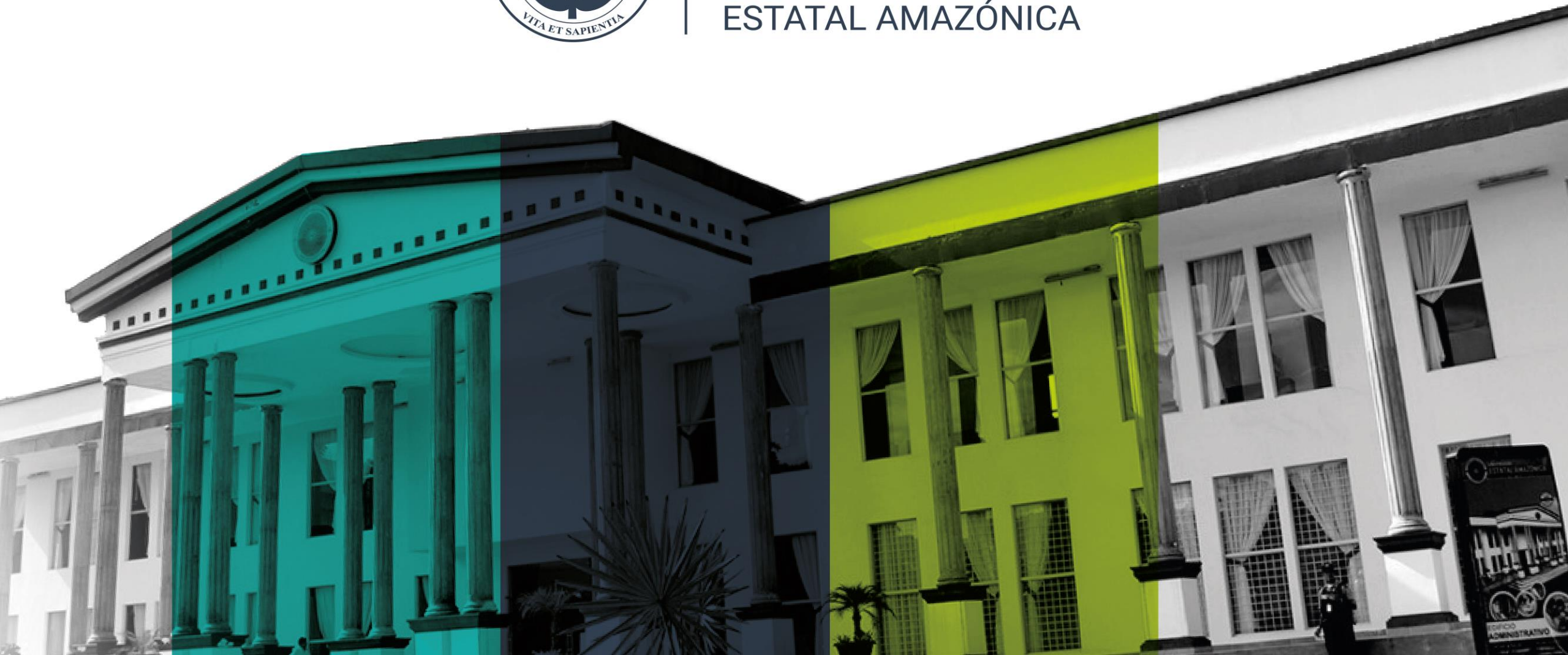
UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA



UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

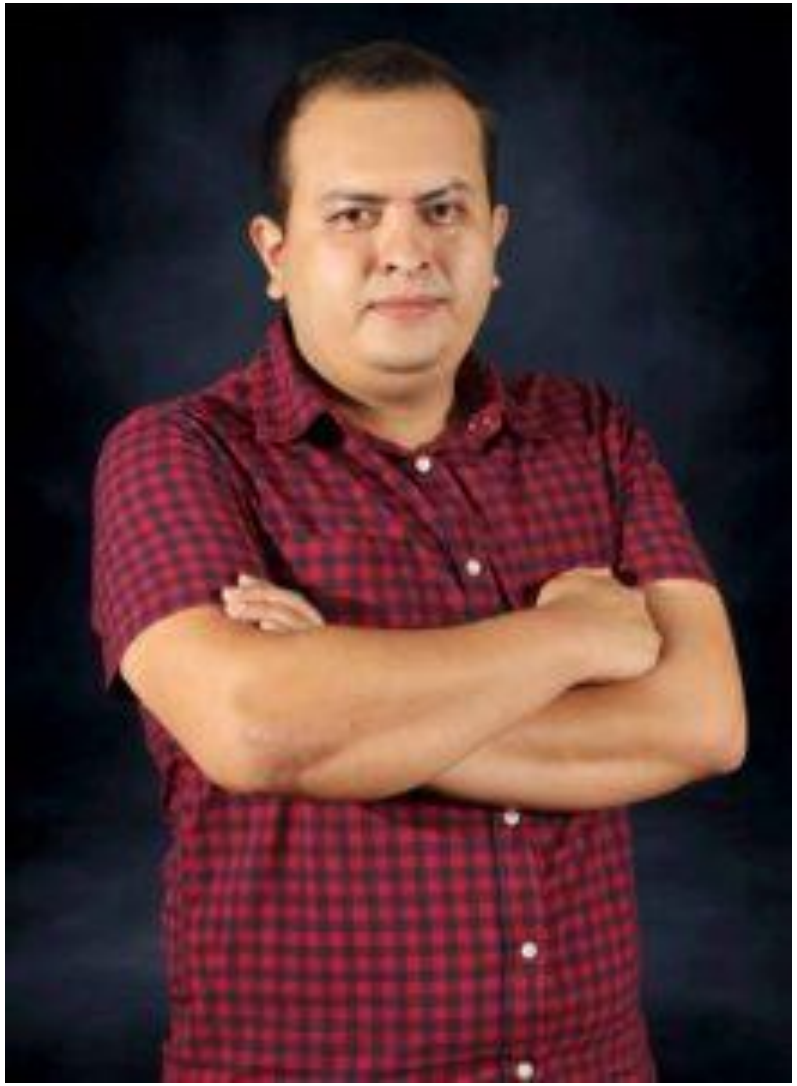
ASIGNATURA

PROGRAMACIÓN ORIENTADA A OBJETOS

Transformamos el mundo desde la Amazonía

Ing. Edwin Gustavo Fernández Sánchez, Mgs.
DOCENTE - PERSONAL ACADÉMICO NO TITULAR OCASIONAL
DIRECTOR DE GESTIÓN DE TECNOLOGÍAS INFORMACIÓN Y COMUNICACIÓN

UNIVERSIDAD ESTATAL AMAZÓNICA



UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

SEMANA
7

DESARROLLO DE LA SEMANA 7: DEL LUN. 20 AL DOM. 26 DE ENERO/2025

Resultado de aprendizaje: Resultado de aprendizaje: Aplicar programas que hacen uso de relación de herencia, interfaces y clases abstractas con métodos y variables polimórficas.

CONTENIDOS

UNIDAD II: Objetos, clases, Herencia, Polimorfismo.

- Tema 2.2: Constructores y destructores
  - Subtema 2.2.1: Fundamentos de Constructores y destructores

UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

Unidad 2

Objetos, clases, Herencia, Polimorfismo.

Tema 2.

Constructores y destructores

UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

Subtema 2.1: Fundamentos de Constructores y destructores

PROGRAMACIÓN ORIENTADA A OBJETOS
(UEA-L-UFB-030)

UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

¿Qué son Constructores y Destructores?

En la Programación Orientada a Objetos (POO), los constructores y destructores son métodos especiales que juegan roles cruciales en la vida de un objeto.

El constructor es un método que se llama automáticamente al crear una nueva instancia de una clase, mientras que el destructor se llama cuando un objeto está a punto de ser destruido o eliminado.

UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

El Rol del Constructor

El propósito principal del constructor es inicializar el objeto. Esto incluye asignar valores a los atributos del objeto y configurar el estado inicial necesario para el funcionamiento del objeto.

En Python, el constructor se define mediante el método especial __init__. Este método se llama automáticamente al crear un objeto, permitiendo que el programador establezca los atributos y realice cualquier configuración inicial.

UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

El Rol del Destructor

Por otro lado, el destructor, definido mediante el método especial __del__ en Python, se utiliza para liberar recursos antes de que el objeto sea destruido.

El destructor puede ser útil para tareas como cerrar conexiones a bases de datos o archivos, o para limpiar recursos que el objeto haya consumido durante su vida útil.

UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

Fundamentos del Constructor

El constructor en la Programación Orientada a Objetos es un bloque de código especial que se ejecuta cuando se crea una nueva instancia de una clase.

En Python, el constructor se define a través del método __init__. Este método tiene la responsabilidad de iniciar los atributos del objeto, estableciendo un estado inicial adecuado.

Características de __init__

El método __init__ puede aceptar parámetros, además del parámetro self, que representa la instancia del objeto. Estos parámetros se utilizan para pasar valores iniciales a los atributos del objeto.

Por ejemplo, en una clase Persona, el constructor podría inicializar atributos como nombre, edad y direccion.

```python
class Persona:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad
```

UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

Importancia del Constructor

El constructor es crucial porque garantiza que el objeto se encuentre en un estado válido antes de ser utilizado. Sin una inicialización adecuada, los objetos podrían comportarse de manera inesperada o causar errores.

Los constructores también mejoran la legibilidad y la estructura del código, proporcionando un punto claro para la configuración inicial del objeto.

Un buen entendimiento del constructor es esencial para cualquier programador de POO. Permite diseñar clases de manera que sus instancias siempre empiecen con un estado definido y consistente.

UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

Pasos para Definir un Constructor

Crear un constructor en Python implica definir el método __init__ dentro de una clase. Este método se invoca automáticamente al crear una nueva instancia de la clase.

El primer parámetro de __init__ es siempre self, que es una referencia a la instancia actual del objeto. Los parámetros adicionales se utilizan para pasar datos al objeto durante su creación

```python
class MiClase:
    def __init__(self, parametro1, parametro2):
        self.atributo1 = parametro1
        self.atributo2 = parametro2
```

Aquí, parametro1 y parametro2 son argumentos que se pasan al constructor, y atributo1 y atributo2 son atributos del objeto que se inicializan con estos valores.

UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

Inicialización de Atributos

- El propósito principal del constructor es **inicializar los atributos del objeto**. Esto establece un estado inicial claro y controlado para el objeto.
- La inicialización de atributos dentro del constructor garantiza que **cada instancia de la clase** comience con sus propios valores de datos específicos.
- Es una buena práctica hacer que los constructores inicialicen todos los atributos esenciales del objeto. Esto **evita errores causados por atributos no inicializados** más adelante.
- También se pueden establecer **valores predeterminados** para algunos parámetros, lo que aumenta la flexibilidad a la hora de crear instancias de la clase.

UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

Aplicación Práctica de Constructores

Cada vez que creamos una nueva instancia de una clase, el constructor de esa clase se invoca automáticamente, permitiendo que los datos necesarios se asignen a los atributos del objeto.

python Copy code

```python
class Libro:
    def __init__(self, titulo, autor):
        self.titulo = titulo
        self.autor = autor

mi_libro = Libro("Cien Años de Soledad", "Gabriel García Márquez")
```

Consideremos una clase Libro con un constructor definido. La creación de un objeto Libro se vería así en Python: Aquí, mi_libro es una instancia de la clase Libro, y se le pasan 'Cien Años de Soledad' y 'Gabriel García Márquez' como argumentos al constructor.

UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

Inicialización de Objetos

- **El constructor garantiza que mi_libro se inicializa correctamente** con su titulo y autor antes de que el objeto se utilice en cualquier otro lugar del programa.
- Esta práctica asegura que los **objetos sean consistentes y predecibles** en su comportamiento, lo cual es crucial para mantener la integridad del código.
- Los constructores también **aportan flexibilidad en la creación de objetos**, permitiendo que se inicialicen con diferentes valores para representar entidades únicas.

UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

Concepto de Destructor

En la Programación Orientada a Objetos, un destructor es un método especial que se ejecuta cuando un objeto está a punto de ser destruido o eliminado del espacio de memoria.

En Python, el destructor se define a través del método __del__. Aunque Python maneja automáticamente la recolección de basura, el destructor proporciona un espacio para definir la limpieza necesaria antes de que el objeto se elimine completamente.

Uso del Destructor en Python

El método __del__ se llama automáticamente cuando todas las referencias a un objeto se eliminan o cuando el programa termina. Su uso principal es para realizar operaciones de limpieza, como cerrar archivos o conexiones a bases de datos, o liberar recursos externos que el objeto podría haber adquirido durante su vida útil.

```python
class Archivo:
    def __init__(self, nombre_archivo):
        self.archivo = open(nombre_archivo, 'r')

    def __del__(self):
        self.archivo.close()
        print("Archivo cerrado correctamente.")
```

UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

Precauciones con los Destructores

- Es importante ser **cauteloso** al usar destructores en Python, ya que la gestión automática de memoria del lenguaje generalmente se encarga de la mayoría de las tareas de limpieza.
- Un **uso indebido** del destructor podría conducir a comportamientos inesperados, especialmente en situaciones con múltiples referencias al mismo objeto o en entornos de ejecución concurrente.
- Aunque **no siempre son necesarios en Python** debido a su eficiente recolector de basura, entender los destructores es importante para ciertos casos donde se requiere una gestión explícita de recursos.
- Proporcionan un mecanismo adicional para asegurar que los recursos se liberen adecuadamente, contribuyendo a un **software más limpio y eficiente**.

UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

Definición de un Destructor

En Python, un destructor se define mediante el método especial **__del__** dentro de una clase. **Este método se invoca automáticamente cuando un objeto está a punto de ser destruido**, generalmente cuando se elimina su última referencia o al finalizar el programa.

Aunque la gestión de memoria en Python se maneja a través de su recolector de basura, el destructor permite definir acciones específicas de limpieza para asegurar que los recursos utilizados por el objeto sean liberados adecuadamente.

UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

Cómo Crear un Destructor

La estructura de un destructor es similar a la de cualquier otro método dentro de una clase, pero con la particularidad de llamarse __del__.

Este método no toma ningún parámetro aparte de self, que es una referencia al objeto que se está destruyendo."

En este ejemplo, la clase ConexiónBaseDatos establece una conexión en su constructor y la cierra en su destructor. Esto asegura que la conexión se cierre correctamente, independientemente de cómo o cuándo se elimine el objeto.

python Copy code

```python
class ConexiónBaseDatos:
    def __init__(self):
        self.conexion = establecer_conexion()  # Suponiendo una funció

    def __del__(self):
        self.conexion.cerrar()  # Suponiendo un método para cerrar la
        print("Conexión a base de datos cerrada.")
```

UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

Consideraciones Importantes

- **Es crucial no depender exclusivamente de los destructores** para la liberación de recursos externos, especialmente en Python, donde el tiempo exacto de la llamada al destructor no es predecible debido a la recolección de basura.
- Sin embargo, **los destructores pueden ser útiles en ciertos escenarios**, como la liberación de recursos no gestionados, la limpieza de conexiones o el cierre de archivos.
- **Implementar un destructor en Python puede ser una herramienta valiosa en situaciones específicas**. Proporciona un método adicional para controlar y gestionar el ciclo de vida de un objeto, contribuyendo a un manejo más eficiente de los recursos y a la estabilidad del programa.

UEA

UNIVERSIDAD
ESTATAL AMAZÓNICA