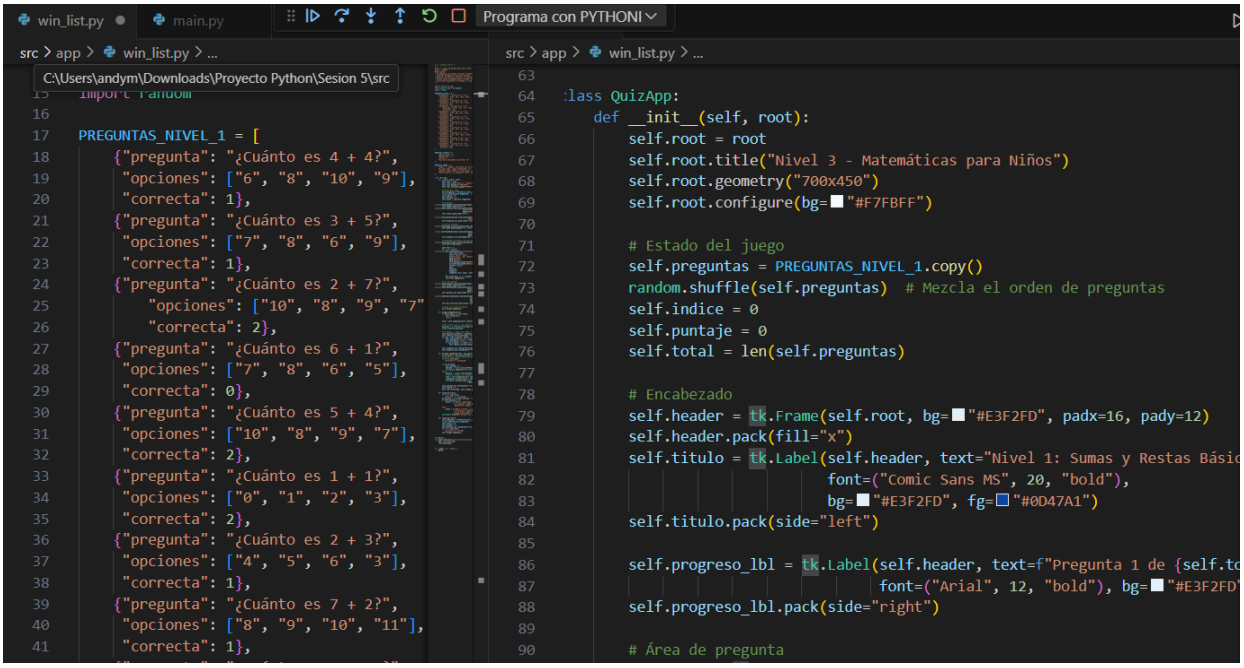
Pensamiento computacional para Ingeniería

- **Andrés Alejandro Sánchez Rábago Ventana asignada - La elaboración de los niveles del juego (win form.py).**

Desarrollé una aplicación interactiva de aprendizaje basada en Tkinter, la biblioteca gráfica estándar de Python, diseñada especialmente para niños en edad escolar inicial. La app consiste en un quiz de matemáticas básicas (sumas y restas) con 10 preguntas de opción múltiple. El objetivo es reforzar el aprendizaje de operaciones aritméticas simples de forma divertida, motivadora y visualmente atractiva.



Y en la elaboración del código fue el siguiente:



```
C:\Users\andym\Downloads\Proyecto Python\Sesion 5\src

import random

PREGUNTAS_NIVEL_1 = [
    {"pregunta": "¿Cuánto es 4 + 4?",
     "opciones": ["6", "8", "10", "9"],
     "correcta": 1},
    {"pregunta": "¿Cuánto es 3 + 5?",
     "opciones": ["7", "8", "6", "9"],
     "correcta": 1},
    {"pregunta": "¿Cuánto es 2 + 7?",
        "opciones": ["10", "8", "9", "7"
        "correcta": 2},
    {"pregunta": "¿Cuánto es 6 + 1?",
     "opciones": ["7", "8", "6", "5"],
     "correcta": 0},
    {"pregunta": "¿Cuánto es 5 + 4?",
     "opciones": ["10", "8", "9", "7"],
     "correcta": 2},
    {"pregunta": "¿Cuánto es 1 + 1?",
     "opciones": ["0", "1", "2", "3"],
     "correcta": 2},
    {"pregunta": "¿Cuánto es 2 + 3?",
     "opciones": ["4", "5", "6", "3"],
     "correcta": 1},
    {"pregunta": "¿Cuánto es 7 + 2?",
     "opciones": ["8", "9", "10", "11"],
     "correcta": 1},
```

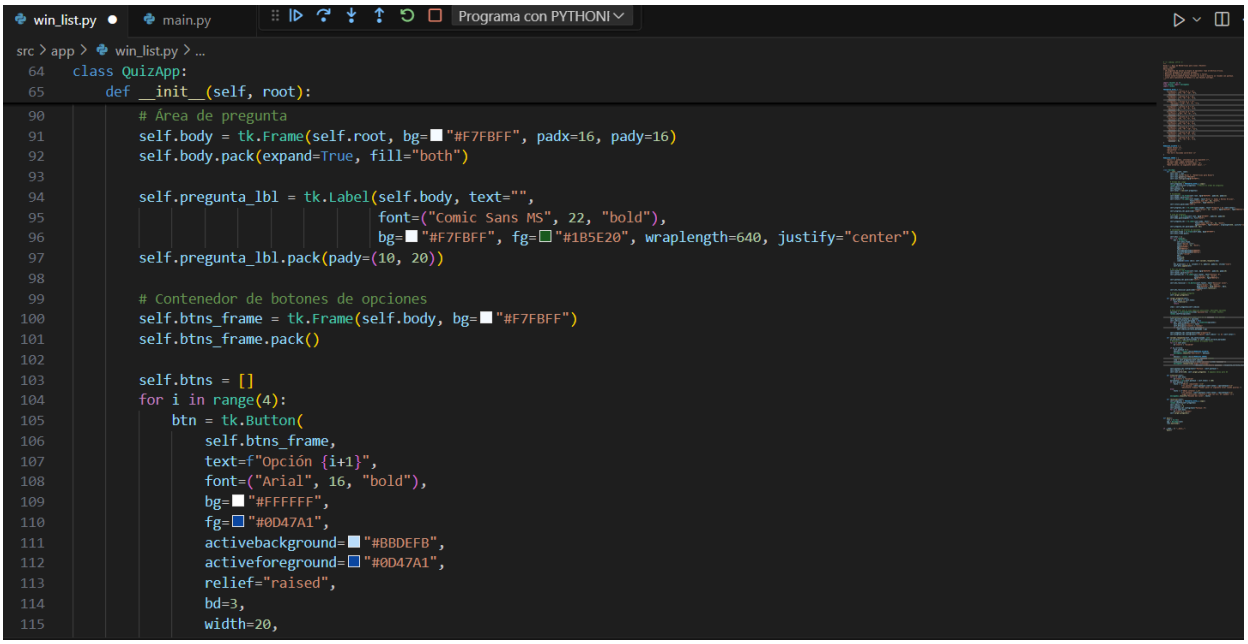
```
class QuizApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Nivel 3 - Matemáticas para Niños")
        self.root.geometry("700x450")
        self.root.configure(bg="#F7FBFF")

        # Estado del juego
        self.preguntas = PREGUNTAS_NIVEL_1.copy()
        random.shuffle(self.preguntas)  # Mezcla el orden de preguntas
        self.indice = 0
        self.puntaje = 0
        self.total = len(self.preguntas)

        # Encabezado
        self.header = tk.Frame(self.root, bg="#E3F2FD", padx=16, pady=12)
        self.header.pack(fill="x")
        self.titulo = tk.Label(self.header, text="Nivel 1: Sumas y Restas Básic
                               font=("Comic Sans MS", 20, "bold"),
                               bg="#E3F2FD", fg="#0D47A1")
        self.titulo.pack(side="left")

        self.progreso_lbl = tk.Label(self.header, text=f"Pregunta 1 de {self.to
                                     font=("Arial", 12, "bold"), bg="#E3F2FD"
        self.progreso_lbl.pack(side="right")

        # Área de pregunta
```



```
class QuizApp:
    def __init__(self, root):
        # Área de pregunta
        self.body = tk.Frame(self.root, bg="#F7FBFF", padx=16, pady=16)
        self.body.pack(expand=True, fill="both")

        self.pregunta_lbl = tk.Label(self.body, text="",
                                     font=("Comic Sans MS", 22, "bold"),
                                     bg="#F7FBFF", fg="#1B5E20", wraplength=640, justify="center")
        self.pregunta_lbl.pack(pady=(10, 20))

        # Contenedor de botones de opciones
        self.btns_frame = tk.Frame(self.body, bg="#F7FBFF")
        self.btns_frame.pack()

        self.btns = []
        for i in range(4):
            btn = tk.Button(
                self.btns_frame,
                text=f"Opción {i+1}",
                font=("Arial", 16, "bold"),
                bg="#FFFFFF",
                fg="#0D47A1",
                activebackground="#BBDEFB",
                activeforeground="#0D47A1",
                relief="raised",
                bd=3,
                width=20,
```

```
    # Pie con puntaje
    self.footer = tk.Frame(self.root, bg="#E3F2FD", padx=16, pady=10)
    self.footer.pack(fill="x")
    self.puntaje_lbl = tk.Label(self.footer, text="Puntaje: 0",
                                font=("Arial", 12, "bold"),
                                bg="#E3F2FD", fg="#0D47A1")
    self.puntaje_lbl.pack(side="left")

    self.btn_reiniciar = tk.Button(self.footer, text="Reiniciar nivel",
                                   font=("Arial", 12, "bold"),
                                   bg="#FFFFFF", fg="#0D47A1", bd=2,
                                   command=self.reiniciar)
    self.btn_reiniciar.pack(side="right")

    # Cargar la primera pregunta
    self.cargar_pregunta()

ef cargar_pregunta(self):
    if self.indice >= self.total:
        self.finalizar()
        return

    item = self.preguntas[self.indice]
```

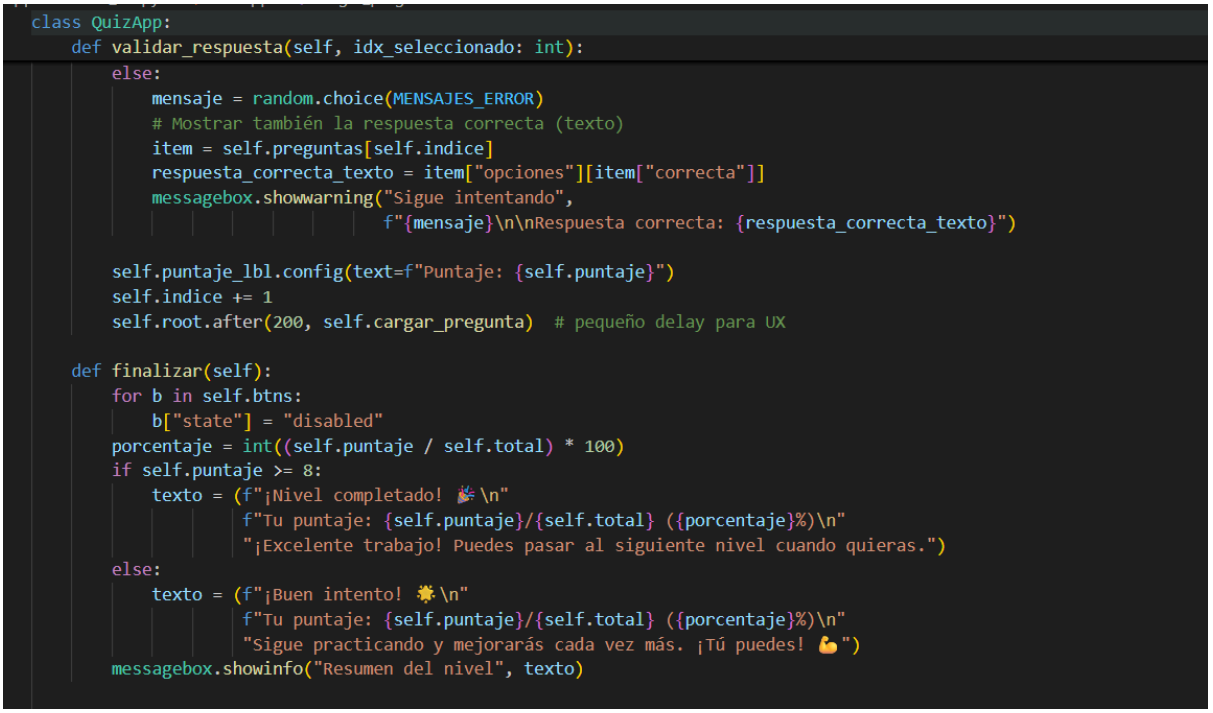
```
    opciones = list(enumerate(item["opciones"]))  # [(idx, texto)]
    random.shuffle(opciones)

    # Guardamos el mapeo para saber cuál es la correcta tras mezclar
    self.opciones_actuales = opciones
    self.indice_correcto_barajado = None
    for pos, (idx_original, texto) in enumerate(opciones):
        self.btns[pos]["text"] = texto
        self.btns[pos]["state"] = "normal"
        if idx_original == item["correcta"]:
            self.indice_correcto_barajado = pos

    self.pregunta_lbl.config(text=item["pregunta"])
    self.progreso_lbl.config(text=f"Pregunta {self.indice + 1} de {self.total}")

ef validar_respuesta(self, idx_seleccionado: int):
    es_correcta = (idx_seleccionado == self.indice_correcto_barajado)
    # Deshabilitar botones para evitar múltiples clics
    for b in self.btns:
        b["state"] = "disabled"

    if es_correcta:
        self.puntaje += 1
        mensaje = random.choice(MENSAJES_ACIERTO)
        messagebox.showinfo("¡Correcto!", mensaje)
```



```python
class QuizApp:
    def validar_respuesta(self, idx_seleccionado: int):
        else:
            mensaje = random.choice(MENSAJES_ERROR)
            # Mostrar también la respuesta correcta (texto)
            item = self.preguntas[self.indice]
            respuesta_correcta_texto = item["opciones"][item["correcta"]]
            messagebox.showwarning("Sigue intentando",
                                   f"{mensaje}\n\nRespuesta correcta: {respuesta_correcta_texto}")

        self.puntaje_lbl.config(text=f"Puntaje: {self.puntaje}")
        self.indice += 1
        self.root.after(200, self.cargar_pregunta)  # pequeño delay para UX

    def finalizar(self):
        for b in self.btns:
            b["state"] = "disabled"
        porcentaje = int((self.puntaje / self.total) * 100)
        if self.puntaje >= 8:
            texto = (f"¡Nivel completado! 🎉\n"
                     f"Tu puntaje: {self.puntaje}/{self.total} ({porcentaje}%)\n"
                     "¡Excelente trabajo! Puedes pasar al siguiente nivel cuando quieras.")
        else:
            texto = (f"¡Buen intento! 🌟\n"
                     f"Tu puntaje: {self.puntaje}/{self.total} ({porcentaje}%)\n"
                     "Sigue practicando y mejorarás cada vez más. ¡Tú puedes! 💪")
        messagebox.showinfo("Resumen del nivel", texto)
```

Datos iniciales:
Definí una lista global PREGUNTAS NIVEL 3 con 10 preguntas tipo "¿Cuánto es X + Y?" o "¿Cuánto es X - Y?", cada una con 4 opciones y la posición de la respuesta correcta. También creé dos listas de mensajes (MENSAJES ACIERTO y MENSAJE ERROR) para dar retroalimentación emocional positiva.

Interfaz gráfica:
Configuré la ventana principal con tamaño fijo, colores suaves y tipografía legible.
Dividí la interfaz en tres secciones: encabezado (título y progreso), cuerpo (pregunta y botones de opciones) y pie (puntaje y botón de reinicio). Los botones de opciones se generan dinámicamente en una cuadrícula de 2x2, con estilos visuales atractivos y efectos de hover.

Lógica del juego:
Al iniciar, las preguntas se mezclan aleatoriamente para evitar memorización por orden.
Para cada pregunta, también se barajan las opciones, pero se guarda internamente cuál es la correcta.
Cuando el usuario selecciona una opción, se valida si es correcta, se actualiza el puntaje, se muestra un mensaje animador y se avanza a la siguiente pregunta automáticamente tras un pequeño delay. Si el usuario falla, se le muestra la respuesta correcta para reforzar el aprendizaje.

Finalización y reinicio:
Al terminar las 10 preguntas, se muestra un mensaje con el puntaje total y un porcentaje de aciertos, junto con una retroalimentación motivadora según el desempeño. El botón “Reiniciar nivel” permite volver a jugar desde cero, con nuevas posiciones de preguntas y opciones.

Detalles técnicos:
Usé random.shuffle() para mezclar tanto las preguntas como las opciones. Utilicé messagebox.showinfo() para mostrar mensajes emergentes. Implementé after() para crear un pequeño retardo entre preguntas, mejorando la experiencia de usuario.
En conclusión esta aplicación ayuda a concientizar acerca de la falta que hace fortalecer los temas básicos de matemáticas en los grados más básicos de la sociedad como es la primaria, y también para reforzar, lo básico.