



Principios de programación y algoritmos en el ámbito biomédico

Una introducción a la inteligencia artificial

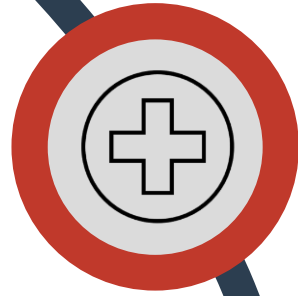
Juan Esteban Morales Mendoza
Bioingeniero

M.D. Bioingeniería - Inteligencia Artificial (en curso)

Joven investigador

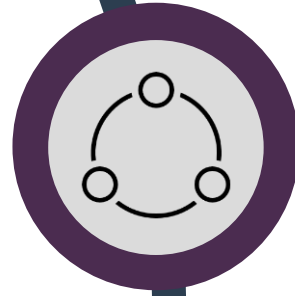
Instituto de Gerencia y Gestión Sanitaria
jemoralesme@unisanitas.edu.co

AGENDA



01 - INTELIGENCIA ARTIFICIAL EN EL ÁMBITO BIOMÉDICO

- Casos reales de IA aplicada para la resolución de problemas en salud
- Algoritmos y programación como pilares de la IA



02 - ALGORITMOS Y PROGRAMACIÓN

- ¿Qué es un algoritmo?
- Representación de un algoritmo (diagrama de flujo y pseudocódigo)
- Relación entre algoritmo y programación



03 - PRINCIPIOS DE PROGRAMACIÓN

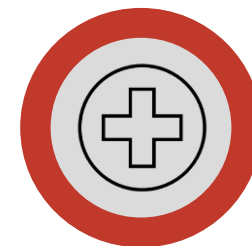
- ¿Qué es un lenguaje de programación?
- Constantes y variables
- Tipos de datos y operadores
- Estructuras de datos
- Estructuras de control



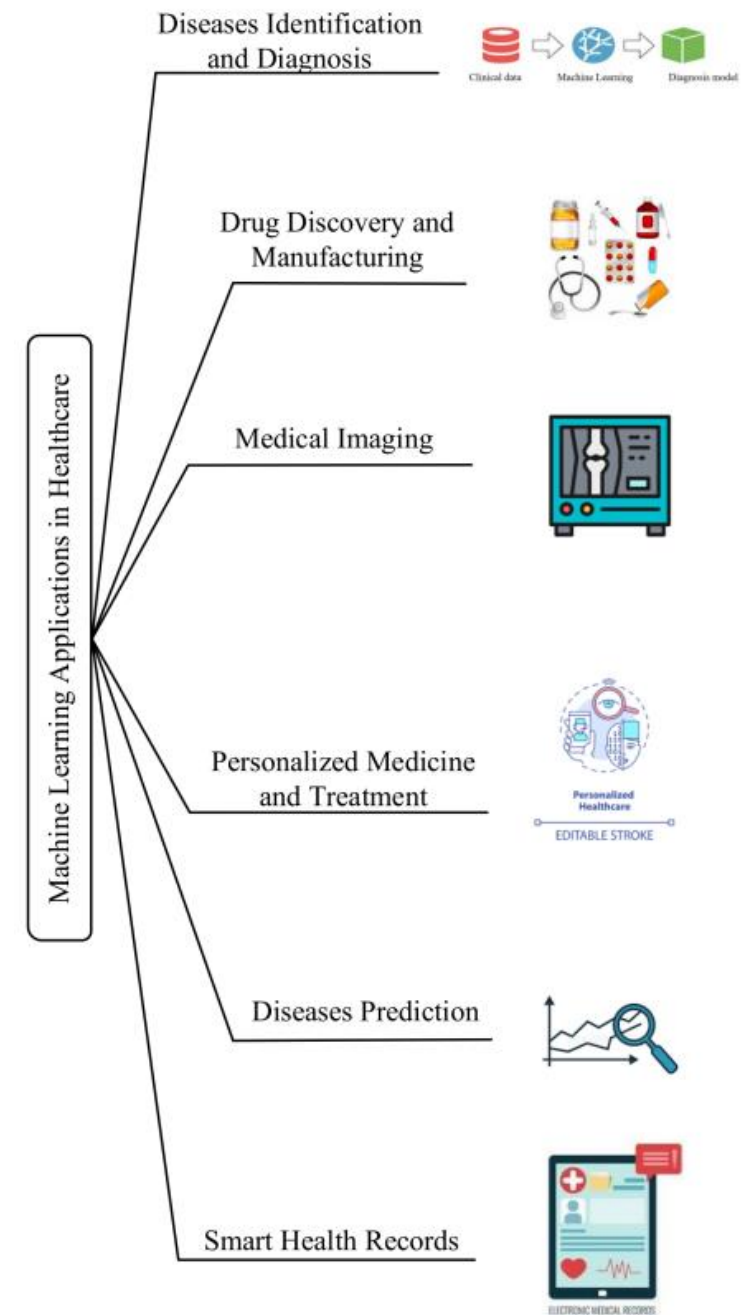
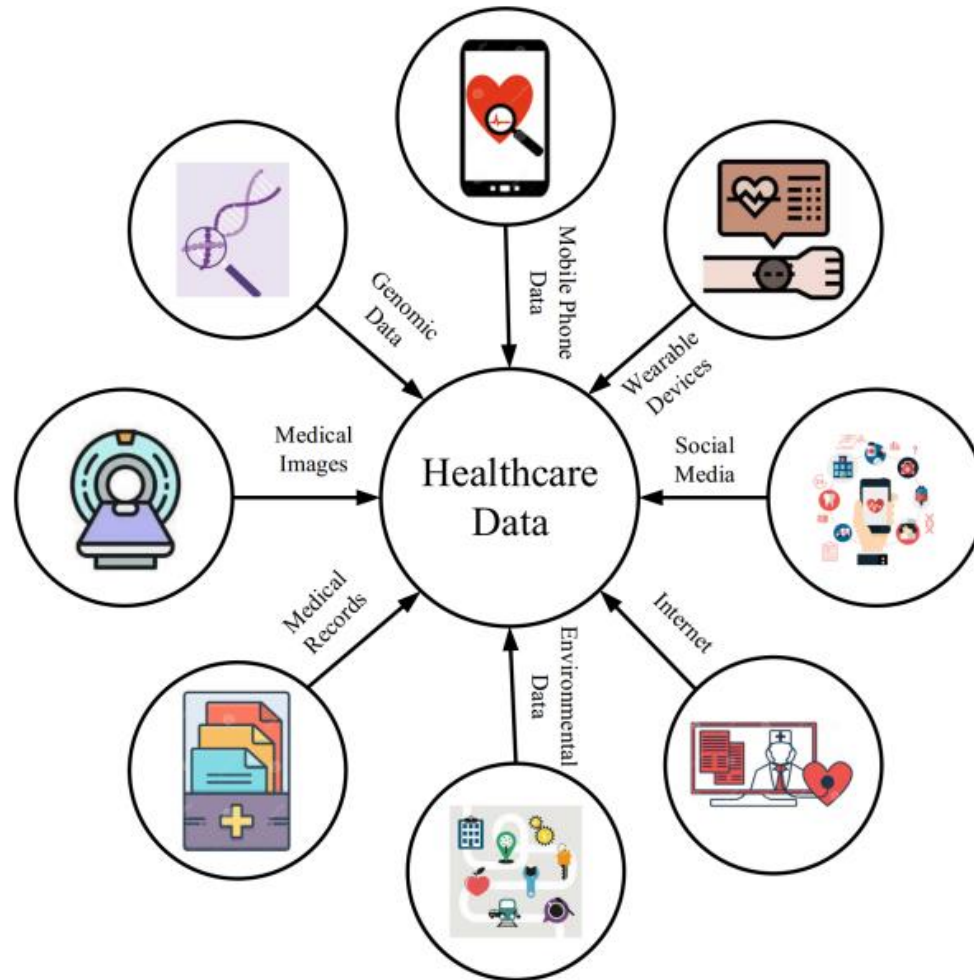
04 - TALLER PRÁCTICO CON INFORMACIÓN CLÍNICA

- Importar datos
- Exploración de datos
- Visualización de datos
- Exportar datos

01 - INTELIGENCIA ARTIFICIAL EN EL ÁMBITO BIOMÉDICO



Datos y aplicaciones IA en el cuidado de la salud



Casos de IA aplicada en salud



Casos de IA aplicada en salud



Hi, I'm Ada.
I can help if you're
feeling unwell.



Casos de IA aplicada en salud



Procesamiento de lenguaje en historias clínicas electrónicas relacionadas con Trastorno Afectivo Bipolar

1) Anonimización de historias clínicas

2) Preprocesamiento del corpus clínico

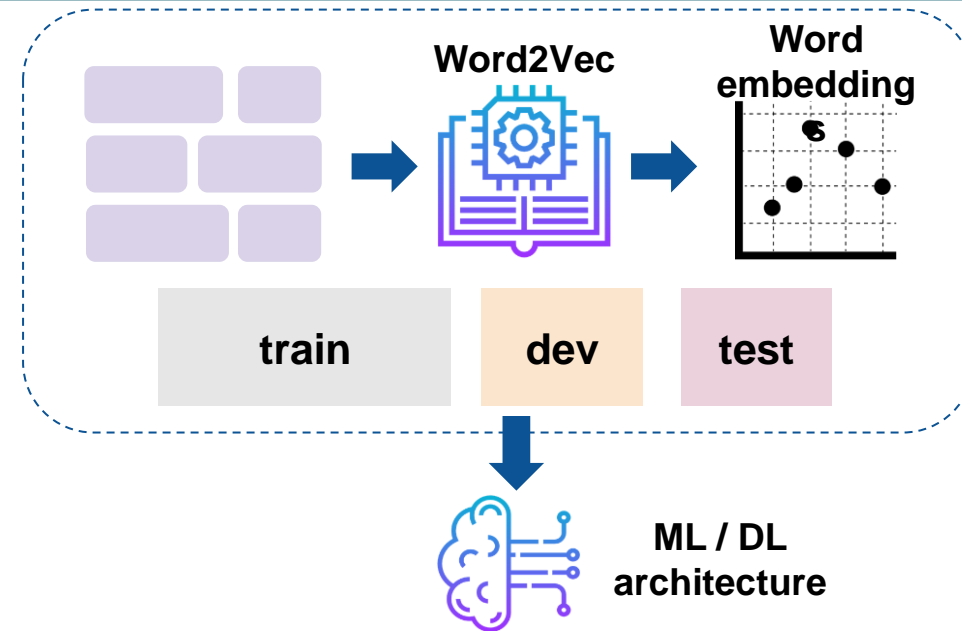
paciente ingresa por sus propios medios
con adecuada presentación personal
afecto modulado y sin ideas delirantes

3) Anotación del corpus clínico

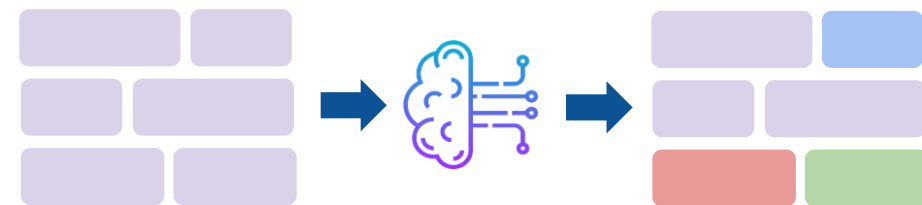
paciente ingresa por sus propios medios
con adecuada presentación personal

AFEC_NORM **NEG** **IDEL**
afecto modulado y sin ideas delirantes

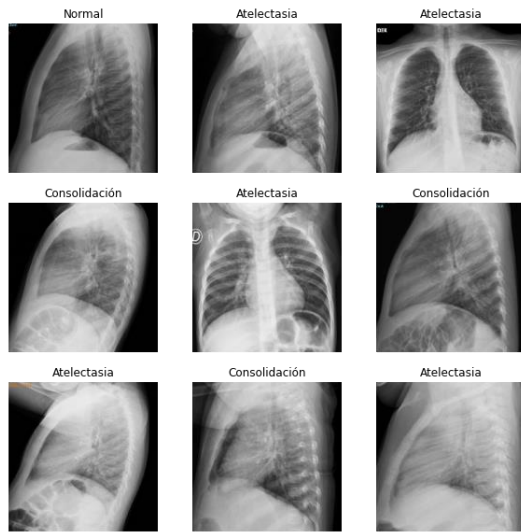
4) Entrenamiento y validación del modelo MER



5) Extracción de variables con el modelo MER

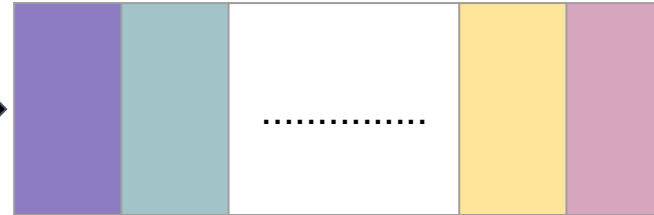


Modelo de predicción basado en aprendizaje profundo para diferenciar consolidación de atelectasia



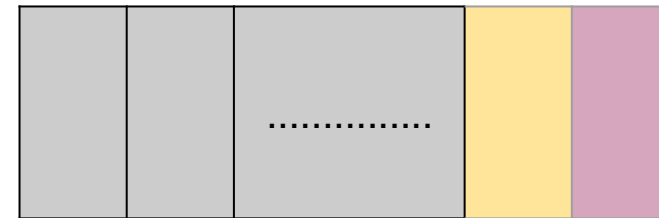
Pediatric
X-Ray
dataset

1



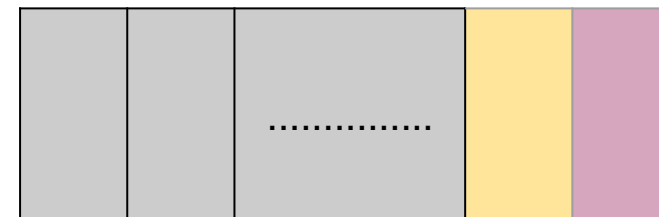
→ Atelectasia
→ Consolidación
→ Normal

2



→ Atelectasia
→ Consolidación
→ Normal

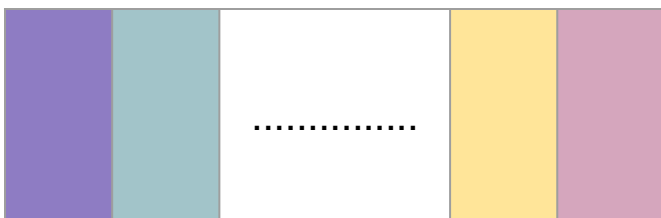
3



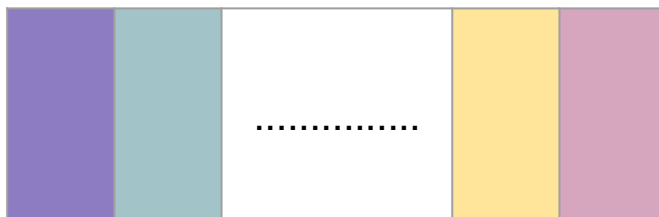
→ Atelectasia
→ Consolidación
→ Normal

Image
Net

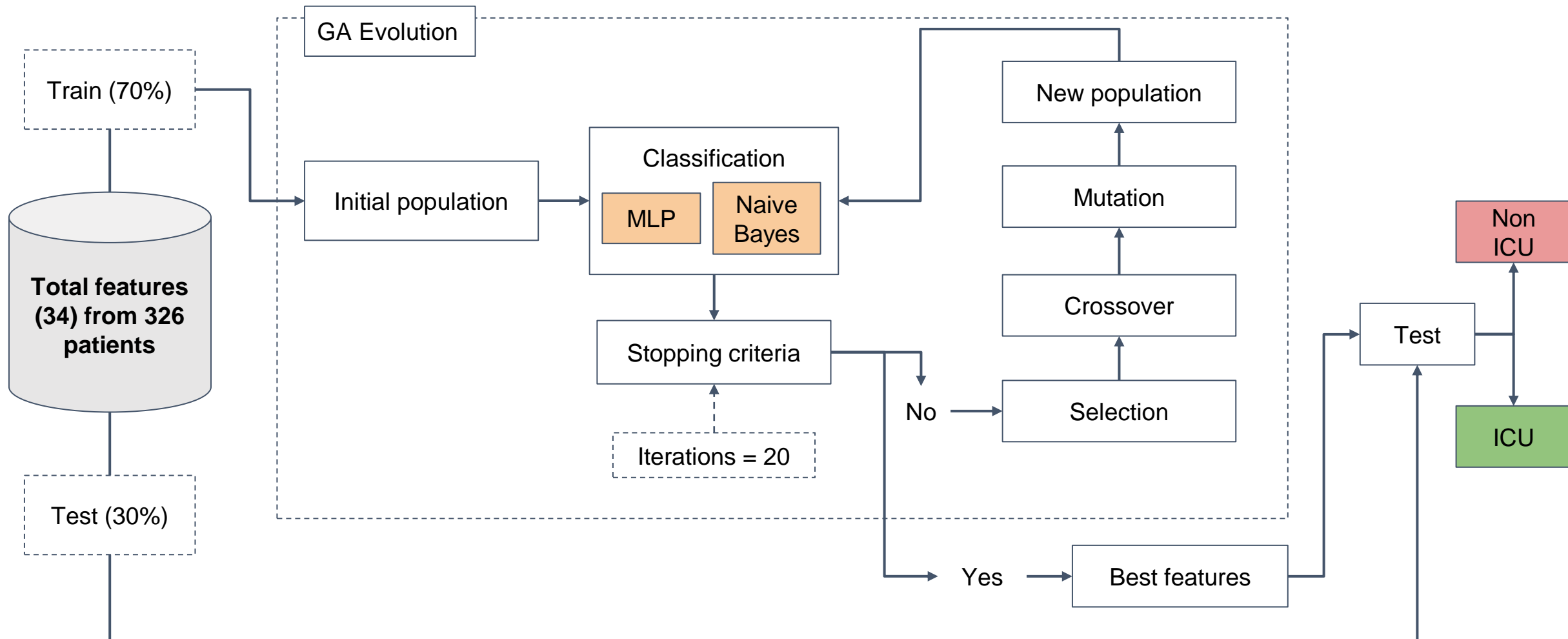
MobileNET V2



Chest X-
Ray

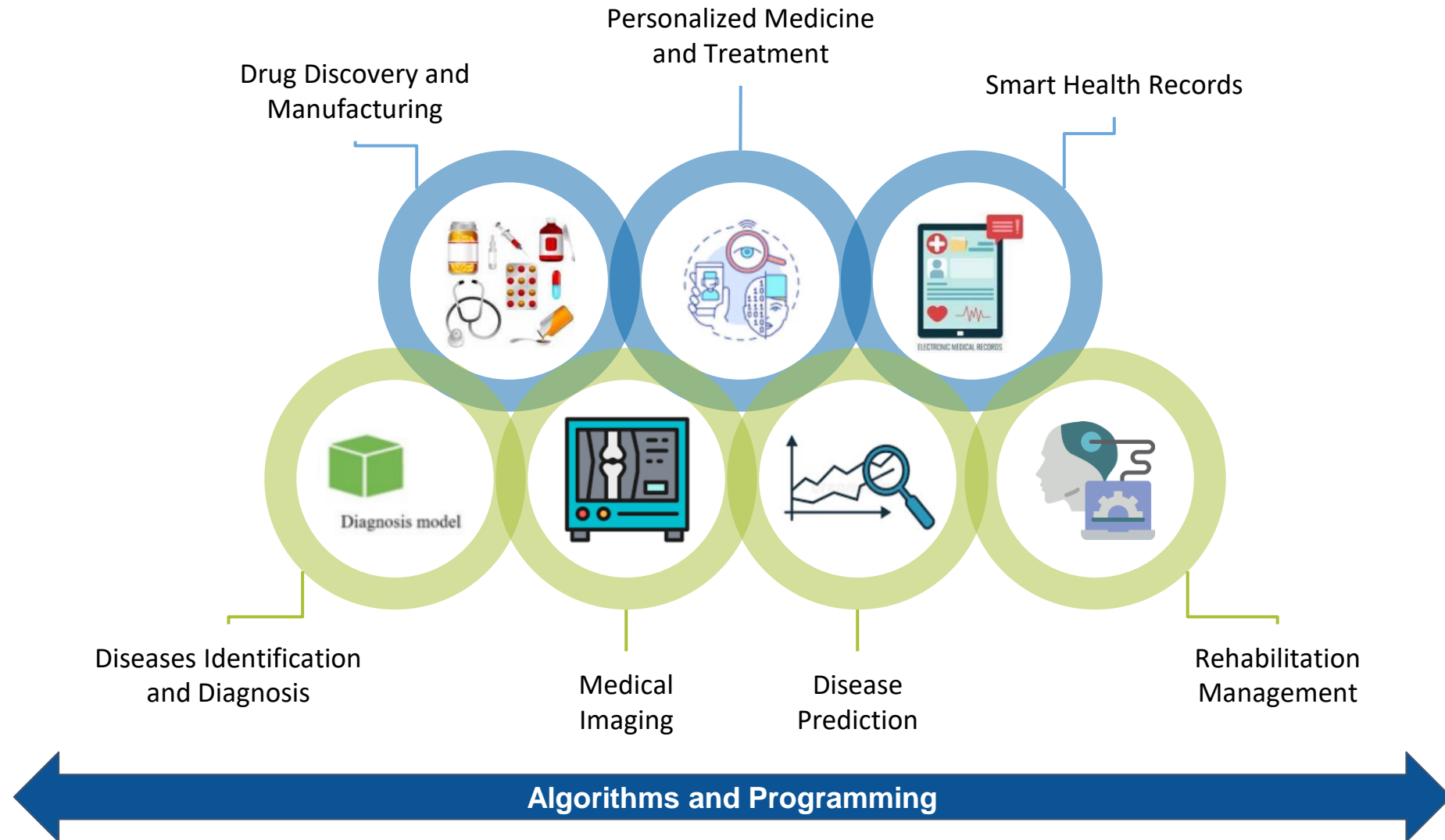


Predicción del ingreso a UCIP en pacientes con diagnóstico de Infección Respiratoria Aguda

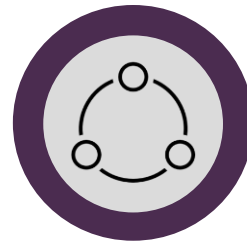


Non ICU: 198 (60.74%); **ICU:** 128 (39.26%)

Algoritmos y programación como pilares de la IA



02 - ALGORITMOS Y PROGRAMACIÓN

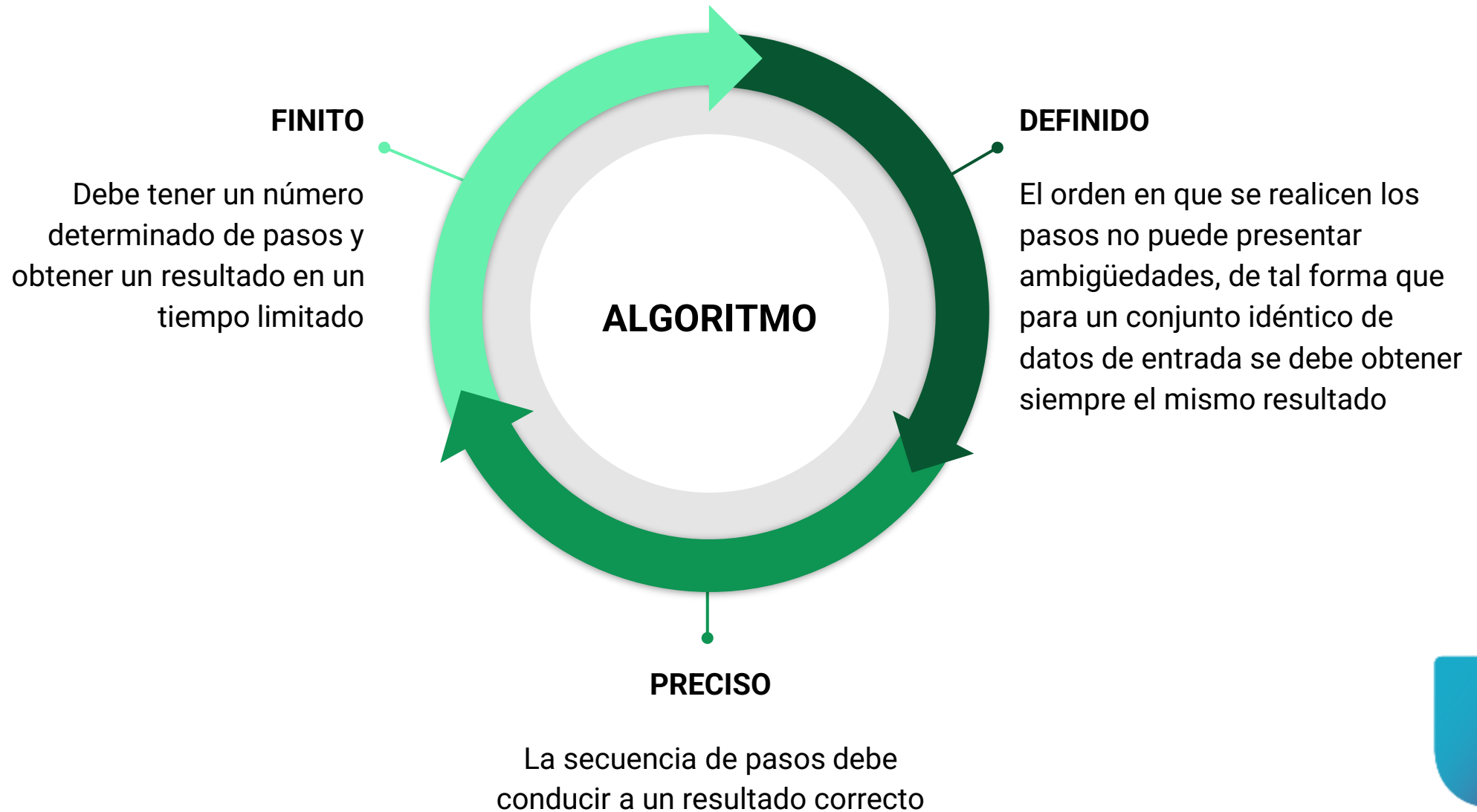


¿Qué es un algoritmo?



Un algoritmo es un conjunto de pasos, procedimientos o acciones que nos permiten alcanzar un resultado o resolver un problema

Características de los algoritmos

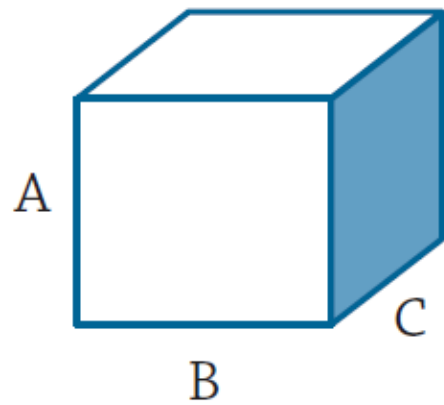


Formas de representación de un algoritmo

Pseudocódigo

- Lenguaje intermedio entre nuestro lenguaje y el lenguaje de programación.
- Representa la solución de un algoritmo de la forma más detallada posible, y a su vez lo más parecida posible al lenguaje que posteriormente se utilizará para la codificación en el computador.

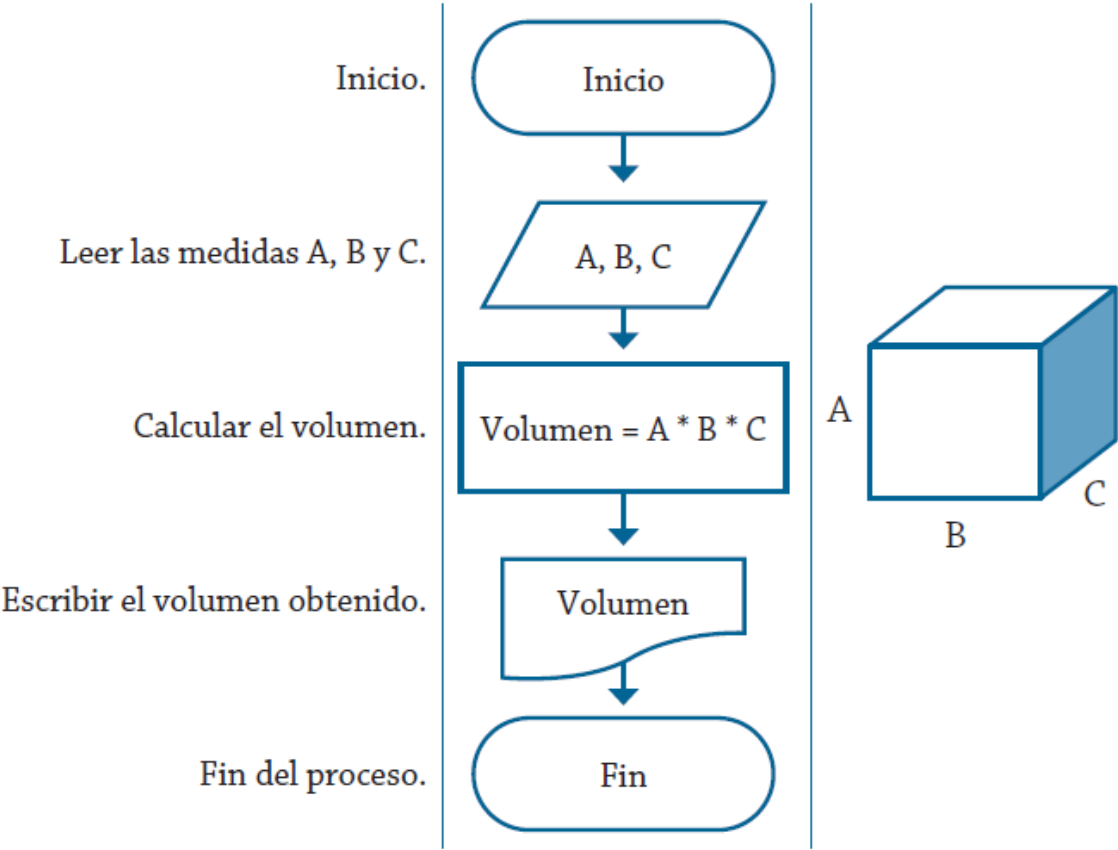
Por ejemplo, el pseudocódigo para determinar el volumen de una caja de dimensiones A, B y C se puede establecer de la siguiente forma:



1. Inicio.
2. Leer las medidas A, B y C.
3. Realizar el producto de $A * B * C$ y guardarlo en V ($V = A * B * C$).
4. Escribir el resultado V.
5. Fin.

Diagrama de flujo

Representación gráfica de un algoritmo mediante símbolos especiales, considerando los pasos o procedimientos de manera secuencial y lógica que se deben realizar para solucionar un problema dado



Símbolo	Significado
	Terminal /Inicio.
	Entrada de datos.
	Proceso.
	Decisión.
	Decisión múltiple.
	Imprimir resultados.
	Flujo de datos.
	Conectores.

Relación entre algoritmos y programación

Como se observa en los ejemplos anteriores, al representar un algoritmo se identifica una secuencia de acciones, con un orden de principio a fin, y se cuestiona qué tan precisas fueron sus descripciones.

Estas consideraciones también aplican cuando se desarrolla un **algoritmo en un entorno computacional**, y es en este punto donde hablamos de **programación**.

Se transforma el pseudocódigo o el diagrama de flujo al código de un lenguaje de programación en particular

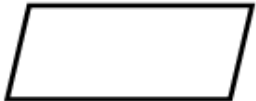
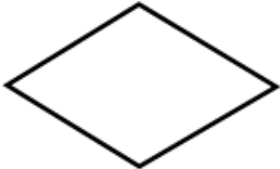



03 - PRINCIPIOS DE PROGRAMACIÓN



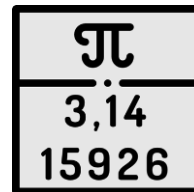
Lenguaje de programación

Conjunto de comandos que puede ser utilizado para **controlar el comportamiento de una computadora**, de acuerdo con reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente

Nombre	Símbolo	C++	Python	Java	Pseudocódigo
Entrada		<code>cin>></code>	<code>input</code>	<code>showInputDialog</code> <code>teclado.nextInt</code>	Leer
Decisión		Pertenece a varias estructuras de control y de ello depende el comando que se debe utilizar, ya sea If, DoWhile o While			Si se cumple la condición De lo contrario
Salida		<code>cout<<</code>	<code>print</code>	<code>System.out.print</code>	Imprimir

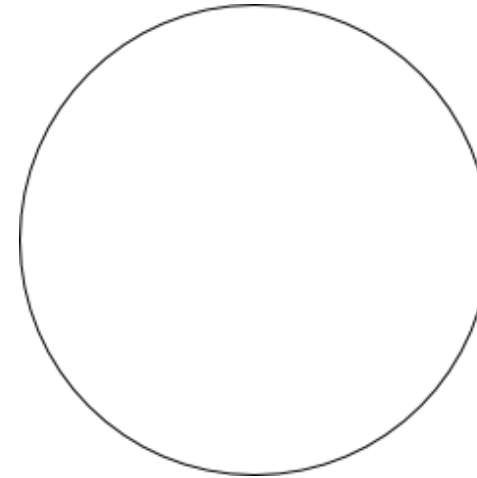
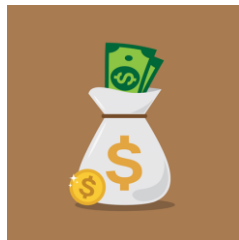
Constante

Dato que no cambia durante la ejecución de un programa



Variable

Objeto que cambia su valor durante la ejecución de un programa

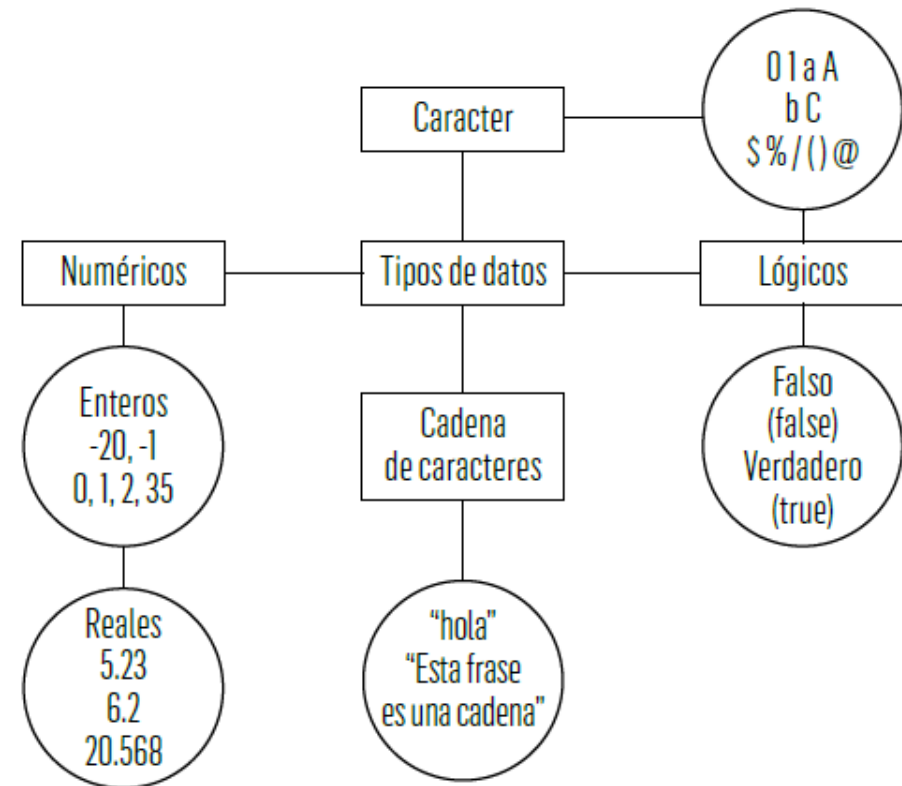


Tipos de datos

Un tipo de dato es un atributo que indica al computador y al programador la clase de dato que se va a procesar.

Los tipos de datos más comunes son:

- 1) **Numéricos:** corresponde a los datos cuantificables (enteros y reales).
- 2) **Carácter:** corresponde a valores que puede tomar cualquier símbolo del teclado, también son conocidos como alfanuméricos.
- 3) **Lógicos:** corresponde a los valores que puede tomar una condición, es decir Verdadero (Sí) o Falso (No).



Operadores

Los operadores son símbolos que permiten hacer una tarea específica con los diferentes tipos de datos

Operadores aritméticos

Operador	Operación	Ejemplo	Resultado
+	Suma	4 + 2	6
-	Resta	4 - 2	2
*	Multiplicación	4 * 2	8
/	División	4 / 2	2
**	Potencia	4 ** 2	16
%	Módulo	4 % 2	0

Operadores relacionales

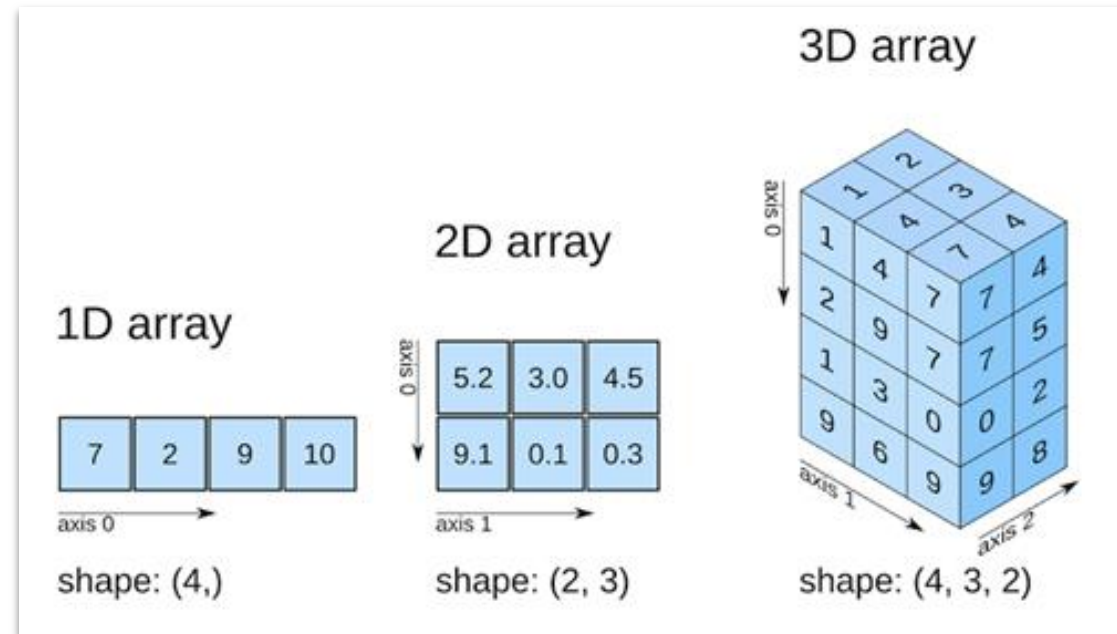
Operador	Comparación	Ejemplo	Resultado
<	Menor que	4 < 2	False
<=	Menor o igual que	4 <= 2	False
>	Mayor que	4 > 2	True
>=	Mayor o igual que	4 >= 2	True
==	Igual que	4 == 2	False
!=	No es igual que	4 != 2	True

Estructuras de datos

Las estructuras de datos son objetos que contienen datos. Por definición, cuando una estructura es **homogénea**, los datos contenidos son de un **mismo tipo**, mientras que una estructura **heterogénea** puede contener datos de **diferentes tipos**

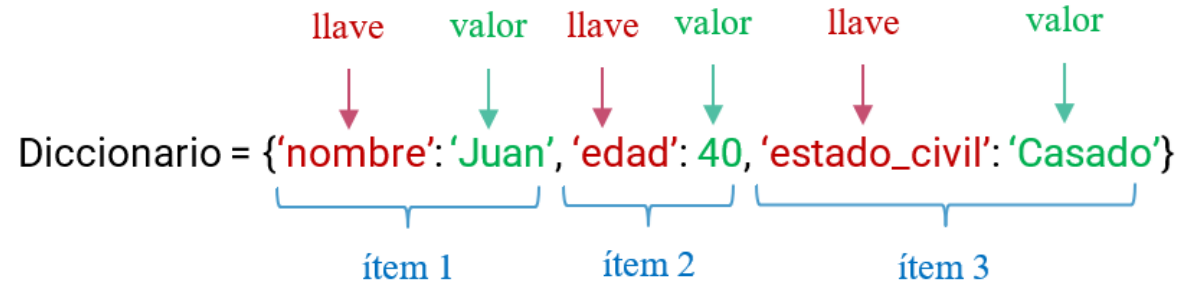
Arreglo

Colección de datos del mismo tipo con un número arbitrario de dimensiones



Diccionario

Colección no ordenada de parejas compuestas de llaves y valores, donde las llaves son únicas y los valores pueden ser de cualquier tipo arbitrario (números, cadenas, entre otros)



- ✓ Los ítems en el diccionario se almacenan sin ningún valor de indexación
- ✓ Las llaves son únicas en los diccionarios
- ✓ Se puede añadir, modificar o remover una pareja (llave y valor) después de su creación

DataFrame

Estructura de datos de dos dimensiones que puede contener datos de diferentes tipos (similar a una tabla). Esta colección de datos suele ser la más utilizada para procesar la información

Column Label/ Header		0	1	2	3	4
Index Label		Name	Age	Marks	Grade	Hobby
0	S1	Joe	20	85.10	A	Swimming
1	S2	Nat	21	77.80	B	Reading
2	S3	Harry	19	91.54	A	Music
3	S4	Sam	20	88.78	A	Painting
4	S5	Monica	22	60.55	B	Dancing

Column Index

Row

Row Index

Column

Element/ Value/ Entry

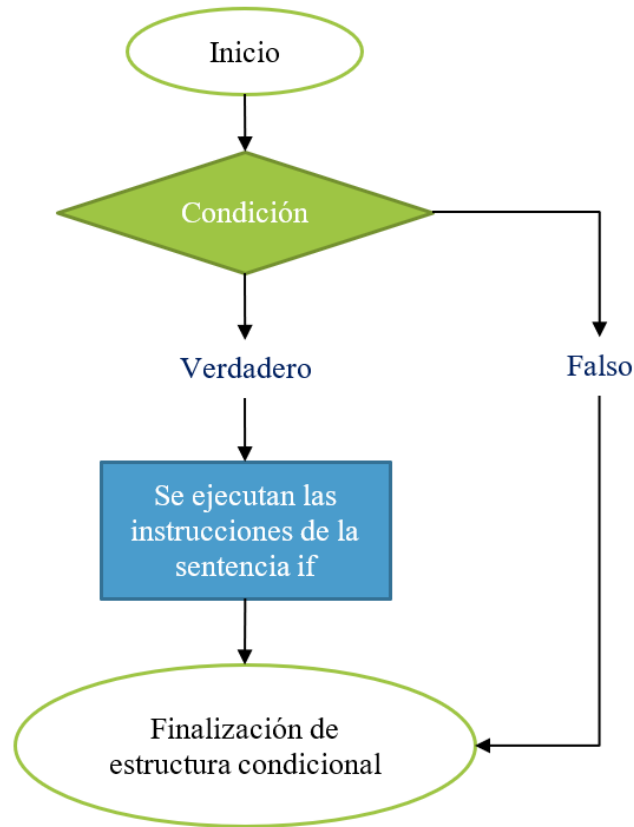
Estructuras de control

Para la construcción de un código de programación es fundamental definir sentencias que especifiquen el **orden de ejecución de las tareas**, estableciendo diferentes estructuras de control (condicionales o iterativas)

Estructura de control	Función
if, else	Evaluar una condición y actuar en consecuencia
for	Ejecutar un ciclo por un número fijo de veces
while	Ejecutar un ciclo mientras se cumpla una condición

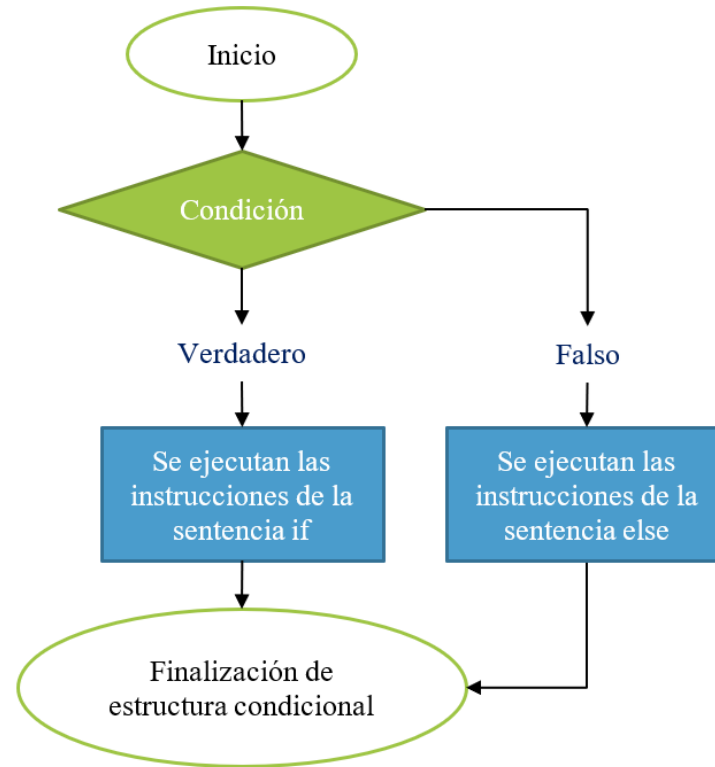
If

Esta sentencia realiza una acción cuando la condición es verdadera, y cuando es falsa la ignora



If-Else

Esta sentencia realiza una acción cuando la condición es verdadera, y cuando es falsa realiza otra acción



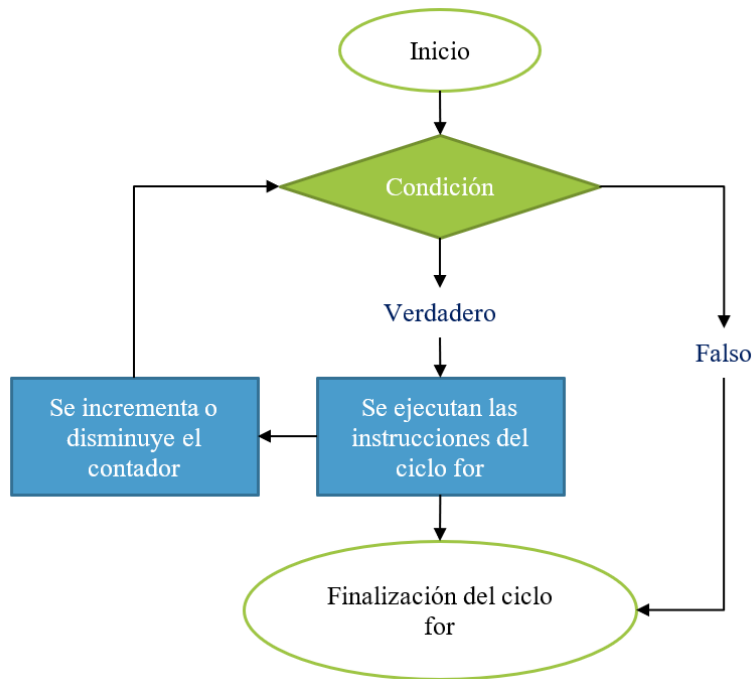
```
1 if 1 > 2 :  
2     print("a")  
3 else :  
4     print("b")
```

▶ Run

✓ Check

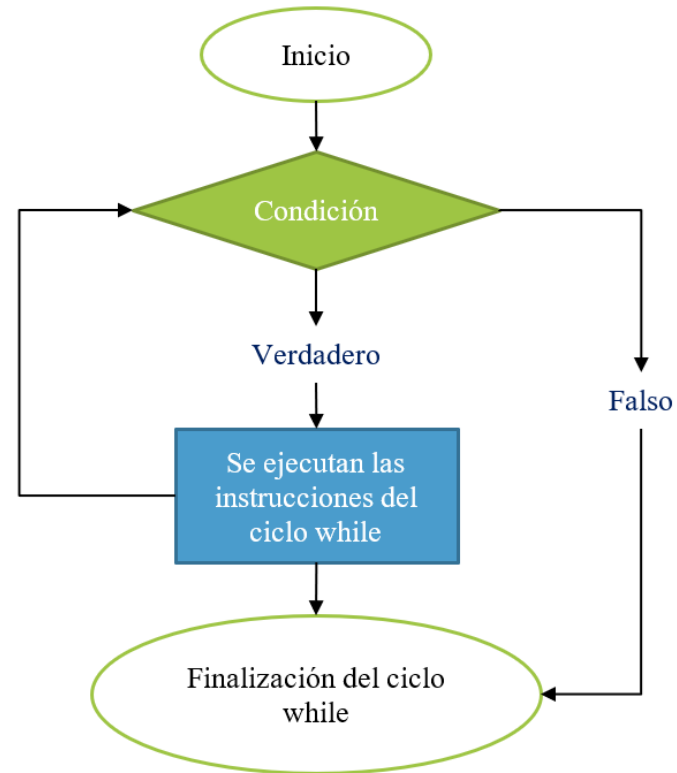
For

Realiza una operación para cada elemento de un grupo de datos, ejecutándose un número preestablecido de veces



While

Permite realizar una acción mientras que una condición se mantenga como verdadera



```
script.py
a_list = [1, 3, 5]
for value in a_list:
    print(value)
```

Output

code

```
1 a = 1
2 while a < 10:
3     print (a)
4     a += 2
```

output

variables

04 - TALLER PRÁCTICO CON INFORMACIÓN CLÍNICA





Gracias

Unisanitas/Vigilada Mineducación
Reconocimiento personería jurídica: Resolución No. 3015 del 23 de diciembre de 2002