PUCE Virtual
EDUCACIÓN FLEXIBLE

Programación I

Condicionales y anidamiento (Parte 2)

Clase 8

Ingeniería en ciberseguridad

La excelencia no se improvisa

1. INTRODUCCIÓN DE LA CLASE (2 párrafos)

En el desarrollo de programas, uno de los aspectos más importantes es la capacidad de tomar decisiones. A través de estructuras de control condicionales, los programadores pueden guiar el flujo de ejecución del programa según diferentes condiciones. Estas estructuras permiten que el programa ejecute ciertos bloques de código dependiendo de si se cumplen ciertas condiciones, lo que es fundamental en casi todas las aplicaciones. En esta clase, profundizaremos en dos conceptos fundamentales: condicionales anidadas y el operador ternario.

Las condicionales anidadas permiten tener decisiones dentro de otras decisiones, lo que crea un flujo más dinámico y flexible. Esta técnica es crucial cuando se necesita hacer múltiples verificaciones y ejecutar distintas acciones según cada resultado. Además, aprenderemos a usar el operador ternario, que proporciona una forma compacta y eficiente de hacer decisiones simples en una sola línea de código. Ambas herramientas son esenciales para mejorar la legibilidad y eficiencia de los programas, especialmente cuando se manejan múltiples condiciones o cuando se busca simplificar estructuras condicionales complejas.

Clase 8: Condicionales y anidamiento (Parte 2)

Resultado o resultados de aprendizaje que será abordado con el contenido de la clase.

Reconocer las estructuras básicas de un lenguaje de programación estructurado, su sintaxis y su utilidad en la solución de problemas de programación.

Reto # 2

Contenido de la Clase:

8. **Condicionales y anidamiento (Parte 2)**
   - 8.1. **Condicionales anidados (ej: menús de opciones)**

Las condicionales anidadas son una herramienta fundamental en programación, ya que permiten tomar decisiones dentro de otras decisiones, lo que a su vez habilita una mayor flexibilidad en el diseño del flujo del programa. Esta estructura es particularmente útil cuando se deben realizar múltiples verificaciones en función de una serie de condiciones. Sin esta capacidad, un programa estaría limitado a realizar solo decisiones simples, incapaz de manejar casos más complejos o interactivos.

¿Qué son las Condicionales Anidadas?

En términos sencillos, una condicional anidada es una estructura if, else o elif que se encuentra dentro de otra estructura if, else o elif. Esto permite que el programa realice una serie de comprobaciones antes de ejecutar una acción específica. El flujo de ejecución se bifurca de manera secuencial y puede manejar situaciones más complejas que involucran varias condiciones.

Por ejemplo, imagina que estamos creando un sistema de control de acceso donde, después de que un usuario ingrese su contraseña, necesitamos verificar si tiene permisos especiales o si debe acceder a un nivel específico de la aplicación. En este caso, usaríamos condicionales anidadas para manejar cada uno de estos pasos de forma eficiente.
Para ilustrar el uso de las condicionales anidadas, supongamos que estamos creando un programa que permita al usuario elegir una operación matemática básica. El menú presentará las opciones de suma, resta, multiplicación y salida, y dependiendo de la elección, el programa pedirá los números y ejecutará la operación correspondiente. Si el usuario ingresa una opción inválida, el programa le indicará que elija una opción válida.

Las condicionales anidadas son una herramienta esencial para crear aplicaciones interactivas. Pueden ser utilizadas en una amplia variedad de escenarios, como la creación de menús de opciones, formularios de validación de datos o incluso en sistemas de control de acceso, donde el flujo de ejecución depende de varios factores evaluados de manera secuencial.

Ejemplo 1 en Python: Menú de Opciones

Imaginemos que estamos construyendo un programa simple que permite al usuario realizar operaciones matemáticas básicas como suma, resta y multiplicación. Para hacer este programa interactivo, vamos a usar un menú que le permita al usuario elegir entre las diferentes operaciones. Dependiendo de la opción que elija, el programa pedirá los números necesarios y realizará la operación correspondiente.

```python
# Menú de opciones
print("Menú de opciones:")
print("1. Sumar")
print("2. Restar")
print("3. Multiplicar")
print("4. Salir")

# Entrada de usuario
opcion = int(input("Elija una opción (1-4): "))

# Condicionales anidados para manejar las opciones
if opcion == 1:
    num1 = float(input("Ingrese el primer número: "))
    num2 = float(input("Ingrese el segundo número: "))
    print(f"La suma es: {num1 + num2}")
elif opcion == 2:
    num1 = float(input("Ingrese el primer número: "))
    num2 = float(input("Ingrese el segundo número: "))
    print(f"La resta es: {num1 - num2}")
elif opcion == 3:
    num1 = float(input("Ingrese el primer número: "))
    num2 = float(input("Ingrese el segundo número: "))
    print(f"La multiplicación es: {num1 * num2}")
elif opcion == 4:
    print("¡Hasta luego!")
else:
    print("Opción no válida. Por favor, elija una opción entre 1 y 4.")
```

Este código es un ejemplo clásico de cómo las condicionales anidadas pueden ser usadas para manejar un menú de opciones. El programa primero le presenta al usuario las opciones disponibles (sumar, restar, multiplicar y salir). Luego, según la opción seleccionada, se ejecuta una operación matemática, y si la opción no es válida, el programa le indica al usuario que debe elegir una opción válida.

Este tipo de programa es sencillo, pero ejemplifica perfectamente el uso de las condicionales anidadas, ya que cada opción (suma, resta, multiplicación) depende de una evaluación previa de la opción seleccionada.

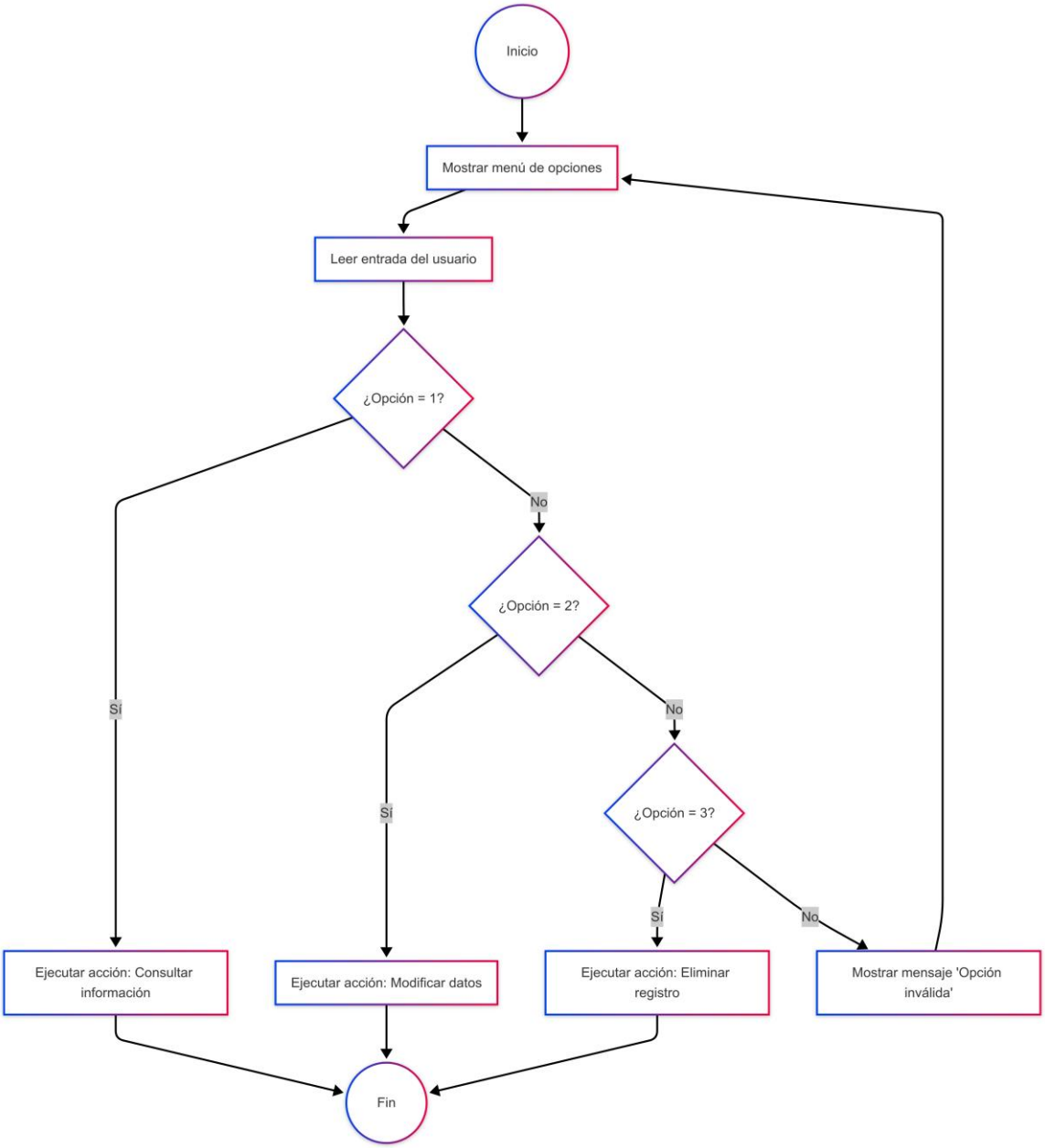


Imagen 1. Diagrama de flujo de un menú interactivo
Damián Nicolalde Rodríguez. (2025). Diagrama de flujo de un menú interactivo.

Las condicionales anidadas se vuelven aún más útiles cuando necesitamos evaluar más de una condición dentro de una misma opción. En este sentido, las condicionales anidadas no solo permiten que un bloque de código esté dentro de otro, sino que también pueden manejar varios niveles de decisiones.

Imaginemos un menú con subopciones, donde dentro de la opción "Operaciones Básicas", el programa ofrece un submenú para elegir la operación matemática (suma, resta, multiplicación).

Aquí, las condicionales anidadas permiten gestionar tanto la selección del menú principal como las subopciones de manera eficiente.

Ejemplo 2: Menú con Subopciones

Expandimos el ejemplo anterior, agregando subopciones dentro de la opción de "Operaciones Básicas". Ahora, el programa pedirá al usuario que elija una operación dentro del submenú después de seleccionar el menú principal.

```
# Menú principal
print("Menú Principal:")
print("1. Operaciones básicas")
print("2. Salir")

# Entrada de usuario
opcion = int(input("Elija una opción (1-2): "))

# Condicionales anidados para manejar las opciones
if opcion == 1:
    print("Submenú de operaciones:")
    print("1. Sumar")
    print("2. Restar")
    print("3. Multiplicar")
    subopcion = int(input("Elija una operación (1-3): "))

    num1 = float(input("Ingrese el primer número: "))
    num2 = float(input("Ingrese el segundo número: "))

    if subopcion == 1:
        print(f"La suma es: {num1 + num2}")
    elif subopcion == 2:
        print(f"La resta es: {num1 - num2}")
    elif subopcion == 3:
        print(f"La multiplicación es: {num1 * num2}")
    else:
        print("Subopción no válida.")
elif opcion == 2:
    print("¡Hasta luego!")
else:
    print("Opción no válida.")
```

Este ejemplo amplía el concepto de las condicionales anidadas al introducir un submenú dentro de una de las opciones del menú principal. Después de que el usuario selecciona la opción

"Operaciones básicas", se le presenta un submenú con tres operaciones: suma, resta y multiplicación. Dependiendo de su selección, el programa ejecutará la operación correspondiente.

Este tipo de menú con subopciones es más dinámico y flexible, permitiendo que el programa se adapte mejor a situaciones más complejas.

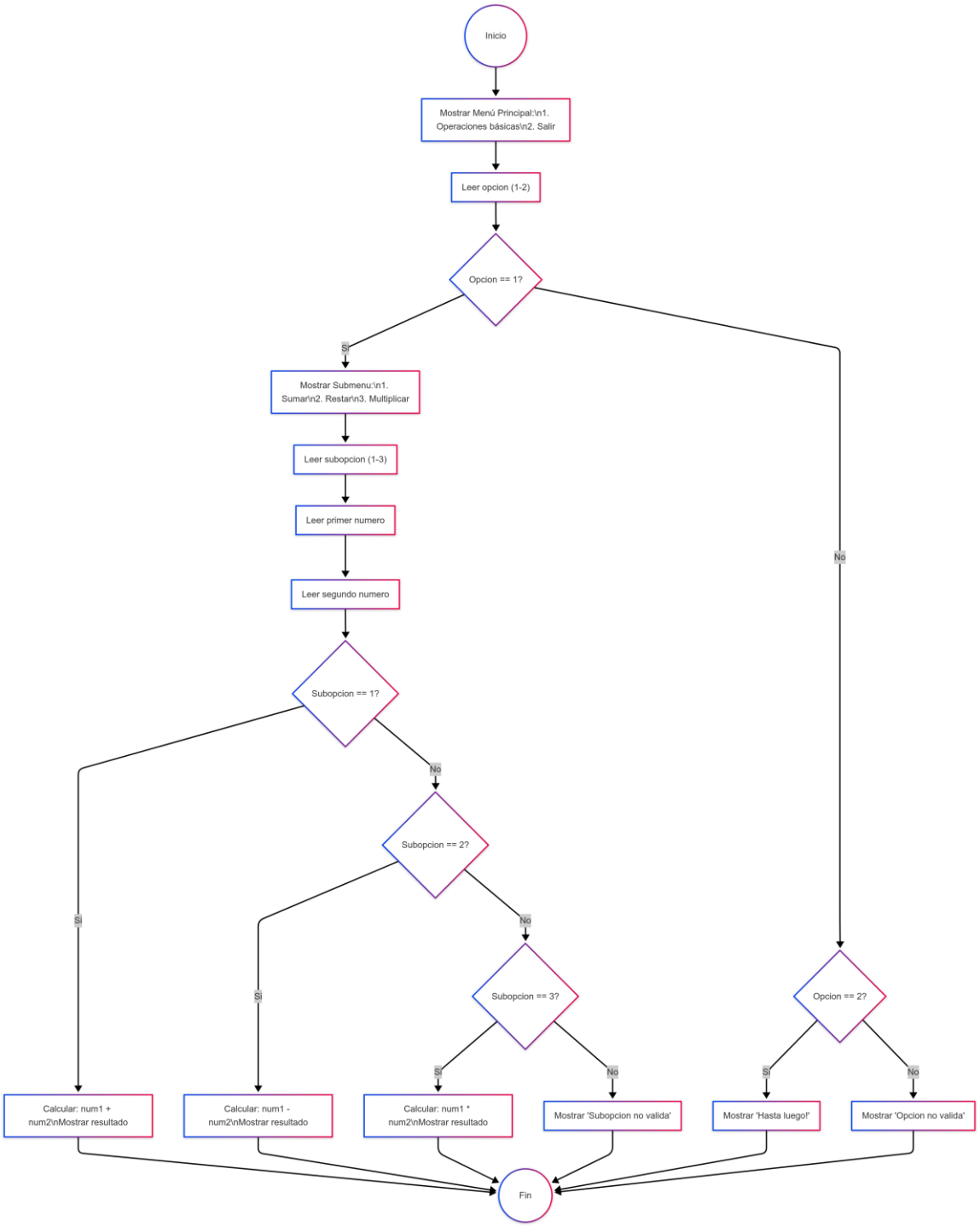


Imagen 2: Diagrama de flujo de un menú con subopciones
Damián Nicolalde Rodríguez. (2025). Diagrama de flujo de un menú con subopciones.

Este diagrama ilustra cómo el programa maneja un menú con subopciones. Las decisiones dentro del submenú se evalúan dentro de las decisiones del menú principal, creando un flujo de ejecución más complejo.

Beneficios de las Condicionales Anidadas

Las condicionales anidadas tienen numerosos beneficios, especialmente cuando se trata de manejar situaciones más complejas donde se deben evaluar varias condiciones antes de tomar una acción.

- Manejo de múltiples condiciones: Permiten manejar múltiples condiciones dentro de un solo bloque de código. Esto es útil cuando se necesita evaluar diversas situaciones antes de tomar una decisión. Los menús interactivos y los sistemas de validación de datos son ejemplos donde se puede aplicar esta estructura de manera eficiente.

- Flexibilidad en la toma de decisiones: Las condicionales anidadas permiten construir programas más dinámicos y adaptativos. Por ejemplo, en sistemas de clasificación, el programa puede evaluar una serie de condiciones antes de asignar un valor o tomar una acción.

- Mejor control del flujo de ejecución: El uso de condicionales anidadas brinda un control más detallado sobre el flujo del programa. Esto es particularmente útil en aplicaciones que requieren un comportamiento específico dependiendo de las condiciones que se cumplan.

Las condicionales anidadas son una herramienta poderosa en la programación que permiten manejar decisiones complejas dentro de un programa. Son especialmente útiles en menús de opciones, formularios de validación o cualquier sistema que requiera decisiones basadas en múltiples condiciones. Al aprender a implementar estas estructuras, los programadores pueden crear aplicaciones más flexibles y robustas que respondan dinámicamente a las entradas del usuario.

Este concepto es fundamental para la creación de programas interactivos y adaptativos, y dominar su uso es esencial para la construcción de aplicaciones más sofisticadas y útiles.

Para complementar y profundizar, se sugieren acceder a este recurso:

- Título del enlace relacionado: Condicionales anidadas en PYTHON
- **Descripción del enlace relacionado:** En este video aprenderán sobre como manipular las sentencias condicionales anidadas.
- **Enlace:** [Tutorial 15] Condicionales anidadas en PYTHON

8.2. Operador ternario para simplificar código

El operador ternario es una estructura condicional compacta que nos permite realizar decisiones simples en una sola línea de código. Es ideal para situaciones en las que solo necesitamos escoger entre dos posibles resultados basados en una condición. Este operador es una forma eficiente de escribir decisiones en programas, especialmente cuando se trata de condiciones que no requieren bloques de código largos.

¿Qué es el Operador Ternario?

El operador ternario permite hacer una evaluación condicional en una sola línea de código, lo cual es mucho más eficiente que usar un bloque if-else tradicional cuando solo tenemos dos posibles resultados. Su sintaxis es la siguiente:

```
resultado = valor_true if condición else valor_false
```

El operador ternario permite realizar una evaluación condicional y retornar un valor de manera más eficiente. En lugar de escribir un bloque completo de código if-else, se puede realizar la evaluación y asignar el valor en una sola línea, lo que hace que el código sea más limpio y fácil de leer. Esta técnica es ideal cuando solo se necesita tomar una decisión sencilla entre dos opciones.

Ventajas del Operador Ternario

- Concisión: Evita la escritura de un bloque de código if-else completo, simplificando las decisiones condicionales simples.

- Legibilidad: El código es más limpio y más fácil de entender cuando solo se trata de una condición binaria.

- Eficiencia: Cuando solo hay dos posibles resultados, el operador ternario es una opción mucho más eficiente que un bloque completo de if-else.

Es importante recordar que el operador ternario es más adecuado para decisiones simples. Si se requieren verificaciones más complejas o múltiples condiciones, el uso de if-else es más apropiado.

Ejemplo:

Vamos a comenzar con un ejemplo sencillo para mostrar cómo usar el operador ternario para evaluar una condición y elegir entre dos opciones. Supongamos que queremos verificar si una persona es mayor de edad.

```python
# Ejemplo de uso del operador ternario

edad = int(input("Ingrese su edad: "))

# Uso del operador ternario
mensaje = "Mayor de edad" if edad >= 18 else "Menor de edad"

print(mensaje)
```

En este ejemplo, el operador ternario evalúa la condición edad >= 18. Si la condición es verdadera (es decir, la edad es mayor o igual a 18), la variable mensaje recibirá el valor "Mayor de edad". Si la condición es falsa (es decir, la edad es menor a 18), se asignará el valor "Menor de edad". Este es un ejemplo típico de una decisión simple, donde solo tenemos dos posibles resultados.

Este tipo de implementación es mucho más compacta y clara que si usáramos una estructura if-else tradicional:

```python
if edad >= 18:
    mensaje = "Mayor de edad"
else:
    mensaje = "Menor de edad"
```

El siguiente diagrama visualiza cómo el operador ternario evalúa la condición y elige entre dos posibles valores en una sola línea de código. Este diagrama ilustra de manera clara la eficiencia y simplicidad del operador ternario.

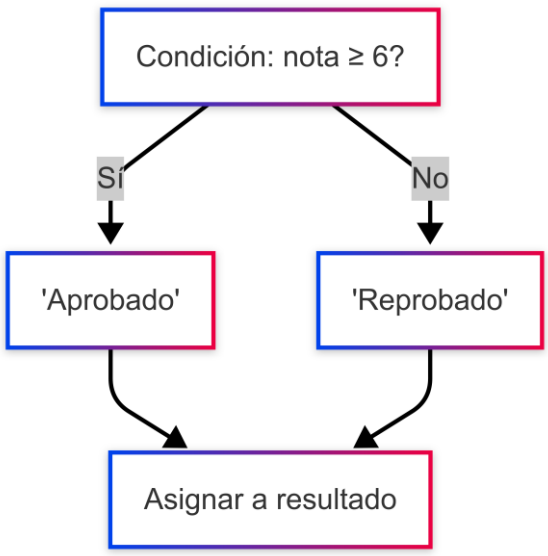


Imagen 3: Ejemplo visual del operador ternario
Damián Nicolalde Rodríguez. (2025). Ejemplo visual del operador ternario.

Ahora, vamos a ver un uso más avanzado del operador ternario, donde evaluaremos múltiples condiciones en cadena utilizando el operador ternario anidado. Imaginemos que tenemos un sistema de clasificación de calificaciones, y queremos asignar una categoría a la calificación ingresada, como "Excelente", "Bueno", "Suficiente" o "Insuficiente".

```
# Clasificación de calificación
nota = float(input("Ingrese la calificación (0-10): "))

# Operador ternario anidado para clasificación de calificación
resultado = "Excelente" if nota >= 9 else "Bueno" if nota >= 7 else "Suficiente" if nota >= 5 else "Insuficiente"

print(f"Calificación: {resultado}")
```

En este ejemplo, el operador ternario está anidado, lo que significa que dentro de una decisión else, evaluamos una nueva condición. Dependiendo de la calificación ingresada:

- Si la calificación es mayor o igual a 9, el resultado será "Excelente".
- Si la calificación es mayor o igual a 7 pero menor que 9, el resultado será "Bueno".
- Si la calificación es mayor o igual a 5 pero menor que 7, el resultado será "Suficiente".
- Si la calificación es menor que 5, el resultado será "Insuficiente".

Este tipo de clasificación es muy eficiente y evita la necesidad de escribir un bloque if-elif-else largo, manteniendo el código más compacto y legible.

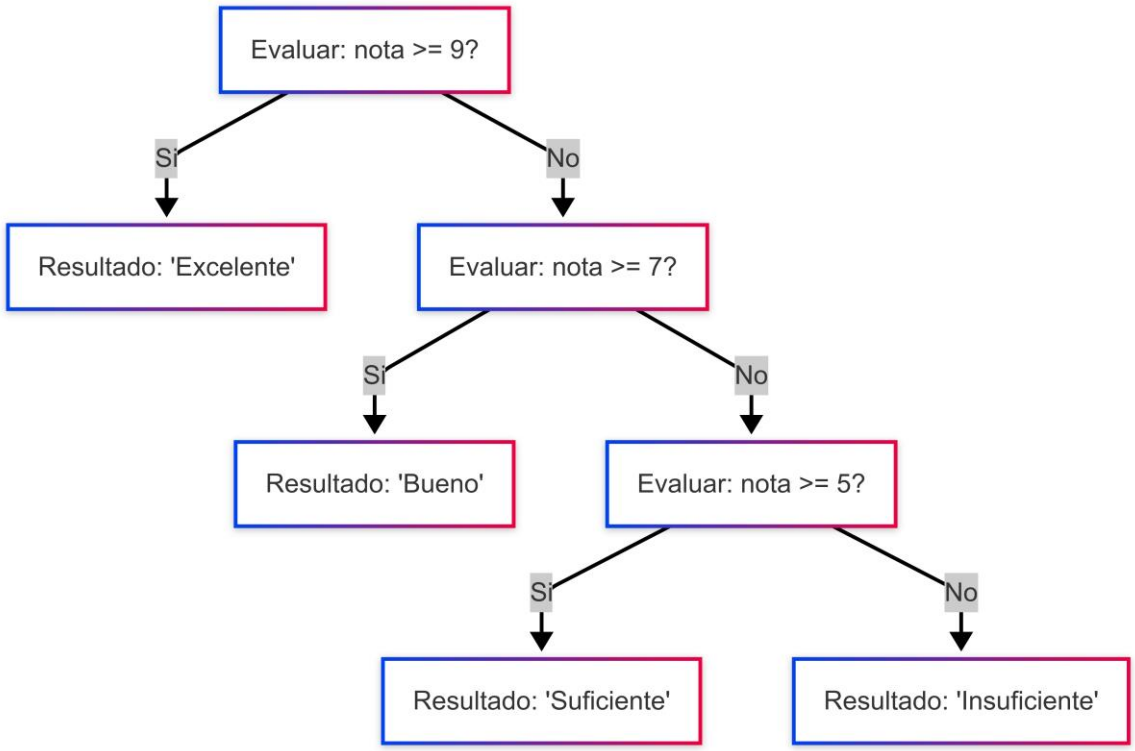


Imagen 4: Diagrama del Operador Ternario Anidado
Damián Nicolalde Rodríguez. (2025). Diagrama del Operador Ternario Anidado.

Para complementar y profundizar, se sugieren acceder a este recurso:

- Título del enlace relacionado: Operador Ternario en Python
- **Descripción del enlace relacionado:** En este video, profundizaremos en uno de los conceptos más útiles y poderosos de Python: el operador ternario. Si eres un programador de Python o estás comenzando a aprender, este es un concepto que definitivamente querrás dominar.
- **Enlace:** OPERADOR TERNARIO Python

En la siguiente tabla, se presentan dos estructuras condicionales ampliamente utilizadas en programación: if-else y el operador ternario. A continuación, se muestra un ejemplo de código para cada una de estas estructuras, junto con una descripción de sus características principales. Esta comparación te ayudará a comprender las diferencias clave entre ambas y cuándo es más apropiado utilizar cada una según la complejidad de la decisión que deseas tomar en tu código.

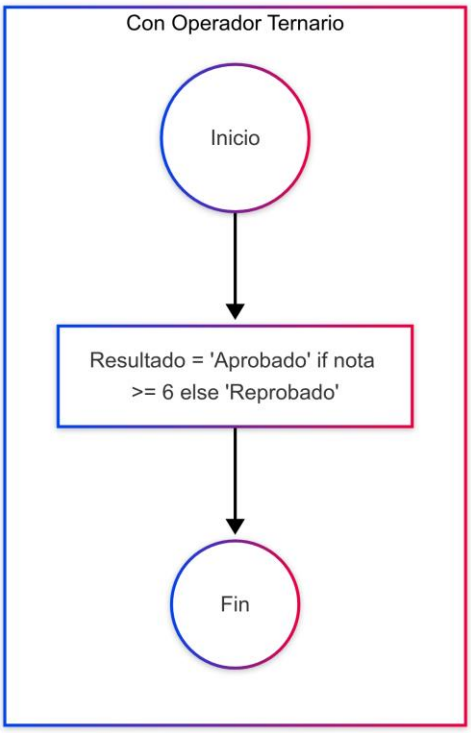
Tabla 1. Comparación entre if-else y operador ternario

Estructura	Ejemplo Código	Características
if-else	if (edad >= 18): print("Mayor de edad") else: print("Menor de edad")	Bloques de código más largos. Ideal para decisiones complejas.
Operador ternario	mensaje = "Mayor de edad" if edad >= 18 else "Menor de edad"	Expresión en una sola línea. Ideal para decisiones simples.

Damián Nicolalde Rodríguez. (2025).

En esta clase, hemos aprendido a manejar dos herramientas esenciales para la toma de decisiones dentro de un programa: las condicionales anidadas y el operador ternario. Las condicionales anidadas permiten que un programa ejecute diferentes bloques de código dependiendo de varias condiciones, como en los menús interactivos. Por otro lado, el operador ternario nos ofrece una forma eficiente y compacta de tomar decisiones simples entre dos posibles resultados, mejorando la legibilidad del código.

Estas herramientas mejoran la flexibilidad del programa, permitiendo que el código responda de manera dinámica a las entradas del usuario o a diversas condiciones. Al comprender y dominar estas estructuras, los estudiantes estarán mejor preparados para escribir programas más complejos, con flujos de ejecución que no solo sigan un camino lineal, sino que también puedan adaptarse a situaciones complejas.


Con Operador Ternario
Inicio
Resultado = 'Aprobado' if nota >= 6 else 'Reprobado'
Fin

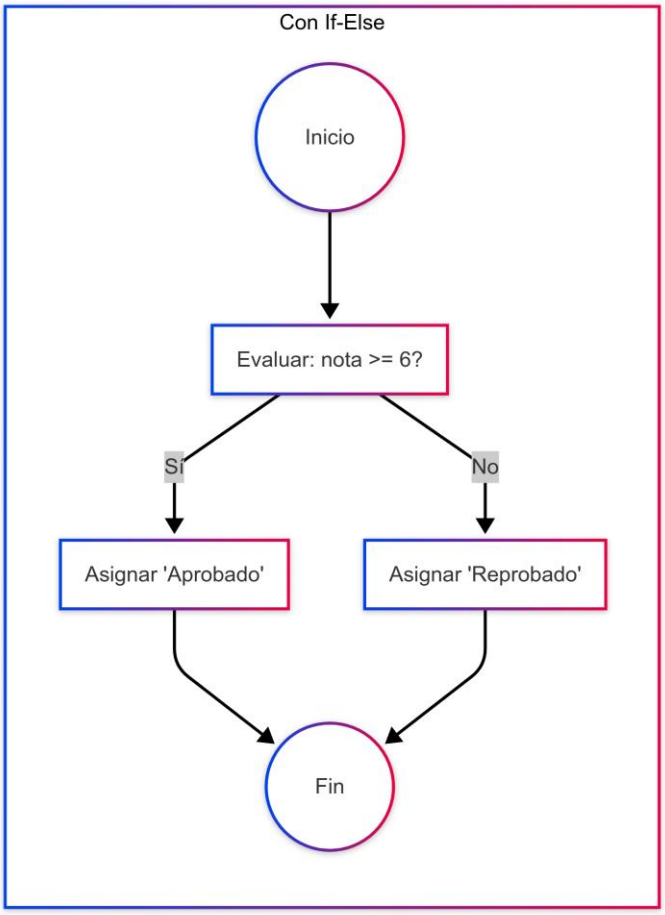

Con If-Else
Inicio
Evaluar: nota >= 6?
Sí
No
Asignar 'Aprobado'
Asignar 'Reprobado'
Fin

Imagen 4: Diagrama de flujo con subopciones en un menú interactivo
Damián Nicolalde Rodríguez. (2025). Diagrama de flujo con subopciones en un menú interactivo.

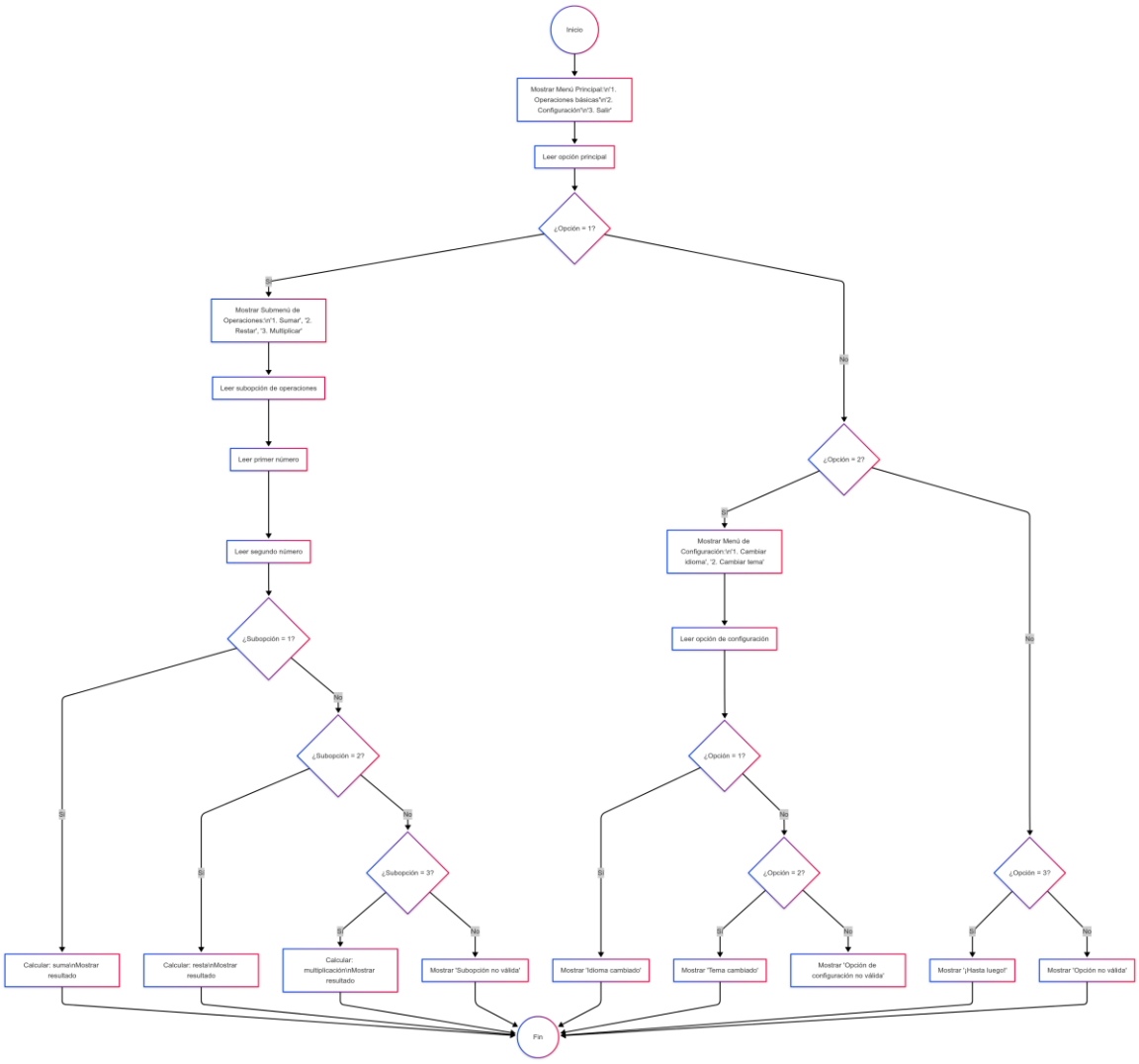


Imagen 5: Ejemplo de menú interactivo con múltiples niveles de decisión
Damián Nicolalde Rodríguez. (2025). Ejemplo de menú interactivo con múltiples niveles de decisión.

Referencias citadas en la Clase 8.

- https://elibro.puce.elogim.com/es/ereader/puce/230298
- https://puce.odilo.us/info/facil-aprendizaje-estructuras-de-datos-algoritmos-c-aprenda-facilmente-estructuras-de-datos-graficamente-03127232
- https://puce.odilo.us/info/aprende-c-en-un-fin-de-semana-03105596

Definición de los términos citados en la Clase 8.

Condicionales Anidadas: Estructuras de control donde una condición está contenida dentro de otra. Son útiles para tomar decisiones más complejas en el flujo de ejecución de un programa.

Operador Ternario: Una forma compacta de escribir un condicional if-else, que evalúa una condición y retorna uno de dos valores posibles en una sola línea de código.

PUCE Virtual
EDUCACIÓN FLEXIBLE

La excelencia no se improvisa

síguenos

MDASC - 04 - 251