CURSO PYTHON JUNIOR
PLAN DE ESTUDIOS

Presentación

El Curso Python Junior Sólido es la etapa final del recorrido Coding Club hacia la programación textual. Durante 70 semanas de 60 minutos, adolescentes de 13 a 16 años que ya dominan entornos visuales (Scratch, RPG Maker, Lua Roblox) se adentran en Python con el objetivo de alcanzar un nivel comparable al PCAP Certified Associate.

El plan se organiza en bloques progresivos que conectan lo visual con la sintaxis escrita, amplían el repertorio con comprehensions, programación funcional, clases y herencia, e incorporan herramientas profesionales como GitHub, linters, pruebas automatizadas y CI. En la segunda mitad del curso los estudiantes desarrollan habilidades de algoritmia intermedia, bases de datos ligeras, micro APIs web, visualización de datos y un primer contacto con asincronía.

Cada bloque culmina en un producto funcional que refuerza el nuevo conocimiento y alimenta un portafolio personal. El curso concluye con un proyecto capstone de 500 1000 líneas: una aplicación documentada, testeada y desplegada públicamente.

Al terminar, los estudiantes pueden escribir código idiomático, depurarlo con criterio, colaborar en repositorios Git y resolver problemas algorítmicos estándar, quedando listos para certificarse en PCEP/PCAP y seguir especializaciones en desarrollo web, ciencia de datos o IA.

Principios didácticos

Puente visual-textual gradual: se preserva el apoyo gráfico mientras se introduce la sintaxis.

- Aprendizaje basado en proyectos (ABP): cada módulo culmina en un producto funcional.
- Andamiaje cognitivo: complejidad creciente sin exceder la capacidad del aprendiz.
- Metacognición y depuración: reflexión explícita sobre el propio código y estrategias de prueba.
- Colaboración y cultura de juego seguro: trabajo en equipos reducidos y retroalimentación positiva.

Contenidos:

1. **Introducción & Diagnóstico**
   I. Entorno (venv, VS Code)
   II. Repaso de lógica básica
   III. Script “Hello World” interactivo

2. **Fundamentos de Sintaxis & Control**
   I. Tipos, if/elif/else
   II. Bucles
   III. Match/case
   IV. F strings
   V. Minijuego de adivinanza

3. **Colecciones & Programación Funcional**
   I. Comprehensions
   II. Lambda
   III. Map/filter
   IV. Itertools
   V. Generadores
   VI. Generador de contraseñas

4. **Modularidad & OOP**
   I. Módulos, empaquetado, clases, herencia, dunder methods
   II. Biblioteca de figuras geométricas

5. **Entorno Profesional**
   I. Pip, linters, pre commit, flujo GitHub
   II. Repo con README y CI de linters

6. **Pruebas & Depuración**
   I. Pytest, unittest.mock, pdb
   II. Manejo de excepciones
   III. Cobertura de tests ≥ 70 %

7. **Estructuras de Datos & Algoritmia**
   I. Complejidad, búsquedas, ordenamientos, recursión
   II. DP light
   III. Problemas ICPC Bronze resueltos

8. Acceso a Datos & SQLite

I. CSV/JSON/TXT
II. SQLModel ORM
III. CRUD
IV. Agenda CLI persistente

9. Web & APIs

I. Flask/FastAPI
II. Jinja2
III. Requests, autenticación token básica
IV. Micro API + cliente consumidor

10. Data Science & Visualización

I. Pandas
II. Limpieza de datos
III. Matplotlib/plotly
IV. Dashboard interactivo

11. Concurrencia & Asíncronía (intro)

I. Asyncio, tareas concurrentes I/O
II. Downloader concurrente

12. Proyecto Capstone

I. Planificación ágil
II. Sprints
III. Despliegue
IV. App de 500 1000 líneas en GitHub/Replit