«Talento Tech»

Inteligencia Artificial

Clase 02

Buenos Aires Ciudad

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Clase N° 2 | Conceptos básicos

Temario:

- Introducción a la sintaxis básica de Python.
- Variables, tipos de datos y operaciones básicas.
- Condicionales: if, elif y else.
- Función Input

Buenos Aires aprende
Agencia de Habilidades para el Futuro

COMENTARIOS

Como cualquier otro lenguaje de programación, **Python** permite escribir comentarios en el código. Los comentarios son útiles para explicar por qué estamos programando algo de un modo concreto o añadir indicaciones. Te aseguro que son de utilidad cuando se retoma un programa o aplicación en el futuro

Los comentarios son ignorados por el intérprete de Python. Solo tienen sentido para los programadores.

Para añadir un comentario a tu código simplemente comienza una línea con el carácter #:

Comentarios en varias líneas

Para escribir comentarios que ocupan varias líneas, simplemente escribe cada una de las líneas anteponiendo el carácter #:

```
# Este comentario ocupa
# 2 líneas
```

También se puede escribir un comentario en varias líneas si lo encierras entre tres comillas simples ''' o dobles """

```
'''Este comentario
también ocupa 2 líneas'''
```

Sin embargo, personalmente no me gusta este modo de definir un comentario en varias líneas porque también es la forma de definir un string en varias líneas.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Trabajando con datos.

Tipos de datos

Cuando trabajamos en programación, inevitablemente trabajamos con datos, es decir, con información que entra y sale de nuestro programa. Estos datos se dividen según lo que queramos ingresar, ya que definirán qué valores y operaciones podemos hacer durante el programa.

Los principales tipos de datos en Python son:

1. Tipo cadena (str)
2. Tipos numéricos
    a. Tipo entero (int)
    b. Tipo real o punto flotante (float)

---

3. Tipo lista (list)
4. Tipo tuplas (tuple)
5. Tipo de datos mapas o diccionario (dict)

NOMBRE	TIPO	EJEMPLO	DEFINICIÓN	TIPO DE DATO
ENTERO	**int**	12 8765 -1232	Los números enteros son aquellos que no contienen decimales, pueden ser positivos o negativos además del cero.	**INMUTABLE**
BOOLEANO	**bool**	True, False	Es un tipo de dato que permite almacenar dos valores True o False	**INMUTABLE**
FLOTANTE	**float**	3.14 1.5e3	Este tipo de dato se representa en lenguaje de programación como float (flotante). Puede, al igual que el entero, ser positivo o negativo, conteniendo uno o más decimales.	**INMUTABLE**
CADENA	**str**	Hola mundo' , '''1234 probando, uno dos tres''	Permite almacenar de caracteres. Para crear una, es necesario incluir el texto entre comillas dobles " " .Las cadenas no están limitadas en tamaño, por lo que el único límite es la memoria de tu ordenado	**INMUTABLE**
DICCIONARIO	**dict**	{'Chrome': 'v79' , 'Firefox': 'v71'}	Un diccionario en Python es una colección de elementos, donde cada uno tiene una llave key y un valor value. Los diccionarios se pueden crear con llaves {} separando con una coma cada par key: value	**MUTABLE**
TUPLA	**tuple**	(1, 3, 5) ("c", "palta", "Tomate")	En Python, una tupla es un conjunto ordenado e inmutable de elementos del mismo o diferente tipo. Las tuplas se representanescribiendo los elementos entre paréntesis y separados por comas	**INMUTABLE**
LISTA	**list**	['Chrome', 'Firefox'] [1, 3, 4, 5, 6, 2]	Las listas en Python son un tipo contenedor compuesto, que se usan para almacenar conjuntos de elementos relacionados del mismo tipo o de distinto.	**MUTABLE**

Buenos Aires aprende
Agencia de Habilidades para el Futuro

En esta clase nos enfocaremos en los datos de tipo cadena o str, y en los tipos de datos numéricos como los enteros o int, y los flotantes o float.

¡Miremos este video para aclarar algunas dudas!
<iframe width="854" height="480"
src="https://www.youtube.com/embed/QwN3Cv1auOA?si=hrywUHryJkI2icPv"
title="YouTube video player" frameborder="0" allow="accelerometer; autoplay;
clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share"
referrerpolicy="strict-origin-when-cross-origin" allowfullscreen></iframe>

Tipo de dato string, int, float

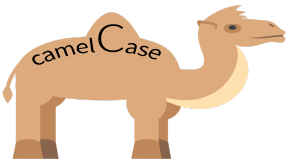
Variables.

Las **variables** son pequeños espacios en la memoria que nos permite guardar datos (de cualquier tipo) para luego utilizarlas en el momento que necesitemos. Esto nos permite avanzar en el programa sin tener que declarar repetidas veces un mismo dato. Para guardarlas es necesario colocarles un nombre, lo que funcionará como una dirección, y cada vez que las necesitemos usar, solo tendremos que llamar a ese dato por el nombre.
Existe un tipo de variable “constante”, la cual se utiliza para definir valores fijos, que no requieran ser modificados. Esto nos evitará problemas futuros, ya que nos aseguramos que su valor será siempre igual. Por ejemplo, sabemos que la variable PI será siempre 3,14.

Nombrando las variables

Para que los lenguajes de programación entiendan que estamos declarando o llamando a una variable, hay varios aspectos que debemos tener en cuenta:

Estos nombres pueden estar formados por letras (minúsculas y mayúsculas) de la “a” a la “z”. También pueden incluir el guión bajo, y se pueden usar números, salvo como primer carácter. Se recomienda utilizar nombres descriptivos y en minúsculas. Para nombres compuestos, separar las palabras por guiones bajos o en el formato **camelCase**. Antes y después del signo = , debe haber un espacio en blanco.



Mayúsculas y minúsculas son diferentes para Python. Por ejemplo, todas las siguientes variables son diferentes:

- nombre = 'David'
- Nombre = 'Diego'
- NOMBRE = 'Rosita'`

Te invito a que hagas la prueba con estas variables y crees otras para practicarlas.

Algunos ejemplos que pueden servirnos al momento de pensar nombres de variables y su descripción de lo que queremos guardar. En muchos casos será muy importante que podamos usar la imaginación para generar las variables correctas, mezclando mayúsculas y minúsculas.

Palabras reservadas de Python

Python tiene una serie de palabras clave que están reservadas para su sintaxis y funcionamiento, por tanto, no pueden usarse como nombres de variables o funciones. Estas palabras clave se utilizan para definir la sintaxis y estructura del lenguaje Python. La lista de palabras reservadas es la siguiente:

and, as, assert, break, class, continue, def, del, elif, else, except, False, finally, for, from, global, if, import, in, is, lambda, None, nonlocal, not, or, pass, raise, return, True, try, yield, while y with

No hace falta que sepas de memoria, con la práctica van a ir apareciendo algunas de estas palabras y vamos a ir entendiendo su funcionamiento.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Función print()

En los programas, para que python nos muestre texto o variables hay que utilizar la función **print().**

La función **print()** permite mostrar texto en pantalla. Las cadenas se pueden delimitar tanto por comillas dobles **(")** como por comillas simples **(')**. Puede admitir varios argumentos seguidos de cualquier tipo.

Concatenar

Podemos crear un mensaje concatenando variables.
Hay varias formas de hacer esto en Python:
Utilizando el signo +:

```
nombre = "Lucas"
print("Mi nombre es " + nombre)
```

```
Mi nombre es Lucas
```

Esto será útil solamente cuando tengamos que concatenar cadenas de caracteres.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

¿Qué pasa si queremos concatenar cadenas con otro tipo de datos?

```
nombre = "Lucas"
edad = 17
print("Mi nombre es " + nombre + " y tengo " + edad)
---------------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call
last)
<ipython-input-4-4f18add92282> in <cell line: 3>()
      1 nombre = "Lucas"
      2 edad = 17
----> 3 print("Mi nombre es" + nombre + "y tengo" + edad)

TypeError: can only concatenate str (not "int") to str
```

Nos devuelve un error.

Para lograr esto podemos usar el modo f-string. Agregamos una f antes de iniciar la cadena y las variables las colocamos dentro de las comillas y entre llaves:

```
nombre = "Lucas"
edad = 17
print(f"Mi nombre es {nombre} y tengo {edad}")
```

```
Mi nombre es Lucas y tengo 17
```

Esto nos permitirá concatenar distintos tipos de datos. ¿Qué otra forma se te ocurre?

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Función type()

Python al ser un lenguaje dinámico, lo que significa que las variables pueden cambiar de tipo durante la ejecución del programa, necesitamos conocer muchas veces si en nuestro código va a salir lo que espero. type es muy útil para identificar el tipo de dato que estamos trabajando, ya que puede ayudar a solucionar errores y asegurar que las operaciones se realicen en tipos compatibles. type toma un elemento entre paréntesis y devuelve el tipo del elemento. Por ejemplo, puede devolver <class 'int'> para enteros, <class 'float'> para flotantes, <class 'str'> para cadenas, etc

```
nombre = "Lucas"
type(nombre) # salida => <class 'str'>
type(edad) # salida => <class 'int'>
```

.

Función input()

La función input() en Python se utiliza para recibir datos del usuario durante la ejecución de un programa. Es especialmente útil para hacer que los programas sean interactivos, permitiendo que el usuario ingrese información que el programa puede procesar y utilizar. Aunque se le puede proporcionar un mensaje al usuario para indicar qué tipo de dato se espera, la función siempre devuelve los datos ingresados como una cadena (string), independientemente del tipo de datos que el usuario ingrese.

```
nombre = input("¿Cuál es tu nombre? ")
print(f"Hola, {nombre}!")
```

Operadores

Operadores matemáticos

Los operadores matemáticos o aritméticos se usan para calcular un valor a partir de dos o más números, incluso cambiar el signo de un número de positivo a negativo o viceversa.

Nos permiten realizar tareas como sumar, restar, multiplicar, dividir y realizar otras operaciones matemáticas básicas.

OPERADOR	DESCRIPCIÓN	USO
+	Realiza la suma de dos o más elementos numéricos	10 + 2 = 12
-	Realiza la resta entre los operandos numéricos	9 - 3 = 6
/	Realiza División entre dos números	15 / 3 = 5
%	Devuelve el resto de una división	14 % 3 = 1 16 % 2 = 0
*	Realiza la multiplicación entre dos elementos	10 * 3 = 30
**	Realiza la potencia de los elementos	2 ** 3 = 8
//	Realiza la división dando como resultado un número entero	21 // 5 = 4

Operadores de comparación

Los operadores de comparación se utilizan para comparar dos valores y evaluar si una cierta relación es verdadera o falsa. Estos operadores se aplican comúnmente en estructuras condicionales y expresiones lógicas. En Python, los operadores de comparación incluyen:

Operadores de comparación

OPERADOR	PROPÓSITO	EJEMPLO
<	Devuelve True si el primer valor es menor que el segundo valor	Valor1 < Valor2
<=	Devuelve True si el primer valor es menor o igual que el segundo valor.	Valor1 <= Valor2
>	Devuelve True si el primer valor es mayor que el segundo valor.	Valor1 > Valor2
>=	Devuelve True si el primer valor es mayor o igual que el segundo valor	Valor1 >= Valor2
==	Devuelve True si el primer valor es igual al segundo valor.	Valor1 = Valor2

Python, al igual que las condiciones lógicas en matemáticas, utiliza estos operadores para evaluar expresiones y devolver un valor booleano, es decir, True (verdadero) o False (falso).

Cuando se utiliza un operador de comparación, la expresión se evalúa y produce un resultado lógico en función de la relación entre los valores involucrados.

Operadores lógicos

Los operadores lógicos son herramientas que nos permiten evaluar y combinar más de una expresión lógica para verificar la validez de ciertas condiciones. Al trabajar con varias expresiones juntas, los operadores lógicos nos permiten construir condiciones más complejas y tomar decisiones basadas en múltiples criterios.
operadores lógicos

OPERADOR	PROPÓSITO	EJEMPLO
And	Devuelve True cuando Expresión1 y Expresión2 son verdaderas.	Expresión1 Y Expresión2
Or	Devuelve True cuando Expresión1 o Expresión2 es verdadera.	Expresión1 O Expresión2
Not	Devuelve True cuando la Expresión no es verdadera.	No Expresión

Por ejemplo:

“Una persona puede entrar a un bar solo si es mayor de edad y tiene el DNI.”

En ese caso tenemos dos datos que evaluar y algunas decisiones que tomar. Si utilizamos “Y” entre nuestras dos condiciones, sabremos que se tienen que cumplir si o si ambas cuestiones, pero si utilizamos O, solo debemos asegurarnos que se cumpla una u otra, sin importarnos cuál de las dos sea la correcta.

Veamos este ejemplo en una tabla un poco más clara:

OPERADORES	CONDICIONES	RESULTADO
Y	Mayor de 18 Y tiene DNI	TRUE
	Mayor de 18 Y no tiene DNI	FALSE
	Menor de 18 Y tiene DNI	FALSE
	Menor de 18 Y no tiene DNI	FALSE
O	Mayor de 18 O tiene DNI	TRUE
	Mayor de 18 O no tiene DNI	TRUE
	Menor de 18 O tiene DNI	TRUE
	Menor de 18 O no tiene DNI	FALSE

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Operadores compuestos

En Python, también se emplean operadores compuestos que combinan una operación aritmética con una asignación. Estos operadores nos proporcionan una forma más concisa y eficiente de realizar una operación sobre una variable x y asignar el resultado a la misma variable. En lugar de realizar la operación por separado y luego asignar el resultado, con los operadores compuestos realizamos ambas acciones en una única expresión

Operadores compuestos

OPERADOR	PROPÓSITO	EJEMPLO
+=	Suma y asignación.	Variable += Valor
-=	Resta y asignación.	Variable -= Valor
*=	Multiplicación y asignación	Variable *= Valor
/=	División y asignación.	Variable /= Valor
%=	Módulo y asignación.	Variable %= Valor
**=	Exponenciación y asignación.	Variable **= Valor

Condicionales

Los condicionales nos permiten tomar decisiones en función de eventos, guiando el flujo del programa y permitiendo ejecutar bloques de código específicos si se cumplen ciertas condiciones y, opcionalmente, otros bloques si no se cumplen. Estas estructuras de control son esenciales para crear lógica y adaptabilidad en los programas.

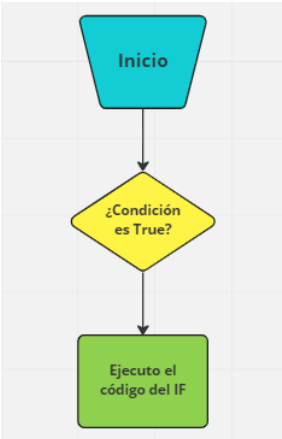
Estructura if

La estructura condicional **"if"** nos permite ejecutar un bloque de código específico si una determinada condición se cumple, dirigiendo la ejecución del programa según ciertas circunstancias.

Veamos algunos ejemplos:

Si es de noche ➡️ puede encender una luz
Si tengo hambre ➡️ puedo preparar un almuerzo
Si llueve ➡️ uso un paraguas

En estos ejemplos, podemos encontrar bien claro cuál es nuestra condición, de la cual va a depender si se ejecuta nuestra acción o no.



Las condiciones serían las siguientes:

1. Que sea de noche
2. Tener hambre
3. Y que esté lloviendo

Si estas condiciones son verdaderas o falsas, nosotros haremos una cosa u otra. Tranquilamente, podemos llevar estas decisiones a nuestro lenguaje.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

En python, como en tantos otros lenguajes, vamos a utilizar un if, veamos como quedaría nuestra sintaxis:

```
x = 6
y = 5
if(x>y):
    print(f'{x} es mas grande que  {y}')
```

```
...      6 es mas grande que 5
```

X e Y son variables que usamos con nuestro operador “mayor a” como condición.

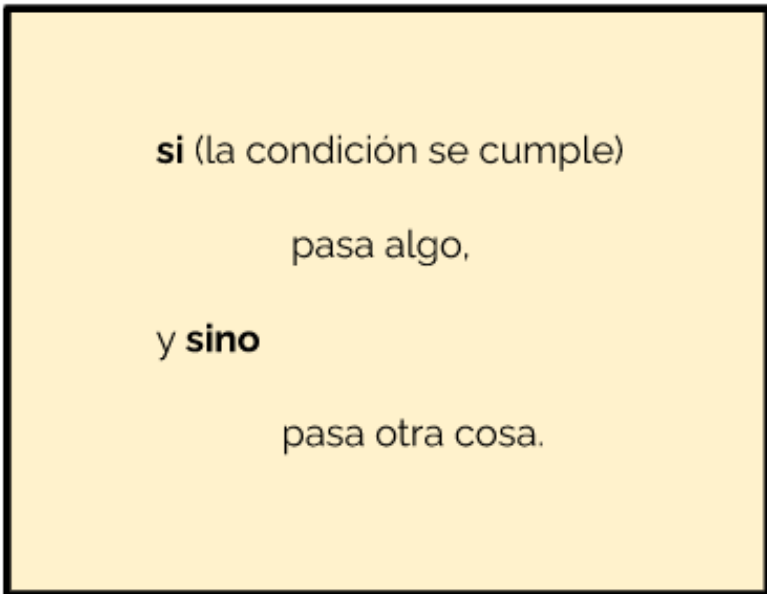
Estructura else

“Sino” en python

Como mencionamos anteriormente, la declaración if en Python se utiliza para operaciones de toma de decisiones. Su estructura permite que se ejecute sólo cuando la condición dada en la declaración if es verdadera.

¿Pero qué pasa si la condición es falsa?

En estos casos existe una palabra reservada traducida como “sino” que se ejecuta cuando la condición del if es falsa. Es decir:



Esta es la declaración else: Esta declaración es opcional y contiene un código para la condición **else**.

Cuando querés justificar una condición que resultó falsa, entonces usa la declaración if else de Python. Es una de las mas importantes, por lo que te recomiendo que si te queda alguna duda sobre el tema, lo consultes con tus compañeros por el canal de Discord o con tu mentor.

Diagrama de flujo if… else

Buenos Aires aprende
Agencia de Habilidades para el Futuro


```
if expresión:
    Declaración
else:
    Declaración
```

Un ejemplo:

```
a = 4
b = 5
if(a<b):
    print(f'{a} es mas chico que {b}')
else:
    print(f'{a} es mas grande que {b}')
```

```
    ...      4 es mas chico que 5
```

En la línea 1 y 2 declaramos las variables a y b como 4 y 5 respectivamente

En la línea 3 “preguntamos” si a es menor a b

Si esta condición es verdadera, entonces imprimimos por consola el texto del print de la línea 4.

En la línea 5 agrupamos todos los valores que no son verdaderos para la pregunta de la línea 3 y devolverá en este caso, el texto del print de la línea 6.

Cuando “otra condición” no funciona

Puede haber muchos casos en los que tu "condición else" no te dé el resultado deseado. Esto puede hacer que se imprima el resultado incorrecto ya que hay un error en la lógica del programa. En la mayoría de los casos, este error sucede cuando se tiene que justificar más de dos condiciones o declaraciones en un mismo programa.

```
c = 4
d = 4
if(c < d):
    texto= "d es mayor que c"
else:
    texto= "C es mayor que d"
    print(texto)
```

```
    ...      c es mas mayor que d
```

Aquí ambas variables valen lo mismo y la salida del programa es "c es mayor que d", lo cual es INCORRECTO . Esto se debe a que verifica la primera condición (condición if), y si falla, imprime la segunda condición (condición else) como predeterminada. Para eso existe la palabra reservada elif, la cuál usaremos después de un if y podemos traducirla como “sino si”.

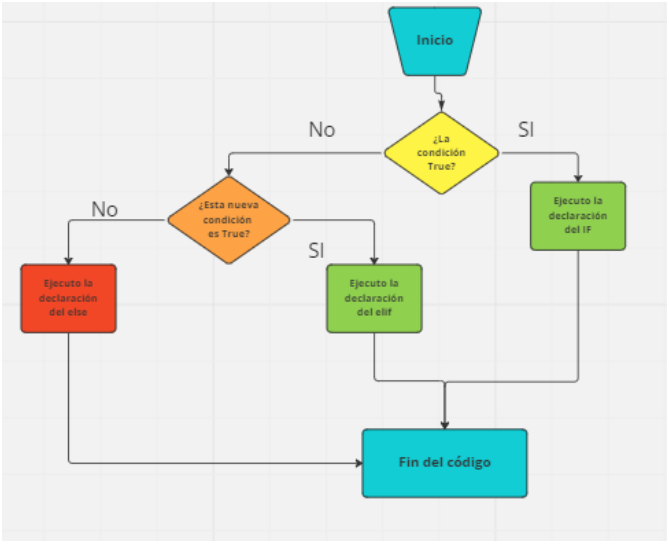
Buenos Aires aprende
Agencia de Habilidades para el Futuro

Estructura elif

“Sino si” en Python

Para corregir el error previamente cometido al usar la declaración "else", podemos emplear la instrucción "elif". Al utilizar la estructura condicional "elif", le estamos indicando al programa que ejecute un bloque de código asociado a una segunda o tercera condición o posibilidad cuando la condición anterior resulta falsa o incorrecta.

```
if expresión:
      Declaración
elif expresión:
      Declaración
else:
      Declaración
```



Un ejemplo:

```
c,d = 4, 4
    if(c < d):
        texto= "d es mayor que c"
    elif (c == d):
        texto= "d es igual a c"
```

```
    else:
        texto= "ces mayor que d"

    print(texto)
```

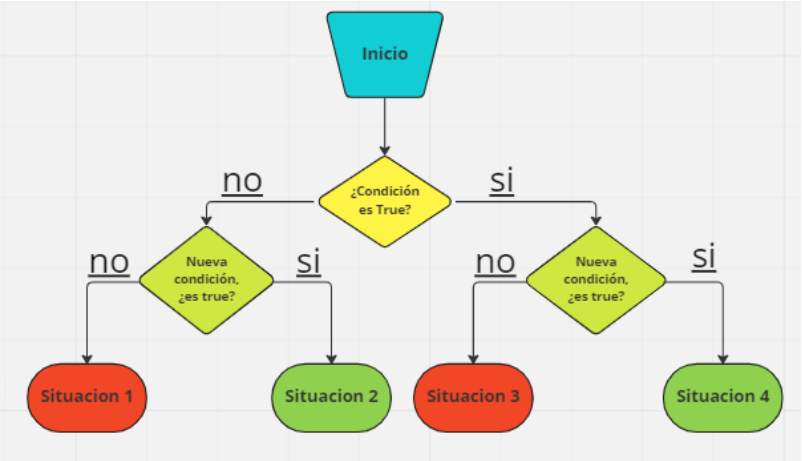
```
        ...      d es igual a c
```

Miremos este corto videito para visualizar como podemos trabajar con los condicionales

https://drive.google.com/file/d/1K4ZYJ2MCSTIyEMaTJgh5EgaznLgiNKRu/view

Anidar declaraciones condicionales

Muchas veces tenemos la necesidad de crear un árbol de preguntas y respuestas para abarcar una mayor cantidad de situaciones y hacer que nuestro código sea más abarcativo. Para esos casos podemos anidar condicionales de la siguiente forma:



Veamos un ejemplo en código como se vería:

```
llueve = True
muchoViento = True
mensaje1 = "Llevo paraguas"
mensaje2 = "Llevo una campera rompevientos"

if (llueve):
    print(mensaje1)
    if(muchoViento):
        print(mensaje1 + " y " + mensaje2)
if (not llueve):
    print("Salgo sin paraguas")
    if (not muchoViento):
        print(mensaje2 + " por las dudas")
```

Este ejercicio tiene cuatro variables:

- Llueve y muchoViento que son True, y dos mensajes de salida de tipo String.

- Como está en el diagrama de arriba, esta situación tiene 4 finalidades, cuatro salidas por la terminal distintas que dependen de si las variables “llueve” y “muchoViento” son True o False.

- Te invitamos a que copies el código y pruebes cambiando el valor de las variables booleanas para imprimir por la terminal los distintos mensajes.

Detalle

Estamos acostumbrados a que con otros lenguajes podemos utilizar un switch para resolver estos casos. Pero esto no es posible con Python, por lo que tendremos que buscar otras opciones..

Si prestamos atención al código, podemos ver que el condicional llueve, que es True, de la linea 7, es el opuesto al condicional “not llueve” de la línea 11.

Esto como vimos, es lo mismo que decir if y else, así que te propongo en tu código, cambiar la estructura del código para trabajarlo de esta forma. ¡Incluso vas a notar que el diagrama para trabajar con condicionales anidados sería el mismo!

Resumen:

Una estructura condicional en Python es manejada por declaraciones if y vimos otras formas en las que podemos usar declaraciones condicionales en Python cómo else y elif.

if: Se utiliza para imprimir un resultado dependiendo de una sola condición, si esta es verdadera o falsa.

else: Imprime la declaración cuando las condiciones no se cumplen.

elif: Empleado para una tercera posibilidad. Permite verificar múltiples condiciones elif para diferentes posibilidades.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Desafío N° 2

1. Crear dos operaciones básicas: 1.- Suma de 3 números enteros. 2.- Resta de números flotantes.
2. Usando condicionales y operadores lógicos, crea un programa simple que evalúe dos números enteros e imprima por pantalla: El mayor de los dos números, el menor de los números o si son iguales.


Buenos Aires
aprende
Buenos
Aires
Ciudad