guayerd

Fundamentos IA

Introducción IA y datos

Clase 4

En colaboración con

IBM **SkillsBuild**

¡Bienvenidos!

¿Nos presentamos?

- ¿Qué recuerdan de la clase anterior?
- ¿Qué esperan aprender?
- ¿Tienen alguna pregunta?

guayerd

En colaboración con

IBM **SkillsBuild**

Contenidos

Por temas

01
- Introducción IA

02
- Fundamentos del dato
- Pensamiento computacional

03
- Introducción Python

04
- Introducción a Python

guayerd

En colaboración con

IBM **SkillsBuild**

Objetivos de la clase

- El arte de preguntar
- Estructuras de control
- Funciones
- Numpy

guayerd

En colaboración con

IBM **SkillsBuild**

Introducción a la IA y los datos

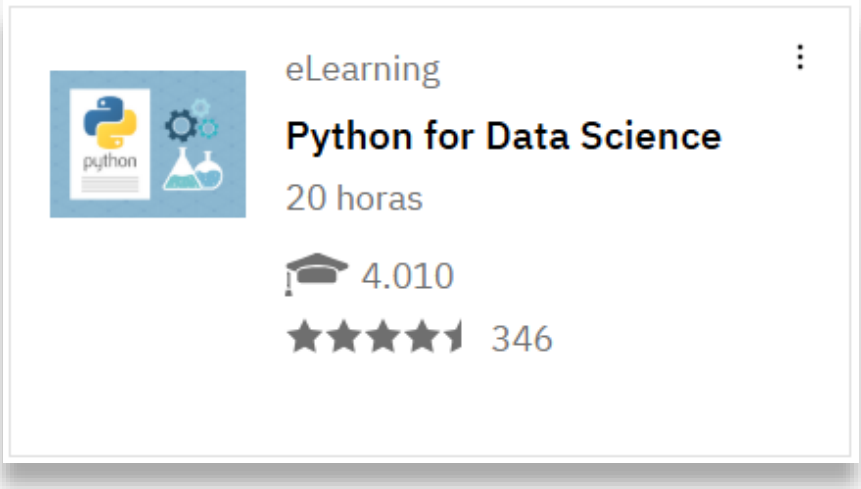
Introducción a Python

guayerd

En colaboración con

IBM **SkillsBuild**

Plataforma Skill Build: Python



guayerd

En colaboración con

IBM **SkillsBuild**

El arte de preguntar

- Orientan decisiones y evitan la improvisación
- Transforman datos en conocimiento útil
- Abren caminos hacia soluciones innovadoras
- Conectan problemas con objetivos claros
- Son la base de todo proceso analítico

Tipos de indagación

Explorar

¿Qué patrones existen?

Comparar

¿En qué se diferencian?

Explicar

¿Por qué sucede esto?

Anticipar

¿Qué ocurrirá después?

Estructurar cualquier análisis

Problema

- Qué [acción/resultado] queremos lograr
- Por qué [este problema] es prioritario
- Cuándo [tiempo/plazo] necesitamos respuestas

Datos

- Qué [información] tenemos disponible
- Cómo [organizar/limpiar] para analizarla
- Qué [limitaciones] debemos considerar

Enfoque

- Qué [relaciones/patrones] explorar primero
- Qué [variables] incluir o descartar
- Cómo [validar] que los resultados sean confiables

Aplicación en Machine Learning

Fórmula

[Verbo de acción] + [objeto específico] + [criterio de éxito]

Aplicación en Machine Learning

Ejemplos:

- ¿Predecir qué clientes abandonarán con +85% precisión?
- ¿Clasificar productos defectuosos minimizando falsos negativos?

Preguntas de validación:

- ¿Los datos representan el problema real?
- ¿El modelo mejora la decisión actual?
- ¿Existen sesgos o consideraciones éticas?

Estructuras de control

Definen **cuándo** y **bajo qué condiciones** se ejecutan las instrucciones.

Tipos

- **Condicionales:** ejecutan código si se cumple una condición
- **Bucles:** repiten código mientras la condición sea verdadera

Son la base para que un programa deje de ser lineal y pase a ser interactivo

Condicionales

Sintaxis

```
if condición:
    # código si la condición es verdadera
elif otra_condición:
    # código si la anterior fue falsa y esta es verdadera
else:
    # código si ninguna condición es verdadera
```

Ejemplo

```
edad = 20
if edad >= 18:
    print("Mayor de edad")
else:
    print("Menor de edad")
```

Bucles

For

Recorre elementos de una secuencia (listas, cadenas, rangos, etc.)

```
for variable in secuencia:
    # código
```

While

Repite mientras una condición sea verdadera

```
while condición:
    # código
```

Bucles

Control

break: interrumpe el bucle

```
for i in range(5):
        if i ==3:
            break
```

continue: salta a la siguiente iteración

```
for i in range(5):
        if i ==2:
            continue
```

pass: no hace nada (placeholder)

```
for i in range(3):
        pass
```

range() genera secuencias

Bucles

Patrones de iteración

Estrategias para recorrer y procesar elementos en una secuencia.

Acumulador: sumar o multiplicar valores

```
suma = 0
for x in numeros:
        suma += x
```

Contador: contar elementos que cumplen condición

```
contar = 0
for x in numeros:
        if x > 10:
                contar += 1
```

Buscador: encontrar un elemento específico

```
for x in numeros:
        if x == 15:
                print("Encontrado")
                break
```

Filtro: procesar sólo ciertos elementos

```
for x in numeros:
        if x % 2 ==0:
                print(x)
```

Funciones

Bloques reutilizables de código que realizan tareas específicas

Ventajas

- Evitan repetir código
- Facilitan el mantenimiento
- Mejoran la organización del programa

Sintaxis

```
# Definir

def nombre_funcion(parametros):
        # código
        return resultado # opcional

# Llamar

nombre_funcion(parametros)
```

Funciones

Parámetro y retorno

Las funciones pueden recibir datos (parámetros) y devolver resultados con return.

Ejemplo

```
def cuadrado(x):
    return x * x

resultado = cuadrado(5)
```

Extra:
Las variables dentro de una función son locales, mientras que las de afuera son globales

Funciones

Integradas

Categoría	Función	Descripción	Ejemplo
E/S	print()	Mostrar en pantalla	print("Hola")
	input()	Recibir datos	nombre = input("Nombre: ")
Conversión	int()	Convertir a entero	int("5") → 5
	float()	Convertir a decimal	float("3.14") → 3.14
	str()	Convertir a texto	str(10) → "10"
Matemáticas	abs()	Valor absoluto	abs(-5) → 5
	round()	Redondear	round(3.7) → 4
	min() / max()	Mínimo y máximo	max(2,5,1) → 5
	sum()	Suma lista	sum([1,2,3]) → 6
Colecciones	len()	Longitud	len("Hola") → 4
	range()	Secuencia números	range(5) → 0,1,2,3,4
	sorted()	Ordenar	sorted([3,1,2]) → [1,2,3]

Registro de temperaturas

1. Solicitar la temperatura de 5 días consecutivos
2. Mostrar la temperatura promedio, máxima y mínima de la semana
3. Contar cuántos días tuvieron temperatura alta (mayor a 25°C)
4. Mostrar un resumen completo de los datos registrados

¿Qué son las librerías?

Conjuntos de herramientas que **amplían las capacidades de Python**, ya que el lenguaje sólo incluye recursos básicos.

Ventajas

- Permiten trabajar eficientemente con grandes volúmenes de datos.
- Evitan tener que programar soluciones que ya están optimizadas.
- Aprovechan años de desarrollo y conocimiento especializado.

NumPy
(Numerical Python)

Librería esencial para el **trabajo con datos numéricos** en Python.

- Procesamiento eficiente de arrays multidimensionales
- Operaciones matemáticas y estadísticas optimizadas
- Base fundamental de librerías como Pandas y Scikit-learn
- Rendimiento superior para cálculos numéricos masivos

Comandos

- **Instalación:** pip install numpy
- **Uso:** import numpy as np

Accede a funciones con alias.nombre_funcion()

Arrays en NumPy

Estructura de datos optimizada para cálculos numéricos, similar a las listas pero con mayor eficiencia.

- Almacena elementos del mismo tipo de dato
- Soporta operaciones en una y múltiples dimensiones
- Permite operaciones matemáticas vectorizadas
- Procesamiento simultáneo de todos los elementos

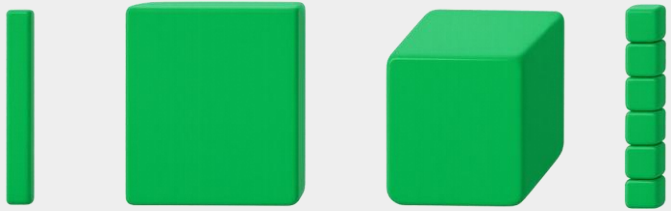
Velocidad de procesamiento hasta 100 veces superior a listas Python tradicionales.

Dimensiones

Número de ejes o direcciones que tiene un array

2D - Matriz
matriz con filas y columnas

4D+ - Tensores multidimensionales
estructuras de datos complejas



1D - Vector
secuencia lineal de elementos

3D - Tensor
múltiples matrices apiladas

Sintaxis básica

De arrays

Importación y creación

```
import numpy as np
vector = np.array([elementos])
matriz = np.array([[fila1], [fila2]])
```

Acceso

```
array[índice]        # Elemento específico
array[inicio:fin]    # Rango de elementos
```

Modificación

```
array[índice] = valor   # Cambiar elemento
array[:] = valor        # Cambiar todos los elementos
```

shape muestra dimensiones

Operaciones vectorizadas

- **Array con escalar:** array + número, array * número, array ** número
- **Array con array:** array1 + array2, array1 > array2 (mismo tamaño)
- **Funciones universales:** np.sqrt(), np.sin(), np.abs(), np.exp(), np.log()
- **Operaciones de comparación:** array > valor (genera array booleano)
- **Operaciones lógicas:** array1 & array2, array1 | array2

Importante:
Una operación se aplica automáticamente a todos los elementos del array

Arrays especiales

NumPy ofrece **funciones para crear arrays con patrones específicos** sin definir elementos manualmente.

- **Valores constantes:** np.zeros(), np.ones(), np.full()
- **Secuencias numéricas:** np.arange(), np.linspace()
- **Aleatorios:** np.random.rand(), np.random.randint()
- **Estructura:** np.eye(), np.empty()

Propiedades básica

De arrays

Comandos esenciales para **inspeccionar y entender la estructura de arrays.**

Información dimensional

- **array.shape:** dimensiones del array
- **array.ndim:** número total de dimensiones
- **array.size:** cantidad total de elementos

Información del contenido

- **array.dtype:** tipo de datos almacenados
- **array.itemsize:** tamaño en bytes de cada elemento
- **array.nbytes:** memoria total ocupada

Los arrays son la base de toda la inteligencia artificial moderna

Primeras ventas

Acabas de abrir tu primera tienda y quieres analizar tus primeras 10 ventas para entender el arranque del negocio.

1. Solicitar al usuario el monto de las primeras 10 ventas
2. Calcular el promedio de estas ventas iniciales
3. Identificar cuáles ventas estuvieron por encima del promedio
4. Calcular el total recaudado en estas primeras ventas
5. Determinar cuál fue tu mejor y peor venta inicial

Proyecto

Tienda Aurelion

- **Documentación:** notebook Markdown
- **Desarrollo técnico:** programa Python
- **Visualización de datos:** dashboard en Power BI
- **Presentación oral:** problema, solución y hallazgos

guayerd

En colaboración con

IBM **SkillsBuild**

Codificación

Trabajo en equipo

En relación al programa en Python que se desarrollará:

1. Crea el **diagrama de flujo**
2. Desarrolla el **programa interactivo**

Preparación demo

próxima clase

guayerd

En colaboración con

IBM **SkillsBuild**

Características

1° demo: asincrónica

- **Tema:** Inteligencia Artificial
- **Medio:** Carpeta personal en Drive
- **Fecha límite:** 20/10
- **Apertura para la entrega:** 6/10
- **Devolución:** En salitas por equipo

Fechas actualizadas

Contenido

1° demo: asincrónica

Documentación (.md)

- Tema, problema y solución
- Dataset de referencia: fuente, definición, estructura, tipos y escala
- Información, pasos, pseudocódigo y diagrama del programa
- Sugerencias y mejoras aplicadas con Copilot

Programa (.py)

- Debe permitir obtener información del proyecto

Instrucciones (.md)

- Instrucciones para Copilot

Criterios

1° demo: asincrónica

- **Tema, problema y solución** claros, vinculados a la base de datos brindada
- **Fuente, definición, estructura, tipos y escala** según lo trabajado en clase 2
- **Pasos, pseudocódigo y diagrama** que represente el desarrollo del programa en Python
- Sugerencias de **Copilot** aceptadas y descartadas
- **Programa en Python** interactivo que permita consultar la documentación de manera interactiva y sin errores de ejecución

Retro

¿Cómo nos vamos?

- ¿Qué fue lo más útil de la clase?
- ¿Qué parte te costó más?
- ¿Qué te gustaría repasar o reforzar?

guayerd

En colaboración con

IBM **SkillsBuild**