ALGORITMIA

Juan Carlos Molina Lozano

Docente

OBJETIVOS DE LA CLASE

1. Explicar la importancia de las condiciones en la toma de decisiones dentro de un algoritmo.
2. Identificar y diferenciar los tipos de estructuras condicionales en Python (if, if-else, if-elif-else).
3. Aplicar estructuras condicionales en la solución de problemas mediante ejemplos y ejercicios prácticos.
4. Desarrollar pensamiento lógico a través de la experimentación con código en Python.

METODOLOGÍA

1. Exposición con ejemplos prácticos.
2. Aprendizaje Activo: actividad en grupo con ejercicios.
3. Discusión Guiada: preguntas estratégicas para fomentar la reflexión.
4. Componente Práctico: resolución de problemas algorítmicos en clase

CONTENIDO

1. Introducción a flujo.
2. Diagramas de Flujo
3. Qué es una condición en un algoritmo?.
4. Estructuras Condicionales.
5. Importancia de las condiciones en los algoritmos.
6. Ejercicios.

INTRODUCCIÓN A FLUJO

Ideas clave:

- Los programas se ejecutan de manera lineal, de arriba a abajo.
- Podemos cambiar el flujo de un programa utilizando condiciones.

¿Qué es el flujo?

El flujo de un programa es el orden en el que se ejecutan las instrucciones. Hasta el momento, los programas que hemos construido han seguido un flujo lineal, lo que significa que las instrucciones se ejecutan en el mismo orden, de la primera línea a la última. Sin embargo, en la mayoría de los programas necesitamos tomar decisiones y ejecutar instrucciones diferentes dependiendo de ciertas condiciones. Podemos manipular el flujo de un programa utilizando la instrucción if.

DIAGRAMAS DE FLUJO

El flujo de un programa es el orden en el que se ejecutan las instrucciones y, si bien podemos seguir el flujo de un programa leyendo el código, a veces es más fácil visualizarlo. Para esto se utilizan los diagramas de flujo, que son una herramienta visual que nos permite representar el flujo de un programa.

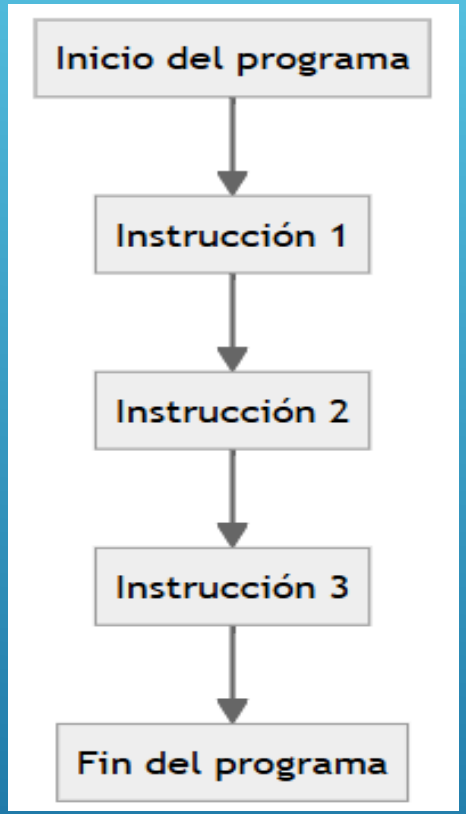
Los diagramas de flujo se componen de bloques que representan instrucciones y flechas que indican el flujo de ejecución. Cada bloque tiene una forma específica que indica el tipo de instrucción que representa. Por ejemplo, un bloque rectangular representa una instrucción, un rombo representa una condición y un óvalo representa el inicio o fin del programa.

DIAGRAMAS DE FLUJO

Código y diagramas de flujo

Veamos un ejemplo lineal primero.

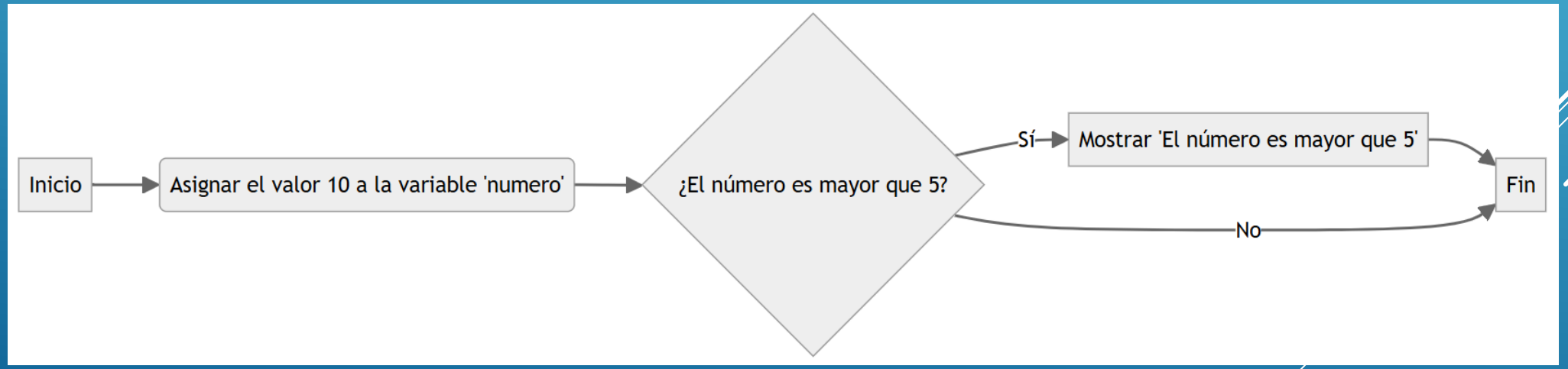
```
print("Inicio del programa")
print("Instrucción 1")
print("Instrucción 2")
print("Instrucción 3")
```



DIAGRAMAS DE FLUJO

Veamos un ejemplo con una condición.

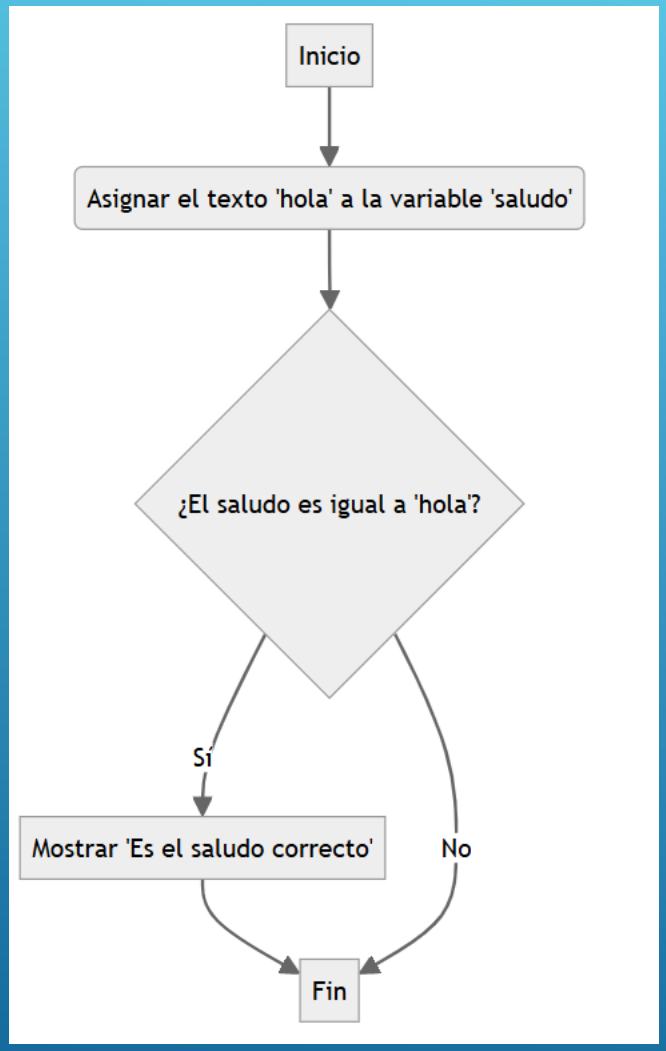
```
numero = 10
if numero > 5:
    print("El número es mayor que 5")
```



DIAGRAMAS DE FLUJO

Ejercicio

Implementa el siguiente código
a partir de un diagrama de flujo



QUÉ ES UNA CONDICIÓN EN UN ALGORITMO?

Una condición en un algoritmo es una expresión lógica que determina si una determinada acción o conjunto de instrucciones debe ejecutarse. En otras palabras, permiten que un programa tome decisiones y siga diferentes caminos dependiendo de ciertos valores o situaciones.

Las condiciones son fundamentales en la programación, ya que sin ellas los programas solo podrían ejecutarse de manera secuencial sin responder a cambios o eventos.

Ejemplo cotidiano:

Si llueve, llevaré un paraguas.

Si mi nota es mayor o igual a 7, aprobé el curso.

Si el semáforo está en rojo, me detengo; de lo contrario, sigo.

Estas decisiones se implementan en los algoritmos mediante estructuras condicionales.

OPERADORES RELACIONALES Y LÓGICOS EN ALGORITMOS

Para formular condiciones en Python, usamos operadores relacionales y operadores lógicos.

1. Operadores Relacionales: Los operadores relacionales comparan valores y devuelven un resultado Verdadero (True) o Falso (False).

Operador	Descripción	Ejemplo
`==`	Igualdad	`5 == 5` → `True`
`!=`	Diferente	`5 != 3` → `True`
`>`	Mayor que	`10 > 5` → `True`
`<`	Menor que	`3 < 8` → `True`
`>=`	Mayor o igual que	`7 >= 7` → `True`
`<=`	Menor o igual que	`4 <= 6` → `True`

Ejemplo en código Python:

```
edad = 18
if edad >= 18:
        print("Eres mayor de edad.")
else:
        print("Eres menor de edad.")
```

```
INICIO
    DEFINIR edad COMO ENTERO
    ESCRIBIR "Ingresa tu edad: "
    LEER edad

    SI edad >= 18 ENTONCES
        ESCRIBIR "Eres mayor de edad."
    SINO
        ESCRIBIR "Eres menor de edad."
    FIN SI
FIN
```

OPERADORES RELACIONALES Y LÓGICOS EN ALGORITMOS

2. Operadores Lógicos: Permiten combinar varias condiciones en una sola expresión.

Operador	Descripción	Ejemplo
`and`	Devuelve `True` si ambas condiciones son verdaderas	`(5 > 3) and (10 > 8)` → `True`
`or`	Devuelve `True` si al menos una condición es verdadera	`(5 > 3) or (10 < 8)` → `True`
`not`	Invierte el valor de la condición	`not (5 > 3)` → `False`

Ejemplo en código Python:

```
usuario = "admin"clave = "1234“
if usuario == "admin" and clave == "1234":
        print("Acceso concedido.")
else:
        print("Acceso denegado.")
```

```
INICIO
    DEFINIR usuario, clave COMO CADENA
    ESCRIBIR "Ingresa tu usuario: "
    LEER usuario
    ESCRIBIR "Ingresa tu clave: "
    LEER clave

    SI usuario = "admin" Y clave = "1234" ENTONCES
        ESCRIBIR "Acceso concedido."
    SINO
        ESCRIBIR "Acceso denegado."
    FIN SI
FIN
```

ESTRUCTURAS CONDICIONALES

Python ofrece tres formas de evaluar condiciones y ejecutar código en consecuencia:

1. Condición simple (if)

Se usa cuando queremos ejecutar un bloque de código solo si una condición es verdadera.

Ejemplo:

```
temperatura = 30
if temperatura > 25:
    print("Hace calor.")
```

```
INICIO
    DEFINIR temperatura COMO ENTERO
    ESCRIBIR "Ingresa la temperatura: "
    LEER temperatura

    SI temperatura > 25 ENTONCES
        ESCRIBIR "Hace calor."
    FIN SI
FIN
```

Si la temperatura es mayor a 25, se imprimirá "Hace calor.".

ESTRUCTURAS CONDICIONALES

2. Condición compuesta (if-else)

Se usa cuando queremos que el programa ejecute un código si la condición es verdadera y otro diferente si es falsa.

Ejemplo:

```
edad = int(input("Ingresa tu edad: "))
if edad >= 18:
    print("Puedes votar.")
else:
    print("No puedes votar.")
```

```
INICIO
    DEFINIR edad COMO ENTERO
    ESCRIBIR "Ingresa tu edad: "
    LEER edad

    SI edad >= 18 ENTONCES
        ESCRIBIR "Puedes votar."
    SINO
        ESCRIBIR "No puedes votar."
    FIN SI
FIN
```

Si el usuario ingresa una edad mayor o igual a 18, verá "Puedes votar.", de lo contrario, verá "No puedes votar.".

ESTRUCTURAS CONDICIONALES

3. Condición múltiple (if-elif-else)

Se usa cuando hay varias condiciones posibles y queremos evaluar cada una de ellas.

Ejemplo:

```
nota = float(input("Ingresa tu nota: "))
if nota >= 9:
        print("Excelente.")
elif nota >= 7:
        print("Bueno.")
elif nota >= 5:
        print("Regular.")
else:
        print("Reprobado.")
```

```
INICIO
    DEFINIR nota COMO REAL
    ESCRIBIR "Ingresa tu nota: "
    LEER nota

    SI nota >= 9 ENTONCES
        ESCRIBIR "Excelente."
    SINO SI nota >= 7 ENTONCES
        ESCRIBIR "Bueno."
    SINO SI nota >= 5 ENTONCES
        ESCRIBIR "Regular."
    SINO
        ESCRIBIR "Reprobado."
    FIN SI
FIN
```

El programa evalúa la nota ingresada y muestra el mensaje correspondiente.

IMPORTANCIA DE LAS CONDICIONES EN LOS ALGORITMOS

Las condiciones permiten que un programa:

- ☑ Responda de manera inteligente a la entrada del usuario.
- ☑ Tome decisiones basadas en datos.
- ☑ Automatice procesos de toma de decisiones.
- ☑ Evite ejecuciones innecesarias de código.

EJERCICIOS

1. Determinar si un número es positivo, negativo
2. Determine un programa para saber si una persona es mayor de edad o no.
3. Desarrollar un programa que permita si un estudiante aprobó la nota o no.
4. Número par o impar: Escribe un programa que pida un número y determine si es par o impar.
5. Calculadora de descuentos: Si el total de la compra es mayor a $100, aplicar un 10% de descuento y mostrar el total final.
6. Determinar el mayor de dos números: Pide al usuario dos números y muestra cual es el mayor.

RESUMEN

1. Que es un Flujo.
2. Diagramas de Flujo
3. Condiciones
4. Estructuras Condicionales.
5. Importancia de las condiciones en los algoritmos.