PROGRAMACIÓN 1

1º AÑO

CLASE Nº 4

Flujo de control I

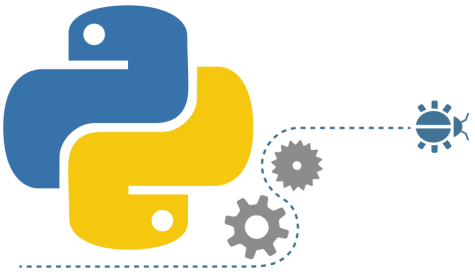
Estructuras condicionales

Contenido

En la clase de hoy trabajaremos los siguientes temas:

- Flujo de control de un programa.
- Estructura secuencial.
- Condicional simple -> Sentencia if.
- Condicional doble -> Sentencia if y else.
- Condicional múltiple -> Sentencia elif.
- Condicionales anidadas.
- Condiciones más complejas.

1. Presentación

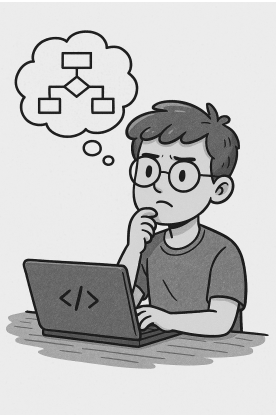


¡Bienvenidos al emocionante mundo de las estructuras selectivas en Python!

Aquí aprenderás cómo tomar decisiones inteligentes en tus programas usando las declaraciones "if", "else" y "elif". Descubrirás cómo crear lógica condicional, evaluar expresiones y guiar el flujo de tu código según condiciones específicas. ¡Prepárate para potenciar tus habilidades de programación y hacer que tus programas tomen decisiones informadas!

Pregunta para reflexión

Antes de comenzar, piensen en esto: Si tuvieras que indicarle a una computadora "qué hacer si está lloviendo y qué hacer si no lo está", **¿Cómo lo explicarías paso a paso?**



(Tomémonos unos minutos para pensar)

Lo que acabas de hacer es pensar en **condiciones**: instrucciones que dependen de ciertas situaciones. Hoy aprenderemos a programarlas.

Vídeo Tutorial

En el siguiente video se presenta una introducción a las sentencias condicionales if, elif y else en Python. Es ideal para comenzar a entender cómo tomar decisiones dentro de un programa.

Programación en Python - Condicionales

2. Desarrollo

2.1 Flujo de control de un programa

El flujo de control se refiere a la secuencia en la que se ejecutan las instrucciones de un programa y cómo se toman decisiones o se repiten ciertas partes del código según ciertas condiciones. En Python, el flujo de control se maneja mediante estructuras como instrucciones condicionales (if, elif, else) y bucles (for, while). Estas estructuras permiten controlar la dirección y la repetición de la ejecución del programa.

2.2 Estructura secuencial

La estructura secuencial es un concepto fundamental en programación que se refiere a la ejecución ordenada de instrucciones en un programa, una después de la otra, siguiendo una secuencia lógica.

En Python, la estructura secuencial permite ejecutar una serie de declaraciones en el orden en que están escritas, sin realizar saltos ni desvíos. En resumen, cada instrucción se ejecuta en el orden escrito.

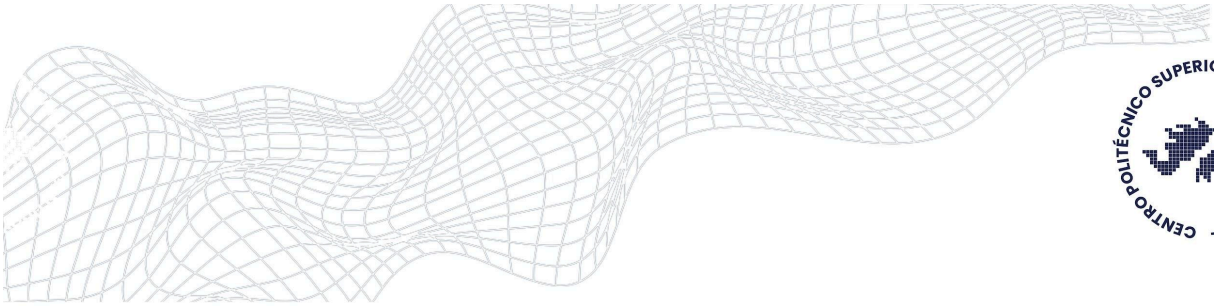
Ejemplo

Supongamos que queremos realizar un programa para sumar dos números, podríamos resolverlo de la siguiente manera:

```python
# Ingreso de datos por el usuario
numero1 = float(input("Ingrese el primer número: "))
numero2 = float(input("Ingrese el segundo número: "))

# Suma de los números
suma = numero1 + numero2

# Mostrar el resultado
print(f"La suma de {numero1} y {numero2} es: {suma}")
```

Donde, en este caso, las instrucciones también se ejecutan en secuencia:

1. El usuario ingresa el primer número.
2. El usuario ingresa el segundo número.
3. Se calcula la suma de los dos números ingresados.
4. Se muestra el resultado de la suma en la pantalla.



2.3 Estructuras condicionales

Las estructuras condicionales o también llamadas selectivas son bloques de programación que permiten que un programa tome decisiones y realice diferentes acciones en función de ciertas condiciones o situaciones.

Estas estructuras son fundamentales para controlar el flujo de ejecución en un programa y permiten que éste se adapte y tome diferentes caminos según las circunstancias.

Condicional simple -> Sentencia if

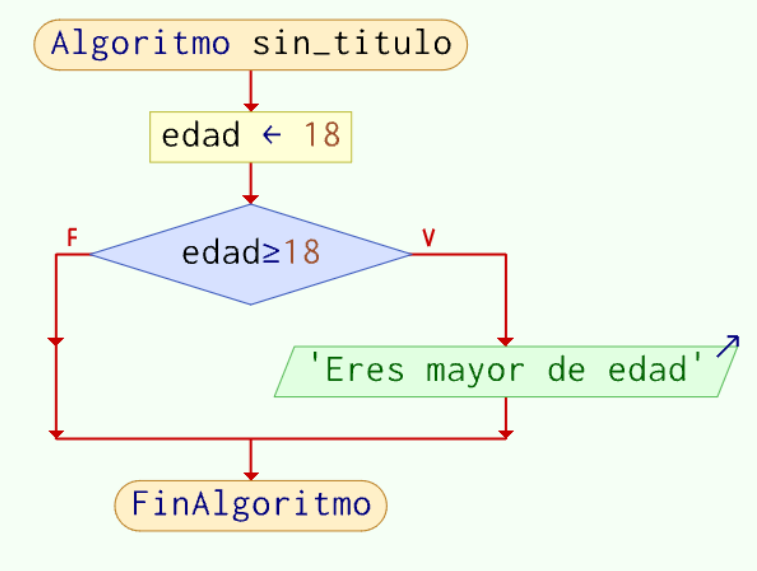
La sentencia "if" (si, en inglés) es una estructura fundamental en programación que permite controlar el flujo de ejecución de un programa en función de una condición. Esta condición puede ser verdadera o falsa, y en función de su evaluación, el programa tomará diferentes caminos y realizará diferentes acciones.

Ejemplo

Supongamos que queremos realizar un programa para evaluar si una persona es mayor de edad, podríamos resolverlo en Python de la siguiente manera:

```python
edad = 18

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

¿Y si lo resolvemos con un diagrama de flujo en PSeInt? Van a notar que seguimos la misma lógica, pero expresada de forma visual, como se muestra a continuación:



Condicional simple -> Sentencias if y else

La instrucción **else** se utiliza junto con **if** para ejecutar un bloque de código cuando la condición del "if" no es verdadera.

La sentencia **if** y **else** son estructuras fundamentales en programación que permiten tomar decisiones en función de una condición. Estas estructuras controlan el flujo de ejecución del programa al evaluar una expresión booleana y ejecutar diferentes bloques de código según si la condición es verdadera o falsa.

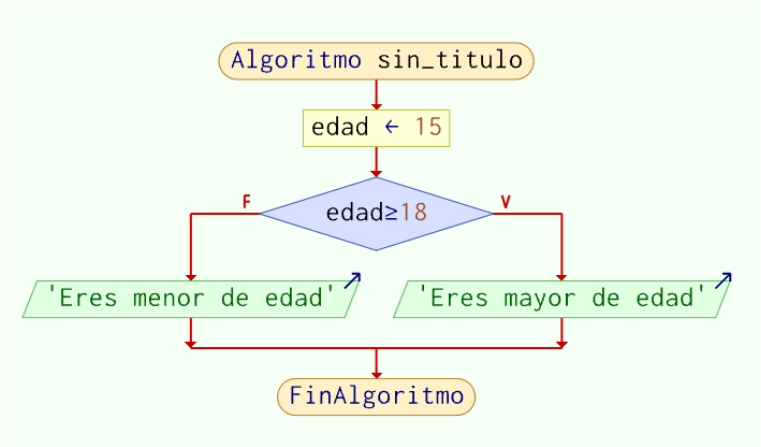
Ejemplo

Supongamos que queremos realizar un programa para evaluar si una persona es mayor de edad, o es menor de edad, podríamos resolverlo en Python de la siguiente manera:

```python
edad = 15

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

En este ejemplo, se evalúa si la variable edad es mayor o igual a 18. **Si** se cumple la condición, el programa imprime que la persona es mayor de edad; **de lo contrario**, indica que es menor de edad.

Ahora bien, si quisiéramos resolver este mismo problema en **PSeInt**, verán que la lógica se mantiene. Podemos representarla utilizando un **diagrama de flujo**, tal como se muestra a continuación:



Condicional múltiple -> Sentencias elif

elif es una abreviatura de **else if** en muchos lenguajes de programación, incluido Python. Se utiliza como parte de una estructura condicional para evaluar múltiples condiciones en orden y ejecutar el bloque de código asociado a la primera condición que sea verdadera.

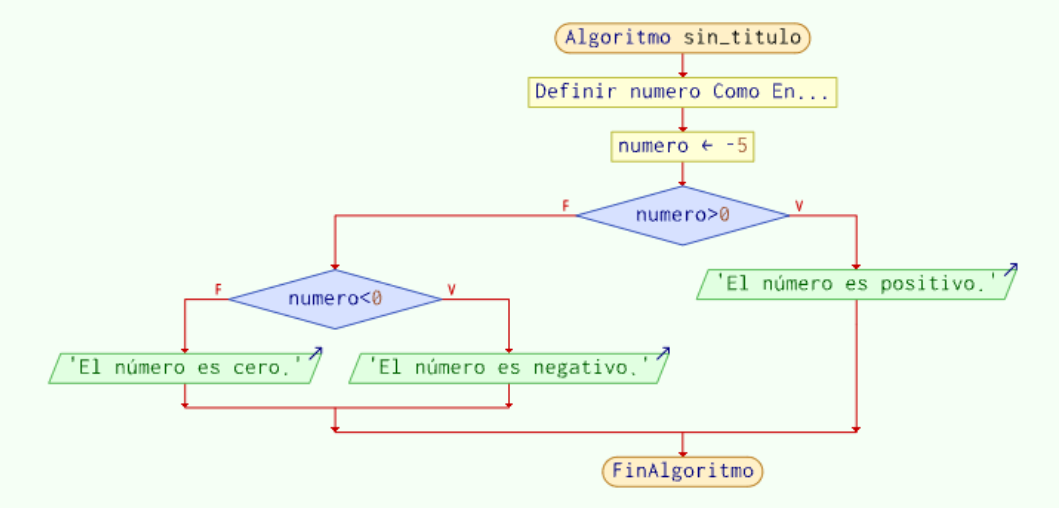
Ejemplo

Supongamos que queremos crear un programa en Python que determine si un número ingresado es positivo, negativo o igual a cero. Para ello, podemos utilizar una estructura condicional if - elif - else, como se muestra a continuación:

```python
numero = -5  # Cambia este valor para probar diferentes números

if numero > 0:
    print("El número es positivo.")
elif numero < 0:
    print("El número es negativo.")
else:
    print("El número es cero.")
```

Este programa evalúa el valor de la variable **numero** y muestra un mensaje correspondiente dependiendo de su signo.

Ahora bien, si quisiéramos implementar este mismo programa en **PSeInt**, podríamos utilizar un **diagrama de flujo** manteniendo exactamente la misma lógica condicional. A continuación, se muestra cómo sería esa estructura visualmente:



Condicionales anidadas

Una estructura de decisión anidada es un concepto de programación que implica la inclusión de múltiples bloques de decisión dentro de otro bloque de decisión. En otras palabras, es la combinación de múltiples estructuras de decisión (como declaraciones "if" o "switch") dentro de una sola estructura, con el propósito de manejar una variedad más compleja de condiciones y escenarios.

Esta anidación de estructuras de decisión permite que un programa evalúe condiciones en varias etapas y tome diferentes rutas de acción en función de la combinación de condiciones que se cumplen. En esencia, las estructuras de decisión anidadas permiten crear lógica más sofisticada y detallada en el flujo de ejecución de un programa.

Ejemplo

Supongamos que queremos clasificar a una persona según su edad. Para ello, usaremos una **estructura de decisión anidada** en Python, que nos permitirá evaluar distintos rangos de edad y mostrar un mensaje apropiado para cada caso.

A continuación, te mostramos cómo se puede resolver este problema:

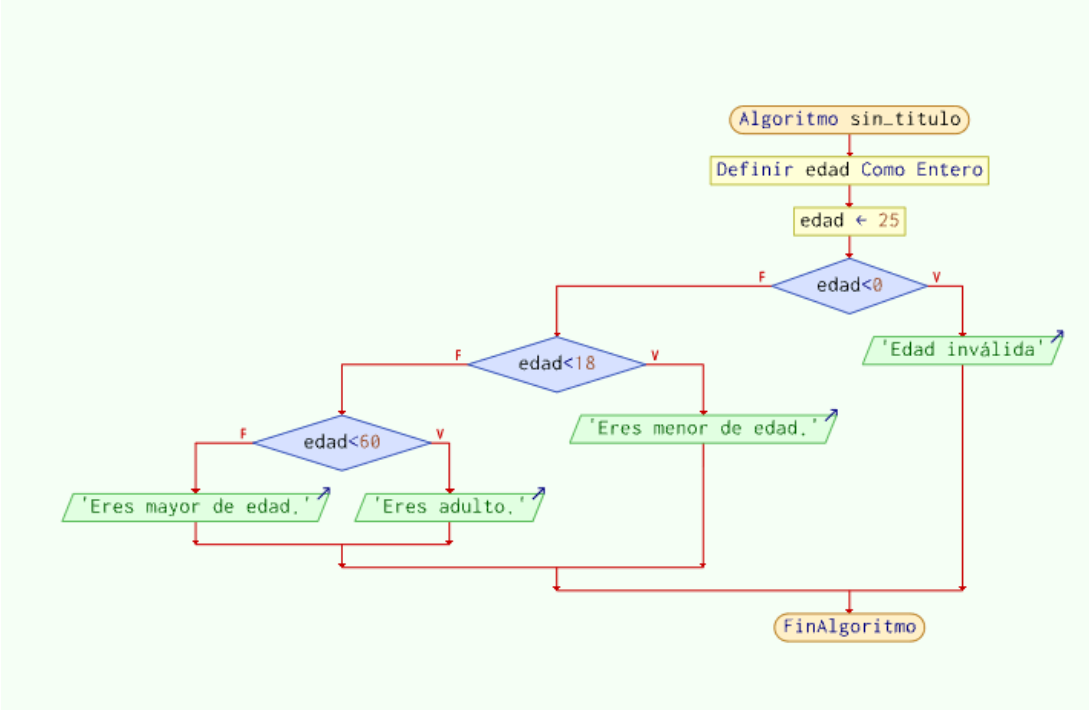
```python
# Ejemplo de estructura de decisión anidada en Python

# Definir la variable edad
edad = 25

# Comenzar la estructura de decisión anidada
if edad < 0:
    print("Edad inválida")
else:
    if edad < 18:
        print("Eres menor de edad.")
    else:
        if edad < 60:
            print("Eres adulto.")
        else:
            print("Eres mayor de edad.")
```

¿Qué hace este código?

1. Primero, verifica si la edad es menor a 0. Si es así, muestra un mensaje indicando que la edad es inválida.
2. Si la edad es válida, evalúa si es menor de 18 años: en ese caso, se trata de un menor de edad.
3. Si tiene 18 años o más, evalúa si es menor de 60 años. Si lo es, se considera adulto.
4. Finalmente, si la edad es igual o mayor a 60, se clasifica como mayor de edad.

Este tipo de estructura es muy útil cuando queremos aplicar múltiples condiciones dependientes entre sí.

Acá te dejamos la **versión en PSeInt** para el ejemplo anterior de clasificación por edad:



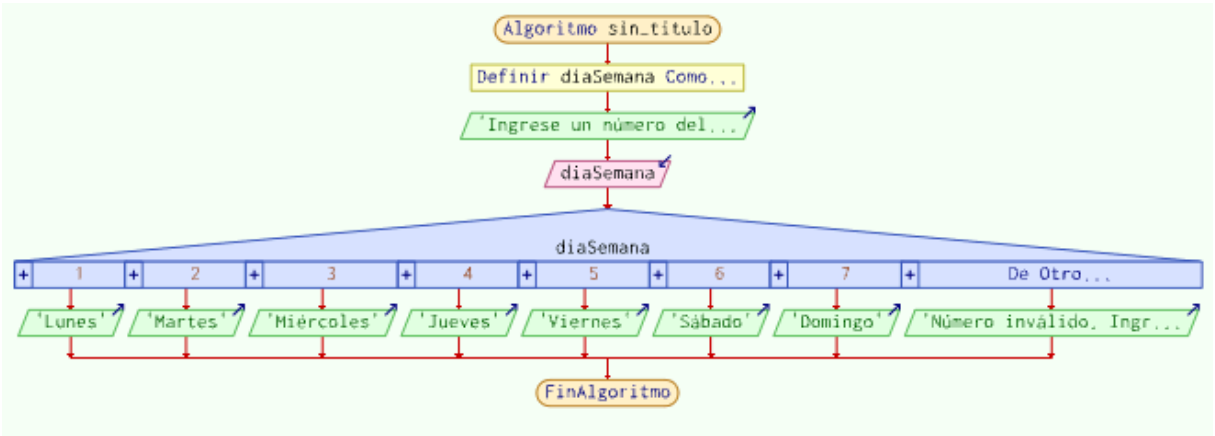
Sentencia Switch

El "switch" es una estructura de control en programación que se utiliza para evaluar una expresión y ejecutar diferentes acciones según el valor que tenga. En lugar de usar varias condiciones if y else, el switch permite organizar el código de forma más clara y eficiente cuando se tienen múltiples opciones posibles.

Cada una de estas opciones se conoce como un "caso", y el programa ejecuta el bloque de código correspondiente al valor que coincida con la expresión evaluada.

Ejemplo

Se le pide al usuario que ingrese un número del 1 al 7. Dependiendo del valor, se mostrará el día correspondiente de la semana. Si el número no está dentro de ese rango, se mostrará un mensaje de error. Aquí un ejemplo en Pseint:



En Python, no existe una sentencia switch como en algunos otros lenguajes de programación. Sin embargo, puedes lograr un comportamiento similar utilizando una serie de declaraciones if-elif-else.

```python
dia_semana = int(input("Ingrese un número del 1 al 7 que repres

if dia_semana == 1:
    print("Lunes")
elif dia_semana == 2:
    print("Martes")
elif dia_semana == 3:
    print("Miércoles")
elif dia_semana == 4:
    print("Jueves")
elif dia_semana == 5:
    print("Viernes")
elif dia_semana == 6:
    print("Sábado")
elif dia_semana == 7:
    print("Domingo")
else:
    print("Número inválido. Ingrese un número del 1 al 7.")
```

En este caso, hemos utilizado la estructura if, elif y else en Python para lograr el mismo comportamiento que la estructura switch.

Condiciones más complejas

En Python, las condiciones pueden volverse más complejas al combinar múltiples expresiones lógicas. Esto se logra utilizando operadores lógicos como "and", "or" y "not" para construir expresiones que evalúen múltiples condiciones simultáneamente. Además, podemos agrupar expresiones lógicas utilizando paréntesis para establecer el orden de evaluación.

- ➔ **Operador "and":** Este operador devuelve True si ambas condiciones son verdaderas.

```
x = 5
y = 10
if x > 0 and y > 0:
    print("Ambas variables son positivas")
```

- ➔ **Operador "or":** Este operador devuelve True si al menos una de las condiciones es verdadera.

```
edad = 25
if edad < 18 or edad >= 65:
    print("Tienes derecho a un descuento")
```

- ➔ **Operador "not":** Este operador invierte el valor de la condición.

```
x = 5
if not x == 0:
    print("x no es igual a cero")
```

Agrupación de expresiones lógicas

Podemos usar paréntesis para agrupar expresiones y controlar el orden de evaluación.

Ejemplo

Veamos estos dos casos de ejemplo. En el primero, se evalúa la variable edad para verificar si es mayor o igual a 18 y menor a 65. Además, mediante el operador lógico OR, se valida si la edad es mayor o igual a 65 y menor a 70 años.

```
edad = 35
if (edad >= 18 and edad < 65) or (edad >= 65 and edad < 70):
    print("Eres elegible para votar")
```

En el segundo ejemplo, se valida la variable número para comprobar si es mayor que cero y menor que cien. Asimismo, utilizando el operador lógico OR, se permite que el número sea menor que cero.

```
numero = 75
if (numero > 0 and numero < 100) or numero < 0:
    print("El número es positivo y menor que 100 o es negativo")
```

Las condiciones más complejas, como las de estos ejemplos, nos permiten hacer comprobaciones más detalladas. Aunque son útiles, hay que tener cuidado de que no se vuelvan difíciles de entender o de manejar.

Actividad de cierre

Vamos a realizar una actividad para entender cómo funciona un algoritmo que clasifica el IMC (Índice de Masa Corporal) usando condicionales, con la ayuda de ChatGPT.

PASO 1 >> Usamos la IA para generar un ejemplo

Abrí ChatGPT y escribí:

🗨️ *"Generá un ejemplo de estructura if-elif-else que clasifique el IMC en categorías."*

Copiá el código que te da y léelo con atención.

PASO 2 >> Analizamos el código

Explicá con tus palabras qué hace el código, línea por línea.

Luego, pensá: ¿cómo podrías resolver lo mismo usando otra forma de condicionales? (por ejemplo, varios if separados o estructuras diferentes).

👉 **Importante**: Esta parte hacela vos, sin usar ChatGPT. Queremos ver cómo lo entendiste.

PASO 3 >> Compartimos en el foro

Subí al foro de intercambio de la Clase Nº4 lo siguiente:

- ➔ El código que te dio Chat GPT.
- ➔ Tu explicación línea por línea.
- ➔ Tu propuesta alternativa (si se te ocurrió una).
- ➔ Un comentario final: ¿te ayudó usar la IA para entender mejor el tema?

Ejercicios de práctica con condicionales

A continuación, resolvé estos tres ejercicios usando tu lógica y todo lo aprendido en esta clase sobre condicionales:

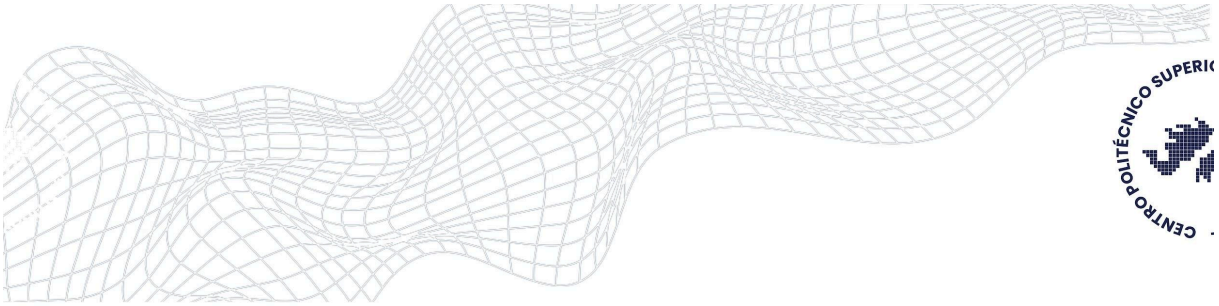
1. **Verificación de edad para ver una película**

   Pedile al usuario su edad y permitile ver la película solo si tiene 13 años o más.

2. **Calificación de un estudiante**

   Según la nota del examen (de 0 a 100), asigná una letra:

   - A: 90 o más



- B: entre 80 y 89
- C: entre 70 y 79
- D: entre 60 y 69
- F: menos de 60

3. **Calculadora de descuento**

   Pedí el precio de un producto y aplica un descuento:

   a. Si cuesta $12.999 o más → 30% de descuento
   b. Si cuesta menos → 20% de descuento

3. Cierre

En este módulo abordamos los fundamentos de las estructuras selectivas en Python. Exploramos cómo implementar alternativas simples, dobles y múltiples mediante condicionales if, if-else e if-elif-else, así como decisiones anidadas. Estas herramientas nos permiten desarrollar algoritmos más eficientes y adaptables, esenciales para resolver problemas complejos en distintos contextos de programación.

4. Bibliografía

- FUNDAMENTOS DE PROGRAMACIÓN. *Algoritmos, estructura de datos y objetos-Cuarta edición - Luis Joyanes Aguilar- McGrawHill - 2008 - ISBN 978-84-481-6111-8*
- *Python 3 – Los fundamentos del lenguaje - Es un libro completo, bien escrito, claro, aunque un poco extenso, por lo que no es apto para lectores apurados. La intención del autor es brindar absolutamente todo lo que se necesita para que el lector aprenda a programar en Python desde cero.*
- *Python para todos: explorando datos en Python 3 - Este libro es ideal para estudiantes y programadores que buscan especializarse en la exploración y el análisis de datos.*
- OpenAI. (2025). *ChatGPT* (versión GPT-4). Recuperado de https://chat.openai.com