Actumlogos
La palabra reflexionada puesta en acción

Python Intermedio: de scripts a código sólido

Curso • 16 horas • Zoom

Este curso te lleva **de escribir scripts que funcionan a construir código sólido**: legible, tipado, validado y listo para crecer. Es el paso que separa a quien resuelve problemas con scripts sueltos de quien entiende cómo se comporta Python por dentro, organiza un proyecto como lo hace un equipo profesional y escribe código en el que otros pueden confiar. Es, también, la base que exige trabajar en serio con datos, Machine Learning, LLMs y agentes de IA. Cada tema se trabaja en dos planos: el concepto, para entender qué hace Python y por qué, y el código, para verlo en acción. El recorrido está pensado como una construcción. Primero entenderás el modelo de objetos (referencias, mutabilidad y el costo real de list, dict y set) y llevarás las funciones a fondo hasta escribir tus primeros decoradores. Después aplicarás la programación orientada a objetos con criterio, estructurarás un proyecto real con uv y pyproject.toml, y harás más confiable tu código con type hints, mypy y Pydantic V2. Con esa base procesarás archivos de gran tamaño sin saturar la RAM, trabajarás datos con NumPy y Polars y consumirás APIs con httpx y async/await. Todo converge en un proyecto integrador que recorre el ciclo de vida completo del código y los datos. Y lo harás con Python 3.15 y sus novedades: lazy imports, profiling nativo con Tachyon y el ABI estable para free-threading.

Proyecto integrador: CLI de enriquecimiento y análisis de datos

- **Inicio y empaquetado** (sesiones 1 y 4): el proyecto se inicializa con uv y pyproject.toml, arranca al instante gracias a los lazy imports de Python 3.15 y deja trazabilidad con logging.
- **Lectura eficiente** (sesión 6): procesa por lotes un archivo JSONL masivo con un generador (yield) y context managers (with open(...)), sin saturar la RAM.
- **Modelado y validación** (sesiones 3 y 5): los registros se modelan con dataclasses y se validan estrictamente en tiempo de ejecución con type hints y Pydantic V2.
- **Metaprogramación y resiliencia** (sesión 2): un decorador @retry propio, construido con *args y **kwargs, protege las llamadas de red propensas a fallos.
- **Enriquecimiento asíncrono** (sesión 8): httpx y async/await lanzan cientos de peticiones concurrentes a una API externa y enriquecen los datos filtrados en una fracción del tiempo habitual.
- **Procesamiento analítico** (sesión 7): los datos enriquecidos se consolidan en un DataFrame de Polars, con expresiones lazy para extraer métricas y agregaciones a máxima velocidad y NumPy para transformaciones contiguas si es necesario.

Herramientas: Python, uv, mypy, Pydantic V2, NumPy, Polars, httpx, asyncio, logging, dataclasses, Tachyon

Contenido

1. **Modelo de objetos**
   - 1.1. Referencias, mutabilidad y copias (shallow vs. deep)
   - 1.2. Costo espacial y temporal de estructuras: list, dict y set
   - 1.3. Novedades de Python 3.15: el impacto del ABI estable para free-threading (ejecución sin GIL)
2. **Funciones a fondo**
   - 2.1. Empaquetado y desempaquetado de argumentos: *args y **kwargs
   - 2.2. Ámbito de variables y closures
   - 2.3. Introducción a la metaprogramación: primeros decoradores
3. **POO práctica**
   - 3.1. Diseño de software: composición vs. herencia
   - 3.2. Métodos mágicos (dunders) para emular comportamiento nativo
   - 3.3. Encapsulamiento con @property
   - 3.4. Estructuras eficientes con dataclasses
4. **Proyecto real en Python**
   - 4.1. Manejo avanzado de excepciones y trazabilidad con logging
   - 4.2. Estructura de paquetes y módulos
   - 4.3. Gestión de dependencias ultrarrápida con uv
   - 4.4. Configuración de proyectos modernos con pyproject.toml
   - 4.5. Novedades de Python 3.15: optimización de arranque con lazy imports y profiling nativo con Tachyon
5. **Tipos y validación**
   - 5.1. Type hints y análisis estático de código con mypy
   - 5.2. Validación de datos robusta en tiempo de ejecución con Pydantic V2
6. **Iteradores y recursos**
   - 6.1. Protocolo de iteración y creación de generadores (yield)
   - 6.2. Gestión segura de recursos con context managers (with)
   - 6.3. Caso práctico: parseo eficiente de archivos JSONL de gran tamaño sin saturar la RAM
7. **Datos**
   - 7.1. NumPy: fundamentos de vectorización y reglas de broadcasting
   - 7.2. Polars: introducción a expresiones y evaluación perezosa (lazy)
8. **APIs y cierre**
   - 8.1. Consumo de servicios externos y peticiones HTTP con httpx
   - 8.2. Serialización y manejo avanzado de JSON
   - 8.3. Introducción a la concurrencia: conceptos básicos de async/await
   - 8.4. Presentación y revisión final del proyecto integrador

Ya viste lo que vas a aprender.

Asegura tu lugar

Inscríbete ahora · $1,600 MXN →

Constancia con registro STPS · Grabaciones por 1 año · Acceso a Zoom al instante

¿Dudas antes de inscribirte? Escríbenos por WhatsApp

Al terminar tendrás:

- Un modelo mental claro de cómo Python maneja objetos, referencias y mutabilidad, para evitar errores silenciosos con las copias y elegir entre list, dict y set según su costo en tiempo y memoria.
- Dominio a fondo de las funciones: *args y **kwargs, ámbito de variables, closures y decoradores propios, como un @retry que protege tus llamadas de red.
- Criterio de diseño orientado a objetos: cuándo usar composición o herencia, cómo emular comportamiento nativo con métodos mágicos y cómo encapsular con @property y dataclasses.
- La habilidad de estructurar un proyecto como lo hace un equipo profesional: paquetes y módulos, excepciones bien manejadas, logging, dependencias con uv y configuración en pyproject.toml.
- Código más confiable gracias a los type hints, el análisis estático con mypy y la validación de datos en tiempo de ejecución con Pydantic V2.
- La capacidad de procesar archivos de gran tamaño sin saturar la RAM con generadores y context managers, y de analizar datos con NumPy (vectorización y broadcasting) y Polars (expresiones y evaluación lazy).
- Experiencia consumiendo APIs con httpx, manejando JSON y lanzando peticiones concurrentes con async/await.
- Las novedades de Python 3.15 que importan: lazy imports para acelerar el arranque, profiling nativo con Tachyon y el impacto del free-threading (ejecución sin GIL).
- Una CLI de enriquecimiento y análisis de datos terminada, que integra todo lo anterior y que puedes mostrar en tu portafolio.
- Constancia de participación verificable en línea.

¿Para quién es este curso?

- Personas que ya programan en Python a nivel básico (variables, condicionales, ciclos, funciones, listas y diccionarios) y quieren dar el salto a escribir código sólido y mantenible.
- Científicos de datos, analistas e ingenieros de Machine Learning cuyos notebooks y scripts funcionan, pero cuesta mantenerlos, reutilizarlos o compartirlos con su equipo.

- Desarrolladores que vienen de otro lenguaje y quieren escribir Python idiomático y aprovechar su ecosistema moderno: uv, mypy, Pydantic y Polars.

Este curso NO es para ti si:

- Nunca has programado en Python: el curso parte de que ya manejas variables, condicionales, ciclos, funciones, listas y diccionarios.
- Ya escribes Python avanzado (metaclases, descriptores, asyncio a fondo, optimización de rendimiento): aquí se consolida el nivel intermedio y la concurrencia se ve solo como introducción.
- Buscas un curso de Machine Learning, desarrollo web o despliegue: aquí el foco es el lenguaje y su ecosistema, y NumPy, Polars y el consumo de APIs se ven como herramientas para trabajar con datos.
- Prefieres solo escuchar teoría: el curso está orientado a la práctica y culmina en un proyecto integrador, así que rinde a quien está dispuesto a escribir código.

Requisitos

Conocimientos básicos de Python: variables, condicionales, ciclos, funciones, listas y diccionarios. Necesitas una computadora de escritorio o laptop de 64 bits (Windows, macOS o Linux) en la que puedas instalar software, y una conexión a Internet de al menos 5 Mbps. Trabajarás en tu propia computadora con Python 3.15 o superior y un editor de código, por ejemplo VS Code.

Clases

8 sesiones de 2 horas (16 h en total): martes, miércoles y jueves de 20:00 a 22:00 (hora de la Ciudad de México), los días 8, 9, 10, 15, 16, 17, 22 y 23 de diciembre de 2026. Todo se imparte por Zoom y queda grabado; tendrás acceso a las grabaciones durante 1 año para repasar a tu ritmo.

Instructor

M. en T. de Cómputo Saulo Gante — LinkedIn

Testimonios

Hemos capacitado a más de 2,000 personas en IA — visita actumlogos.com

Medios de pago

Tarjetas de crédito/débito, Link, PayPal, transferencia SPEI, Mercado Pago, Apple Pay, Google Pay. Ver checkout

Actumlogos
La palabra reflexionada puesta en acción

Constancia de participación con folio único, verificable en línea

Al terminar recibirás una constancia de participación firmada por el instructor y el director de Actumlogos — agente capacitador registrado ante la Secretaría del Trabajo y Previsión Social. Cada constancia incluye un folio único con el que cualquier persona puede verificar su autenticidad en línea, y podrás compartirla directamente en LinkedIn.

Registro STPS: ZAGE-810930-FW2-0005

Este es un ejemplo para ilustrar qué esperar:





¿Buscas un tema diferente o te perdiste alguna edición?

Únete a nuestra comunidad en WhatsApp para decidir los próximos temas y enterarte con anticipación.

Lista de Espera