

Pruebas Lógicas

Pruebas Lógicas	1
¿Qué aprenderás?	2
Introducción	2
Operaciones o pruebas lógicas	3
Diferencia entre = y ==	8



¡Comencemos!

¿Qué aprenderás?

- Reconocer los operadores matemáticos, lógicos y de comparación para la construcción de expresiones.

Introducción

A lo largo de este capítulo, aprenderemos acerca de las pruebas lógicas, las cuales permitirán indicar al computador si realizar o no una acción dependiendo del resultados de estas pruebas.

Las pruebas lógicas son la base de las estructuras condicionales en Python, una vez que entendemos cómo resolver pruebas lógicas, podremos implementar programas más complejos.

Operaciones o pruebas lógicas

Anteriormente definimos el tipo de dato booleano, los que pueden tomar solo dos valores: **Verdadero o Falso**.

Pero, ¿por qué es necesario disponer de un tipo de dato que parece ser demasiado sencillo de entender? Luego de haber aprendido qué es un diagrama de flujo y qué es el pseudocódigo, la manera más sencilla de entender el funcionamiento de este tipo de dato es que será la respuesta a una pregunta (representadas con el rombo en los diagramas de flujo), técnicamente, una operación lógica.

Toda pregunta lógica tiene dos posibles respuestas:

True (cuando la respuesta es afirmativa) y;

False (en caso de ser negativa).

Las operaciones lógicas o también llamadas pruebas lógicas, serán la manera en la que podemos representar en forma de código una pregunta. En general, esta se presenta en forma de comparación entre 2 valores por medio de un **operador matemático de comparación**.

Los operadores más conocidos son los siguientes:

Operador	Nombre	Ejemplo	Resultado
==	Igual a	2 == 2	True
!=	Distinto a	2 != 2	False
>	Mayor a	3 > 4	False
>=	Mayor o igual a	3 >= 3	True
<	Menor a	4 < 3	False
<=	Menor o igual a	3 <= 4	True

Tabla 1. Operadores de Comparación Numéricos
Fuente: Desafío Latam.

En la Tabla 1 se pueden apreciar distintos Operadores de comparación que permitirán tomar decisiones cuando se comparan valores numéricos.

Para analizar el funcionamiento de estos operadores utilizaremos el siguiente caso:

- Juan tiene 27 años
- No tiene Hijos
- Salió al colegio a los 17 años
- Estudio por 6 años en la Universidad
- Lleva 3 años pololeando o de noviazgo
- Tiene 4 años de Experiencia Laboral

¿Qué preguntas podemos hacer? Definamos algunas variables:

```
nombre = "Juan"
edad = 27 # años
n_hijos = 0
graduacion_colegio = 17 # años duracion_uni = 6 # años
duracion_pololeo = 3 # años
exp_laboral = 4 # años
```

- ¿Juan es mayor de edad?

```
print(edad >= 18)
```

True

Dado que Juan es mayor a 18, podemos deducir que es mayor de edad.

- ¿Juan se graduó del colegio antes de los 18 años?

```
print(graduacion_colegio < 18)
```

True

Debido a que la edad de graduación es menor que 18, interpretamos que la graduación ocurrió antes de dicha edad.

- ¿Juan no tiene 4 años de experiencia laboral?

```
print(exp_laboral != 4)
```

False

En este caso, sabemos que Juan tiene 4 años de experiencia laboral, por lo tanto, esto resultará en False.

- ¿Juan tiene hijos?

```
print(n_hijos > 0)
```

False

Este corresponde a un caso un poco distinto, ya que la manera en la que se definió n_hijos es por la cantidad de hijos, por lo tanto, si tiene un número mayor a 0 quiere decir que efectivamente tiene hijos.

- ¿Juan no tiene hijos?

```
print(n_hijos == 0)
```

True

Al modificar levemente la pregunta, podemos utilizar un operador completamente distinto. Si es que el número de hijos es cero, entonces no tiene hijos.



NOTA: Para preguntar si “algo es igual”, se utiliza doble signo igual (`==`). Ten en cuenta que utilizar sólo un signo igual implica una asignación, es decir, definir una variable, lo cual no devuelve un valor True o False.

Existen otro tipo de preguntas que no se responden de manera tan directa y hay que recurrir al ingenio para hacerlas entender en nuestro código. A estas las llamaremos **Preguntas Indirectas**:

- ¿Al graduarse de la Universidad ya había cumplido 22?

No tenemos una variable directa que pueda responder esto, pero podemos armarla:

```
edad_grad_uni = graduacion_colegio + duracion_uni
```

```
print(edad_grad_uni >= 22)
```

True

En este caso, la interpretación es: al sumar la edad de graduación más la duración de la universidad obtengo un estimado de mi edad de graduación. Si esta edad, es al menos 22, quiere decir que al graduarse ya tenía 22.

- ¿Juan comenzó a pololear a los 25?

Acá podemos utilizar un procedimiento similar al anterior, ya que se define un estimado de la edad de inicio del pololeo:

```
edad_inicio_pololeo = edad - duracion_pololeo
```

```
print(edad_inicio_pololeo == 25)
```

False

En este caso la respuesta es no, ya que comenzó a pololear a los 24.

- ¿Juan lleva más tiempo trabajando o estudiando?

Si bien esta es una pregunta que no requiere la creación de variables extras, dependiendo del planteamiento es la interpretación. Básicamente estamos transformando la pregunta:

- ¿Juan tiene más tiempo trabajando?

```
print(exp_laboral > duracion_pololeo)
```

True

En este caso la respuesta es directa, sí, lleva más tiempo trabajando.

- ¿Juan tiene más tiempo pololeando?

```
print(exp_laboral < duracion_pololeo)
```

False

En este caso la respuesta es no, por lo tanto, se deduce que lleva más tiempo trabajando.

- ¿Juan ya estaba pololeando cuando comenzó a trabajar?

En este caso, podemos utilizar la variable `edad_inicio_pololeo` que ya habíamos creado y complementar con un inicio de trabajo:

```
edad_inicio_trabajo = edad - exp_laboral
```

Luego, replantear la pregunta:

- ¿Juan comenzó a pololear antes de trabajar?

```
print(edad_inicio_pololeo < edad_inicio_trabajo)
```

En este caso, la respuesta es directa. Debido a que NO comenzó a pololear antes de trabajar.

- ¿Juan comenzó a trabajar antes de pololear?

```
print(edad_inicio_pololeo > edad_inicio_trabajo)
```

True

En este caso la respuesta es indirecta. Como comenzó a trabajar antes de pololear, quiere decir que no estaba pololeando antes.

Algunas de estas pruebas también tienen sentido con **Strings**, por ejemplo:

- ¿El sujeto se llama Juan?

```
print(nombre == "Juan")
```

True

- ¿El sujeto no se llama José?

```
print(nombre != "José")
```

True

Diferencia entre = y ==

Es importante diferenciar el operador asignación (=) del operador de igualdad (==). Ambos pueden combinarse dentro del código y se debe comprender la función de cada uno. Para esto, seguiremos trabajando con el caso de Juan.

```
me_llamo_juan = nombre == "Juan"
```

True

En este caso se realiza la prueba lógica **¿el nombre es Juan?** (fíjate que es una pregunta, que se responde con True o False) y la respuesta se almacena en la variable `me_llamo_juan`, que en este caso es cierta.

```
print(type(me_llamo_juan))
```

bool

Y si revisamos el tipo de dato notamos que es Booleano.

En conclusión, **cada prueba lógica es un tipo de dato Boolean, el cual se puede asignar a una variable** (o incluso a una estructura de datos).

En variados casos, la respuesta a una pregunta se puede responder con varias operaciones lógicas combinadas. Para esto existen los **operadores lógicos, los cuales están basados en la lógica proposicional**.

Estos permiten combinar varias pruebas lógicas y dependiendo de los resultados de ellas, entregarán una respuesta final.

Los operadores lógicos más comunes son **AND, OR** y **^** ("o exclusivo", también llamado XOR).

Si suponemos que A y B son pruebas lógicas, la combinación de ellas se resuelven según la siguiente tabla:

Resultado Prueba A	Resultado Prueba B	A and B	A or B	A ^ B
True	True	True	True	False
True	False	False	True	True
False	True	False	True	True
False	False	False	False	False

Tabla 2: Resultado de Operaciones Lógicas
Fuente: Desafío Latam

En resumen, **AND** es Falso a menos que ambos sean verdaderos. **OR** es verdadero a menos que ambos sean Falsos y **^** es verdadero si A y B tienen distintos resultados.

- ¿Juan es mayor de edad y está pololeando?

```
print(edad > 18 and duracion_pololeo > 0)
```

```
True
```

Dado que Juan es mayor que 18 y además está pololeando porque lleva más de 3 años, entonces la condición final es True.

Supongamos que Juan ahora quiere cambiarse de trabajo, y sólo puede postular si tiene una carrera de al menos 6 años o si tiene más de 5 años de experiencia laboral:

- ¿Puede Juan postular?

```
print(duracion_uni >= 6 or exp_laboral > 5)
```

True

Juan cumple, con solo una de las dos condiciones, debido a que la condición utiliza un **OR**, cumplir una de las dos condiciones es suficiente.

Supongamos que Juan ahora quiere postular a un beneficio, solo puede postular si es que cumple un requisito: ser menor a 28 o tener menos de 3 años de experiencia laboral.

```
menor_28 = edad < 28  
exp_menor_3 = exp_laboral < 3  
  
print(menor_28 ^ exp_menor_3)
```

True

Este caso también resulta en verdadero, dado que solo se cumple una de las dos condiciones. Si por alguna razón ambas condiciones se cumplieran esto resultaría en falso.

En este caso se crearon variables para representar cada condición, esto es solo para demostrar que es posible almacenar condiciones, como se demostró anteriormente.