Ciencia de **DATOS**

Estructuras de control de flujo

Estructuras de control de flujo

Hasta ahora gran parte del código de Python que has escrito es incondicional. Es decir, el código no hace ninguna elección y se ejecuta de manera secuencial en donde las sentencias siempre se ejecutan una tras otra, exactamente en el orden que has especificado.

Estructuras de control de flujo

Pero normalmente los programas serán más complicados que eso. Con frecuencia, un programa necesita omitir algunas instrucciones, ejecutar una serie de instrucciones de forma repetitiva o elegir entre conjuntos alternativos de instrucciones para ejecutar.

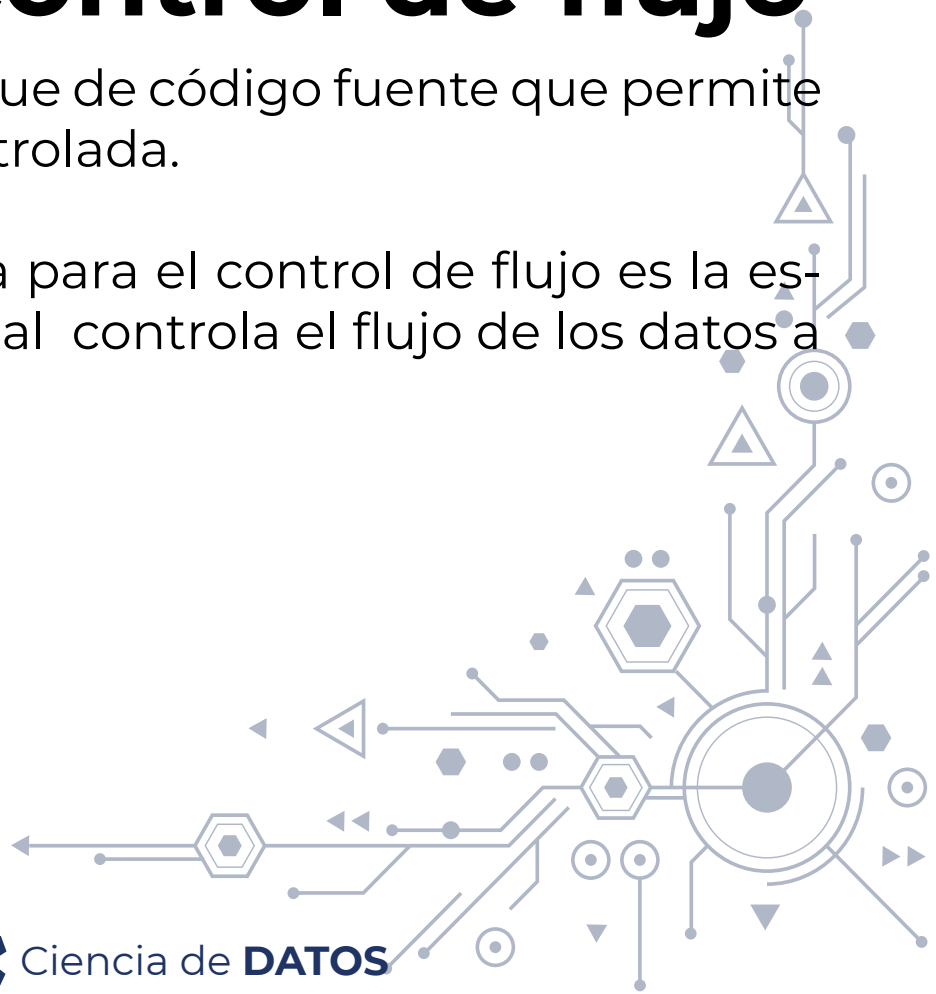
Estructuras de control de flujo

Ahí es donde entran las estructuras de control. Una estructura de control dirige el orden de ejecución de las declaraciones en un programa (conocido como el flujo de control del programa).

Estructuras de control de flujo

Una estructura de control es un bloque de código fuente que permite agrupar instrucciones de forma controlada.

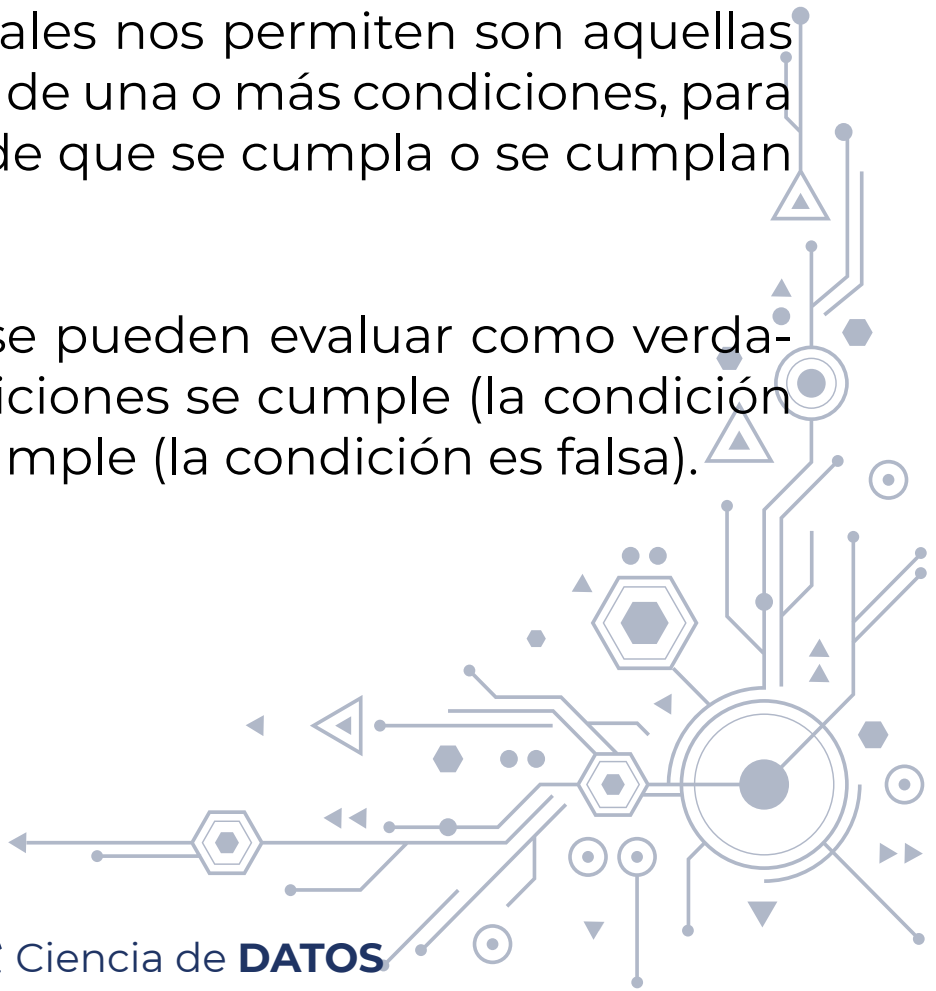
Una de las estructuras más utilizada para el control de flujo es la estructura de control condicional la cual controla el flujo de los datos a partir de condiciones.



Estructuras de control de flujo condicional

Las estructuras de control condicionales nos permiten son aquellas que permiten realizar una evaluación de una o más condiciones, para decidir qué acción ejecutar en caso de que se cumpla o se cumplan las condiciones.

La condición o las condiciones solo se pueden evaluar como verdaderas o falsas. Si la condición o condiciones se cumple (la condición es verdadera), o la condición no se cumple (la condición es falsa).



Estructuras de control de flujo condicional

En el mundo real, comúnmente debemos evaluar la información que nos rodea y luego elegir un curso de acción u otro en función de lo que observamos:

Si hace buen tiempo, cortaré el césped. (Está implícito que si el clima no es agradable, entonces no cortaré el césped).

Condicional if

En un programa de Python, la estructura de control condicional **if** es la que se utiliza para tomar este tipo de decisiones. Te permitirá la ejecución condicional de una sentencia o grupo de sentencias en función del valor de una expresión.

Condicional if

Veamos el tipo más básico de la estructura de control condicional if:

```
if <expresión>:
    <sentencias>
```

Como se muestra arriba:

<expresión> es una expresión evaluada de forma booleana (verdadero o falso).

<sentencias> es una o varias sentencias validas de Python, las cuales deben estar propiamente indentadas.

Condicional if

Si la <expresión> es verdadera, entonces las <sentencias> serán ejecutadas. Si la <expresión> es falsa entonces las <sentencias> serán omitidas y no serán ejecutadas.
Recuerda que los (:) que siguen a la <expresión> son obligatorios. Algunos lenguajes de Programación requieren que la <expresión> se coloque entre paréntesis, sin embargo Python no.

Ejemplo condicional if

Aquí verás algunos ejemplos de la estructura de control condicional if:
Suponiendo que:

```
X=0
y=4
if x < y:
    print('si')
```

En este caso la estructura de control condicional **if** evalúa si es el caso de que **x** fuera menor que **y**, como la condición es verdadera entonces se ejecutará la sentencia **print** mandando como resultado la cadena si.

Ejemplo condicional if

```
X=0
y=4
if x > y:
    print('si')
```

En este caso la estructura de control condicional **if** evalúa si es el caso de que **x** fuera mayor que **y**, como la condición es falsa entonces no se ejecutará la sentencia **print** y seguirá el flujo del programa.

Operadores de comparación

.Recuerda que para evaluar la condición puedes utilizar los operadores de comparación:

. ==	a == b	Igual a.	Verdadero si el valor de a es igual al valor de b, falso en cualquier otro caso
.!=	a != b	No igual a.	Verdadero si **a** no es igual a **b**, falso en cualquier otro caso
.<	a < b	Menor que.	Verdadero si **a** es menor a **b**, falso en cualquier otro caso
.<=	a <= b	Menor o igual a.	Verdadero si **a** es menor o igual a **b,** falso en cualquier otro caso
.>	a > b	Mayor que.	Verdadero si **a** es Mayor a **b**, falso en cualquier otro caso
.>=	a >= b	Mayor o igual a.	Verdadero si **a** es mayor o igual a **b,** falso en cualquier otro caso

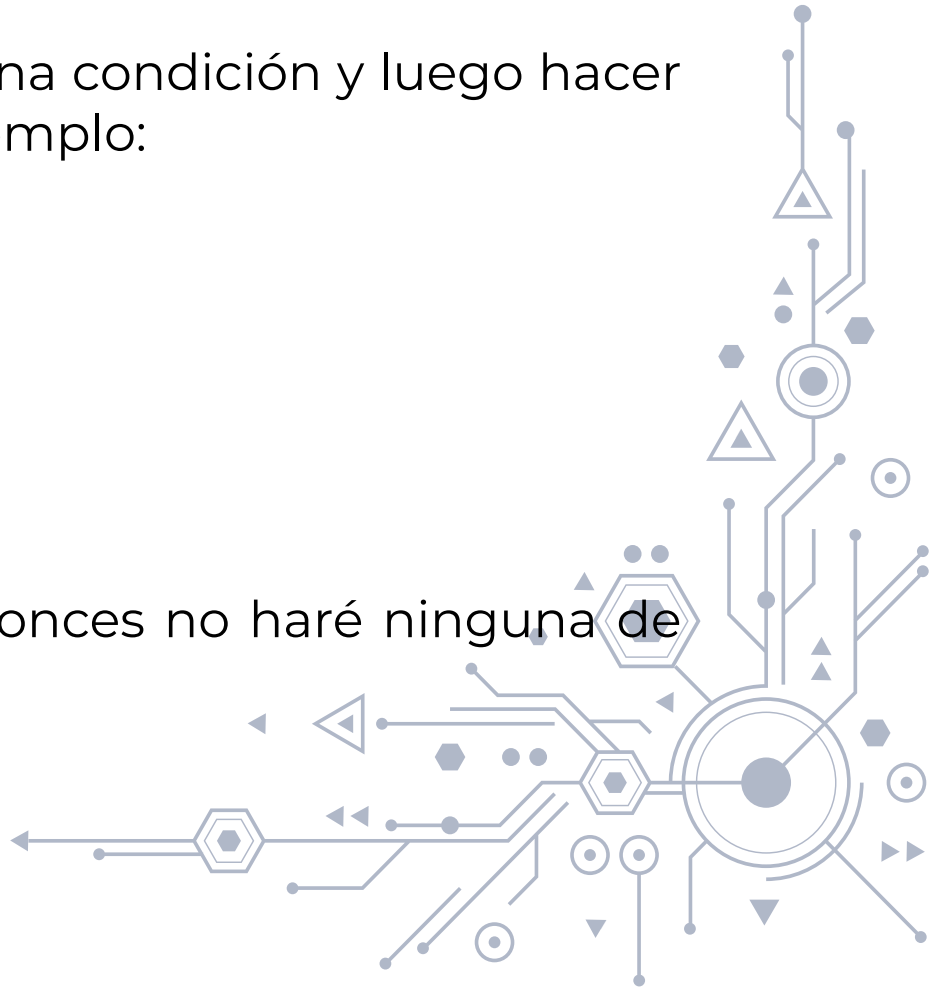
Ejecutando un Bloque de Sentencias

Supongamos que deseas evaluar una condición y luego hacer más de una cosa si es verdadera, ejemplo:

Si hace buen tiempo, entonces:

Cortar el césped
Desyerbar el jardín
Sacar al perro a pasear

(Si el clima no es agradable, entonces no haré ninguna de estas cosas).



Ejecutando un Bloque de Sentencias

En todos los ejemplos anteriores, cada if <expresión>: ha sido seguido por una sola <sentencia>. Tiene que haber alguna manera de decir "Si <expresión> es verdadero, haga todo lo siguiente".

El enfoque habitual adoptado por la mayoría de los lenguajes de programación es definir un dispositivo sintáctico que agrupa varias declaraciones en una declaración o bloque compuesto. Un bloque se considera sintácticamente como una sola entidad. Cuando es el destino de una sentencia **if** y <expresión> es verdadero, se ejecutan todas las sentencias del bloque. Si <expresión> es falso, entonces ninguna de las sentencias se ejecuta.

La importancia de la indentación

Recuerda que la indentación es muy importante en **Python**, la usaremos para definir los bloques de código que se ejecutarán si la condición de nuestra estructura de control condicional **if es verdadera.**

La importancia de la indentación

Si queremos ejecutar un bloque de código si la condición de nuestro if es verdadera debemos indentar todo como sigue:

```
1  if <expresión>:
2    <sentencia>
3    <sentencia>
4    ...
5    <sentencia>
6  <siguiente sentencia>
```

En este ejemplo todas las sentencias de la línea 2 a la 5 (en verde) se van a ejecutar si la condición de nuestro **if** es verdadera de lo contrario la siguiente sentencia (en azul). Nota que el bloque de código en verde tiene la misma indentación y es lo que se va a ejecutar si la condición de nuestro **if** es verdadera.
El bloque de código en algunas otras fuentes de información de Python también se conoce como **SUITE**

Las Clausulas ELSE y ELIF

Ahora sabes cómo usar una estructura de control **if** para ejecutar condicionalmente una sola declaración o un bloque de varias declaraciones.

En ocasiones es necesario evaluar una condición y tomar una ruta si es verdadera, pero especificar una ruta alternativa si no lo es. Esto se logra con una cláusula **else.**

Las Clausulas ELSE y ELIF

```
if <expresión>:
    <sentencia(s)>
else:
    <sentencia(s)>
```

Si la <expresión> es verdadera, se ejecuta el primer bloque o suite y se omite el segundo. Si la <expresión> es falsa, se salta el primer bloque de código y se ejecuta el segundo. De cualquier forma, la ejecución se reanuda después de la segunda suite. Ambos bloques o suites están definidos por indentación o sangría.

Ejemplos

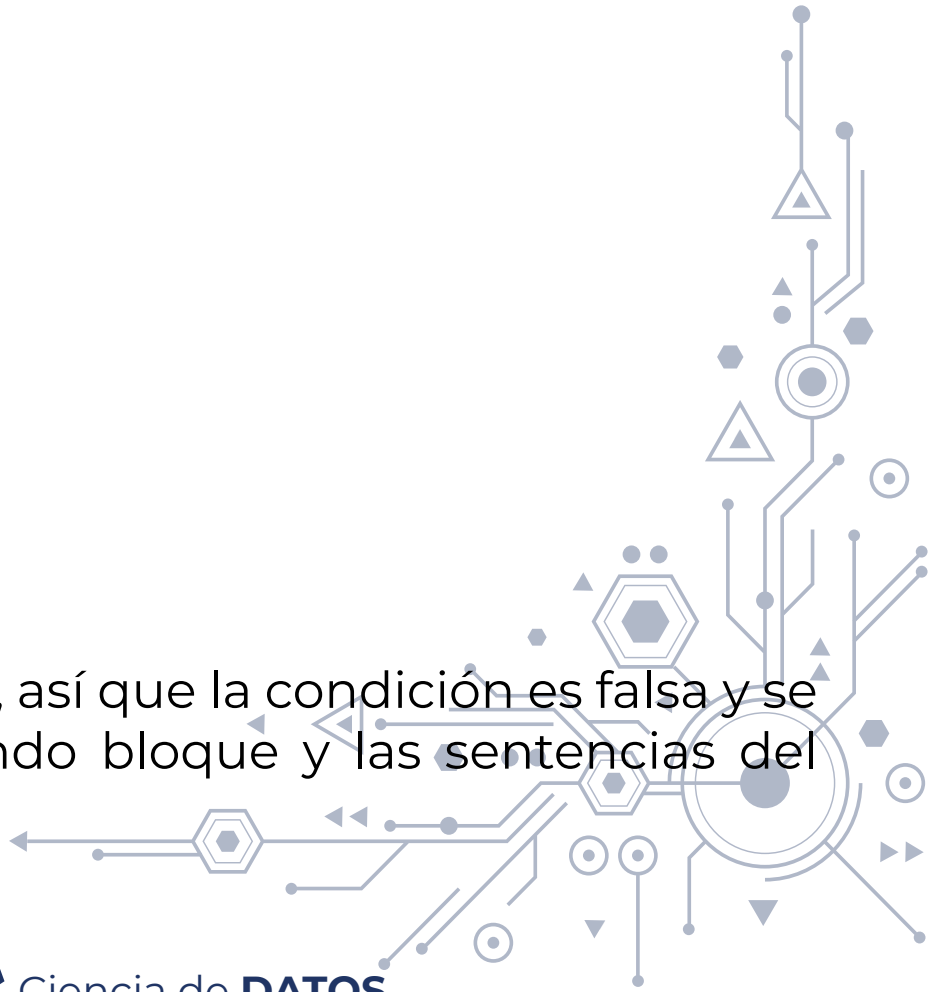
```
x = 20
if x < 50:
    print('(primer bloque)')
    print('x es menor')
 else:
    print('(segundo bloque)')
    print('x es mayor')
```

Resultado:
(primer bloque)
x es menor
En este ejemplo x es menor que 50, así que la condición es verdadera y se ejecutan las sentencias del primer bloque y las sentencias del segundo bloque son ignoradas.

Ejemplos

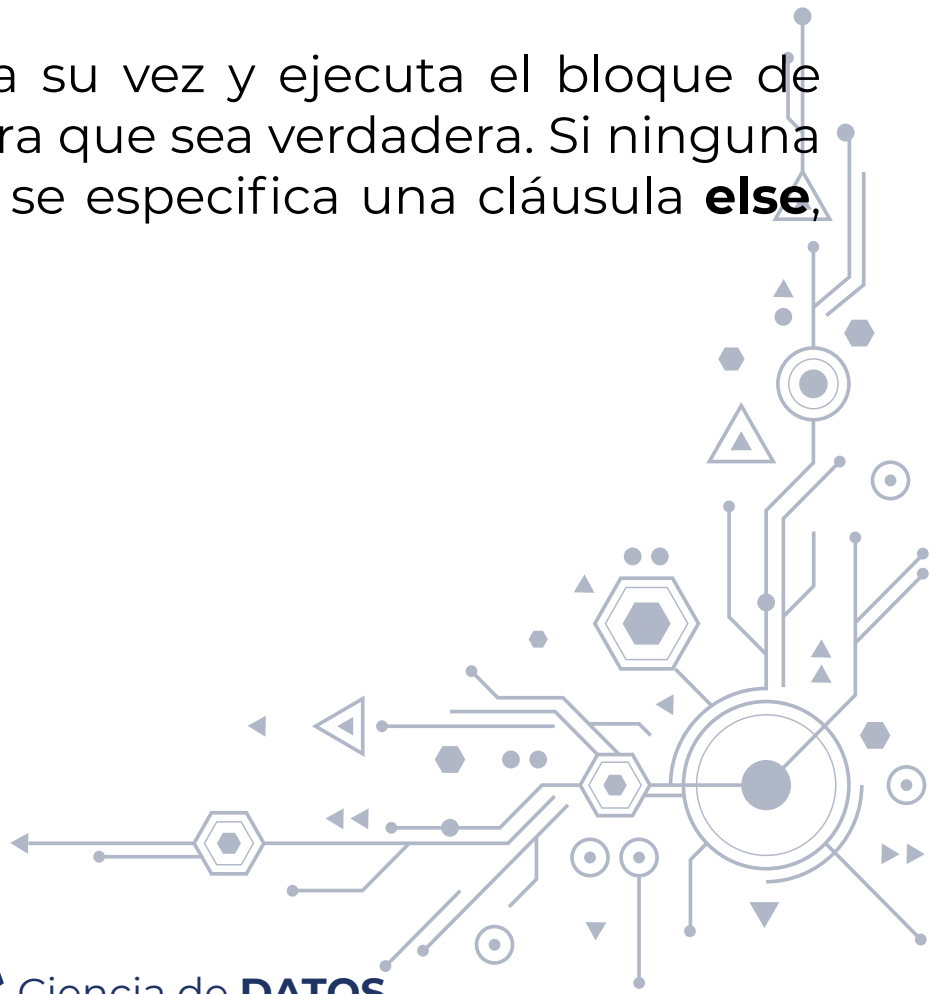
```
x = 120
if x < 50:
    print('(primer bloque)')
    print('x es menor')
 else:
    print('(segundo bloque)')
    print('x es mayor')
```

Resultado:
(primer bloque)
x es menor
En este ejemplo x es mayor que 50, así que la condición es falsa y se ejecutan las sentencias del segundo bloque y las sentencias del primer bloque son ignoradas.



elif

También existe una sintaxis para la ejecución de condiciones cuando tenemos varias alternativas. Para esto, use una o más cláusulas **elif** (abreviatura de else if).
Python evalúa cada <expresión> a su vez y ejecuta el bloque de código correspondiente a la primera que sea verdadera. Si ninguna de las expresiones es verdadera y se especifica una cláusula **else**, entonces se ejecuta su bloque.

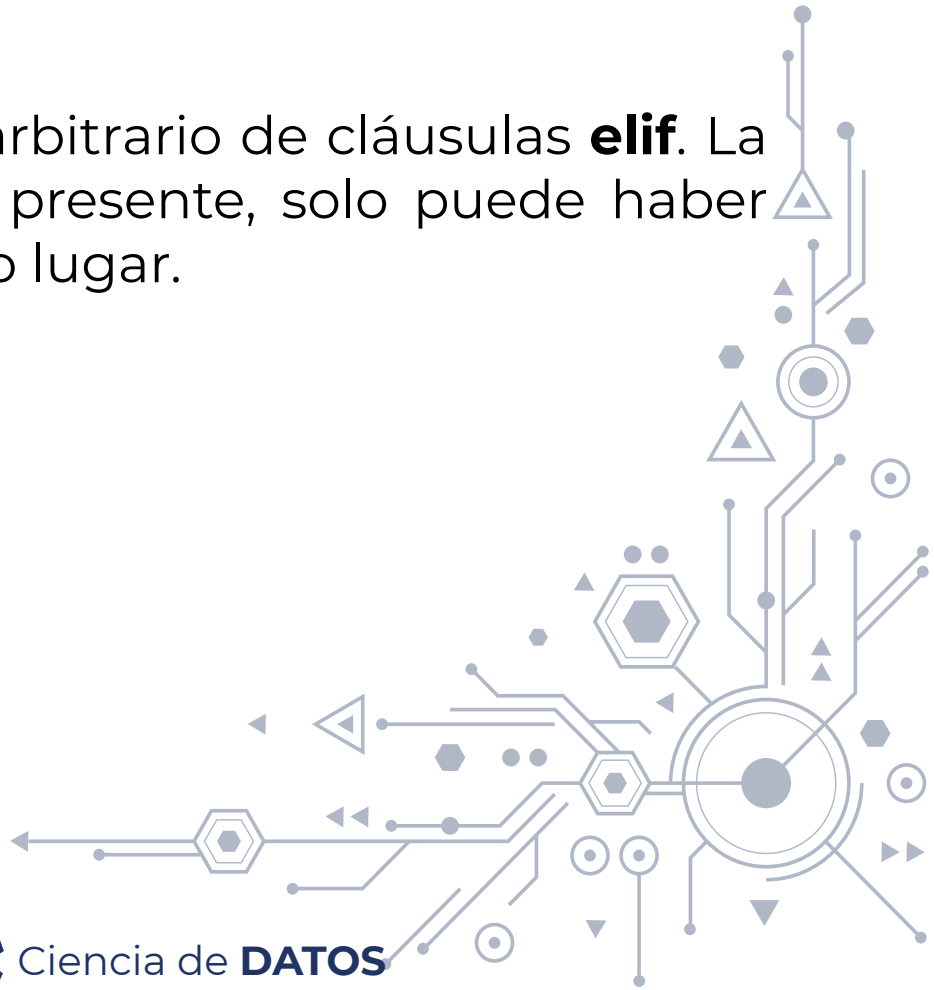


Sintaxis elif

```
if <expresión>:
    <sentencia(s)>
elif <expr>:
    <sentencia(s)>
elif <expr>:
    <sentencia(s)>
    ...
else:
    <sentencia(s)>
```

Sintaxis elif

Se puede especificar un número arbitrario de cláusulas **elif**. La cláusula **else** es opcional. Si está presente, solo puede haber uno, y debe especificarse en último lugar.

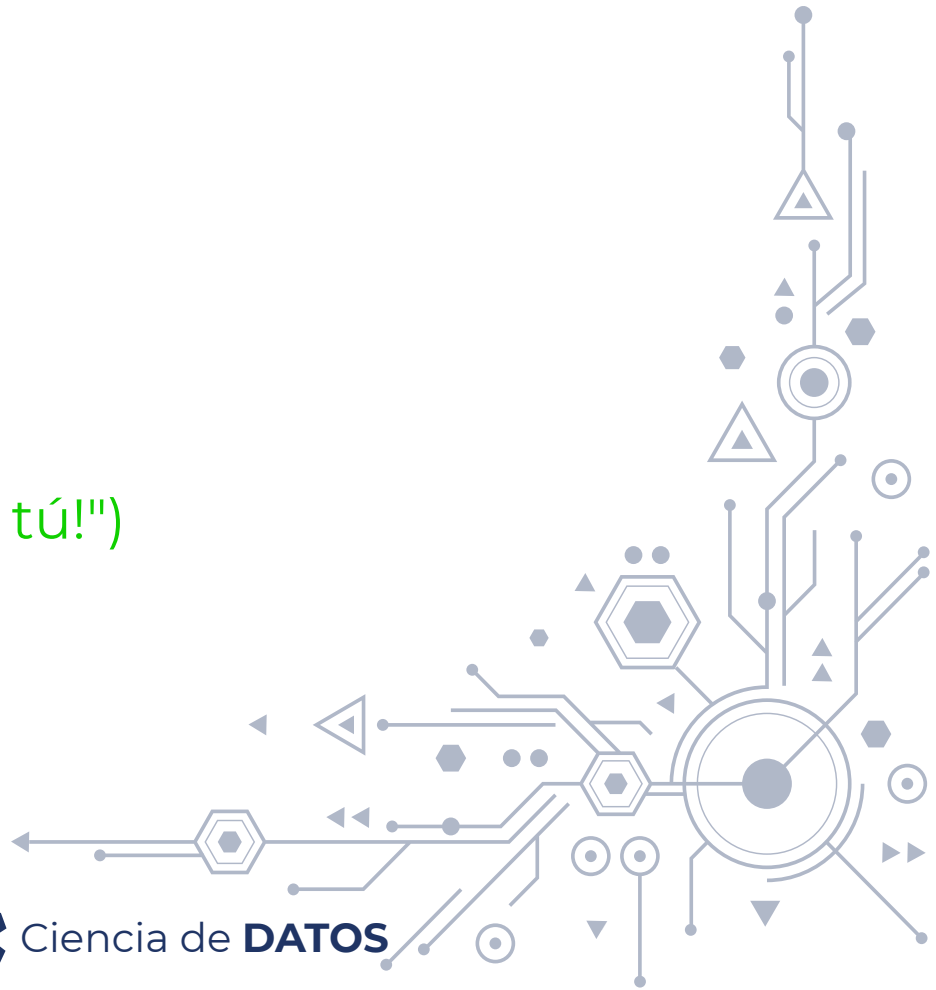


Ejemplo clausula elif

```
name = 'Juan'
if name == 'Pedro':
    print('Hola Pedro')
elif name == 'Luis':
    print('Hola Luis')
elif name == 'Juan':
    print('Hola Juan')
elif name == 'Antonio':
    print('Hola Antonio')
else:
    print("¡No se quien eres tú!")
```

Resultado:

Hola Juan



En conclusión

Se presentó el concepto de estructuras de control. Estas son sentencias compuestas que alteran el flujo de control del programa: el orden de ejecución de las sentencias del programa.

Se aprendió a agrupar declaraciones individuales en un bloque o conjunto.

Encontraste tu primera estructura de control, la declaración **if**, que hace posible ejecutar condicionalmente una sentencia o bloque de sentencias basado en la evaluación de los datos del programa.