Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😁

alkemy

Clase grabada

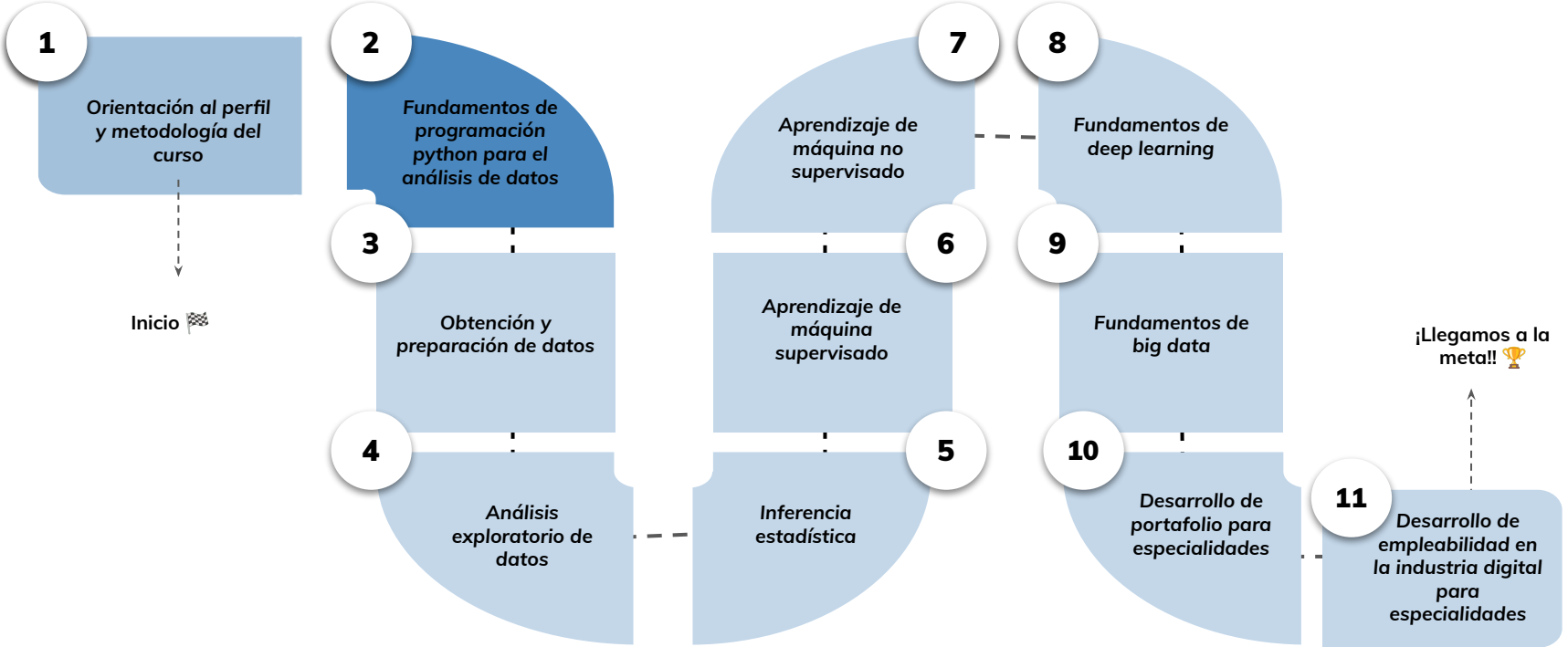
Sentencias condicionales - Parte 2

Aprendizaje Esperado 3: Codificar una rutina utilizando estructuras condicionales y expresiones booleanas para resolver un problema de baja complejidad de acuerdo al lenguaje Python.

alkemy

Hoja de ruta

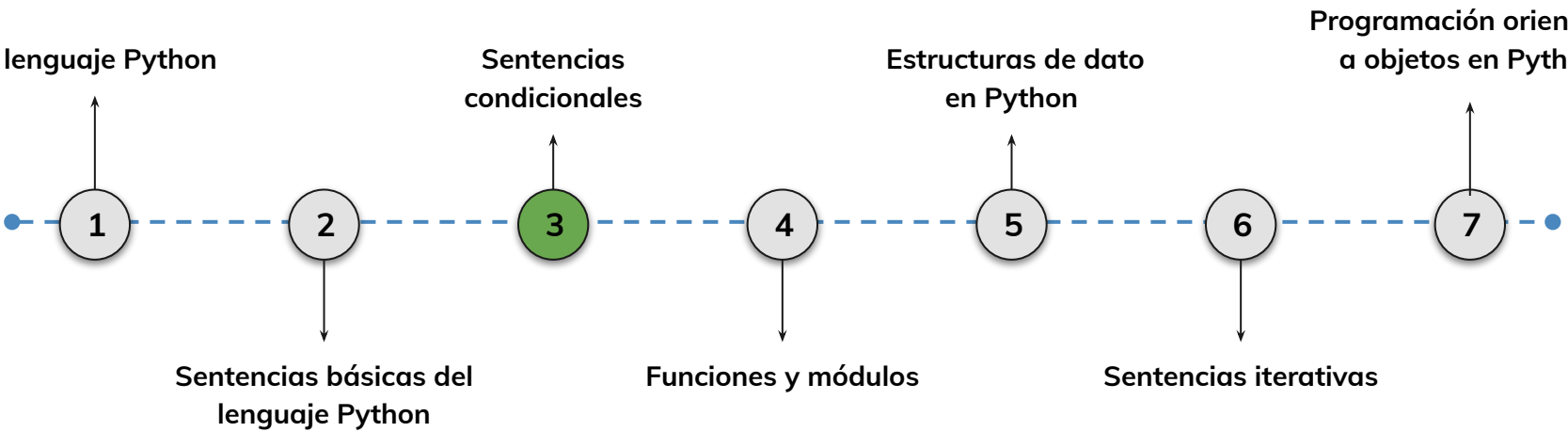
¿Cuáles skills conforman el programa? **Fundamentos de Ciencia de Datos**



alkemy

Roadmap de lecciones

*¿Cuáles **lecciones** estaremos estudiando en este módulo?*



alkemy

Learning Path

¿Cuáles temas trabajaremos hoy?

3.

Construcción de decisiones múltiples y compactas con condicionales en Python

En esta clase profundizaremos en el uso de sentencias condicionales, incorporando estructuras if-elif-else y expresiones ternarias para resolver problemas con múltiples caminos posibles de forma clara y eficiente.

- if-elif-else
  - Evaluación secuencial de múltiples condiciones
  - Organización lógica y clasificación de datos
- Expresiones ternarias
  - Sintaxis compacta para decisiones simples
  - Casos de uso y buenas prácticas

alkemy

Sentencias condicionales

Objetivos de aprendizaje

¿Qué aprenderás?

- Aplicar estructuras if-elif-else para manejar múltiples condiciones
- Usar expresiones ternarias para codificar decisiones simples
- Identificar cuándo conviene usar cada tipo de estructura condicional
- Mejorar la legibilidad del código mediante condiciones claras y bien estructuradas
- Detectar errores comunes en condicionales y corregirlos

Repaso clase anterior

¿Quedó alguna duda?

En la clase anterior trabajamos 📚:

- → *Aplicamos estructuras if, else y operadores booleanos para tomar decisiones.*
- → *Evaluamos condiciones con and, or, not y operadores de comparación.*
- → *Diseñamos rutinas que mostraban diferentes respuestas según la entrada del usuario.*
- → *Practicamos validaciones y estructuras simples de flujo lógico.*

alkemy

Ejemplo básico de if

En este ejemplo, el mensaje "Tienes acceso" solo se mostrará si la variable edad es mayor o igual a 18. Si la condición es falsa, el programa simplemente continuará con la siguiente instrucción después del bloque if.

```
temperatura = 30

if temperatura > 25:
    print("Hace calor.")
# Salida: Hace calor.
```

Condiciones Booleanas en if

La condición en un if puede incluir comparaciones simples o expresiones booleanas. Si la condición es verdadera, se ejecuta el bloque de código dentro del if; si no, se omite.

En este ejemplo, se utilizan operadores booleanos para crear una condición más compleja que determina si un estudiante puede inscribirse en un curso.

```
edad = 20
tiene_permiso = True

if edad >= 18 and tiene_permiso:
    print("Puedes conducir.")
# Salida: Puedes conducir.
```

Uso Práctico de if en Validaciones

El if es útil para verificar entradas del usuario, controlar flujos de datos y validar requisitos antes de proceder con operaciones. Esto asegura que solo se ejecute el código si las condiciones son adecuadas.

```
# Validación de entrada de usuarioentrada = input("Ingrese un número: ")if entrada.isdigit():    numero = int(entrada)    print(f"El cuadrado de {numero} es {numero**2}")
```

En este ejemplo, el programa solo calcula el cuadrado si la entrada es un número válido.

Buena Práctica: Identación en el Bloque if

Python requiere una identación adecuada en los bloques if. Todo el código dentro del if debe estar correctamente identado para evitar errores y asegurar la ejecución adecuada del bloque.

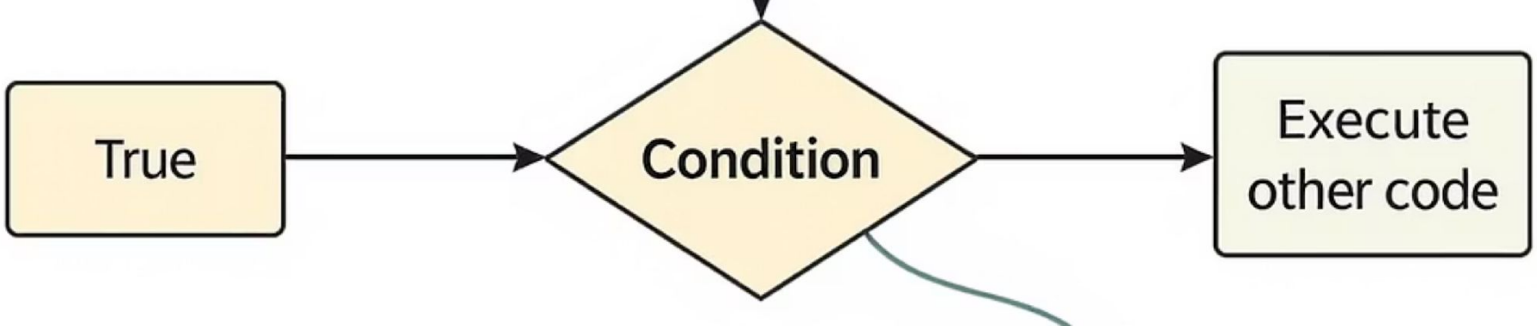
Correcto

```
if condicion:    print("Línea 1")
print("Línea 2")print("Fuera del if")
```

Incorrecto

```
if condicion:print("Error: sin identación")
print("Identación inconsistente")
```

La identación correcta es fundamental en Python, ya que define los bloques de código.



La Sentencia if-else

Estructura de if-else

La sentencia if-else permite manejar dos posibles caminos de ejecución. Si la condición es verdadera, se ejecuta el bloque if; si es falsa, se ejecuta el bloque else. Esto permite que el programa maneje ambas opciones de manera estructurada.

Ejemplo de if-else

En este ejemplo, el programa muestra un mensaje diferente dependiendo de si el usuario es mayor de edad o no. La estructura if-else garantiza que siempre se ejecute uno de los dos bloques.

```
hora = 10

if hora < 12:
    print("Buenos días.")
else:
    print("Buenas tardes.")
# Salida: Buenos días.
```

Cuando Usar if-else

if-else es útil en casos donde el programa debe responder a una condición de manera exclusiva, permitiendo un flujo claro y alternativo en el código.

Situaciones binarias

Cuando solo hay dos posibles resultados (sí/no, verdadero/falso)

Manejo de errores simple

Para proporcionar un camino alternativo cuando una operación falla

Validaciones con respuesta

Cuando necesitas proporcionar retroalimentación tanto para entradas válidas como inválidas

Comparaciones y Alternativas en if-else

if-else también permite comparar variables y ejecutar distintos bloques de acuerdo a los resultados, mejorando la adaptabilidad de las aplicaciones.

```
# Comparación de dos númerosa = 10b = 20if a > b:    print(f"{a} es mayor que {b}")else:    print(f"{a} no es mayor que {b}")
```

Este código compara dos valores y muestra un mensaje apropiado según el resultado de la comparación.

Ejemplo de Validación de Edad con if-else

Este ejemplo muestra cómo utilizar if-else para validar la edad de un usuario y proporcionar mensajes específicos según el resultado de la validación.

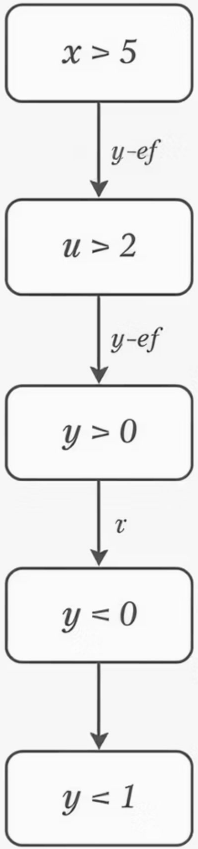
```
edad = 17

if edad >= 18:
    print("Acceso permitido.")
else:
    print("Acceso denegado.")
# Salida: Acceso denegado.
```

La Sentencia if-elif-else

Estructura de if-elif-else

La sentencia if-elif-else permite evaluar múltiples condiciones de forma secuencial. Si la primera condición es verdadera, se ejecuta su bloque; si no, se evalúa la siguiente condición elif, y así sucesivamente.



Ejemplo de if-elif-else

En este ejemplo, el programa evalúa la calificación de un estudiante y muestra un mensaje correspondiente según el rango en el que se encuentre. La estructura if-elif-else permite manejar múltiples casos de manera ordenada.

```
dia = "miércoles"

if dia == "lunes":
    print("Hoy es lunes.")
elif dia == "martes":
    print("Hoy es martes.")
else:
    print("Es otro día.")
# Salida: Es otro día.
```

Cuando Usar if-elif-else

Es útil cuando hay varias condiciones posibles y solo una debe ejecutarse, permitiendo manejar casos alternativos.

Múltiples condiciones excluyentes

Cuando necesitas evaluar varias posibilidades donde solo una debe ser verdadera

Categorización de datos

Para clasificar valores en diferentes categorías o rangos

Menús de opciones

Para implementar diferentes acciones basadas en la selección del usuario

Verificación de Rangos de Valores

En programación, if-elif-else es común para verificar rangos de valores o categorías.

Este ejemplo muestra cómo categorizar a una persona según su edad, utilizando rangos específicos para cada categoría.

```
nota = 85

if nota >= 90:
    print("Excelente")
elif nota >= 75:
    print("Bueno")
else:
    print("Necesita mejorar")
# Salida: Bueno
```

Flujo de Evaluación de Condiciones

El if-elif-else evalúa de forma secuencial, permitiendo un flujo estructurado para distintas situaciones.

Evaluación de la condición if

Si es verdadera, se ejecuta su bloque y se omiten las demás condiciones

Evaluación de la primera condición elif

Si la condición if es falsa y esta condición es verdadera, se ejecuta su bloque

Evaluación de condiciones elif adicionales

Se evalúan en orden hasta encontrar una verdadera o llegar al else

Ejecución del bloque else (opcional)

Se ejecuta solo si todas las condiciones anteriores son falsas

Ejemplo Completo de if-elif-else

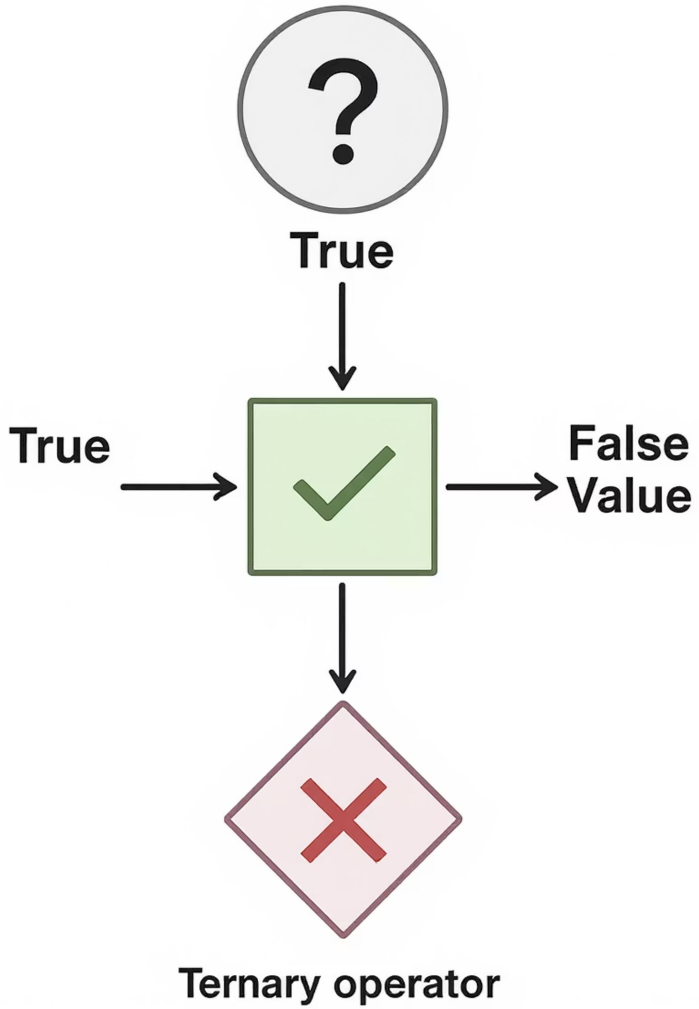
```
# Determinar el tipo de triángulo según sus ladoslado1 = 5lado2 = 5lado3 = 5if lado1 == lado2 == lado3:
print("Triángulo equilátero")elif lado1 == lado2 or lado1 == lado3 or lado2 == lado3:
print("Triángulo isósceles")else:    print("Triángulo escaleno")
```

Este código evalúa las longitudes de los lados de un triángulo y determina su tipo según las relaciones entre ellos.

Expresiones Ternarias

Concepto de Expresiones Ternarias

Las expresiones ternarias permiten escribir una condición if-else en una sola línea. Son útiles para casos simples y mejoran la legibilidad del código en comparaciones cortas.



Ejemplo de expresión ternaria

En este ejemplo, la expresión ternaria asigna "Mayor de edad" o "Menor de edad" a la variable mensaje, dependiendo del valor de edad.

Esta sintaxis compacta es equivalente a una estructura if-else tradicional.

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

Cuando Usar Expresiones Ternarias

Son ideales para asignaciones rápidas o impresiones condicionales, evitando la necesidad de una estructura if-else extensa.

Asignaciones condicionales simples

Cuando necesitas asignar uno de dos valores posibles a una variable

Retornos condicionales en funciones

Para devolver diferentes valores basados en una condición simple

Formateo de texto condicional

Para incluir diferentes textos en un string según una condición

Ventajas en Código Compacto

La ternaria mejora la legibilidad cuando se usa con moderación, evitando estructuras largas.

Estructura if-else tradicional

```
if condicion:    resultado = valor1else:
resultado = valor2
```

Expresión ternaria

```
resultado = valor1 if condicion else valor2
```

La expresión ternaria reduce cuatro líneas de código a una sola, manteniendo la claridad de la lógica.

Ejemplo de Condición de Aprobación con Ternaria

Este ejemplo muestra cómo utilizar una expresión ternaria para determinar si un estudiante ha aprobado o no un curso, basándose en su calificación.

```
nota = 70
resultado = "Aprobado" if nota >= 60 else "Reprobado"
print(resultado)
# Salida: Aprobado
```

Comparación de Código: Tradicional vs Ternaria

Esta imagen muestra la comparación entre una estructura if-else tradicional y su equivalente usando una expresión ternaria.

Ambos códigos producen el mismo resultado, pero la expresión ternaria es más concisa.

Limitaciones de las Expresiones Ternarias

Aunque son útiles, las expresiones ternarias tienen algunas limitaciones:

No son adecuadas para lógica compleja

Si necesitas múltiples condiciones o bloques de código extensos, es mejor usar if-elif-else

Pueden reducir la legibilidad si se anidan

Las expresiones ternarias anidadas son difíciles de leer y mantener

No permiten múltiples instrucciones por bloque

Solo pueden ejecutar una expresión para cada caso (verdadero/falso)

Ejemplo Práctico: Validación de Entrada

```
# Validación de entrada con expresión ternariaedad_str = input("Ingrese su edad: ")edad = int(edad_str) if edad_str.isdigit() else 0print("Edad válida" if edad > 0 else "Edad inválida")
```

Este ejemplo muestra cómo utilizar expresiones ternarias para validar la entrada del usuario y proporcionar retroalimentación adecuada.

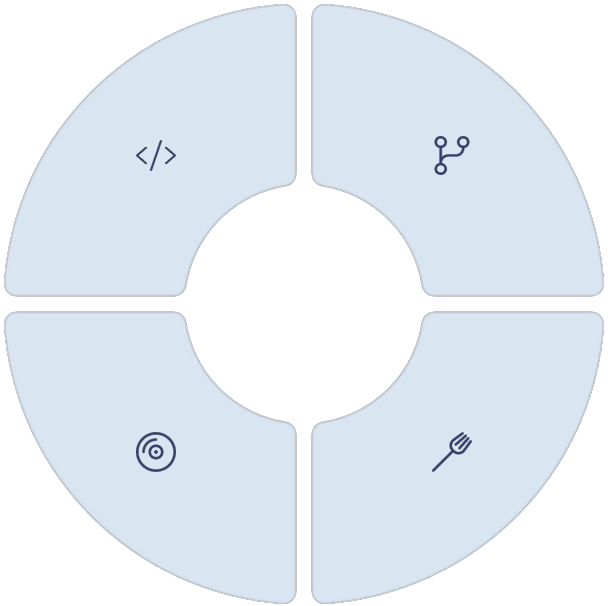
Resumen de Sentencias Condicionales

if

Ejecuta un bloque de código si la condición es verdadera

Ternaria

Forma compacta de if-else en una sola línea



if-else

Ejecuta un bloque si la condición es verdadera, otro si es falsa

if-elif-else

Evalúa múltiples condiciones en secuencia

alkemy

Operadores de Comparación y Booleanos

1

2

Operadores de Comparación				
Operador	Nombre	Fórmula	Ejemplo	Resultado
=	Igual a	=A1=A2	=14=20	FALSO
>	Mayor que	=A1>A2	=20>31	FALSO
<	Menor que	=A1<A2	=13<22	VERDADERO
>=	Mayor o igual que	=A1>=A2	=10>=14	FALSO
<=	Menor o igual que	=A1<=A2	=15<=15	VERDADERO
<>	Diferente de	=A1<>A2	=15<>12	VERDADERO

Operadores de Comparación

>, >=, <, <=, ==, != para comparar valores

OPERADOR	DESCRIPCIÓN	EJEMPLO
AND (Intersección)	Reduce y especifica la búsqueda.	Contabilidad AND Auditoría 
OR (Union)	Amplia la búsqueda.	Contabiblidad OR Auditoría 
NOT (Exclusión)	El término o expresión que le sigue.	Contabilidad NOT Auditoría. 

Operadores Booleanos

and, or, not para combinar condiciones

alkemy

Buenas Prácticas en Sentencias Condicionales

Mantener las condiciones simples y legibles

Dividir condiciones complejas en variables con nombres descriptivos

Usar paréntesis para clarificar la precedencia

Especialmente en expresiones con múltiples operadores booleanos

Evitar la anidación excesiva de condicionales

Considerar refactorizar o usar funciones para reducir la complejidad

Ser consistente con la identación

Usar siempre el mismo estilo de identación (4 espacios es lo recomendado en Python)

Casos de Uso Comunes

Validación de Formularios

Verificar que los datos ingresados cumplan con los requisitos antes de procesarlos

Filtrado de Datos

Seleccionar registros que cumplan con ciertos criterios en análisis de datos

Lógica de Juegos

Determinar acciones basadas en el estado del juego y las entradas del usuario

Errores Comunes a Evitar

Confundir = (asignación) con == (comparación)

Usar = en una condición asigna un valor en lugar de compararlo

Olvidar los dos puntos (:) después de la condición

La sintaxis correcta requiere dos puntos al final de cada línea de condición

Identación inconsistente

Python utiliza la identación para definir bloques, por lo que debe ser consistente

No considerar todos los casos posibles

Asegurarse de que todas las situaciones estén cubiertas en la lógica condicional

Conclusión

Las sentencias condicionales en Python son fundamentales para permitir que un programa tome decisiones basadas en condiciones específicas. Al utilizar estructuras como if, if-else, y if-elif-else, los programadores pueden controlar qué bloques de código se ejecutan en función de los datos o del contexto, lo que otorga flexibilidad y adaptabilidad a las aplicaciones.

Estas estructuras condicionales, junto con los operadores de comparación y booleanos (and, or, not), permiten evaluar expresiones lógicas complejas, lo cual es esencial para manejar una amplia variedad de escenarios y casos en la ejecución de un programa.

Beneficios de las Sentencias Condicionales

La comprensión y uso adecuado de los operadores y expresiones booleanas mejora la claridad y eficiencia del código, ayudando a evitar errores comunes y a hacer el código más mantenible. Las condicionales también permiten manejar validaciones de entrada, personalización de respuestas y adaptación de comportamientos en función de diversas condiciones, lo cual es crucial en aplicaciones interactivas o sistemas complejos.

Mediante el uso de paréntesis y expresiones ternarias, los desarrolladores pueden crear condiciones más legibles y eficientes, optimizando tanto el flujo de trabajo como la experiencia del usuario.

Reflexión Final

En conclusión, las sentencias condicionales son la base para una programación lógica y adaptable, y su dominio es clave para progresar hacia estructuras de control más complejas, como bucles y funciones. Practicar con diferentes tipos de condicionales y situaciones permite desarrollar una lógica de programación más sólida y ayuda a resolver problemas de forma estructurada y efectiva.

Con una base fuerte en el uso de sentencias condicionales, los programadores pueden abordar proyectos más complejos y crear aplicaciones que respondan de manera eficaz a las necesidades de los usuarios.

Live Coding

¿En qué consistirá la Demo?

Vamos a construir una rutina que clasifique a una persona según su edad e indique si tiene acceso a un beneficio.

1. *Solicitar edad y categoría (estudiante, docente, visitante)*
2. *Evaluar múltiples condiciones usando if-elif-else*
3. *Usar una expresión ternaria para mostrar un mensaje resumen*
4. *Incorporar validaciones adicionales (por ejemplo, entrada inválida)*
5. *Aplicar indentación clara y lógica secuencial*
6. *Mostrar resultados personalizados según combinación de condiciones*
7. *Incluir una comparación de sintaxis tradicional vs ternaria*
8. *Comentar el código para explicar la lógica aplicada*

Tiempo: 25 Minutos

#Momentode Preguntas...

- *¿Qué diferencia hay entre elif y else?*
- *¿Qué sucede si ninguna condición en un if-elif-else se cumple?*
- *¿Cuándo conviene usar una expresión ternaria en lugar de if-else?*
- *¿Qué errores comunes se deben evitar en las estructuras condicionales?*

Momento:

Time-out!

⏰5 -10 min.

Ejercicio N° 1

¿Qué categoría eres?

¿Qué categoría eres?

Contexto: 🙌

Queremos desarrollar una rutina que clasifique a una persona según su edad y brinde recomendaciones personalizadas.

Consigna: ✍️

Codificar una rutina utilizando estructuras condicionales y expresiones booleanas para resolver un problema de baja complejidad de acuerdo al lenguaje Python.

Tiempo 🕛**:** *30 Minutos*

Paso a paso: ⚙️

1. Solicitar al usuario:

- edad
- rol o categoría (estudiante, docente, visitante)

2. Clasificar edad:

- Menor a 13 → "Infancia"
- 13 a 17 → "Adolescencia"
- 18 a 64 → "Adultez"
- 65 o más → "Persona mayor"

3. Evaluar combinación de rol y edad para habilitar acceso o no a un beneficio
4. Usar ternaria para mostrar si accede o no al sistema
5. Imprimir resumen final de edad, rol y acceso
6. Validar entradas con if-else y operadores de comparación

alkemy

¿Alguna consulta?

alkemy

Resumen

¿Qué logramos en esta clase?

- ✓ ***Aplicamos estructuras if-elif-else para clasificar datos***
- ✓ ***Usamos expresiones ternarias para codificar lógica simple***
- ✓ ***Comprendimos cuándo usar estructuras tradicionales vs compactas***
- ✓ ***Identificamos errores comunes y los evitamos con buenas prácticas***
- ✓ ***Creamos una rutina con múltiples condiciones y salida personalizada***

¡Ponte a prueba!

Momento de ejercitación

Te invitamos a aprovechar esta última sección del espacio sincrónico para realizar de manera individual las ***actividades disponibles en la plataforma.*** *Estas propuestas son claves para afianzar lo trabajado y* ***forman parte obligatoria del recorrido de aprendizaje.***

💪 Análisis de caso ——— 💪 Selección Múltiple

💪 Comprensión lectora

Si al resolverlas surge alguna duda, compartela o tráela al próximo encuentro sincrónico.

alkemy

¡Muchas gracias!

alkemy