Sintaxis PYTHON

A diferencia de muchos otros lenguajes, no se declara el tipo de la variable al crearla.

```
unSaludo = "Hola Mundo"  # cadenas de caracteres

unNumero = 23  # número entero
```

Ejemplo de asignar múltiples valores a múltiples variables

A continuación, se creará múltiples variables (*entero*, *coma flotante*, *cadenas de caracteres*) asignando múltiples valores:

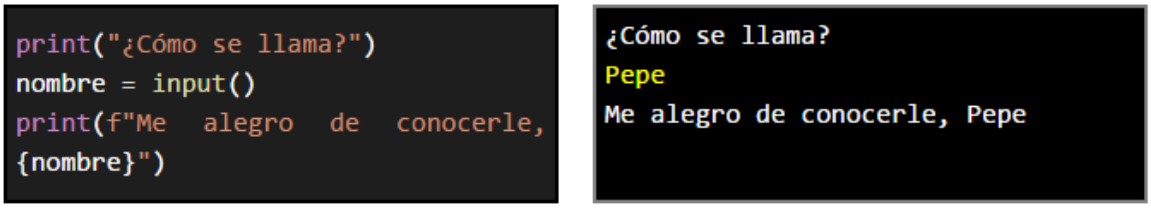
```
>>> a, b, c = 5, 3.2, "Hola"
>>> print a
5
>>> print b
3.2
>>> print c
'Hola'
```

Si usted quiere asignar el mismo valor a múltiples variables al mismo tiempo, usted puede hacer lo siguiente:

```
>>> x = y = z = True
>>> print x
True
>>> print y
True
>>> print z
True
```

La función input()

La función input() permite obtener texto escrito por teclado. Al llegar a la función, el programa se detiene esperando que se escriba algo y se pulse la tecla Intro, como muestra el siguiente ejemplo:



```
print("¿Cómo se llama?")
nombre = input()
print(f"Me alegro de conocerle, {nombre}")
```

```
¿Cómo se llama?
Pepe
Me alegro de conocerle, Pepe
```

Si se quiere que Python interprete la entrada como un número entero, se debe utilizar la función int() de la siguiente manera:

```
cantidad = int(input("Dígame una cantidad en pesos: "))
```

De la misma manera, para que Python interprete la entrada como un número decimal, se debe utilizar la función float() de la siguiente manera:

```
cantidad = float(input("Dígame una medida (hasta con 2 decimales): "))
```

La función print

La función *print()* admite un argumento, llamado **sep**, que contiene la cadena de caracteres que emplearemos para separar cada elemento. Si no se especifica, toma su valor por defecto, un espacio en blanco, sep=" "

```
>>> print("pepino","tomate",sep="_")
Pepino_tomate
```

Puedes indicar también cómo quieres que termine la impresión empleando un nuevo argumento opcional, **end**. Si no se especifica, toma su valor por defecto, que como hemos dicho, provoca un salto de línea, representando por el carácter de escape \n.

```
>>> print("pepino","tomate","lechuga",end="--")
pepino tomate lechuga--
```

Concatenar strings en Python usando el operador '+'

```
>>> nombre = 'Juan José'
>>> apellidos = 'Lozano Gómez'
>>> nombre_completo = nombre + apellidos
>>> nombre_completo
'Juan JoséLozano Gómez'
```

Si tratamos de concatenar, por ejemplo, un string con un int, el intérprete lanzará un error. Para ello puedes usar el método `str()`, que devuelve una representación de tipo `string` del objeto que se le pasa como parámetro.

```
>>> suma = 1 + 2
>>> suma
3
>>> res = 'El resultado de 1 + 2 es: ' + str(suma)
>>> res
'El resultado de 1 + 2 es: 3'
```

Formatear strings usando el método 'format'

Sustituyendo valores en función de la posición a través del marcador `{}`:

```
>>> var1 = 'J2logo'
>>> var2 = 'Hola'
>>> '{} {}, ¿cómo estás?'.format(var2, var1)
'Hola J2logo, ¿cómo estás?'
```

Pero también puedes especificar el nombre de las variables, de manera que el orden en el que se pasen los argumentos al llamar a format no importa:

```
>>> var1 = 'J2logo'
>>> var2 = 'Hola'
>>> '{var2} {var1}, ¿cómo estás?'.format(var1=var1, var2=var2)
'Hola J2logo, ¿cómo estás?'
```

O indicando el índice de los parámetros al invocar al método format:

```
>>> var1 = 'J2logo'
>>> var2 = 'Hola'
>>> '{1} {0}, ¿cómo estás?'.format(var1, var2)
'Hola J2logo, ¿cómo estás?'
```

Usando f-Strings en Python 3.6+

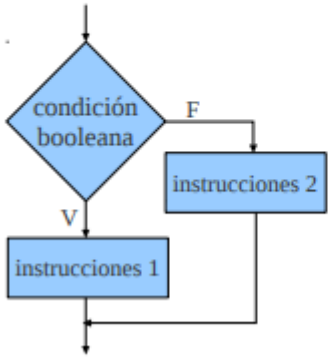
A partir de la versión 3.6 de Python, el lenguaje introdujo un nuevo modo de formatear strings que es más eficiente que todos los que hemos visto anteriormente. Son las cadenas literales formateadas o f-Strings.

```
nombre = 'J2logo'
f'Hola {nombre}'
'Hola J2logo'
```

```
valor1=int(input("Ingrese primer valor:"))
valor2=int(input("Ingrese segundo valor:"))
suma=valor1+valor2
print(f"La suma de {valor1} y {valor2} es {suma}")
```

El comando if then ... else ...

Sintaxis: if *condición-boolena*:
instrucciones 2
else
instrucciones 2



Semántica

Se evalúa la condición booleana
Si es Verdadera
se ejecutan las *instrucciones 1*
si es Falsa
se ejecutan las *instrucciones 2*

Sintaxis de la sentencia condicional if ... elif ... else ...

```
if condición_1:
    bloque 1
elif condición_2:
    bloque 2
else:
    bloque 3
```

Esta estructura es equivalente a la siguiente estructura de if ... else ... anidados:

```
if condición_1:
    bloque 1
else:
    if condición_2:
        bloque 2
    else:
        bloque 3
```

Operaciones aritméticas

- Resta: -
- `Suma: +`
- Multiplicación: *
- División entera:

  cociente: //

  resto (módulo): %

  (2 // 3 es 0, 2 % 3 es 2)
- División (para reales): / (2.0 / 3.0 es 0.66666)
- Exponente: ** (a ** b es a^b)

Expresiones booleanas

Constantes: True, False

Operadores:

- `and` - conjunción
- `or` - disyunción
- `not` – negación

Operadores relacionales

= = (igual)

! = (distinto)

< = (≤, menor o igual)

> = (≥, mayor o igual)

> (mayor)

< (menor)