## \*\*Nivel 1: Principiante\*\*

### 1. \*\*Fundamentos de Programación\*\*

- \*\*Lenguajes recomendados:\*\* Python y SQL.

- \*\*Temas clave:\*\*

- Sintaxis básica, variables, tipos de datos.

- Estructuras de control (condicionales, bucles).

- Funciones, clases y objetos (POO).

- Manejo de excepciones y debugging.

- \*\*Proyectos:\*\*

- Script básico de análisis de datos (por ejemplo, estadísticas simples de ventas).

- Conexión básica a una base de datos con SQL.

- \*\*Herramientas:\*\*

- Jupyter Notebooks

- Git y control de versiones básico.

### 2. \*\*Matemáticas y Estadística Básica\*\*

- \*\*Temas clave:\*\*

- Álgebra lineal básica.

- Probabilidad y distribuciones.

- Descriptiva (media, mediana, varianza, desviación estándar).

- \*\*Proyectos:\*\*

- Análisis estadístico básico de un conjunto de datos (por ejemplo, dataset de iris).

- \*\*Herramientas:\*\*

- Python con librerías como `NumPy` y `Pandas`.

### 3. \*\*Introducción a los Datos y su Análisis\*\*

- \*\*Temas clave:\*\*

- Tipos de datos (estructurados, no estructurados).

- Limpieza y preprocesamiento de datos.

- Exploración de datos (EDA: Exploratory Data Analysis).

- \*\*Proyectos:\*\*

- Limpieza de un dataset sucio y visualización de insights básicos.

- \*\*Herramientas:\*\*

- `Pandas`, `Matplotlib`, `Seaborn`.

---

## \*\*Nivel 2: Intermedio\*\*

### 1. \*\*Ingeniería de Datos (Parte I)\*\*

- \*\*Temas clave:\*\*

- Bases de datos relacionales y no relacionales.

- SQL avanzado: joins, subqueries, optimización de consultas.

- Diseño de bases de datos, normalización.

- ETL (Extract, Transform, Load) y Data Pipelines.

- \*\*Proyectos:\*\*

- Diseñar y construir un pequeño pipeline ETL para mover datos entre bases de datos.

- \*\*Herramientas:\*\*

- `MySQL`, `PostgreSQL`, `MongoDB`, `Apache Airflow`.

### 2. \*\*Ciencia y Análisis de Datos (Parte I)\*\*

- \*\*Temas clave:\*\*

- Técnicas avanzadas de análisis estadístico.

- Inferencia estadística y pruebas de hipótesis.

- Introducción a la regresión (lineal, logística).

- Reducción de dimensionalidad (PCA, SVD).

- \*\*Proyectos:\*\*

- Crear un modelo predictivo simple utilizando regresión lineal.

- \*\*Herramientas:\*\*

- `scikit-learn`, `StatsModels`.

### 3. \*\*Visualización de Datos\*\*

- \*\*Temas clave:\*\*

- Principios de visualización de datos.

- Creación de gráficos complejos y Dashboards.

- Visualización de datos geoespaciales.

- \*\*Proyectos:\*\*

- Desarrollar un dashboard interactivo para mostrar insights de un conjunto de datos.

- \*\*Herramientas:\*\*

- `Matplotlib`, `Seaborn`, `Plotly`, `Tableau`, `PowerBI`.

---

## \*\*Nivel 3: Avanzado\*\*

### 1. \*\*Ingeniería de Datos (Parte II)\*\*

- \*\*Temas clave:\*\*

- Data Warehousing (almacenamiento de datos) y Data Lakes.

- Procesamiento distribuido (Hadoop, Spark).

- Integración y orquestación de pipelines de datos en producción.

- \*\*Proyectos:\*\*

- Diseñar un sistema de almacenamiento de datos y procesamiento distribuido.

- \*\*Herramientas:\*\*

- `Apache Hadoop`, `Apache Spark`, `AWS S3`, `Kafka`.

### 2. \*\*Ciencia de Datos (Parte II)\*\*

- \*\*Temas clave:\*\*

- Técnicas avanzadas de modelado predictivo (árboles de decisión, random forests, SVM).

- Análisis de series temporales.

- Técnicas de muestreo, validación cruzada.

- Modelos de clustering (K-means, DBSCAN).

- \*\*Proyectos:\*\*

- Implementar un modelo predictivo para series temporales (forecasting de ventas).

- \*\*Herramientas:\*\*

- `scikit-learn`, `XGBoost`, `TensorFlow`.

### 3. \*\*Big Data y Cloud Computing\*\*

- \*\*Temas clave:\*\*

- Introducción a infraestructuras en la nube (AWS, Google Cloud, Azure).

- Procesamiento de Big Data en la nube.

- Serverless computing y gestión de datos en la nube.

- \*\*Proyectos:\*\*

- Crear y gestionar un pipeline de datos en la nube con AWS o Google Cloud.

- \*\*Herramientas:\*\*

- `AWS Lambda`, `GCP BigQuery`, `Azure Data Factory`.

---

## \*\*Nivel 4: Experto\*\*

### 1. \*\*Inteligencia Artificial y Machine Learning Avanzado\*\*

- \*\*Temas clave:\*\*

- Redes neuronales profundas (Deep Learning).

- Redes convolucionales (CNN), recurrentes (RNN), LSTM.

- Transfer learning y autoencoders.

- Generative Adversarial Networks (GANs).

- Reinforcement learning.

- \*\*Proyectos:\*\*

- Desarrollar un modelo de clasificación de imágenes usando CNNs.

- Implementar un modelo de texto predictivo con RNN o LSTM.

- \*\*Herramientas:\*\*

- `TensorFlow`, `PyTorch`, `Keras`.

### 2. \*\*Optimización de Modelos y MLOps\*\*

- \*\*Temas clave:\*\*

- Optimización de hiperparámetros (GridSearch, RandomSearch).

- Interpretación de modelos y explicabilidad (SHAP, LIME).

- MLOps: Implementación de pipelines de Machine Learning en producción.

- Despliegue de modelos en la nube.

- \*\*Proyectos:\*\*

- Implementar un pipeline completo de MLOps para un modelo de ML en producción.

- \*\*Herramientas:\*\*

- `MLflow`, `Kubeflow`, `Docker`, `Kubernetes`.

### 3. \*\*Ingeniería de Datos (Parte III)\*\*

- \*\*Temas clave:\*\*

- Optimización de arquitecturas de datos.

- Integración de IA en sistemas de Big Data.

- Seguridad de datos y privacidad en sistemas distribuidos.

- \*\*Proyectos:\*\*

- Diseñar una arquitectura escalable que integre IA y procesamiento de datos en tiempo real.

- \*\*Herramientas:\*\*

- `Apache Flink`, `Snowflake`, `Presto`.

---

### \*\*Proyectos Finales\*\*

- \*\*Científico de Datos:\*\* Crear un modelo predictivo o de clustering de datos de alto impacto utilizando técnicas avanzadas de machine learning.

- \*\*Analista de Datos:\*\* Desarrollar un dashboard completo que integre análisis de datos y visualización para una toma de decisiones empresarial.

- \*\*Ingeniero de Datos:\*\* Diseñar una arquitectura de datos escalable que integre procesamiento en tiempo real, big data y AI.

- \*\*Experto en IA:\*\* Desarrollar una aplicación basada en Deep Learning, como un clasificador de imágenes o un chatbot basado en modelos de lenguaje.

---

Con este pensum, tendrás una guía integral para convertirte en un experto en análisis, ingeniería de datos y creación de inteligencias artificiales, comenzando desde los conceptos más básicos hasta las aplicaciones más avanzadas.

¡Me alegra que te haya gustado el pensum! Empezaremos con la \*\*Clase 1\*\* del nivel principiante: \*\*Fundamentos de Programación en Python\*\*. Nos enfocaremos en los conceptos esenciales para que puedas entender cómo programar desde cero.

**Clase 1: Fundamentos de Programación en Python**

#### \*\*Objetivos:\*\*

1. Entender qué es un lenguaje de programación.

2. Aprender los conceptos básicos de programación en Python: variables, tipos de datos y operaciones básicas.

3. Practicar con ejemplos sencillos.

4. Dejar una tarea práctica que te acerque a los proyectos finales.

---

### \*\*1. ¿Qué es un lenguaje de programación?\*\*

Un lenguaje de programación es una herramienta que nos permite decirle a la computadora qué hacer. Python es uno de los lenguajes más populares hoy en día, especialmente en ciencia de datos, análisis de datos e inteligencia artificial, debido a su simplicidad y poder.

\*\*Ejemplo básico:\*\*

Cuando escribimos `print("Hello, world!")` en Python, estamos diciéndole a la computadora que nos muestre la frase "Hello, world!" en pantalla.

---

### \*\*2. Variables y Tipos de Datos en Python\*\*

Una \*\*variable\*\* es un contenedor que almacena datos. En Python, no necesitas especificar el tipo de dato al crear una variable (es un lenguaje de tipado dinámico), ya que Python lo asigna automáticamente.

#### \*\*Tipos de datos comunes:\*\*

1. \*\*Enteros (int):\*\* Números sin decimales. Ej: `3`, `45`, `-100`.

2. \*\*Flotantes (float):\*\* Números con decimales. Ej: `3.14`, `-0.001`.

3. \*\*Cadenas de texto (str):\*\* Textos entre comillas. Ej: `"hola"`, `"Python"`.

4. \*\*Booleanos (bool):\*\* Verdadero o falso. Ej: `True`, `False`.

#### \*\*Sintaxis:\*\*

```python

# Ejemplo de variables en Python

edad = 25 # Variable de tipo entero

pi = 3.14159 # Variable de tipo flotante

nombre = "Juan" # Variable de tipo string

es\_estudiante = True # Variable booleana

```

---

### \*\*3. Operaciones Básicas\*\*

Python permite realizar operaciones básicas fácilmente, como suma, resta, multiplicación y división.

```python

# Operaciones matemáticas

x = 10

y = 5

suma = x + y # Suma

resta = x - y # Resta

multiplicacion = x \* y # Multiplicación

division = x / y # División

# Mostrando los resultados

print("Suma:", suma)

print("Resta:", resta)

print("Multiplicación:", multiplicacion)

print("División:", division)

```

---

### \*\*4. Comentarios\*\*

Los comentarios son partes del código que no se ejecutan. Se utilizan para hacer anotaciones.

```python

# Esto es un comentario de una línea

```

Para comentarios de varias líneas:

```python

"""

Esto es un comentario

de varias líneas.

"""

```

---

### \*\*5. Entrada de Datos\*\*

Python permite solicitar información al usuario mediante `input()`. Todo lo que se recibe a través de `input()` es de tipo `str` (cadena de texto), pero se puede convertir a otros tipos.

```python

# Solicitar nombre y edad del usuario

nombre = input("¿Cuál es tu nombre? ")

edad = int(input("¿Cuántos años tienes? "))

# Mostrar los datos

print("Hola", nombre, ", tienes", edad, "años.")

```

---

### \*\*Ejemplo Completo: Calculadora de Edad\*\*

Apliquemos lo que hemos aprendido creando un programa simple que pregunte al usuario su nombre y edad, y luego calcule cuántos años tendrá dentro de 5 años.

```python

# Pedir al usuario su nombre y edad

nombre = input("Por favor, ingresa tu nombre: ")

edad = int(input("Por favor, ingresa tu edad: "))

# Calcular la edad dentro de 5 años

edad\_futura = edad + 5

# Mostrar el resultado

print("Hola", nombre, ". En 5 años tendrás", edad\_futura, "años.")

```

---

### \*\*Tarea Práctica 1: Calculadora de Tiempo de Trabajo\*\*

Para la primera tarea práctica, crearás un programa que le pregunte al usuario cuántos años ha trabajado en su empresa, y luego calcularás cuántos días ha trabajado (suponiendo que trabaja todos los días del año). Este ejercicio te ayudará a consolidar los fundamentos y es un paso pequeño hacia la lógica que utilizarás en análisis de datos.

#### \*\*Requerimientos:\*\*

1. Pregunta al usuario su nombre.

2. Pregunta cuántos años ha trabajado en su empresa.

3. Calcula cuántos días ha trabajado en total (suponiendo que trabaja los 365 días del año).

4. Muestra un mensaje con el total de días trabajados.

#### \*\*Formato del código (Guía):\*\*

```python

# Pedir al usuario su nombre

nombre = input("¿Cuál es tu nombre? ")

# Pedir cuántos años ha trabajado

anios\_trabajados = int(input("¿Cuántos años has trabajado en tu empresa? "))

# Calcular cuántos días ha trabajado en total

dias\_trabajados = anios\_trabajados \* 365

# Mostrar el resultado

print(nombre, ", has trabajado un total de", dias\_trabajados, "días.")

```

---

### \*\*Conclusión de la Clase 1\*\*

En esta primera clase, aprendiste los fundamentos de Python, incluyendo variables, tipos de datos y operaciones básicas. Todo lo que hemos cubierto hoy te servirá como base para trabajar con datos más complejos en el futuro. Además, la tarea práctica te ayudará a afianzar estos conceptos y a empezar a pensar en lógica algorítmica.

---

### \*\*Próximas Clases:\*\*

- En la siguiente clase, aprenderemos sobre estructuras de control de flujo (condicionales y bucles) para hacer nuestros programas más dinámicos.

Si tienes alguna pregunta sobre lo que hemos visto o la tarea, ¡estaré encantado de ayudarte!