

Bootcamp Inteligencia Artificial

Nivel Explorador

Semana 2: Python Para Inteligencia Artificial


Aritmética Cadenas Input

Agenda

1. Operadores en Python
2. Matemáticas en Python
3. Ejercicios matemáticos
4. Cadenas

1.1 Operadores en Python

Los operadores se utilizan para realizar operaciones con variables y valores. En el siguiente ejemplo, utilizamos el operador + para sumar dos valores:



```
print(10 + 5)
```

15

1.2 Operadores en Python

Python divide los operadores en los siguientes grupos:

- Operadores aritméticos
- Operadores de asignación
- Operadores de comparación
- Operadores lógicos
- Operadores de identidad
- Operadores de pertenencia
- Operadores a nivel de bits

1.3 Operadores aritméticos en Python

Operator	Name	Example
+	Addition	$x + y$
-	Subtraction	$x - y$
*	Multiplication	$x * y$
/	Division	x / y
%	Modulus	$x \% y$
**	Exponentiation	$x ** y$
//	Floor division	$x // y$

1.3 Operadores aritméticos en Python

Ejemplos



```
x = 5
y = 3
print(x + y)
print(x - y)
print(x * y)
print(x / y)
print(x % y)
print(x ** y)
print(x // y)
```

```
8
2
15
1.6666666666666667
2
125
1
```

1.4 Operadores de asignación en Python

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
=	x = 3	x = x 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

1.5 Operadores de asignación en Python

```
▶ x = 5
  print(x)
```

☞ 5

```
▶ x += 3
  print(x)
```

☞ 8

```
▶ x -= 3
  print(x)
```

5

```
▶ x *= 3
  print(x)
```

15

```
▶ x /= 3
  print(x)
```

5.0

```
▶ x %= 3
  print(x)
```

☞ 2.0

```
▶ x //= 3
  print(x)
```

0.0

```
▶ x //= 3
  print(x)
```

0.0

```
▶ x = 5
  x &= 3
  print(x)
```

1

1.6 Operadores de asignación en Python

```

▶ x = 5
  x |= 3
  print(x)

```

7

```

▶ x = 5
  x ^= 3
  print(x)

```

↪ 6

```

▶ x = 5
  x >>= 3
  print(x)

```

0

```

▶ x = 5
  x <<= 3
  print(x)

```

40

1.7 Operadores de comparación en Python

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

1.7 Operadores de comparación en Python



```
x = 5  
y = 3  
print(x == y)  
print(x != y)  
print(x > y)  
print(x < y)  
print(x >= y)  
print(x <= y)
```

☞ False
True
True
False
True
False

1.8 Operadores lógicos en Python

Operator	Description	Example
and	Returns True if both statements are true	<code>x < 5 and x < 10</code>
or	Returns True if one of the statements is true	<code>x < 5 or x < 4</code>
not	Reverse the result, returns False if the result is true	<code>not(x < 5 and x < 10)</code>

1.9 Operadores lógicos en Python

```
▶ x = 5  
print(x > 3 and x < 10)
```

True

```
▶ x = 5  
print(x > 3 or x < 4)
```

True

```
▶ x = 5  
print(not(x > 3 and x < 10))
```

False

1.10 Operadores de identidad en Python

Los operadores de identidad se utilizan para comparar los objetos, no si son iguales, sino si son realmente el mismo objeto, con la misma ubicación de memoria.

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

1.11 Operadores de identidad en Python



```
x = ["manzana", "limon"]  
y = ["manzana", "limon"]  
z = x  
print(x is z)  
print(x is y)  
print(x == y)
```



```
True  
False  
True
```

1.12 Operadores de pertenencia en Python

Los operadores de pertenencia se utilizan para comprobar si una secuencia se presenta en un objeto:

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

1.13 Operadores de pertenencia en Python



```
x = ["manzana", "limon"]  
print("manzana" in x)
```

True



```
x = ["manzana", "limon"]  
print("pera" not in x)
```

True

1.14 Operadores Bitwise en Python

Los operadores Bitwise se utilizan para comparar números (binarios)

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits

2.1 Matemáticas de Python

Python cuenta con un conjunto de funciones matemáticas incorporadas, incluyendo un extenso módulo matemático, que permite realizar tareas matemáticas con números.

Las funciones `min()` y `max()` se pueden utilizar para encontrar el valor más bajo o más alto de un iterable



```
x = min(5, 10, 25)
y = max(5, 10, 25)
print(x)
print(y)
```

```
5
25
```

2.2 Matemáticas de Python

La función `abs()` devuelve el valor absoluto (positivo) del número especificado:

```
▶ x = abs(-7.25)  
print(x)
```

7.25

La función `pow(x, y)` devuelve el valor de `x` a la potencia de `y`

```
▶ x = pow(4, 3)  
print(x)
```

64

2.3 El módulo Math

Python tiene también un módulo incorporado llamado math, que amplía la lista de funciones matemáticas.

Para utilizarlo, debes importar el módulo math.

Cuando hayas importado el módulo de matemáticas, puedes empezar a utilizar los métodos y constantes del módulo.

El método `math.sqrt()`, por ejemplo, devuelve la raíz cuadrada de un número



```
import math  
x = math.sqrt(64)  
print(x)
```

8.0

2.4 El módulo Math

El método `math.ceil()` redondea un número hacia arriba a su número entero más cercano, y el método `math.floor()` redondea un número hacia abajo a su número entero más cercano, y devuelve el resultado.



```
import math
x = math.ceil(1.4)
y = math.floor(1.4)
print(x)
print(y)
```

2
1

2.5 El módulo Math

La constante `math.pi`, devuelve el valor de PI (3,14...)



```
import math  
x = math.pi  
print(x)
```

```
3.141592653589793
```

Para ver los metodos del módulo `math`, se puede dirigir al siguiente enlace:

[Enlace](#)

3.1 Ejercicios matemáticos

Traducir las siguientes expresiones matemáticas a Python y evaluarlas.
Trata de utilizar el menor número de paréntesis posible.

a) $2 + (3 \cdot (6/2))$

c) $(4/2)^5$

e) $(-3)^2$

b) $\frac{4 + 6}{2 + 3}$

d) $(4/2)^{5+1}$

f) $-(3^2)$

(Nota: El resultado de evaluar cada expresión es: a) 11; b) 2; c) 32; d) 64; e) 9; f) -9.)

3.2 Ejercicios matemáticas

¿Qué resultados se obtendrán al evaluar las siguientes expresiones Python?

a) $2 + 3 + 1 + 2$

c) $(2 + 3) * 1 + 2$

e) $+- - -6$

b) $2 + 3 * 1 + 2$

d) $(2 + 3) * (1 + 2)$

f) $-+ - +6$

¿Qué resultará de evaluar las siguientes expresiones? Presta especial atención al tipo de datos que resulta de cada operación individual.

a) $1 / 2 / 4.0$

g) $4.0 ** (1 / 2) + 1 / 2$

b) $1 / 2.0 / 4.0$

h) $4.0 ** (1.0 / 2) + 1 / 2.0$

c) $1 / 2.0 / 4$

i) $3e3 / 10$

d) $1.0 / 2 / 4$

j) $10 / 5e-3$

e) $4 ** .5$

k) $10 / 5e-3 + 1$

f) $4.0 ** (1 / 2)$

l) $3 / 2 + 1$

4.1 Cadenas

- Las cadenas (o strings) son un tipo de datos compuestos por secuencias de caracteres que representan texto. Estas cadenas de texto son de tipo str y se delimitan mediante el uso de comillas simples o dobles.
- En el caso que queramos usar comillas (o un apóstrofo) dentro de una cadena tenemos distintas opciones. La más simple es encerrar nuestra cadena mediante un tipo de comillas (simples o dobles) y usar el otro tipo dentro de la cadena. Otra opción es usar en todo momento el mismo tipo de comillas, pero usando la barra invertida (\) como carácter de escape en las comillas del interior de la cadena para indicar que esos caracteres forman parte de la cadena.

4.2 Cadenas

El intérprete de Python tiene incorporadas distintas funciones relacionadas con las cadenas. Una de ellas es la función `len()`, que nos indica el número de elementos de un objeto. En el caso de que el objeto sea una cadena nos indica el número de caracteres que la componen.

Cada uno de los caracteres de una cadena (incluidos los espacios) tiene asignado un índice. Este índice nos permite seleccionar su carácter asociado haciendo referencia a él entre corchetes (`[]`) en el nombre de la variable que almacena la cadena. Si consideremos el orden de izquierda a derecha, el índice empieza en 0 para el primer carácter, etc. También se puede considerar el orden de derecha a izquierda, en cuyo caso al último carácter le corresponde el índice -1, al penúltimo -2 y así sucesivamente

4.3 Operaciones con String

Una operación que podemos realizar es la concatenación que consiste en unir distintas cadenas mediante el uso del signo más (+).

También podemos concatenar con el signo de multiplicación (*), que en este caso significa adjuntarle un determinado número de copias a la cadena.

4.3 Operaciones con String

➤ Comparación:

- Se usan los operadores convencionales (<, <=, >, >=, ==, !=) para comparar cadenas usando el orden lexicográfico.
- En el orden lexicográfico, se comparan de izquierda a derecha uno a uno los caracteres, mientras sean iguales.
- En el caso que no sean iguales, si el carácter de la primera cadena es menor que el de la segunda a la primer cadena se le considera menor, pero si es mayor, a la primer cadena se le considera mayor.
- Si todos los caracteres son iguales, se considera que las cadenas son iguales

4.4 ASCII

- ❖ Código Estadounidense Estándar para el Intercambio de Información (American Standard Code for Information Interchange)
- En su versión original usa 7 bits, definiendo 128 caracteres.
- En la versión extendida usa 8 bits (esto es 1 byte), definiendo 256 caracteres.
- Es la base de los archivos de texto plano (o sin formato).
- Es el esquema base para la escritura de programas en casi todos los
- lenguajes de programación (incluido Python).

4.5 UNICODE

Esquema de codificación cuyo objetivo es dar a cada carácter usado por cada uno de los lenguajes humanos su propio código, es decir, permitir la “internacionalización” de la computación.

- **UTF – 8:** definido por ocho (8) bits (un byte). Toma como base el ASCII, ANSI de Windows y el ISO – 8859 – 1. Muy usado en HTML.
- **UTF – 16:** definido por 16 bits (2 bytes). Usa una representación de longitud variable que permite su optimización en procesos de codificación a texto (usando un subconjunto de (ASCII o UTF – 8).
- **UTF-32:** definido por 32 bits (4 bytes). Es el m´as simple pues usa una representación de longitud fija

4.6 Métodos de String

- Longitud (len): longitud de una cadena
- Subcadena (slice): obtiene una cadena de otra cadena
- Contando (count): cuenta cuantos caracteres hay en una cadena
- Buscando (find, rfind): obtiene la primera y ultima ocurrencia en una cadena
- Mayúscula y minúsculas: s.lower() s.upper()
- Removiendo caracteres (strip, lstrip, rstrip)
- Dividiendo cadenas (split)
- Remplazando (replace)

4.7 Ejercicios con Strings

- Realizar un programa que comprueba si una cadena leída por teclado comienza por una subcadena introducida por teclado.
- Pide una cadena y un carácter por teclado (valida que sea un carácter) y muestra cuantas veces aparece el carácter en la cadena.
- Suponiendo que hemos introducido una cadena por teclado que representa una frase (palabras separadas por espacios), realiza un programa que cuente cuantas palabras tiene.
- Pide una cadena y dos caracteres por teclado (valida que sea un carácter), sustituye la aparición del primer carácter en la cadena por el segundo carácter.

Preguntas

