

PROGRAMACIÓN DE INTELIGENCIA ARTIFICIAL



PROGRAMACIÓN DE INTELIGENCIA ARTIFICIAL

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 entre comillas triples (dobles o sencillas)

```
"""Esto es un bloque  
de comentarios  
"""
```

```
1 # Visualiza en consola mensaje de bienvenida  
2  
3 print ("¡Hola mundo!")  
4
```

72 caracteres por línea

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

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 [Python Standard 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**

```
1 # Envía al dispositivo de salida los parámetros
2 # separados por espacio en blanco
3 # y comenzando en nueva línea
4 print ("¡", "Hola", "mundo", "!")
5
6 # sin argumentos imprime línea en blanco
7 print()
```

¡ Hola mundo !

```
>>> print("Mi nombre \n es \"Bond\" \n James 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 nueva línea**
\" -> **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 **SIEMPRE** han de ser los **últimos argumentos**.

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

PROGRAMACIÓN DE INTELIGENCIA ARTIFICIAL

PYTHON

int -> 43 -50

float -> 4.5 .5 (0.5) 4. (4.0) -3.15

NOTACIÓN CIENTÍFICA: *base E exponente*

$3E8 \rightarrow 3 \cdot 10^8$ $6.45E-34 \rightarrow 6.45 \cdot 10^{-34}$

Python siempre escoge la representación más corta

octal -> `>>> 0o123`
`83`

hexadecimal -> `>>> 0x123`
`291`

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 dígito**)

➤ 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 -> x=1, y= 2, z= 3

x, y, z = z, x, y -> x=3, y= 1, z= 2

VARIABLES

/ devuelve float
+, -, //, % devuelven float si al menos uno de los operandos es float.

CIRCUITO LÓGICO

evaluación cuando encuentra una expresión **False**

evaluación cuando encuentra una expresión **True**

- $\&$ *and* a nivel de bits
- $|$ *or* a nivel de bits
- \wedge *xor* a nivel de bits
- \sim *not* a nivel de bits
- \ll desplazamiento de bits a la izquierda
- \gg desplazamiento de bits a la derecha

```
>>> print ( 20 & 24, 20 | 24, 20^24, ~20)
16 28 12 -21
```

```
>>> print (16 << 3, 16 >> 2)
128 4
```

- # 16 << 3 equivale a multiplicar por 2**3
- # 16 >> 2 equivale a dividir entre 2**2

PROGRAMACIÓN DE INTELIGENCIA ARTIFICIAL

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 evalúa como **True**
- Si la condición se evalúa a **False** se ejecuta el **bloque else**

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

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

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

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

5
11
False

5

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

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

PROGRAMACIÓN DE INTELIGENCIA ARTIFICIAL

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="-")
```

3-4-

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

0-1-2-3-4-

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

c-a-d-e-n-a-
Eso 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")
```

c-a-d-