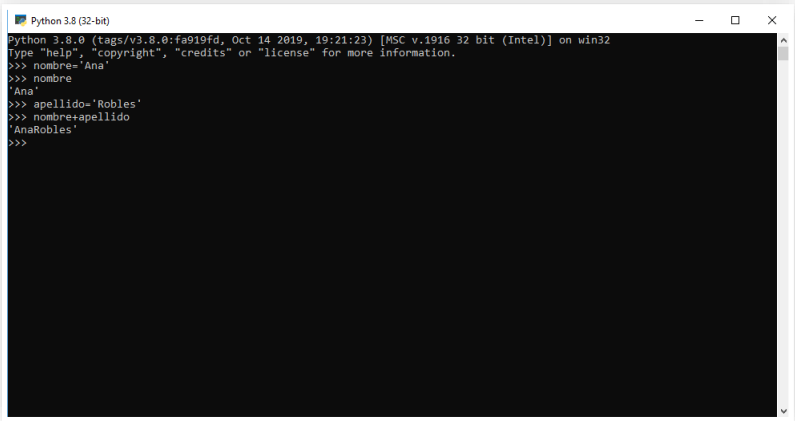
Manejo de strings

En Python es muy fácil asignar cadenas de caracteres o *strings* a variables. En este caso, deberás poner el valor que desees asignar a una variable entre comillas sencillas, por ejemplo:

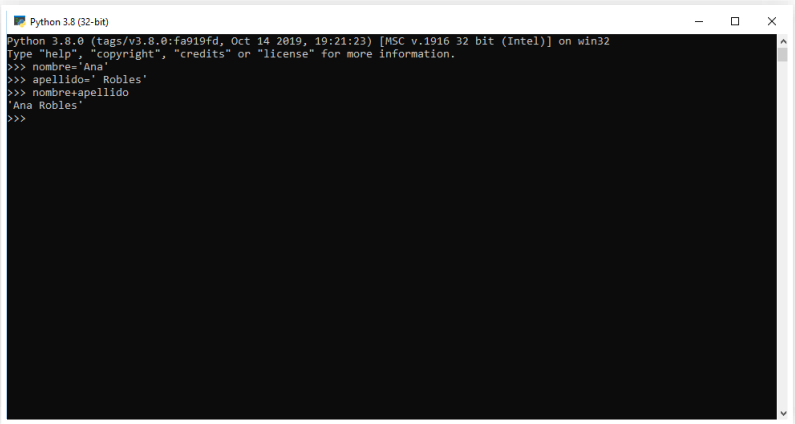
nombre= 'Ana'
apellido='Robles'

Si lo que buscas es concatenar ambos valores basta con utilizar el operador de suma (+) tal y como se hace con los valores numéricos:

nombre+apellido



En este caso podrás observar que la suma de las cadenas de caracteres se hizo sin incluir espacio. Por otro lado, si lo que deseas es agregar un espacio entre nombre y apellido, basta con que se lo agregues en la asignación de cualquiera de las variables, ya sea al final del nombre o al inicio del apellido. En el ejemplo se lo agregarás al inicio del apellido, como se muestra a continuación:



El procedimiento que utilizamos en la multiplicación numérica también puede ser utilizado en operaciones con strings, por ejemplo, si deseas repetir tres veces una palabra concatenándola, necesitarás escribir un código como el siguiente:

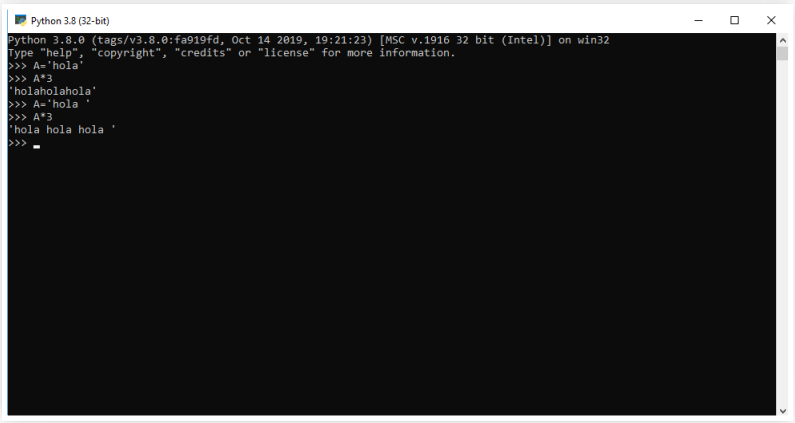
A='Hola'

A*3

Para que las palabras no queden pegadas se agrega un espacio al final:

A='Hola '

A*3



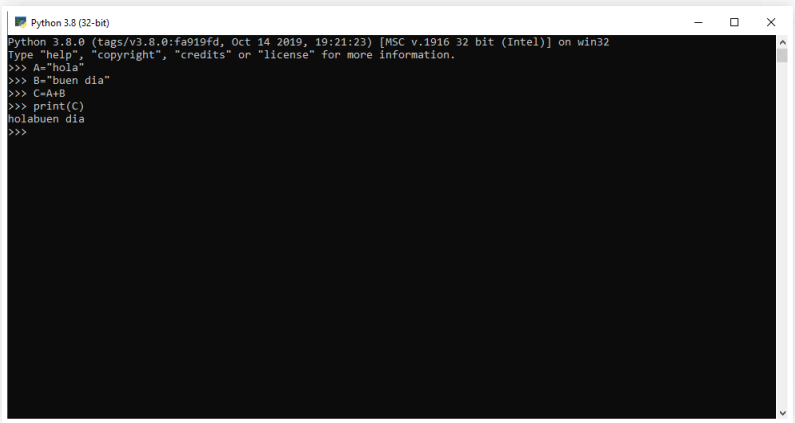
Los strings también pueden asignarse utilizando doble comilla (“). Un ejemplo similar al anterior es el siguiente:

A=”hola”

B= “buen día”

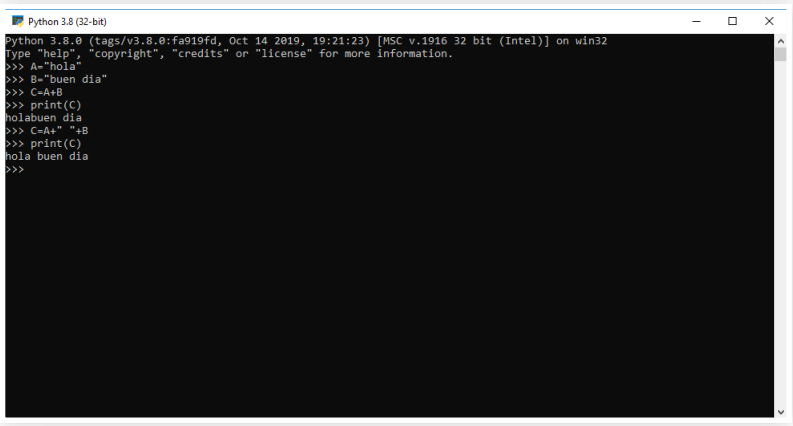
C=A+B

print(C)



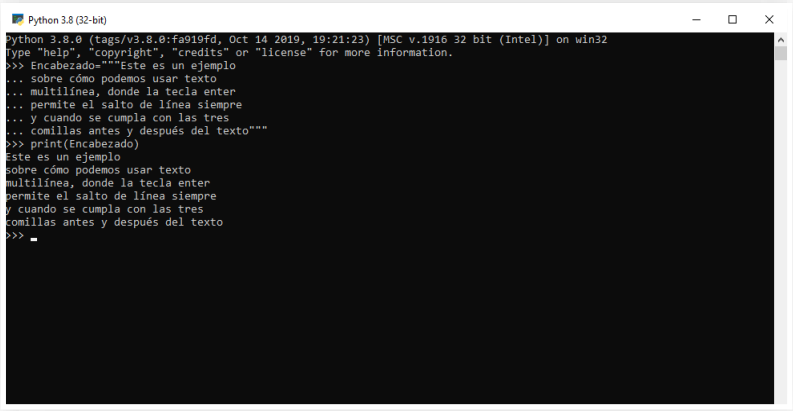
Otra forma de agregar el espacio entre hola y buen día es la siguiente:

C=A+ “ “+B



Si deseas asignar un texto extenso a una variable (con salto de línea) lo puedes hacer, pero se requerirá utilizar tres comillas al inicio del texto y tres al final. Con esta instrucción la tecla Enter servirá para salto de línea, tal y como se muestra en el siguiente ejemplo:

Encabezado="""Este es un ejemplo
... sobre cómo podemos usar texto
... multilínea, donde la tecla enter
... permite el salto de línea siempre
... y cuando se cumpla con las tres
... comillas antes y después del texto"""

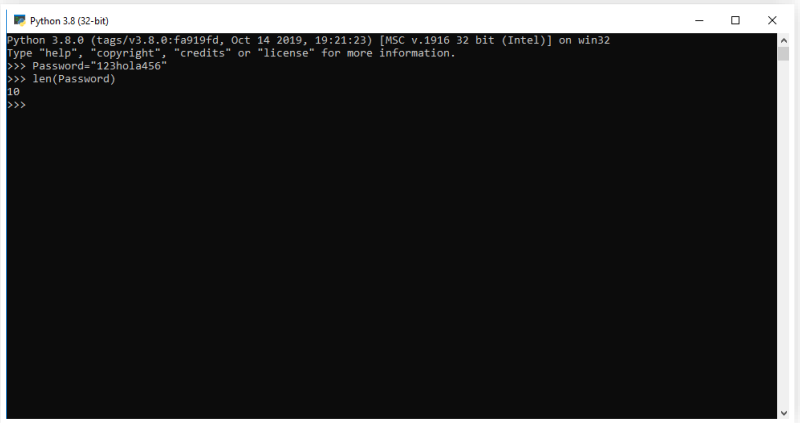


Para conocer la longitud de una cadena podrás utilizar la función *len()*, por ejemplo, crea una variable que tenga el siguiente texto:

Password=”123hola456”

Para conocer su longitud escribirás

len(Password)



En las cadenas (strings) podrás extraer porciones de caracteres, debido a que cada uno de los caracteres de una cadena es un elemento de un arreglo, donde cada uno de ellos guarda una posición determinada por un número entero.

Por ejemplo, define la siguiente variable:

Datos=”05 de Enero de 2020”

Si quisieras extraer el mes de ese texto deberás contar en qué posición comienza ese dato:

Cadena	0	5		d	e		E	n	e	r	o		d	e		2	0	2	0
Posición	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Para este ejemplo el mes inicia en el carácter ubicado en la posición 6, terminando en la 10. Sin embargo, si utilizas la posición 10 para detener la extracción de la porción el carácter *o* de enero no estaría incluido, ya que el límite superior no se incluye. Por lo tanto, la extracción debe quedar como lo siguiente:

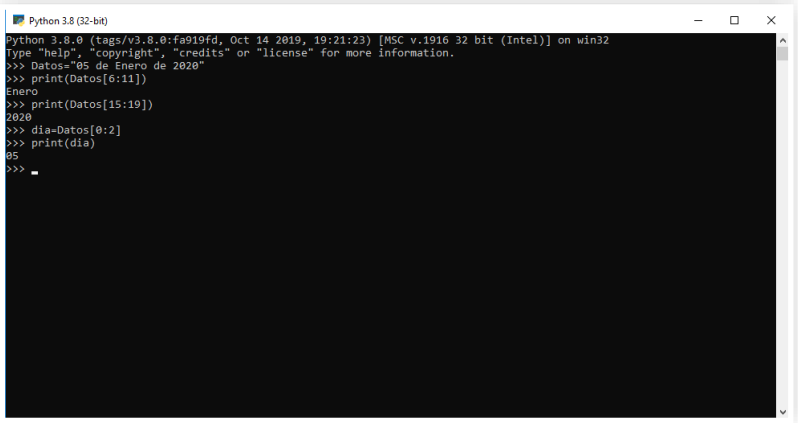
print(Datos[6:11])

Si lo que buscas es extraer el año, entonces la instrucción quedaría de la siguiente manera:

print(Datos[15:19])

Esta información no solo puede imprimirse en pantalla, también puede asignarse a una nueva variable, por ejemplo, para guardar el día:

dia=Datos[0:2]

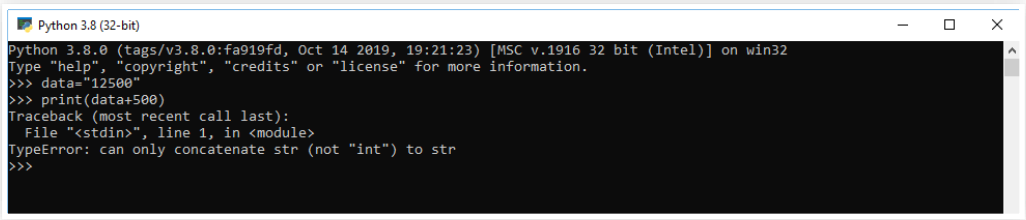


Conversión de texto a entero

Cuando asignas un dato numérico entre comillas le estás diciendo al programa que se trata de una cadena o *string*. Sin embargo, existe la posibilidad de convertir esos datos de texto a un número entero *Integer* o también a un número con decimales *float*, por ejemplo, asignarás a la variable data el valor 12500 como texto:

data=”12500”

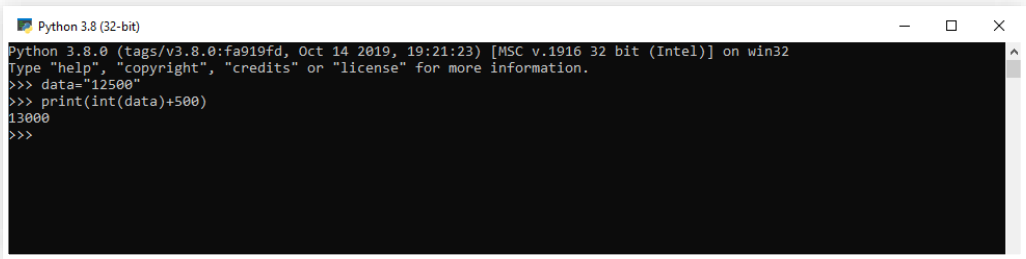
Luego, intentarás sumarle 500 a esa cantidad, lo cual marcará error al tratarse de dos tipos de datos con distinto formato:



En este momento, data es una variable que contiene un tipo de dato *string*. Asimismo, este se podrá convertir a entero utilizando la función int():

Int(data)

Realizarás nuevamente la operación con el siguiente resultado:



Ahora es tu turno para practicar con los siguientes ejercicios:

Práctica:

Define una variable denominada lista y asígnale la siguiente información:

- Carla 80
- Mario 90
- Perla 100
- Arturo 70

A continuación, extrae en una variable llamada Hombres la información de Mario y Arturo y en otra variable llamada Mujeres la información de Carla y Perla. En resumen, de la variable Lista crea dos nuevas variables: una llamada hombres, donde esté la información de Mario y Arturo; y otra llamada mujeres, donde se asigne la información de Carla y Perla. Al final imprime los resultados.

Condicionales

Son una de las herramientas de programación sumamente útiles para la automatización de procesos. Los condicionales son operadores que sirven para comparar valores o verificar si se cumplen condiciones. El resultado que arrojan las condicionales son del tipo booleano, es decir, que solo pueden tener dos tipos de valores falso (*false*) o verdadero (*true*). A los valores falsos también se les conoce como **cero lógico** y a los valores verdaderos como **uno lógico**.

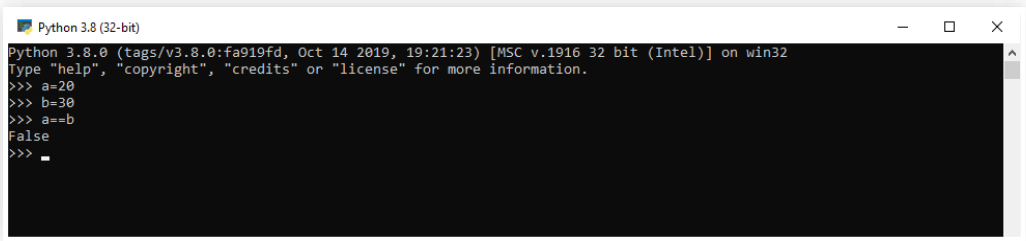
Para comparar si un valor es igual a otro se debe usar un condicional, que en este caso debe ser un doble igual (==), ya que un solo operador de igual (=) significa asignación, como lo vimos en prácticas anteriores. Por tanto, siempre que necesites hacer una comparación de valores deberás utilizar el doble igual.

Por ejemplo, crearás dos variables a=20 y b=30, comparando si son iguales:

a=20

b=30

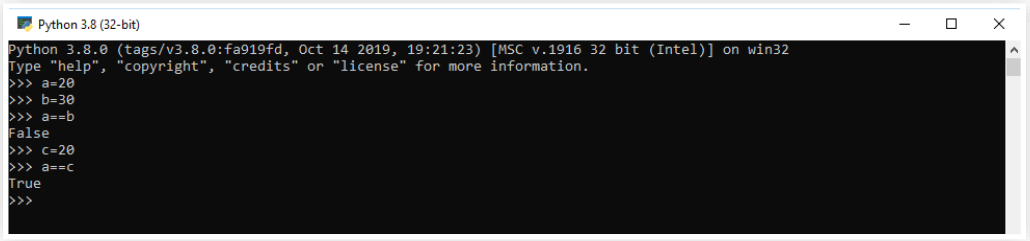
a==b

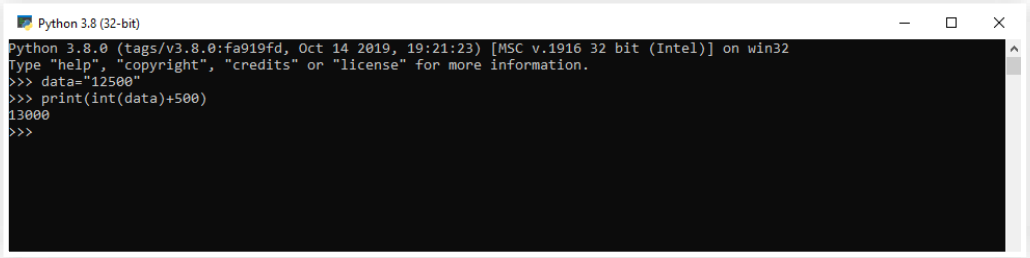


Puedes observar que arrojó un valor falso.

Ahora asignarás un valor de 20 a la variable c y la compararemos con a:

c=20
a==c





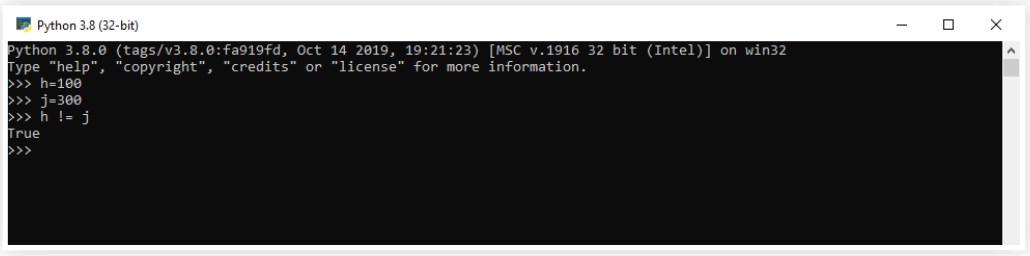
El operador que se utiliza para saber cuando una comparación es desigual es el signo de admiración seguido del signo de igual (!=).

Por ejemplo, si h=100 y j=300 podrás saber el resultado que arroja un programa cuando desees saber si son desiguales.

h=100

j=300

h != j



Lo mismo puedes realizar con los operadores de comparación mayor que (>), menor que (<), mayor o igual que (>=) y menos o igual que (<=).

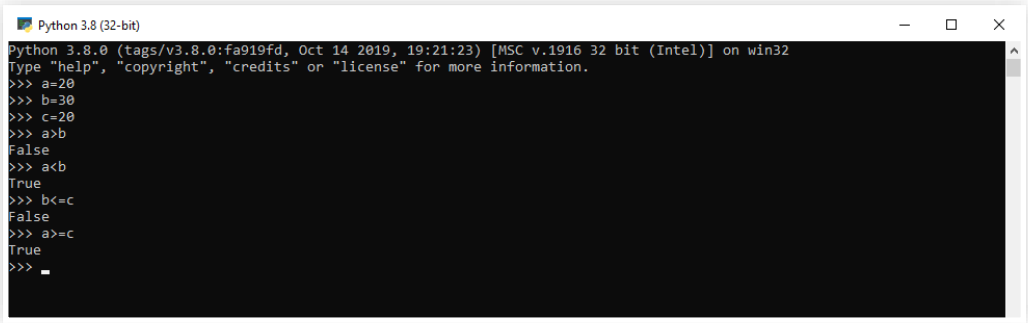
Realiza las siguientes comparaciones para ver sus resultados:

a>b

a<b

b<=c

a>=c



Operadores lógicos

Los operadores lógicos básicos como *and*, *or* y *not* son ampliamente utilizados en la programación porque sirven para complementar los operadores condicionales, ya que permiten la inclusión de varias variables a la hora de tomar una decisión.

El operador *and* implica que la única forma de que la salida sea verdadera (*true*) será cuando todas sus entradas sean verdaderas también, es decir, que para dos variables o valores de entrada tenemos lo siguiente:

Entrada 1	Entrada 2	Salida
False	False	False
False	True	False
True	False	False
True	True	True

Para tres valores de entrada:

Entrada 1	Entrada 2	Entrada 3	Salida
False	False	False	False
False	False	True	False
False	True	False	False
False	True	True	False
True	False	False	False
True	False	True	False
True	True	False	False
True	True	True	True

Para el operador *or* la salida será *True*, siempre y cuando se tenga por lo menos una entrada *True*. En el caso de dos variables:

Entrada 1	Entrada 2	Salida
False	False	False
False	True	True
True	False	True
True	True	True

Para tres valores de entrada:

Entrada 1	Entrada 2	Entrada 3	Salida
False	False	False	False
False	False	True	True
False	True	False	True
False	True	True	True
True	False	False	True
True	False	True	True
True	True	False	True
True	True	True	True

Para un valor de entrada:

Entrada	Salida
False	True
True	False

Por ejemplo, crearás tres variables (x, y, z) y les asignarás valores (1, 2, 3), respectivamente. A continuación, evaluarás si el resultado es verdadero o falso con base en las siguientes condiciones:

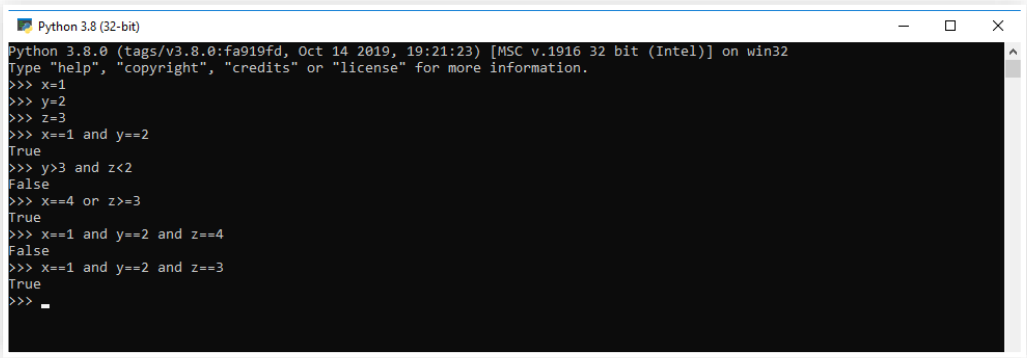
x==1 and y==2

y>3 and z<2

x==4 or z>=3

x==1 and y==2 and z==4

x==1 and y==2 and z==3



Ahora es tu turno para practicar con los siguientes ejercicios:

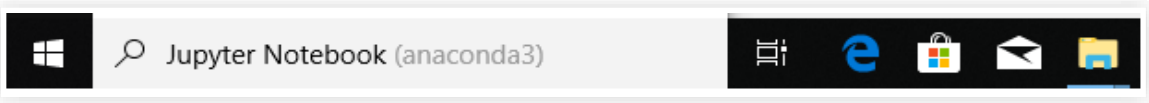
Práctica:

Realiza las siguientes comparaciones y registra los resultados que se obtienen:

A=30, B=60 y C=90
A != B and B != C
A>=B and B<=C
A==20 or B== 70 or C==90

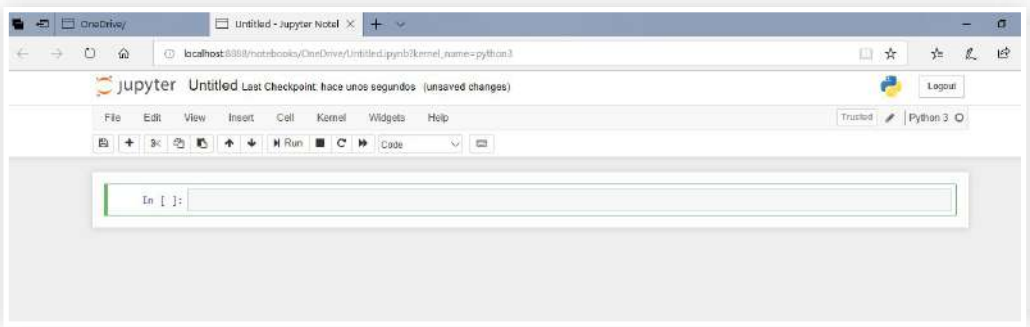
if-else

En este laboratorio utilizarás Jupyter Notebook de Anaconda para trabajar con programas en Python. En la lección teórica (dentro del tema 2) se encuentra el documento con los pasos a seguir para instalarlo. Una vez que tengas el programa podrás abrirlo desde el buscador de tu computadora:

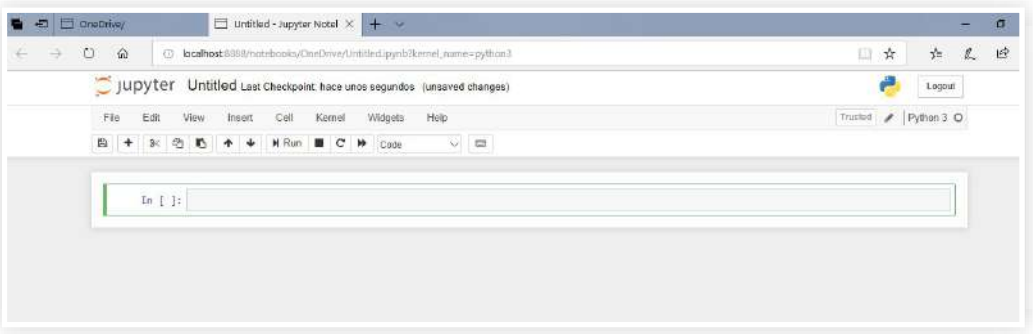


Este se abrirá en el navegador de Internet que tengas como *default*. Posteriormente, en la parte derecha donde dice New seleccionarás Python 3 para crear un nuevo *notebook* para programar.

Te aparecerá un espacio para trabajar en la parte izquierda llamada In [ ]:



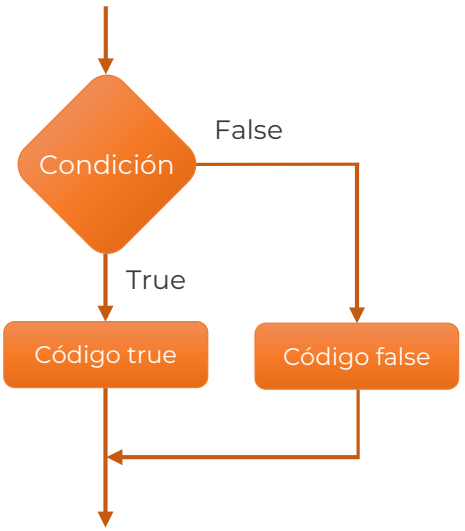
En la *dropdown list* podrás ver que la opción de *Code* viene por *default*, lo que significa que la línea está lista para recibir código:



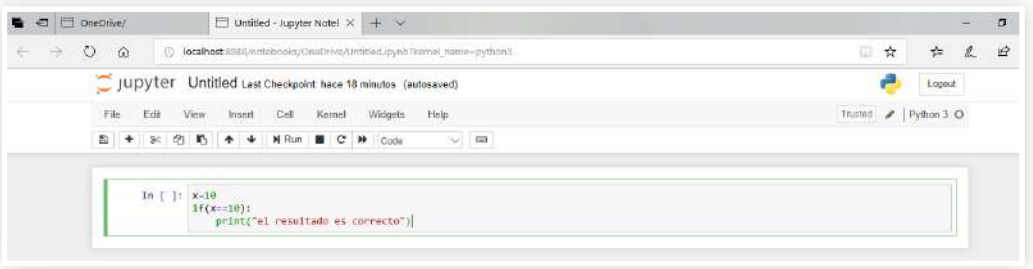
La estructura if-else es sumamente utilizada para la toma de decisiones automatizada. Asimismo, en la siguiente figura podrás encontrar cómo se ejecuta, además de poder observar lo siguiente:

1) Si tiene una condición que puede ser verdadera o falsa.
2) Si esta es *true* se ejecutará el código que se encuentra en el caso verdadero.
3) Si esta es *false* se ejecutará el código que está en falso.
4) Solo puede escoger uno de los dos caminos (nunca ambos al mismo tiempo).
5) Al final cada camino converge a la misma salida.

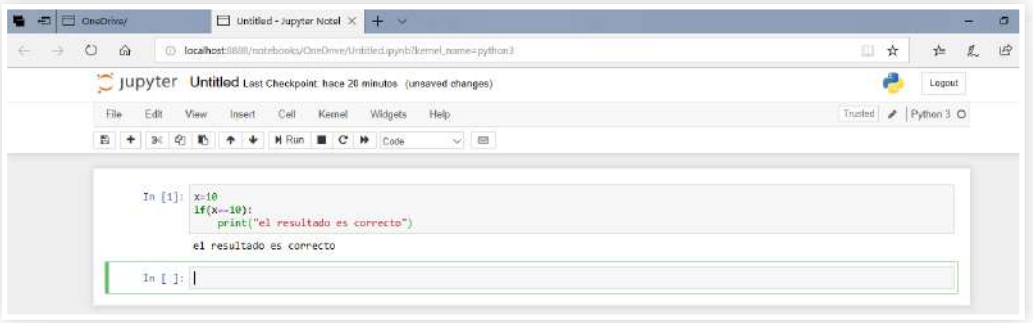
Nota: es importante señalar que *false* no necesita ser programado, ya que es posible poner un *if* sin un *else*. Sin embargo, no se puede programar un *else* sin un *if*.



A continuación, escribe el siguiente código:



Para correr el programa haz clic en el ícono de *Run*. Como podrás observar la variable *x* es igual a 10, por lo tanto en pantalla se imprime lo siguiente: el resultado es correcto.



Ahora harás un nuevo programa donde utilizarás un *if* con un *else,* como en el siguiente caso:

```
In [12]: y=50
         if y>100 :
             print('y es mayor que 100')
         else:
             print('"y" no es mayor que 100')

         "y" no es mayor que 100

In [ ]:
```

En este programa se asignó el valor de 50 a la variable ***y***, después se pregunta si ***y*** es mayor que 100, si es así, se debe imprimir "***y*** es mayor que 100". Sin embargo, si esa condición es falsa, entonces se ha de imprimir "***y*** no es mayor que 100". En este caso lo que se imprimió fue la segunda opción, ya que 50 no es mayor que 100.

Existe la opción de poner *if-else* anidados, es decir, que después de que una condición no sea verdadera (y que el programa siga el camino del *false,* ejecutando el *else*) se pueda preguntar nuevamente si se cumple alguna otra condición por el camino falso, a lo cual se le denomina como *if-else-if*.

En Python esto se le conoce como *elif*, que es una versión abreviada del *if-else-if*. Realiza el siguiente ejemplo:

```
In [*]: dato=int(input("Por favor ingrese un número entre 1 y 10:"))
if 0<dato<4:
    print("el valor es pequeño")
elif 3<dato<7:
    print("el valor es mediano")
elif 6<dato<10:
    print("el valor es grande")
else:
    print("el valor es 10")

Por favor ingrese un número entre 1 y 10:
```

Para este caso utiliza el comando *input* para pedirle al usuario que ingrese un número entre 1 y 10, donde el valor ingresado lo asignarás a la variable Dato. Después, pregunta con un *if* si el valor ingresado es mayor que 0 y menor que 4 (que serían los valores 1, 2 y 3). Posteriormente usarás *elif* (ya que el valor no cumplió con estar dentro del rango anterior) y preguntarás si el valor es mayor que 3 y menor que 7 (aplica para 4, 5 y 6).
En el siguiente caso se cuestiona si el valor es mayor que 6 y menor que 10 (para 7, 8 y 9). Finalmente, si ninguno de los casos anteriores se cumple, entonces quiere decir que el valor ingresado es 10.

Al correr el programa ingresa el número 3:

```
In [17]: dato=int(input("Por favor ingrese un número entre 1 y 10:"))
if 0<dato<4:
    print("el valor es pequeño")
elif 3<dato<7:
    print("el valor es mediano")
elif 6<dato<10:
    print("el valor es grande")
else:
    print("el valor es 10")

Por favor ingrese un número entre 1 y 10:3
el valor es pequeño
```

Ahora ingresa un 5:

```
In [18]: dato=int(input("Por favor ingrese un número entre 1 y 10:"))
if 0<dato<4:
    print("el valor es pequeño")
elif 3<dato<7:
    print("el valor es mediano")
elif 6<dato<10:
    print("el valor es grande")
else:
    print("el valor es 10")

Por favor ingrese un número entre 1 y 10:5
el valor es mediano
```

Luego un 8:

```
In [19]: dato=int(input("Por favor ingrese un número entre 1 y 10:"))
if 0<dato<4:
    print("el valor es pequeño")
elif 3<dato<7:
    print("el valor es mediano")
elif 6<dato<10:
    print("el valor es grande")
else:
    print("el valor es 10")

Por favor ingrese un número entre 1 y 10:8
el valor es grande
```

Para finalizar ingresa un 10:

```
In [20]: dato=int(input("Por favor ingrese un número entre 1 y 10:"))
         if 0<dato<4:
             print("el valor es pequeño")
         elif 3<dato<7:
             print("el valor es mediano")
         elif 6<dato<10:
             print("el valor es grande")
         else:
             print("el valor es 10")

         Por favor ingrese un número entre 1 y 10:10
         el valor es 10
```

Ahora es tu turno para practicar:

Ejercicio:

Realiza un programa que pregunte al usuario dos números enteros, en el cual el usuario introduzca un 1 si desea sumar los números, 2 si desea restarlos y 3 si quiere multiplicarlos. El programa debe arrojar el resultado correcto con base en la opción seleccionada.