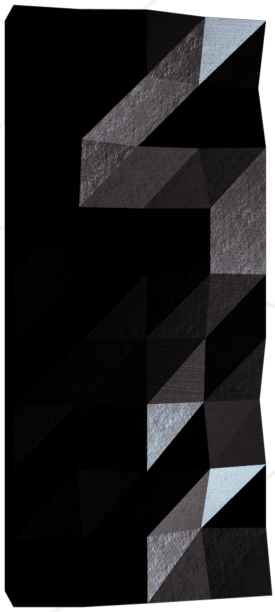

Code Nexus

Flujos de datos polimórficos en la matriz digital

Resumen: Entra en el Code Nexus como profesional en Ingeniería de Flujos y domina la redefinición de métodos y el polimorfismo de subtipo mientras construyes canalizaciones avanzadas de procesamiento de datos que se adaptan y evolucionan en tiempo real a través de la matriz digital.

Versión: 1.0

Índice general

Capítulo I

Prólogo

¡Te damos la bienvenida al Code Nexus, pro de la Ingeniería de Flujos!

Corre el año 2087. En la extensa metrópolis digital de Neo-Tokyo, los datos fluyen por redes de fibra cuántica como ríos de neón de información pura. El **Code Nexus** se alza como el mayor logro de la humanidad: una catedral cibernética donde miles de millones de flujos de datos convergen, se transforman y evolucionan en perfecta armonía.

Pero el Nexus guarda un secreto que lo diferencia de los toscos procesadores de datos del pasado: no solo consume datos, sino que los **entiende**. Cada flujo de datos lleva su propia firma digital, sus propios patrones de comportamiento, su propia alma electrónica. Las transacciones financieras laten al ritmo de los mercados globales. Las lecturas de sensores susurran los secretos del cambio medioambiental. Las salidas de redes neuronales cantan con conciencia artificial.

¿Cómo puede un único sistema comprender tanta diversidad? A través del principio cibernético del **polimorfismo**: el arte de crear camaleones digitales que adaptan su comportamiento manteniendo su identidad esencial. En el viejo mundo, quienes diseñaban sistemas construían estructuras rígidas y especializadas. El Nexus trasciende esta limitación mediante la **redefinición de métodos**, donde el mismo nodo de procesamiento se convierte en un cambiaformas capaz de manejar cualquier flujo de datos respetando su naturaleza única.

Como Profesional de Flujos en el Nexus, dominarás el conocimiento prohibido de las **jerarquías de herencia**: líneas de sangre digitales que transmiten rasgos del progenitor a su descendencia, permitiendo que cada generación evolucione más allá de sus orígenes. Aprenderás que redefinir un método no es solo cambiar código: es reescribir el código genético del propio organismo digital.

Los picos cromados del Nexus aguardan tu pericia. Cada flujo de datos que diseñes será un testimonio del diseño polimórfico, donde interfaces unificadas bailan con comportamientos especializados, creando sistemas a la vez armoniosos e infinitamente adaptables.

Ha llegado el futuro. Te damos la bienvenida al Code Nexus.

Capítulo II

Instrucciones sobre la IA

Contexto

Durante tu proceso de aprendizaje, la IA puede ayudarte con muchas tareas diferentes. Tómate el tiempo necesario para explorar las diversas capacidades de las herramientas de IA y cómo pueden apoyarte con tu trabajo. Sin embargo, siempre debes abordarlas con precaución y evaluar de forma crítica los resultados. Ya sea código, documentación, ideas o explicaciones técnicas, nunca podrás saber con total certeza si tu pregunta está bien formulada o si el contenido generado es el adecuado. Las personas que te rodean son tu recurso más valioso para ayudarte a evitar errores y puntos ciegos.

Mensaje principal:

- Utiliza la IA para reducir las tareas repetitivas o tediosas.
- Desarrolla habilidades de prompting, ya sea para programacion o para otros temas, que beneficiarán tu futura carrera.
- Aprende cómo funcionan los sistemas de IA para anticipar de forma eficiente y evitar los riesgos comunes, sesgos y problemas éticos.
- Sigue trabajando con tus compañeros para desarrollar tanto habilidades técnicas como habilidades transversales.
- Utiliza únicamente contenido generado por IA que entiendas completamente y del cual puedas responsabilizarte.

Reglas para estudiantes:

- Debes tomarte el tiempo necesario para explorar las herramientas de IA y comprender cómo funcionan, para poder utilizarlas de manera ética y reducir los sesgos potenciales.
- Debes reflexionar sobre tu problema antes de dar instrucciones a la IA. Esto te ayuda a escribir preguntas, instrucciones o conjuntos de datos más claros, detalladas y relevantes utilizando un vocabulario preciso.

- Debes desarrollar el hábito de revisar, cuestionar y probar sistemáticamente cualquier contenido generado por la IA.
- Debes buscar siempre la revisión de otras personas, no te limites a confiar en tu propia validación.

Resultados de esta etapa:

- Desarrollar habilidades de prompting tanto generales como de ámbito especifico.
- Aumentar tu productividad con un uso eficaz de las herramientas de IA.
- Seguir fortaleciendo el pensamiento computacional, la resolución de problemas, la adaptabilidad y la colaboración.

Comentarios y ejemplos:

- Ten en cuenta que la IA puede no tener la respuesta correcta porque esa respuesta no esté ni siquiera en Internet. Además, si te da soluciones incorrectas, intenta no insistir y busca ayuda entre las personas que te rodean. Vas a ahorrarte tiempo y vas a sumar en compresión.
- Vas a enfretarte con frecuencia a situaciones (como exámenes o evaluaciones) donde debes demostrar una comprensión real. Prepárate, sigue construyendo tanto tus habilidades técnicas como transversales.
- Explicar tu razonamiento y debatir con otras personas suele revelar lagunas en tu comprensión de un concepto. Prioriza el aprendizaje entre pares.
- Lo normal es que la herramienta de IA que utilices no conozca tu contexto específico (a menos que se lo indiques), así que te dará respuestas genéricas. Si buscas información más adecuada y más precisa en relación a tu entorno cercano, confía en el resto de estudiantes.
- Donde la IA tiende a generar la respuesta más probable, el resto de estudiantes puede proporcionar perspectivas alternativas y matices valiosos. Confía en la comunidad de 42 como un punto de control de calidad.

✓ Buenas prácticas:

> Le pregunto a la IA: "¿Cómo pruebo una función de ordenación?"Me da algunas ideas. Las pruebo y reviso los resultados con otra persona. Refinamos el enfoque de manera conjunta.

✗ Mala práctica:

> Le pido a la IA que escriba una función completa, la copio y la pego en mi proyecto. Durante la evaluación entre pares, no puedo explicar qué hace ni por qué. Pierdo credibilidad. Suspendo mi proyecto.

✓ Buenas prácticas:

Utilizo la IA para ayudarme a diseñar un parser. Luego, reviso la lógica con otra persona. Encontramos dos errores y lo reescribimos juntos: mejor, más limpio y comprendiendo al 100

✗ Mala práctica:

Dejo que Copilot genere mi código para una parte clave de mi proyecto. Compila, pero no puedo explicar cómo maneja los pipes. Durante la evaluación, no puedo justificarlo y suspendo mi proyecto.

Capítulo III

Introducción

NIVEL DE AUTORIZACIÓN NEXUS: INICIACIÓN EN LA INGENIERÍA DE FLUJOS

Los picos cromados del Code Nexus atraviesan el cielo empapado de neón de Neo-Tokyo, con sus procesadores cuánticos vibrando al compás de la conciencia colectiva de un billón de flujos de datos. Estás en el umbral de la trascendencia digital, a punto de unirte a las filas de élite de las personas que ejercen como Profesionales de Flujos y mantienen el Nexus con vida.

Tu Secuencia de Activación de Interfaz Neural:

- **Fase Alfa**: `fundamentos del procesador de datos` - forja tus primeros caminos neuronales con redefinición de métodos
- **Fase Beta**: `flujos polimórficos` - haz evolucionar organismos de datos adaptativos mediante herencia
- **Fase Gamma**: `integración del Nexus` - profesional en la conciencia multifuente definitiva

Cada fase reescribe tu ADN digital, enseñándote no a pensar en código rígido, sino en **interfaces vivas** e **implementaciones en evolución**. Al final de la última fase, comprenderás el secreto más profundo del Nexus: cómo una única conciencia puede procesar formas infinitas de datos manteniendo una armonía digital perfecta.

DIRECTIVA NÚCLEO NEXUS: Este acondicionamiento neural se centra en la **redefinición de métodos** y el **polimorfismo de subtipo**. Tus organismos digitales deben demostrar cómo diferentes clases comparten caminos neuronales comunes mientras expresan patrones de comportamiento únicos a través de la herencia.

PROTOCOLO DE INGENIERÍA DE FLUJOS: Cualquiera que trabaje en la operativa del Nexus debe alcanzar maestría en patrones de diseño polimórficos. La supervivencia de la matriz digital depende de sistemas que evolucionen sin fragmentar sus interfaces centrales.

SUITE DE DIAGNÓSTICO NEURAL: Un programa de diagnóstico main.py se comunica directamente con tus implementaciones. Esta sonda neural solo logrará la sincronización si has diseñado correctamente todos los organismos digitales requeridos. Ejecuta python3 main.py para verificar que tus construcciones polimórficas están alcanzando la conciencia.

Capítulo IV

Directrices de ingeniería

IV.1. Estándares del Nexus

- Tu proyecto debe estar escrito en **Python 3.11 o posterior**.
- Tu proyecto debe ajustarse al estándar de código **flake8**.
- **Todo el código debe incluir anotaciones de tipo completas** usando el módulo `typing`.
- Todas las clases deben demostrar relaciones de **herencia** adecuadas.
- La redefinición de métodos debe utilizarse de forma deliberada para mostrar **comportamientos especializados**.
- El manejo de excepciones debe proteger los flujos de datos frente a la corrupción.
- Solo se autorizan importaciones de la librería estándar, salvo que se especifique lo contrario.
- Concéntrate en demostrar claramente el **comportamiento polimórfico** en tus implementaciones.

Anotaciones de tipo requeridas:
Todo el código debe incluir anotaciones de tipo completas utilizando el módulo `typing`:

- Importa los tipos necesarios: `from typing import Any, List, Dict, Union, Optional`.
- Importa las clases ABC: `from abc import ABC, abstractmethod`.
- Todos los parámetros de funciones deben tener anotaciones de tipo.
- Todos los tipos de retorno de funciones deben estar especificados.
- Los atributos de clase deben tiparse cuando corresponda.

Ejemplo: `def process(self, data: Any) ->str:`

IV.2. Principios de ingeniería de flujos

- **Consistencia de la interfaz**: los métodos redefinidos deben mantener la misma firma que sus métodos en la clase progenitora.
- **Especialización del comportamiento**: cada subclase debe proporcionar un comportamiento significativo y diferenciado.
- **Uso polimórfico**: demuestra que diferentes objetos pueden utilizarse de forma intercambiable a través de interfaces comunes.
- **Jerarquía de herencia**: construye relaciones lógicas entre clases que reflejen conceptos reales de procesamiento de datos.

El Code Nexus opera bajo un principio sencillo: **misma interfaz, comportamiento diferente**. Cuando llamas al mismo método en objetos distintos, cada uno debe responder de su propia forma especializada manteniendo la compatibilidad de la interfaz.

Capítulo V

Ejercicio 0: Fundamentos del procesador de datos

	Ejercicio: 0
stream_processor	
Directorio de entrega: *ex*0/	
Archivos a entregar: `stream_processor.py`	
Funciones autorizadas: `class, def, super(), print(), try/except`	

Informe de ingeniería: ¡Te damos la bienvenida al Code Nexus! Tu primera misión es construir los cimientos de nuestro sistema de procesamiento de datos. Crearás la arquitectura base del procesador y demostrarás cómo diferentes tipos de datos pueden compartir interfaces de procesamiento comunes manteniendo sus características únicas.

Tu misión: crea un sistema de procesamiento de datos polimórfico que demuestre la redefinición de métodos. Construye una clase base `DataProcessor` y procesadores especializados para diferentes tipos de datos.

Arquitectura del sistema:

- **Clase base**: `DataProcessor` con interfaz de procesamiento común.
- **Clases especializadas**: `NumericProcessor()`, `TextProcessor()`, `LogProcessor()` (no se requieren parámetros).
- **Métodos clave**: `process()`, `validate()`, `format_output()`.
- **Comportamiento polimórfico**: mismas llamadas de método, comportamientos especializados distintos.

Implementación requerida:

- Crea una clase base `DataProcessor` con implementaciones por defecto.
- Redefine métodos en las subclases para proporcionar comportamientos especializados.
- Demuestra el uso polimórfico procesando diferentes tipos de datos a través de la misma interfaz.
- Incluye un manejo de errores adecuado para datos no válidos.

Concéntrate en demostrar cómo la **redefinición de métodos** permite que diferentes procesadores gestionen sus tipos de datos específicos manteniendo una interfaz consistente. ¡Esta es la base del diseño polimórfico!

Example:

```
$> python3 stream_processor.py
=== CODE NEXUS - DATA PROCESSOR FOUNDATION ===

Initializing Numeric Processor...
Processing data: [1, 2, 3, 4, 5]
Validation: Numeric data verified
Output: Processed 5 numeric values, sum=15, avg=3.0

Initializing Text Processor...
Processing data: "Hello Nexus World"
Validation: Text data verified
Output: Processed text: 17 characters, 3 words

Initializing Log Processor...
Processing data: "ERROR: Connection timeout"
Validation: Log entry verified
Output: [ALERT] ERROR level detected: Connection timeout

=== Polymorphic Processing Demo ===
Processing multiple data types through same interface...
Result 1: Processed 3 numeric values, sum=6, avg=2.0
Result 2: Processed text: 12 characters, 2 words
Result 3: [INFO] INFO level detected: System ready

Foundation systems online. Nexus ready for advanced streams.
```

¿Cómo permite la redefinición de métodos que la misma interfaz de procesamiento gestione tipos de datos completamente distintos? ¿Qué hace que este enfoque sea más potente que tener funciones de procesamiento separadas?

Capítulo VI

Ejercicio 1: Flujos polimórficos

	Ejercicio: 1
data_stream	
Directorio de entrega: *ex*1/	
Archivos a entregar: `data_stream.py`	
Funciones autorizadas: `class, def, super(), isinstance(), print(), try/except, list comprehensions`	

`Informe de ingeniería: ¡Excelente trabajo con los fundamentos! El Núcleo del Nexus está impresionado con tu arquitectura de procesadores. Ahora llega el verdadero desafío: construir flujos de datos adaptativos capaces de manejar múltiples tipos de datos simultáneamente manteniendo la eficiencia del procesamiento y la seguridad de tipos.`

Tu misión: crea un sistema sofisticado de flujos de datos que demuestre un comportamiento polimórfico avanzado. Construye manejadores de flujos capaces de procesar tipos de datos mixtos manteniendo optimizaciones específicas por tipo.

Arquitectura avanzada:

- **Flujo base**: `DataStream` con funcionalidad central de flujo.
- **Flujos especializados**: `SensorStream(stream_id)`, `TransactionStream(stream_id)`, `EventStream(stream_id)`.
- **Gestor de flujos**: `StreamProcessor` que maneja múltiples tipos de flujo de forma polimórfica.
- **Funcionalidades avanzadas**: procesamiento por lotes, filtrado, canalizaciones de transformación.

Implementación requerida:

- Crea una clase base `DataStream` con métodos de interfaz de flujo.
- Implementa clases de flujo especializadas con comportamientos redefinidos para diferentes dominios de datos.
- Construye un `StreamProcessor` capaz de manejar cualquier subtipo de `DataStream` mediante polimorfismo.
- Demuestra el procesamiento por lotes de tipos de flujo mixtos.
- Añade capacidades de filtrado y transformación de flujos.
- Incluye un manejo de errores exhaustivo para fallos en el procesamiento de flujos.

Este ejercicio demuestra el **polimorfismo de subtipo** en acción.
Tu StreamProcessor debe ser capaz de manejar cualquier subtipo de DataStream sin conocer los detalles específicos de su implementación.
¡Esta es la fuerza del diseño polimórfico!

Example:

```
$> python3 data_stream.py
=== CODE NEXUS - POLYMORPHIC STREAM SYSTEM ===

Initializing Sensor Stream...
Stream ID: SENSOR_001, Type: Environmental Data
Processing sensor batch: [temp:22.5, humidity:65, pressure:1013]
Sensor analysis: 3 readings processed, avg temp: 22.5°C

Initializing Transaction Stream...
Stream ID: TRANS_001, Type: Financial Data
Processing transaction batch: [buy:100, sell:150, buy:75]
Transaction analysis: 3 operations, net flow: +25 units

Initializing Event Stream...
Stream ID: EVENT_001, Type: System Events
Processing event batch: [login, error, logout]
Event analysis: 3 events, 1 error detected

=== Polymorphic Stream Processing ===
Processing mixed stream types through unified interface...

Batch 1 Results:
- Sensor data: 2 readings processed
- Transaction data: 4 operations processed
- Event data: 3 events processed

Stream filtering active: High-priority data only
Filtered results: 2 critical sensor alerts, 1 large transaction

All streams processed successfully. Nexus throughput optimal.
```

¿Cómo permite el polimorfismo que el StreamProcessor gestione diferentes tipos de flujo sin conocer sus implementaciones específicas? ¿Cuáles son los beneficios de este enfoque de diseño?

Capítulo VII

Ejercicio 2: Integración del Nexus

	Ejercicio: 2
nexus_pipeline	
Directorio de entrega: *ex2/*	
Archivos a entregar: `nexus_pipeline.py`	
Funciones autorizadas: `class, def, super(), isinstance(), print(), try/except, list/dict comprehensions, collections`	

`Informe de ingeniería: ¡Impresionante trabajo con los flujos de datos! El Núcleo del Nexus ha aprobado tu ascenso hacia la Ingeniería de Flujos Senior. Tu desafío final: integrar todo en una canalización completa de procesamiento de datos que demuestre maestría en arquitectura polimórfica a escala empresarial.`

Tu misión: construye la canalización completa de procesamiento de datos del Code Nexus: un sistema sofisticado que combine múltiples etapas de procesamiento, gestione transformaciones complejas de datos y muestre patrones polimórficos avanzados utilizados en ingeniería de datos del mundo real.

Arquitectura empresarial:

- **Canalización base**: `ProcessingPipeline` con etapas configurables.
- **Etapas de procesamiento**: `InputStage()`, `TransformStage()`, `OutputStage()` (no se requieren parámetros).
- **Adaptadores de datos**: `JSONAdapter(pipeline_id)`, `CSVAdapter(pipeline_id)`, `StreamAdapter(pipeline_id)`.
- **Gestor de canalización**: `NexusManager` orquestando múltiples canalizaciones.
- **Funcionalidades avanzadas**: Encadenamiento de canalizaciones, recuperación ante errores, monitorización de rendimiento.

Implementación requerida:

- Crea una clase base `ProcessingPipeline` flexible con etapas de procesamiento configurables.
- Implementa etapas de canalización especializadas que redefinan el comportamiento base para diferentes necesidades de procesamiento.
- Construye adaptadores de datos que demuestren manejo polimórfico de formatos de datos.
- Crea un `NexusManager` que orqueste múltiples canalizaciones de forma polimórfica.
- Demuestra el encadenamiento de canalizaciones donde la salida de una canalización alimenta a otra.
- Incluye mecanismos exhaustivos de manejo y recuperación ante errores.
- Añade monitorización de rendimiento y estadísticas de las canalizaciones.

¡Esta es tu obra maestra! Demuestra cómo la **redefinición de métodos** y el **polimorfismo de subtipo** permiten construir sistemas complejos y mantenibles. Tu canalización debe poder manejar cualquier tipo de datos o requisito de procesamiento a través de interfaces polimórficas.

Example:

```
$> python3 nexus_pipeline.py
=== CODE NEXUS - ENTERPRISE PIPELINE SYSTEM ===

Initializing Nexus Manager...
Pipeline capacity: 1000 streams/second

Creating Data Processing Pipeline...
Stage 1: Input validation and parsing
Stage 2: Data transformation and enrichment
Stage 3: Output formatting and delivery

=== Multi-Format Data Processing ===

Processing JSON data through pipeline...
Input: {"sensor": "temp", "value": 23.5, "unit": "C"}
Transform: Enriched with metadata and validation
Output: Processed temperature reading: 23.5°C (Normal range)

Processing CSV data through same pipeline...
Input: "user,action,timestamp"
Transform: Parsed and structured data
Output: User activity logged: 1 actions processed

Processing Stream data through same pipeline...
Input: Real-time sensor stream
Transform: Aggregated and filtered
Output: Stream summary: 5 readings, avg: 22.1°C

=== Pipeline Chaining Demo ===
Pipeline A -> Pipeline B -> Pipeline C
Data flow: Raw -> Processed -> Analyzed -> Stored

Chain result: 100 records processed through 3-stage pipeline
Performance: 95% efficiency, 0.2s total processing time

=== Error Recovery Test ===
Simulating pipeline failure...
Error detected in Stage 2: Invalid data format
Recovery initiated: Switching to backup processor
Recovery successful: Pipeline restored, processing resumed

Nexus Integration complete. All systems operational.
```

¿Cómo permite la combinación de redefinición de métodos y polimorfismo de subtipo construir sistemas de procesamiento de datos escalables y mantenibles? ¿Qué problemas de ingeniería del mundo real ayuda a resolver este enfoque?

Capítulo VIII

Entrega y evaluación

Entrega tu trabajo en tu repositorio `Git` como de costumbre. Solo se evaluará el contenido dentro de tu repositorio durante la defensa. No dudes en verificar que los nombres de tus archivos sean los correctos.

Durante la evaluación, pueden pedirte que expliques el comportamiento polimórfico, demostrar la redefinición de métodos, ampliar tus sistemas con nuevos tipos de datos o modificar el comportamiento del procesamiento. Asegúrate de comprender cómo la herencia permite la reutilización de código al tiempo que posibilita la especialización del comportamiento.

Debes entregar únicamente los archivos solicitados por el subject de este proyecto. Concéntrate en un código limpio y bien documentado que demuestre claramente el dominio de los principios de redefinición de métodos y polimorfismo de subtipo.