🎮 Curso: Programación en Python con Videojuegos – PyRacers

📅 **Duración:** 16 semanas (1 clase semanal)
👥 **Edad sugerida:** 11 a 16 años
📌 **Objetivo:** Aprender Python desde cero mientras se desarrolla un videojuego completo con Pygame, integrando conceptos de programación, lógica, estructuras de datos y diseño de software.

🧠 Clases 1 a 8 – Introducción a Python + Fundamentos del Juego

Estas clases introducen Python mientras los estudiantes construyen los cimientos del videojuego **PyRacers**.

📍 Clase 1 – ¿Qué es programar? Primer contacto con Python y Pygame

Teoría:

- ¿Qué es la programación? ¿Qué es Python?
- Diferencias con otros lenguajes (JavaScript, Scratch).
- Sintaxis básica: `print()`, comentarios `#`, identación.

Práctica:

- Configurar entorno con `run.py`.
- Analizar `main.py` y `screen.py`: loop principal, inicialización.

- Ejecutar el menú de inicio.

📍 Clase 2 – Variables, tipos de datos y estructuras básicas

Teoría:

- Variables, constantes.
- Tipos de datos: `int`, `float`, `str`, `bool`.
- Entrada de usuario con `input()` y operadores matemáticos.

Práctica:

- Mostrar fondo (`road.py`), aplicar coordenadas.
- Usar variables para posición del jugador.

📍 Clase 3 – Condicionales y eventos

Teoría:

- `if`, `else`, `elif`.
- Operadores lógicos (`and`, `or`, `not`) y relacionales.

Práctica:

- Usar `pygame.key.get_pressed()` para mover al jugador.
- Detectar teclas izquierda/derecha (`player.py`).

📍 Clase 4 – Ciclos y estructuras repetitivas

Teoría:

- `for`, `while`, `range()`.
- Instrucciones `break` y `continue`.

Práctica:

- Crear enemigos estáticos en el juego (`EnemyFactory`).
- Movimiento descendente básico con `StillMovement`.

📍 Clase 5 – Funciones y parámetros

Teoría:

- `def nombre_funcion(parámetros):`
- Uso de `return`, reutilización de código.

Práctica:

- Implementar enemigos zigzag con `ZigZagMovement`.
- Modularizar creación de enemigos en `game.py`.

📍 Clase 6 – Listas y grupos

Teoría:

- Listas: `append()`, `remove()`, `len()`, `for item in lista`.
- Uso de `pygame.sprite.Group`.

Práctica:

- Agrupar enemigos y power-ups.
- Colisiones con `pygame.sprite.spritecollide()`.

📍 Clase 7 – Clases y objetos

Teoría:

- Definición de clase: `class`, `__init__`.
- Métodos y atributos. Qué es `self`.

Práctica:

- Analizar `Player`, `Enemy`, `PowerUp`.
- Mostrar salud visual del jugador con `updateHealth`.

📍 Clase 8 – Herencia, modularidad y arquitectura del juego

Teoría:

- Herencia: `class SubClase(ClasePadre)`.
- Modularidad: separar lógica en archivos.
- Observer pattern: `Publisher` y `Subscriber`.

Práctica:

- Aplicar `notifyAll()` para afectar entidades con power-ups.

🚀 Clases 9 a 16 – Desarrollo avanzado y finalización del juego

---

📍 Clase 9 – Sistema de power-ups

- Crear Blue y Pink con sus efectos (`Frozen`, `Limitless`).
- Aplicar efecto a enemigos con `updateSub`.

📍 Clase 10 – Sonidos del juego

- Agregar `pygame.mixer.init()` en `sounds.py`.
- Sonidos de colisión, arranque, motor acelerado.

📍 Clase 11 – Combustible y aceleración

- Crear sistema de combustible con tecla `Z`.
- Disminuir fuel al acelerar y aumentar distancia recorrida.

📍 Clase 12 – Sistema de vidas con RainbowCar

- Detectar colisiones con auto multicolor.

- Sumar vidas hasta un máximo y actualizar imagen.

📍 Clase 13 – Flujo del juego completo

- Revisar lógica de `runGame()`, `frame_count`, tiempo.
- Validar condiciones de fin del juego y reinicio.

📍 Clase 14 – Refactorización y optimización

- Reorganizar funciones, limpiar código.
- Comentar correctamente cada módulo.

📍 Clase 15 – Personalización y toque final

- Cambiar sprites, imágenes y sonidos.
- Probar distintas dificultades o fondos.

📍 Clase 16 – Presentación de proyectos

- Cada alumno muestra su juego terminado.
- Reciben feedback y certificado.
- Posible mini-torneo o votación de “mejor diseño”.

💻 Requisitos Técnicos

📌 Hardware mínimo

- PC o notebook con Windows / Linux / macOS.
- Procesador Intel Core i3 (8va gen) o AMD Ryzen 3 o superior.
- 4 GB de RAM (8 GB recomendados).
- Cámara web y micrófono (solo para soporte online).
- Resolución mínima: 1366x768 px.

🔧 Software necesario

- **Python 3.10+**
- **Librerías:** `pygame`, `os`, `random`, `subprocess`.
- **Editor recomendado:**
    - Thonny IDE (ideal para principiantes)
    - VS Code + extensiones de Python
    - PyCharm (opcional)

🌐 Alternativa sin instalación

- **Google Colab** + entorno con Pygame preinstalado (*requiere cuenta de Google*).
- *Solo para pruebas básicas gráficas, no ideal para videojuegos interactivos.*