¡Hola, Python!

Diego Morales

2024-03-19

Tabla de contenidos

Prefacio

Bienvenido a mi curso de programación introductoria a Python.

He creado este libro con el objetivo de compartir mis conocimientos con todo el mundo de la manera más efectiva posible.

And what if instead of it being in the right hands, it was in everyone's hands? -Steve Jobs [Film] (2015)

1 Introducción

En este capítulo creamos nuestro primer código de programación en Python, el cual mostrará un saludo en pantalla al ejecutar el programa.

1.1 Mi primer programa

La función `print` permite *desplegar mensajes* en la pantalla. Escribe el siguiente código teniendo cuidado de escribir correctamente el nombre de la función, luego, dentro de paréntesis deberás colocar tu mensaje utilizando comillas simples, `'`, o dobles, `"`, al inicio y al final.

```
print('Hello, world!')
```

```
Hello, world!
```

Si Python interpretó correctamente tu instrucción, entonces, deberías ver el mensaje anterior en la salida.

> **Nota**
>
> Es una práctica común crear el código para desplegar un mensaje de `Hello, world!` cuando se aprende un lenguaje nuevo de programación para comprobar que todo se encuentra configurado correctamente.

2 Variables

Una variable es un espacio de memoria en el cual podemos almacenar valores.

Por ejemplo, en el siguiente código. Se asigna el valor `4` a una variable llamada `x`.

```
x = 4
```

2.1 Características

Las variables poseen las siguientes características: 1. Puede almacenar distintos tipos de datos. 2. Su valor se puede modificar. 3. Se referencian por un nombre único (idealmente, significativo).

Observaciones sobre nombres de variables: 1. Python diferencia entre mayúsculas y minúsculas. 2. No pueden iniciar con números. 3. No pueden contener espacios. 4. No se pueden utilizar **palabras reservadas** (más adelante aprenderemos qué significan).

```
a = 100
print(a)
```

```
100
```

Si asignamos un nuevo valor a la misma variable, el valor anterior es reemplazado.

```
a = 100
a = 200
print(a)
```

```
200
```

Si se desean crear variables utilizando nombres significativos conformados por varias palabras, se recomienda utilizar la "notación de camello" (*camelCase*):

```
correoElectronico = 'juan_lopez2020@gmail.com'
```

También pueden separarse las palabras utilizando **guion bajo**, sin embargo, este tipo de notación suele utilizarse para funciones.

```
correo_electronico = 'juan_lopez2020@gmail.com'
```

3 Tipos de datos

Existen una gran cantidad de tipos de datos en Python: *Texto, numéricos, secuencia, mapeos, conjuntos, booleanos, binarios.*

3.1 Entero (int)

Dato numérico utilizado para representar números enteros.

```
cantidadAlumnos = 256
print(cantidadAlumnos)
```

```
256
```

3.2 Flotante (float)

Dato numérico utilizado para representar números reales, es decir, números con cifras decimales.

```
alturaMetros = 1.75
print(alturaMetros)
```

```
1.75
```

3.3 Texto (str)

Cadena de caracteres: letras, números y símbolos. El valor **se debe colocar dentro de comillas simples o dobles** para indicar que es un `str`.

```
movie = "The Lord of the Rings: The Return of the King (2003)"
cantidadAlumnos2 = "256"
alturaMetros2 = "1.75"
print(movie)
print(cantidadAlumnos2)
print(alturaMetros2)
```

```
The Lord of the Rings: The Return of the King (2003)
256
1.75
```

> **Advertencia**
>
> Se debe tener especial cuidado al manejar números representados como cadenas de caracteres ya que Python no los interpretará como valores numéricos (`int` o `float`).

3.4 Función type

La función `type()` se puede utilizar para conocer el tipo de dato de una variable.

```
print(type(cantidadAlumnos))
print(type(cantidadAlumnos2))
```

```
<class 'int'>
<class 'str'>
```

3.5 Función input

La función `input()` se utiliza para solicitar al usuario una entrada. El programa se detiene, hasta que el usuario presiona la tecla **Enter** en su teclado (luego de escribir el ingreso). Esta función, retorna una cadena de caracteres (`str`) con el ingreso del usuario.

```
nom = input("¿Cuál es su nombre? ")
```

Por lo tanto, puedes utilizar esta función para asignar valores a variables al iniciar la ejecución de tu programa.

Ahora veamos qué sucede si solicitamos al usuario un número que utilizaremos para realizar una operación más adelante.

```
age = input("Ingrese su edad: ")
```

Esa línea de código sería equivalente a que tomemos el valor devuelto por la función `input` y lo asignemos en la variable `age`. Sería igual al siguiente código.

```
age = '28'
birthYear = 2024 - age
print("Usted nació en el año ", birthYear)
```

Python nos muestra un mensaje de error (TypeError) debido a que la operación de resta (`-`) no está soportada entre variables numéricas (`int` o `float`) y cadenas de caracteres (`str`).

> **!** Importante
>
> Es completamente normal (y esperado) obtener errores al escribir código, incluso para programadores experimentados. Suelen aparecer cuando Python no es capaz de interpretar una instrucción (SintaxError) o porque se está realizando una acción inválida (como la anterior). Lo importante es comprender el error y corregirlo, lo cual será cada vez más sencillo con la práctica.

En este caso, podríamos realizar una conversión de datos para que la variable `age` se interprete como un entero.

```
age = '28'
age = int(age) # conversion a entero (casting)
birthYear = 2024 - age
print("Usted nació en el año ", birthYear)
```

```
Usted nació en el año  1996
```

3.6 Conversión de datos

Para convertir entre distintos tipos de datos, simplemente debemos especificar el tipo de dato y colocar el valor dentro de paréntesis.

1. **int()**
2. **float()**
3. **str()**

Por ejemplo, si queremos convertir una variable **`x = "99"`** a un valor numérico.

```
x = "99"
print(type(x))
x = int(x)
print(type(x))
```

```
<class 'str'>
<class 'int'>
```