Elementos básicos del lenguaje

Veamos en este documento algunos elementos básicos del lenguaje para empezar a hacer nuestros primeros programas. Veremos qué tipos de datos maneja Python, cómo declarar variables para almacenar información, y cómo hacer algunas operaciones aritméticas básicas. También veremos cómo mostrar información por pantalla y pedírsela al usuario por teclado.

1. Comentarios

Los comentarios en Python pueden ser de una sola línea o de varias, como en otros lenguajes. Para los comentarios de una sola línea, debemos comenzarlos con el símbolo de la almohadilla `#`. Por ejemplo:

```
# Mostramos un mensaje
print ("Hola mundo")
```

También podemos definir comentarios de múltiples líneas. En este caso, comienzan y terminan por una triple comilla simple `'''`:

```
'''
Esto es un comentario de varias líneas
Segunda línea del comentario
'''
print ("Hola mundo")
```

2. Tipos de datos básicos y variables

Existen tres tipos de datos básicos en Python:

- **Números**: pueden ser enteros (3), reales (4.33) o incluso complejos (5 + 3j)
- **Textos**: Python no dispone del tipo carácter (*char*) que sí existe en otros lenguajes, así que para los textos podemos emplear indistintamente comillas dobles (`"Hola mundo"`) o simples (`'Hola mundo'`). También podemos emplear secuencias de escape habituales en otros lenguajes de programación, como el salto de línea `\n` o la tabulación `\t` (`"Hola\tmundo"`)
- **Booleanos**: pueden ser `True` o `False`

2.1. Declaración e inicialización de variables

Debido a que Python es un lenguaje débilmente tipado (es decir, no es necesario definir el tipo de dato que vamos a almacenar en una variable), simplemente declaramos las variables y les asignamos un valor. Dependiendo del valor asignado, la variable toma el tipo de dato adecuado. Por ejemplo:

```
edad = 23
mensaje = "Hola mundo"
```

Python es un lenguaje que ofrece particularidades que no muchos lenguajes tienen. Por ejemplo, si queremos intercambiar el valor de dos variables, en la mayoría de lenguajes necesitaríamos una tercera variable auxiliar que almacene temporalmente uno de los dos datos a intercambiar. Así nos quedaría el código en Java, por ejemplo, para intercambiar los valores de *n1* y *n2*:

```
int n1 = 3, n2 = 8;
...
int aux = n1;
n1 = n2;
n2 = aux;
```

Sin embargo, en Python es tan sencillo como hacer esto:

```
n1 = 3
n2 = 8

n1, n2 = n2, n1
```

2.2. Conversión de datos

Si queremos convertir de un tipo básico a otro, podemos emplear estas instrucciones útiles:

- `int(valor)` convierte a entero el valor indicado, que puede ser un número real o un texto, por ejemplo
- `float(valor)` funciona como `int`, pero convierte a número real el valor indicado.
- `str(valor)` obtiene una representación textual del valor indicado

```
texto = "23"
numero = int(texto) # 23
```

3. Operadores

Veremos a continuación de forma resumida los principales operadores que ofrece el lenguaje Python

3.1. Operadores aritméticos

Tenemos estos operadores aritméticos disponibles en Python:

Operador	Descripción
`+`	Suma
`-`	Resta
`*`	Multiplicación
`**`	Potencia
`/`	División
`//`	División entera
`%`	Resto o módulo

3.2. Operadores relacionales

A la hora de comparar dos elementos tenemos los típicos operadores de cualquier lenguaje: `==`, `!=`, `<`, `<=`, `>`, `>=`

3.3. Operadores lógicos

Los operadores lógicos que ofrece Python para poder enlazar comprobaciones simples y formar otras más complejas son:

- `and` (Y lógica, que en muchos lenguajes se representa con `&&`)
- `or` (O lógica, que en muchos lenguajes se representa con doble barra vertical `||`)
- `not` (negación lógica, que en muchos lenguajes se representa con `!`)

Por ejemplo:

```
3 > 2 and 5 > 1
```

4. Entrada/Salida básica

Veamos ahora qué mecanismos ofrece Python para sacar información por pantalla y pedirle datos al usuario

4.1. Salida con *print*

Hemos visto que la instrucción `print` puede utilizarse para mostrar mensajes por pantalla. Podemos mostrar un mensaje simple...

```
print ("Hola mundo")
```

... o combinar varias partes de un mensaje, separadas por comas (Python añade un espacio entre cada parte):

```
print ("La suma de", num1, "y", num2, "es", resultado)
```

También podemos proporcionar algún formato de salida, de forma similar a como funciona la instrucción *printf* en lenguajes como C o Java. Por ejemplo, de este modo definimos un hueco o espacio horizontal para un número (3 espacios de hueco)

```
print ("Mi nombre es Nacho y tengo %3d años" % (edad))
```

O podemos especificar cuántos dígitos enteros y decimales mostrar en una expresión o variable real. En el siguiente ejemplo mostramos el número con 3 dígitos enteros (si no hay se rellenan con espacios) y 2 decimales (si no hay se rellenan con ceros).

```
print ("El resultado final es %3.2f" % (resultado))
```

Si necesitamos mostrar más de un dato, podemos especificarlo entre paréntesis, separados por comas:

```
print ("La suma de %d y %d es %5d" % (num1, num2, resultado))
```

Como ocurre con C o Java, podemos emplear los símbolos especiales `%d`, `%f` or `%s` para representar datos enteros, reales o textos en la expresión a mostrar con *print*, respectivamente.

Alternativamente, podemos anteponer una `f` al texto que queremos mostrar, y eso nos va a permitir intercalar variables dentro, encerradas entre llaves. En el caso de que alguna de las variables necesite un formato específico (más o menos cifras enteras o decimales) lo especificamos con dos puntos, y una sintaxis similar a la anterior.

```
print(f'La suma de {num1} y {num2} es {resultado:5d}')
```

Debemos tener en cuenta, no obstante, que por defecto `print` salta a la siguiente línea después de mostrar la información. Si no queremos que sea así, necesitamos añadir un parámetro llamado `end` en la instrucción, indicando con qué carácter queremos terminar. Por ejemplo, de este modo finalizamos con una cadena vacía, para no pasar a la siguiente línea:

```
print ("Hola", end="")
print ("Adiós")
#Muestra "HolaAdiós"
```

4.2. Obtener la entrada del usuario

Para obtener la entrada del usuario por teclado, empleamos la instrucción `input`. Esta instrucción recoge todo lo que escribe el usuario hasta que pulse *Intro* o *Enter*, y lo obtiene como información textual. Si le hemos pedido un número entero o real, por ejemplo, deberemos convertirlo al dato correspondiente usando las instrucciones `int` o `float` vistas antes:

```
print("Escribe un número entero:")
numero = int(input())
```

Alternativamente, la propia instrucción `input` también admite entre paréntesis el texto que le queremos mostrar al usuario para explicarle lo que tiene que hacer. Así, el código anterior puede abreviarse de este modo:

```
numero = int(input("Escribe un número entero:"))
```

Ejercicio 1:

Crea un programa llamado `Porcentajes.py` que le pregunte al usuario cuántos chicos y chicas hay en su clase, y calcule el porcentaje de chicos y chicas. Aquí tienes un ejemplo de cómo debería funcionar:

```
Escribe el número de chicos:
12
Escribe el número de chicas:
8
Hay un 60% de chicos y un 40% de chicas
```

PISTA: para sacar el símbolo del porcentaje en un texto, debemos duplicarlo `%%`.

AYUDA: vídeo con la solución del ejercicio

Ejercicio 2:

Crea un programa llamado `Saludo.py` que le pregunte al usuario su nombre y edad, y muestre un mensaje de saludo personalizado, de este modo:

```
Hola Nacho, tienes 43 años
```

Ejercicio 3:

Crea un programa llamado `Media.py` que le pida al usuario 4 números enteros y calcule su media (real). La media debe mostrarse en pantalla con 3 cifras decimales.

Ejercicio 4:

Descarga este archivo, súbelo a tu cuenta de Google Colab (menú *Archivo* > *Subir cuaderno* desde la aplicación de Colab) y resuelve el ejercicio 1 de nuevo, esta vez desde Colab.