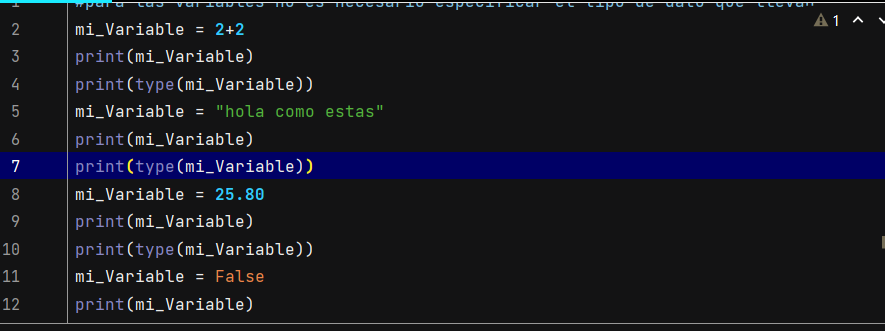
**TIPOS DE DATOS PYTHON**

Existen 5 ramas de las cuales se desprenden otras 6 subdivisiones de tipos de datos en lenguaje Python.

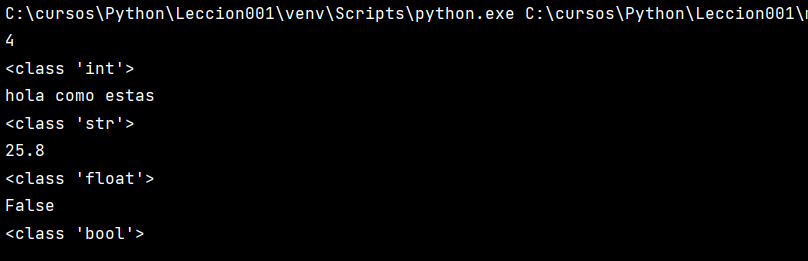
Estos son:

Para saber el tipo de variable de la variable necesaria usamos type(variable) para conseguir su tipo de dato. Empieza a variar dependiendo del tipo de dato que nosotros le ingresamos a la variable. Punto importante es iniciar y cerrar con el mismo tipo de comilla, o dos abiertas y cerradas o una abierta y cerrada pero nunca usar ambas en el mismo término.

Por ejemplo, definimos diferentes tipos de dato para la misma variable construyendo y reutilizándola y a su vez mostrando su type así:

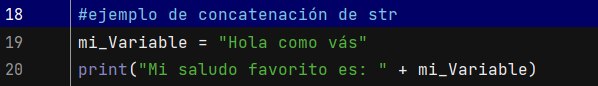


El resultado en consola al correr el script es:



**Concatenación.**

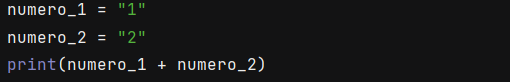
Para concatenar dos variables debemos utilizar tipo string(cadena) para que no haya problemas de sintaxis. En este proceso el símbolo “+” se utiliza para concatenar y así formar oraciones. He aquí un ejemplo:



Se ejecuta así:

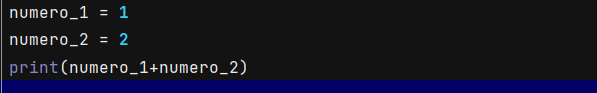


Muchas veces se cae en el error de intentar sumar dos valores y al ver el resultado solo se concatenan, por lo que hay que usar bien el sintaxis correcto del dato, es decir, no utilizar, por ejemplo:

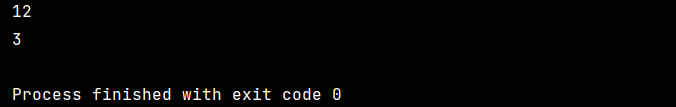


Con las comillas le damos a entender al programa que es un texto tipo string y lo va a concatenar literalmente uno tras el otro cada dato que le pidamos sumar, por lo que daría 12 en esta situación.

La manera correcta o bueno, la sintaxis correcta seria:

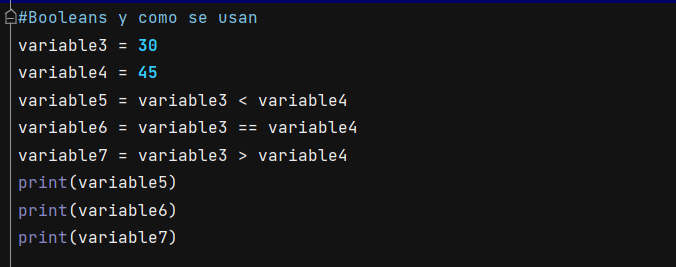


Ambos resultados se muestran a continuación:

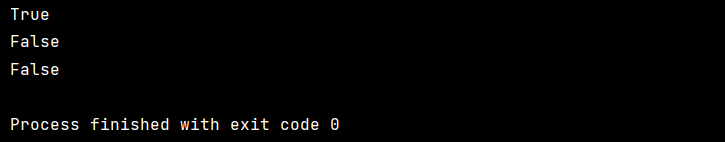


**Booleanos**

Se utilizan para obtener resultados con solo dos posibles respuestas, True or False. Es decir, cuando le preguntamos por ejemplo a la maquina si dos variables son iguales, diferentes, una mayor que otra o una menor que otra, etc. He aquí la demostración:



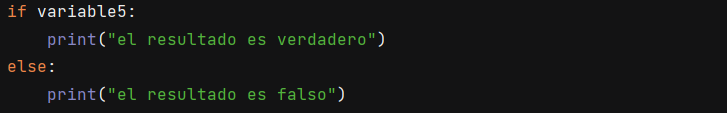
Utilizamos varios comprobantes como es el mayor que, menor que e igual que en Python se representa con dos signos iguales consecutivos (==). Al correr este script el resultado de consola es:



Permiten enrutar la decisión basada en el resultado obtenido y elegir una ruta adecuada para el tratamiento de las variables obtenidas.

Pequeño paréntesis:

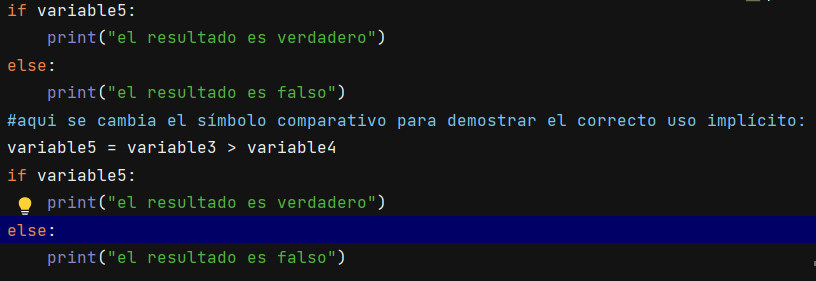
Si la variable es verdadera o falsa, va implícito en el operador if, es decir, si pones:



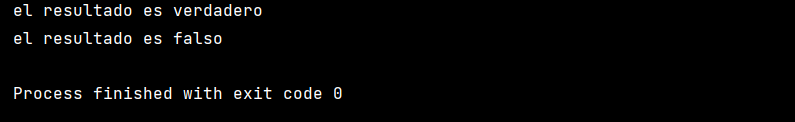
Da como resultado:



Aquí una vez más, pero aplicado a ambas situaciones cambiando su símbolo comparativo:



Y su resultado como es de esperar:



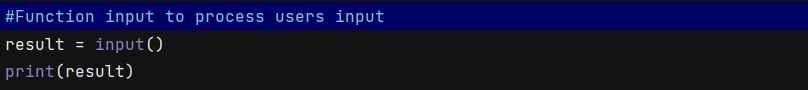
Aquí tenemos una ruta básica de verificación de datos con if y datos de tipo bool:

Como podemos ver, básicamente es un filtro de información para continuar procesando los datos de manera adecuada y obtener el resultado esperado o, por el contrario, también se puede aplicar para que, al no tener el resultado esperado pueda reiniciarse el proceso hasta que se obtengan, o bien simplemente dejar una nota de cuales son los datos que se necesitan y su veracidad antes de ingresarlos en el sistema y el mapa de ruta de información a utilizar.

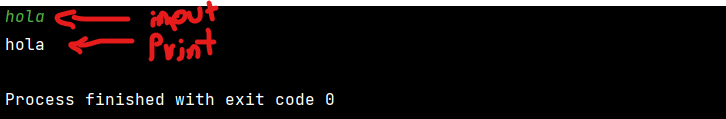
**INGRESO DE DATOS**

**Función [input()]:**

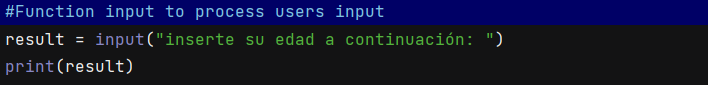
Ingresa por parte del usuario información al código en desarrollo. Veamos un pequeño ejemplo.



Aquí lo que se le pide al usuario es información, con el fin de luego, con la función print(), sea impreso lo mismo que escribió la persona. Se vería así su funcionamiento:



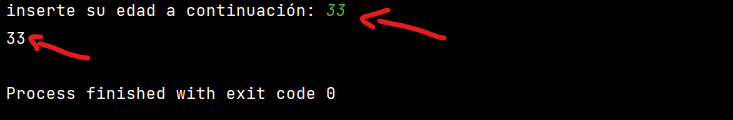
Ahora bien, podría ponerse un pequeño mensaje dentro del input() para que se vea un cuadro de texto dentro de la consola pidiendo la información requerida. Aplicado así:



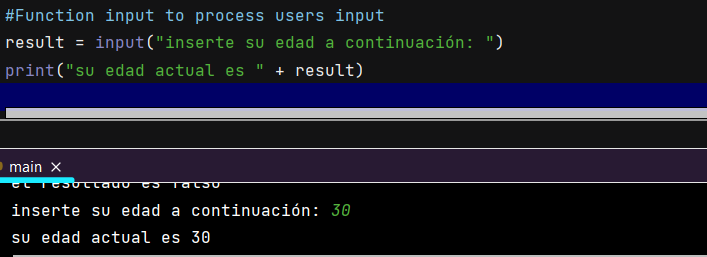
Lo que nos mostraría en la consola sería:

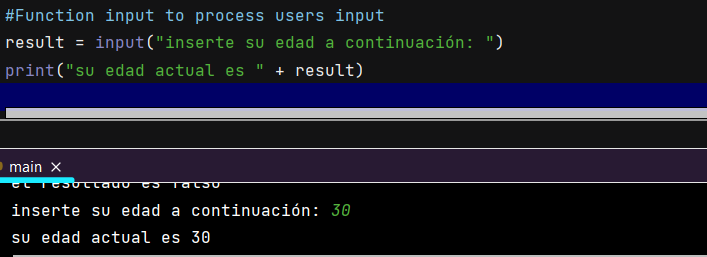


Quedando así el resultado:

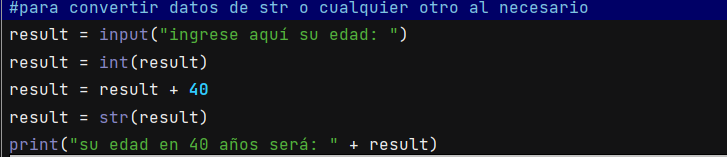


Más estética quedaría:



Resultado:

Tener en cuenta que los datos ingresados en input() serán tomados por defecto como str() por lo que serían concatenados en operaciones matemáticas. En caso de que se necesiten números o decimales deberá hacerse la correcta conversión de los datos. Esto se logra de la siguiente manera:



Cabe resaltar que para utilizar el símbolo + si debe realizarse esa operación de conversión regresiva, sin embargo, puede realizarse simplemente poniendo lo mismo en print() y utilizando entre los términos el símbolo coma. Quedaría así:



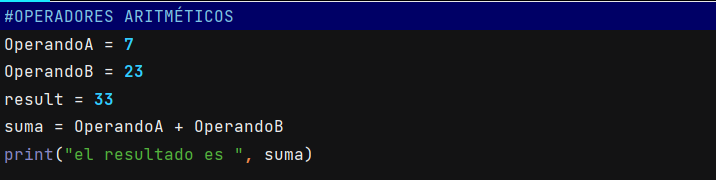
En este último caso, no sería necesario realizar la conversión regresiva de int() a str() que vemos en la 4 línea de código.

Por defecto el sistema recibirá una información tipo str, por lo que será convertida a int() por medio de la correcta utilización de esa función, para después sumarla en la parte de abajo con el número requerido, luego se transforma nuevamente a str() para poder ser concatenado de manera efectiva con el texto a mostrar con print. La ejecución de este script sería:

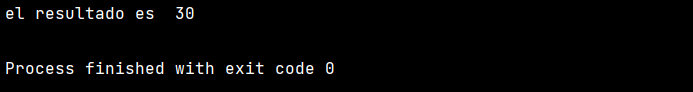


**OPERADORES ARITMÉTICOS**

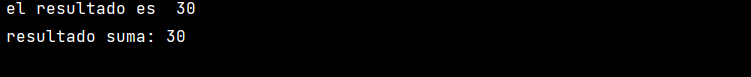
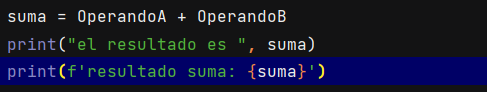
Utilizados para relacionar dos o más tipos de datos de diferentes maneras dependiendo de su tipo y su contenido, veámoslo así:



Run:



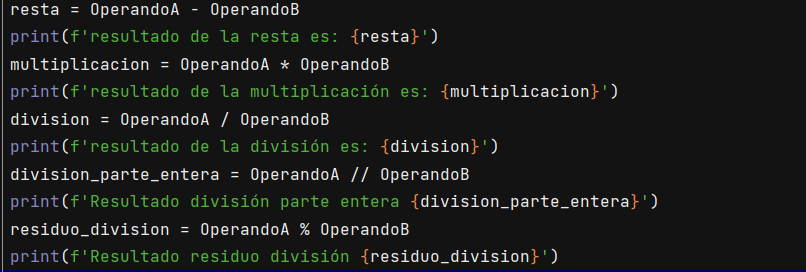
Otra manera de realizarlo es con la f literal en Python. Esta se aplica utilizando la variable a imprimir, todo dentro de comillas simples y las variables adicionales dentro de llaves, de la siguiente manera:



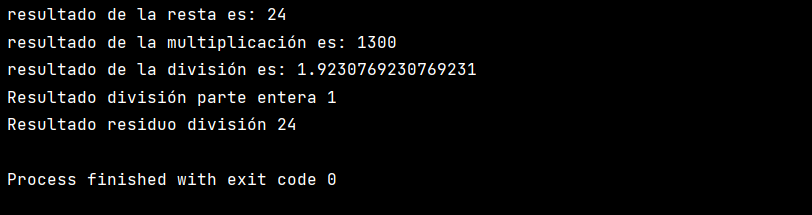
Esto de incluir variables dentro de una cadena str() se le conoce como interpolación.

* No olvidar poner las comillas solo sencillas no dobles
* Poner la f antes de las comillas
* Poner las variables que hay que meter dentro de las str() entre llaves, no corchetes ni paréntesis.

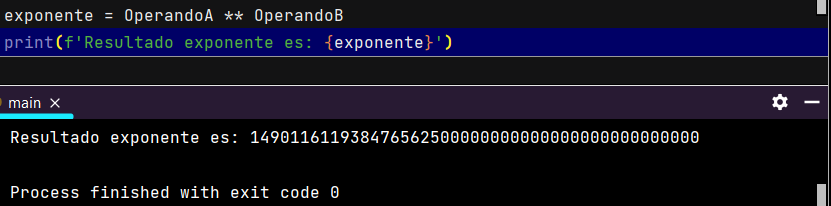
Continuando con Operadores Aritméticos, existen suma(+), resta(-), multiplicación(\*), división(/), resultado de división parte entera(//), residuo de la división(%), exponente(\*\*). A continuación, un ejemplo ilustrativo de como se vería con un ejercicio de interpolación en cada uno de los casos:



El resultado de estas interpolaciones es:



Por último, el exponente (\*\*)



Este pondrá la variable B como exponente de la variable A y el resultado aparecerá en la consola al ser ejecutado.

**OPERADORES DE ASIGNACIÓN**

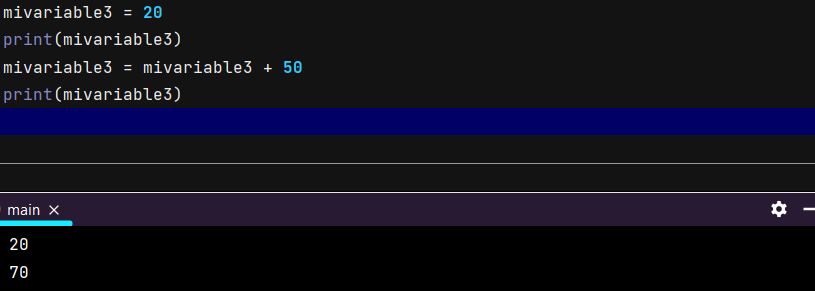
Los operadores de asignación son los que le dan una propiedad o información determinada a otra información, como ocurre cuando se usa la expresión:

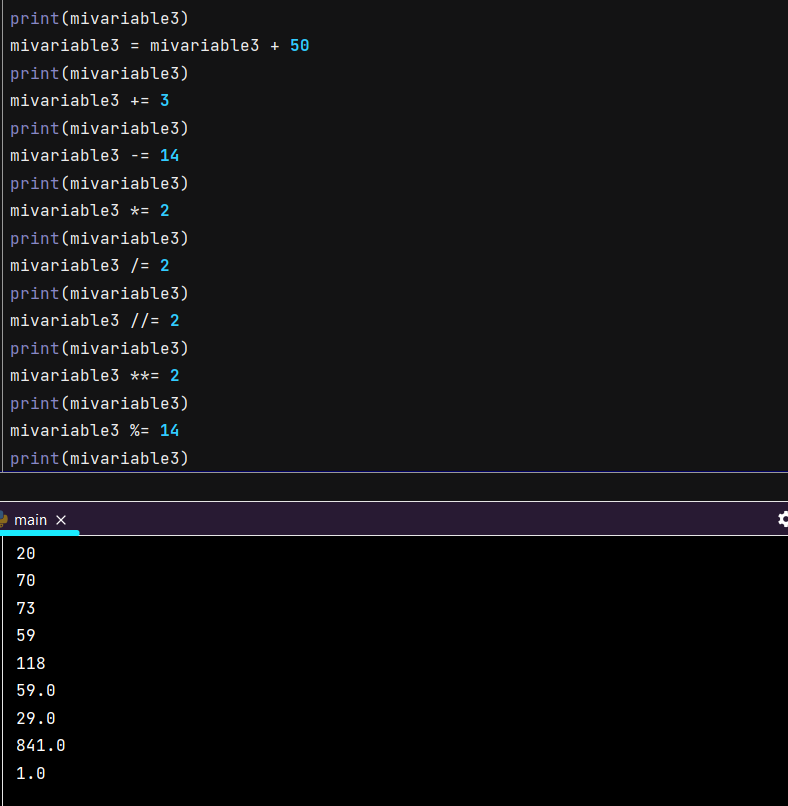
mi\_variable = 1

el operador de asignación es el igual básicamente.

Los operadores de reasignación son los que, como su nombre lo indica, reasignan información o características a una misma variable.

Por ejemplo:



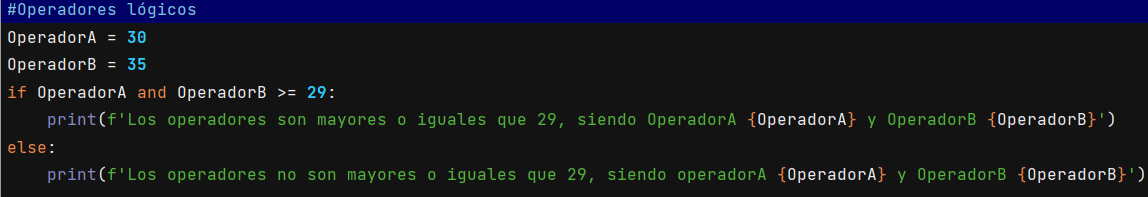
Otro shortcut que podemos utilizar como operador de reasignación es la utilización de (+=)0 , (-=), (\*=), (/=), etc. Veamos un ejemplo de muchas aplicaciones y sus resultados:

**OPERADORES LOGICOS**

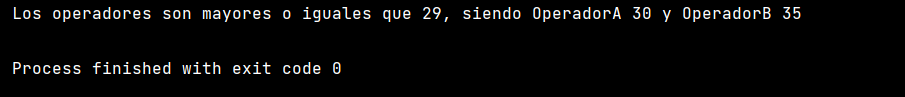
Estos son los encargados de verificar la veracidad o la posibilidad de incluir una variable para el resto del script o el código en general, dependiendo de si cumple una afirmación, dos o más, o incluso si no las cumple. Los operadores lógicos son:

1. *and*

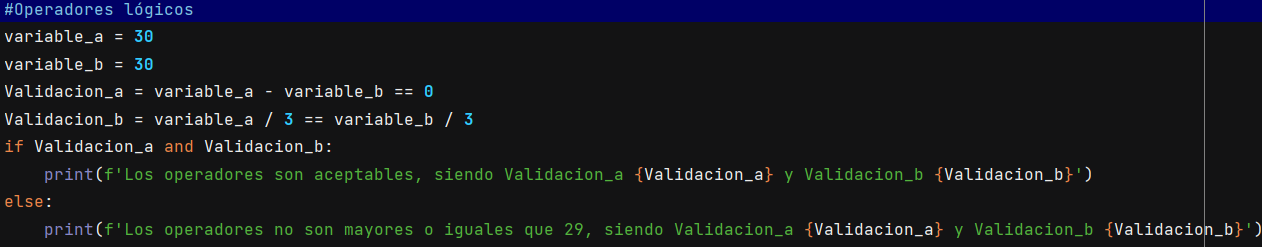
Se utiliza para cumplir dos requisitos o más al mismo tiempo y, al ser estos aceptados devolverá al código el valor de *True*. Como se observa en el siguiente script:



Al correrlo en la consola debe devolver el valor True de ambos, así:

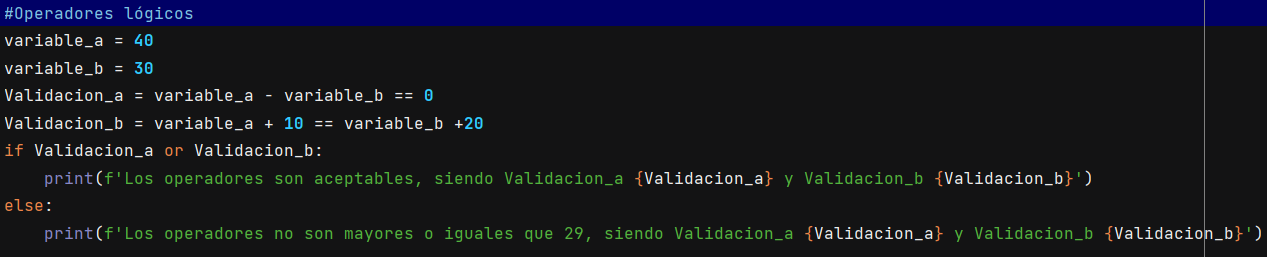


Otra manera de correrlo sería sin un parámetro de comparación como el >=29, quedando así:

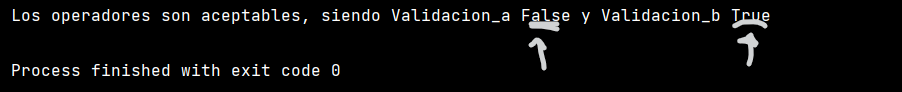


1. *or*

*Es utilizado prácticamente al igual que el anterior, solo que este enviará como resultado positivo si solo alguno de los argumentos (Validacion\_a;Validacion\_b) se cumplen. Como se observa a continuación:*



Resultando así:



1. *not*