Hola y bienvenido al Capítulo 3.

Ahora hablaremos de la ejecución condicional.

La ejecución condicional es donde empezamos a agregar más inteligencia.

En la ejecución secuencial, usted solo escribe y Python lo hace.

Pero es en la secuencial donde puede elegir una de dos rutas.

Ir a la izquierda, a la derecha o recto.

Puede tomar decisiones y

es una parte esencial de hacer que las computadoras parezcan inteligentes.

Así que la clave aquí es la instrucción **if.**

La instrucción if es la palabra reservada que indica que haremos

algo condicionalmente y no es tan difícil de leer.

La instrucción if tiene una condición: es una pregunta.

A diferencia de x=5, que cambia potencialmente el valor de x,

x menor que 10 hace una pregunta. Pregunta: ¿es x menor que 10?

Y eso devuelve Verdadero o Falso.

Aquí, ya que x es 5, x es menor que 10 y resulta verdadero.

Estas instrucciones finalizan en ":".

Y empiezan en un bloque de texto con sangría.

Es como un esquema que dice básicamente: si esto es cierto,

ejecute esta instrucción.

Si no, omita la instrucción.

En este caso, x es menor que 10, así que ejecuta la instrucción.

Luego vuelve y dice:

si x es mayor que 20, devuelve Falso y se omite la instrucción.

Por eso vemos que se imprime Smaller y Finis

y esta línea nunca se ejecuta.

Por eso se llama condicional.

Según esta pregunta que se realiza,

la línea dentro del bloque con sangría se ejecutará o no se ejecutará.

Así que se puede ver esto como conducir un automóvil, ¿verdad?

Ejecuta esta instrucción y luego la siguiente, luego se hace la pregunta.

¿Es x menor que 10? Si la respuesta es sí, gira de esta forma, avanza hasta aquí abajo

y ejecuta este código.

Y luego vuelve a unirse.

Y luego dice, ¿es x menor que 20?

Ya que x es 5, la respuesta es no.

Y eso se omite por completo, luego se ejecuta esto y termina.

Este código nunca se ejecuta, ¿cierto?

Cuando decimos condicional, decimos: sí, esa se ejecutaba, pero esa no.

Por eso le decimos condicional porque puede o no ejecutarse.

El otro elemento importante sobre esto es la noción de sangría.

Comienza con un if, tiene dos puntos y, luego, sangría.

La sangría puede ser una variable.

Preferimos una sangría de cuatro espacios.

Ese es el patrón recomendado.

Veremos en breve que las sangrías pueden ocupar más de una línea.

Esta es una sangría de una línea donde hay una línea condicional, pero

luego veremos cómo podemos hacer esto con más de una línea.

Reproduce el video desde :2:41 y sigue la transcripción2:41

Ahora hablemos de los operadores de comparación.

De nuevo, aquí el problema deriva de los limitados teclados de las computadoras

de las décadas de 1950 y 1960, cuando se creó la mayoría de estos lenguajes.

Así, "<" está en los teclados,

"<=" significa "menor o igual que".

"==" es probable que sea la parte más difícil:

"==" es un signo de pregunta.

Recuerde que "=" es una instrucción de asignación con una orientación.

x = 1, ¿cierto? x = 1.

Pone 1 en x.

Si se dice x == 1, se está preguntando:

¿es x igual a 1 y no daña a x?

Eso probablemente tome un tiempo.

Reproduce el video desde :3:19 y sigue la transcripción3:19

Si desarrollara un lenguaje, yo diría asignar x en 1

y esto sería un solo signo igual.

Pero no diseñé Python, así que no funciona así.

El "==" es la versión del signo de pregunta de igualdad.

"Mayor o igual que" y "mayor que" y "no igual" es un signo de exclamación.

Otra palabra para exclamación es "bang".

Decimos "!=" o "no igual".

El signo de exclamación dice "no igual", lo enfatiza.

No lo sé.

Así es como lo recuerdo.

Reproduce el video desde :3:51 y sigue

Recuerde que nada de esto daña los datos que están viendo.

Evalúan y, luego, devuelven un Verdadero o Falso.

Este es un ejemplo de estas cosas en acción.

Para x = 5, todos serán Verdaderos.

Si x = 5, recuerde que es el signo de pregunta.

Si x > 4 y la respuesta es sí, imprimir "Greater than 4".

Mayor que o igual a (>=) 5, también es Verdadero, y esto se ejecuta.

También puede, si es una sola línea de cosas,

puede subir esto al final aquí.

Si x < 6, imprimir "Less than 6".

Reproduce el video desde :4:25 y sigue la transcripción4:25

Si x es menor que o igual a (<=) 5, imprimir "Less than or equal 5".

Si x no es igual a 6, imprimir 6.

Vea este patrón de sangría, fin de sangría.

Sangría, fin de sangría.

Sangría, fin de sangría.

Sangría, fin de sangría.

Esta es una parte importante de Python.

No muchos lenguajes hacen de la sangría de líneas

algo sintácticamente importante, pero así funciona Python.

Si aplica sangría, no funcionará de la manera que esperaba.

En especial si viene de un lenguaje de programación como JavaScript,

o Java, o C, donde el espaciado no importa.

En Python, el espaciado sí importa.

Reproduce el video desde :5:3 y sigue la transcripción5:03

Este es un ejemplo de un bloque de texto con sangría.

Ahora puede ver cómo funciona la sangría.

x es 5, imprimir "Before 5".

Si x = 5, esto será Verdadero.

El signo ":" inicia un bloque con sangría.

Al mantener la misma sangría,

todo esto se vuelve parte del bloque de ejecución condicional.

Son tres líneas de código que se ejecutan,

ejecuta esta línea, luego ejecuta secuencialmente esta línea, ejecuta secuencialmente esta línea.

Y nosotros indicamos cuándo queremos salir de este bloque y

seguir al eliminar la sangría, ¿sí?

Cuando es Verdadero, ejecuta todo, ahora se ejecuta secuencialmente,

secuencialmente, secuencialmente.

Y ahora dice, oh, if x == 6.

Esto será Falso, omitirá todo esto.

Nada de esto se ejecuta.

Omite todos los bloques con sangría.

Esta sangría es una forma de, en efecto,

hacer bloques más grandes de código condicional, o bloques de varias líneas de código condicional.

Reproduce el video desde :6:8 y sigue la transcripción6:08

La sangría es importante en Python.

Es más importante en Python que en cualquier otro

lenguaje de programación.

En otros lenguajes de programación siempre usamos la sangría para

registrar bloques de texto, pero

no es que el lenguaje de programación se quejará, si usted se equivoca.

Así que debe pensar por qué aumenta la sangría, que es como

si, tras una instrucción if, for, while, que veremos en breve,

mantiene la sangría, o sea que permanece igual, o elimina la sangría o

reduce la sangría, y esa es la forma de terminar un bloque.

Así que la reduce.

Las líneas en blanco no importan.

Los comentarios en sí en una línea no son relevantes para la sangría.

Mientras aplica la sangría,

es muy natural pulsar la tecla TAB en su computadora y continuar.

Así que hace: if blah, blah, blah y, en la siguiente línea, pulsa TAB.

El problema es que Python puede confundirse mucho si, a veces,

usted usa 4 espacios y, a veces, usa la tecla TAB.

El peligro es que puede aparecer bien en la pantalla, pero

Python aun así se quejará.

Así que si usa un editor de texto que pueda poner tabulaciones en su documento,

busque la opción y desactívela.

Un editor de programador, no un procesador de texto,

hablamos de un editor de programador, como Notepad o Text Wrangler,

tiene un ajuste, convertir o no tabulaciones en espacios.

O pone una tabulación en el archivo o pone espacios.

Así que es un buen momento, si usa uno de estos editores de texto,

para desactivar las tabulaciones o hacer que se expandan.

Para no terminar, en pocas semanas o días,

con errores de Python que no comprende y lo ponen furioso.

La buena noticia es que, si usa nuestro editor de texto

recomendado Atom, cuando guarde el archivo con extensión .py,

automáticamente usa… usted pulsa la tecla TAB y mueve los espacios.

Eso es muy bueno y es solo uno de los muchos motivos para recomendar

y usar Atom para editar archivos.

Debe entender que los espacios y tabulaciones pueden verse iguales en la

pantalla, pero para Python, pueden o no ser equivalentes.

Y puede terminar con errores que Python devolverá,

si se equivoca al tabular, ¿sí?

Estas son algunas de las pantallas.

Según la herramienta que use,

el editor que use, es posible que deba buscarlo.

Pero es como "autoexpandir tabulaciones" o "reemplazar tabulaciones por espacios", etc.

La idea es no poner tabulaciones en el documento.

La mejor práctica es usar espacios y

no poner tabulaciones en el documento al escribir programas en Python.

Debe pensar en esto, sobre la sangría y

su eliminación, ¿cuándo entra y cuándo salir?

Así es como definimos los bloques.

Tenemos código secuencial, que mantiene la misma sangría.

Vemos el "if" y los dos puntos ":", o sea que empezamos una sangría.

Para seguir dentro de ese bloque de código, debe mantener la sangría.

Para finalizar ese bloque de código, borra la sangría.

Piense en esto como si borrara la sangría, lo hago a propósito,

no de casualidad.

Si esto estuviera adentro, funcionaría de otra forma.

Que este print quede afuera significa que no es parte de este bloque if.

Estas dos instrucciones forman parte del código condicional;

print es lo que sigue al if, ¿sí?

Esto va a funcionar sí o sí; estos dos pueden funcionar o no.

Aquí, porque es Verdadero, van a ejecutarse y saldrán.

Esto es código condicional,

esto es código secuencial.

Reproduce el video desde :9:46 y sigue la transcripción9:46

Si avanzamos hacia abajo, aún no hablamos sobre bucles, pero la palabra clave

for es un bucle que indica ejecutar esto 5 veces y termina en dos puntos ":".

Luego entra,

hay un código secuencial, hay un if, esto se llama anidación,

Reproduce el video desde :10:2 y sigue la transcripción10:02

un bloque dentro de un bloque.

El if tiene otros dos puntos ":" y avanzamos aun más.

Luego borramos sangría al nivel de este if; y

eliminamos la sangría al nivel de for.

Así que debe combinar estas cosas y

decidir cómo terminar su código condicional o finalizar el código de bucle,

cuyo funcionamiento veremos en la sig