

PYTHON

Los comentarios se inician con #
Todo lo que sigue en la misma
línea es comentario.

Un bloque de varias líneas de comentarios se encierra ente comillas triples (dobles o sencillas)

> """Esto es un bloque de comentarios

Visualiza en consola mensaje de bienvenida

3 print ("¡Hola mundo!")

72 caracteres por linea

Salto de línea es fin de instrucción. Aunque se pueden escribir varias instrucciones en la misma línea separadas por , o por ;

NO SE RECOMIENDA

Haciendo uso de \ se puede romper el código en varias líneas

$$x = 1 + 2 + 3 + 4 +$$

 $5 + 6 + 7 + 8$

Si estamos dentro de un bloque rodeado con paréntesis () , basta con saltar de línea para romper el código en varias líneas

$$x = (1 + 2 + 3 + 4 + 5 + 6 + 7 + 8)$$

Python es case-sensitive

Una función provoca un **efecto** y/o evalúa un valor y lo devuelve como **resultado**.

Una función se invoca por su **nombre** (en este caso print) seguido de un par de **paréntesis** de apertura y cierre.

Si la función tiene **argumentos**, estos van dentro de los paréntesis, separados por comas

Función print ()

PROGRAMACIÓN DE INTELIGENCIA ARTIFICIAL

PYTHON

Las funciones integradas, al contrario que las funciones definidas por el usuario, están siempre disponibles y no tienen que ser importadas. Consulta su lista completa en <u>Python Standard</u> Library.

La función print() es una función integrada:

- √ toma sus argumentos (cero o más)
- ✓ los convierte a un formato legible si es necesario
- ✓ envía los datos resultantes a la pantalla o ventana de consola

Envía al dispositivo de salida los parámetros
separados por espacio en blanco
y comenzando en nueva línea
print ("¡", "Hola", "mundo","!")

5

6 # sin argumentos imprime línea en blanco 7 print()

¡ Hola mundo!

>>> print("Mi nombre \n es \"Bond\"\nJames Bond")
Mi nombre
es "Bond"
James Bond

En las cadenas de texto de Python la **barra diagonal inversa** (\) anuncia que el siguiente carácter tiene un significado diferente:

\n -> carácter de nuevalínea \" -> carácter comilla doble

La función print admite los ARGUMENTOS DE PALABRA CLAVE sep y end.

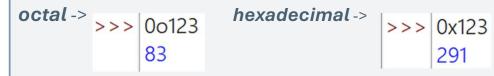
- sep especifica el carácter separador entre los argumentos emitidos. Por defecto es " "
- end especifica el carácter a imprimir al final de la instrucción. Por defecto es salto de línea

Los argumentos de palabra clave <u>SIEMPRE</u> han de ser los <u>últimos</u> argumentos.

```
print("H","o","|","a", sep="-", end="**")
H-o-l-a**
```

PYTHON

Python siempre escoge la representación más corta



bool -> True False
En contextos numéricos True=1 False=0

str -> "Cadena de caracteres"

TIPOS DE DATOS

> CASE-SENSITIVE

- El nombre de la variable se forma con letras, dígitos y carácter '_' (NO PUEDE empezar por digito)
- > Una variable CAMBIA DE TIPO según su contenido
- Las variables **SE CREAN CUANDO SE LES ASIGNA UN VALOR**

a = 3 -> asignación de valor a una variable

x = y = z = 0 -> todas las variables a 0

$$x, y, z = 1,2,3 \rightarrow x=1, y=2, z=3$$

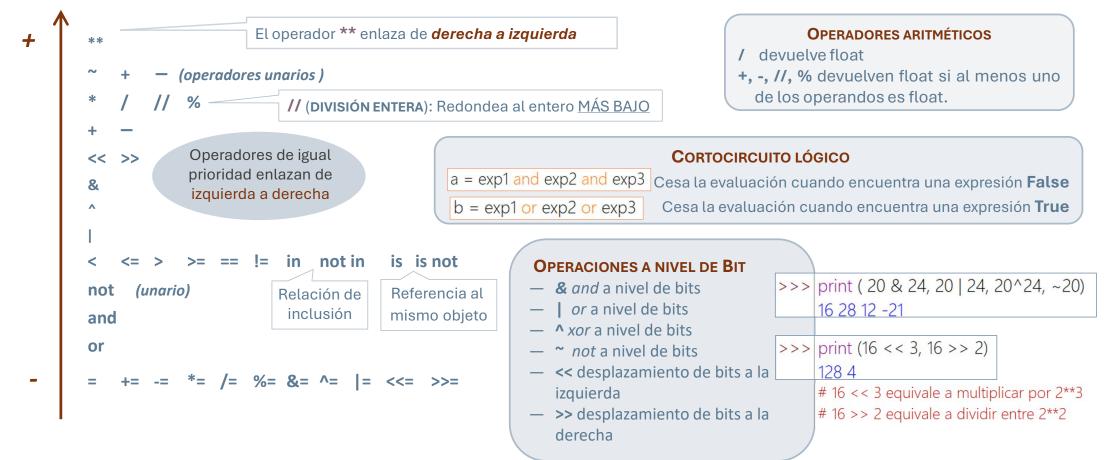
$$x, y, z = z, x, y \rightarrow x=3, y=1, z=2$$

VARIABLES

Prioridad de operadores

PROGRAMACIÓN DE INTELIGENCIA ARTIFICIAL

PYTHON



PYTHON

SENTENCIA if

Se ejecuta un bloque de código si la expresión booleana se evalúa como True o como False

```
if b != 0:

c = a/b

d = c + 1

print(d)
```

```
if a > b:
    print("a es mayor que b")
else:
    print("a no es mayor que b")
```

```
if nota >= 9:
    print('Sobresaliente')
elif nota >=7:
    print('Notable')
elif nota >= 5:
    print("Aprobado")
else:
    print('Suspenso')
```

Los bloques de código se marcan con la sangría

SENTENCIA while

- Se ejecuta el bloque de código mientras la expresión booleana se evalúe como True
- Si la condición se evalúa a False se ejecuta el bloque else

```
i = 2
i = 2
                              while i < 10:
while i < 10:
                                 i + = 3
  i + = 3
                                 if i == 8:
  if i == 8:
                                    break
     continue
                  11
                                 print(i)
   print(i)
                  False
else:
                              else:
   print("False")
                                 print("False")
```

```
while i < 10:
print(i)
i += 2
```

```
while i < 10:
    print(i)
    i += 2
else:
    print("Cond False")</pre>
```

continue: omite el resto del bloque y vuelve a evaluar la decisión

break: termina la ejecución del bucle (no entra en el else)

PYTHON

SENTENCIA for

La ejecución del bloque de código está condicionada por el valor de un iterable

range(inicio, fin, paso)

Genera una serie de valores:

- inicio (opcional): valor incluido en la salida.
 Si se omite, el valor inicial es 0
- fin: valor no incluido en la salida
- paso (opcional): incremento entre valores. Si se omite, se incrementa una unidad

```
for n in range(3,5):
    print(n, end="-")

for n in range(5):
    print(n, end="-")

for n in range(5,2,-1):
    print(n, end="-")

5-4-3-
```

```
for letra in "cadena":
print(letra, end="-")
c-a-d-e-n-a-
```

```
for letra in "cadena":

print(letra, end="-") c-a-d-e-n-a-
else:

print("\nEso es todo")
```

continue: omite el resto del bloque y vuelve a evaluar la decisión break: termina la ejecución del bucle (no entra en el else)

```
for letra in "cadena":

if letra == "e":

continue:

print(letra, end="-")

else:

print("\nEso es todo")

c-a-d-n-a-
Eso es todo
```

```
for letra in "cadena":

if letra == "e":

break

print(letra, end="-")

else:
print("\nEso es todo")
```