PYTHON INICIAL

Clase 2 – Variables y operaciones

Variable

En los lenguajes de programación, una **variable** es un nombre que se refiere a un objeto que reside en la memoria.

En Python las variables no deben declararse previamente, se crean directamente mediante una sentencia de asignación.

a = 20

b = "una cadena de texto"

Python es un lenguaje de **tipos dinámicos**. Esto quiere decir que el tipo no está asociado a la variable, sino a su valor. Por lo tanto, el tipo de la variable no debe declararse, sino que se adapta dinámicamente al tipo del objeto que se le asigna.

A = 440

A = "la4"

La primera sentencia crea la variable A de tipo int con el valor 440, y la segunda sentencia redefine la variable con el tipo str y el valor "la4".

En Python, el nombre de una variable (*identificador*) debe cumplir ciertas condiciones:

- puede contener solamente letras (mayúsculas y minúsculas), números, y guión bajo; todas las letras deben pertenecer al código de caracteres ASCII básico, y no contener caracteres extendidos
- el primer carácter no puede ser un número
- las **palabras reservadas** del lenguaje no pueden usarse como identificadores:

PYTHON INICIAL

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Nombrando variables

Si bien cualquier nombre de variable que respete las reglas mencionadas arriba, va a ser válido y reconocido por el lenguaje, conviene observar algunas recomendaciones y convenciones.

- conviene utilizar nombres que describan el contenido de la variable, evitando nombres demasiado largos; hay varios estilos para nombrar variables, en Python el más usado es el llamado snake_case.
- los nombres que comienzan con guión bajo (simple o doble, _ o __) se reservan para variables con significado especial

1. Tipos de datos básicos en Python

Los tipos de datos fundamentales en Python incluyen:

- **Numéricos**:
  - int: enteros (por ejemplo, 10)
  - float: números decimales (por ejemplo, 3.14)
  - complex: números complejos (por ejemplo, 2 + 3j)
- **Texto**:
  - str: cadenas de texto (por ejemplo, "Hola")

Además, Python es dinámicamente tipado; no necesitas declarar el tipo — se asigna automáticamente al asignar un valor a una variable.

PYTHON INICIAL

Para consultar el tipo de una variable en tiempo de ejecución, puedes utilizar la función type()

Ejemplo

Imprima el tipo de datos de la variable x:

```
x = 5
print(type(x))
```

Enteros

El tipo **int** representa números enteros positivos y negativos, de precisión ilimitada (limitada solo por el tamaño de la memoria disponible, no por el lenguaje).

7

-728

2387672347650910239486849367819823783745

Coma flotante

El tipo **float** representa números decimales, también positivos y negativos de precisión ilimitada.

2.238476

-3.173982340234958656580

1.

.1

Aritmética en Python

Python tiene integrados todos los operadores aritméticos básicos.

Mediante módulos externos, se pueden incorporar operadores adicionales.

operadores aritméticos

- suma: +
- resta: -
- multiplicación: *
- división: /

Siempre devuelve un número de coma flotante.

PYTHON INICIAL

12/3

14/5

-14/5

- división entera: ***//***

Siempre devuelve un entero.

14//5

-14//5

-14//-5

- módulo: **%**

Devuelve el resto de la división entera.

14%5

-14%5

-14%-5

- potencia: **

2**10

Orden de precedencia

Cuando se escribe más de una operación sucesiva, las mismas se realizan siguiendo este orden de precedencia:

- potencia: **
- negación: **-**
- multiplicación, división, división entera, módulo: ***, /, //, %**
- suma, resta: **+**, **-**

Si se quiere cambiar el orden de precedencia, se pueden encerrar operaciones entre paréntesis, y serán evaluadas primero.

3+4*2

(3+4)*2

-2**2

(-2)**2

PYTHON INICIAL

2**1/12

2**(1/12)

Ejemplo: calculadora simple

```
a = int(input("Primer número: "))
b = int(input("Segundo número: "))
suma = a + b
print("La suma es", suma)
```

Calculadora de edad: escribir un código Python donde el usuario ingrese su año de nacimiento (int), y el programa calcule su edad actual (int) usando operaciones aritméticas simples.

```
año_nacimiento = int(input("Ingresa tu año de nacimiento: "))
año_actual = 2025
edad = año_actual - año_nacimiento
print("Tu edad es:", edad)
```

PYTHON INICIAL

Cadenas de caracteres

El tipo **string** es una **secuencia** de **caracteres**.

Cada carácter se representa internamente como un número. Qué carácter representa cada número, está determinado por la **codificación**. Python utiliza la codificación Unicode.

Una cadena o string se crea encerrando uno o más caracteres entre comillas, simples o dobles.

'las cadenas de caracteres se pueden definir con comillas simples...'

"...o con comillas dobles"

"'con tres comillas (simples o dobles)
se pueden definir cadenas de varios renglones"'

Las cadenas de caracteres pueden contener dígitos, pero siguen siendo de un tipo distinto.

'500'

Cuando dentro de la cadena se utiliza la misma comilla de delimitación, la misma tiene que ser antecedida por el carácter de escape .

"esta cadena utiliza la misma comilla de delimitación \" "

'esta cadena utiliza la misma comilla de delimitación \' '

Cadenas de escape

Son caracteres o combinaciones de caracteres que tienen un significado especial en las cadenas. Si se quiere utilizarlas literalmente, deben ser antecedidas del carácter de escape.

- **\n** : salto de línea
- **\t** : tabulador
- \\ : retrobarra
- **\'** : comilla simple
- **\"** : comilla doble

Operaciones con cadenas

- suma: **+**

Equivale a la concatenación de cadenas. Sólo es válida entre dos o más cadenas.

La propiedad de que un mismo operador (**+**) tenga diferentes significados según el contexto, se llama **polimorfismo**.

PYTHON INICIAL

'foo'+'bar'

'foo'+'bar'+'vaz'

'500'+'300'

'500'+300

- multiplicación: *

Equivale a la repetición. Sólo es válida entre una cadena y un entero.

3*"ta"

'5'*8

'foo'*2.5

'foo'*'bar'

- concatenación

"foo" "bar"