Cadenas de texto

Ya hemos comentado en documentos anteriores que Python permite manejar cadenas de texto utilizando indistintamente comillas simples o dobles. Podemos crearlas con un texto predefinido en el programa, o pedirlas al usuario a través de la instrucción `input`:

```
texto = "Hola"
texto2 = 'Buenas'
texto3 = input("Dime tu nombre: ")
```

1. Algunas operaciones básicas

Además, podemos hacer algunas operaciones básicas sobre las cadenas de texto, tales como concatenar con `+`:

```
texto = "Tengo " + edad + " años"
```

Podemos convertir también cualquier dato simple a cadena usando la instrucción `str`:

```
texto = "Tengo " + str(edad) + " años"
```

Podemos repetir un texto un número determinado de veces, usando el operador `*`:

```
texto = "Hola" * 3      #HolaHolaHola
```

Finalmente, podemos acceder a cada uno de los caracteres de la cadena usando los corchetes (desde la posición 0) y podemos determinar el tamaño de la cadena con la instrucción `len(cadena)`:

```
texto = "Hola";
print(texto[0])     # H
print(len(texto))   # 4
```

2. Otras operaciones sobre cadenas

Además de estas operaciones básicas, también podemos utilizar algunas opciones que también ofrecen otros lenguajes. Por ejemplo, tenemos la instrucción `split` que devuelve una lista con los elementos que ha podido **extraer de una cadena**, a partir de un delimitador especificado:

```
partes = texto.split(",")
```

El paso inverso, es decir, **unir** partes de un texto con un separador, se puede hacer a través de la instrucción `join`

```
partes = ["Uno", "Dos", "Tres"]
texto = ','.join(partes) # Uno,Dos,Tres
```

La instrucción `replace` **reemplaza** todas las ocurrencias de un texto (primer dato) por otro texto (segundo dato), devolviendo una cadena con el resultado final:

```
texto = "Java es el mejor lenguaje"
texto2 = texto.replace("Java", "Python") # Python es el mejor lenguaje
```

Las instrucciones `lower` y `upper` convierten todo el texto en **minúsculas / mayúsculas**, respectivamente, devolviendo una cadena con el resultado:

```
texto = "Hola, buenas"
textoMayus = texto.upper() # HOLA, BUENAS
```

La instrucción `find` permite **buscar** si un texto está contenido dentro de otro, y en qué posición aparece por primera vez. Obtendrá un índice negativo si el texto no se encuentra. Alternativamente, podemos utilizar la expresión `in` para ver si un texto está dentro de otro, sin importar la posición.

```
texto = "Hola, buenas"
posicion = texto.find("buenas")
if posicion >= 0:
    print("Texto encontrado en posición", posicion)
if "buenas" in texto:
    print("El texto contiene \"buenas\"")
```

Si queremos **extraer una subcadena** a partir de una cadena principal, debemos indicar entre corchetes el índice a partir del cual queremos empezar a cortar (inclusive), y el índice donde queremos terminar de cortar

(exclusive), separados por dos puntos.

```
texto = "Hola, buenas"
subcadena = texto[6, 12]  # buenas
```

Estos índices pueden tomarse desde el inicio de la cadena (valores positivos) o desde el final (valores negativos). Estas dos instrucciones son equivalentes para la cadena de entrada indicada:

```
texto = "Hola, buenas tardes"
subcadena = texto[6:12]  # buenas
subcadena2 = texto[6:-7] # buenas
```

A la hora de **comparar** cadenas para ver cuál es mayor o menor alfabéticamente, podemos emplear los operadores de comparación.

```
texto1 = "Hola"
texto2 = "buenas"
if texto1 < texto2:
    print("Es menor \"Hola\"")
```

> **Ejercicio 1**:
>
> Crea un programa llamado **SumaSecuencia.py** que le pida al usuario una secuencia de números separados por espacios y calcule la suma total de esos números.

> **Ejercicio 2**: Crea un programa llamado **CuentaTexto.py** que le pida al usuario un texto, y luego le diga cuántas veces aparece la palabra *Python* en ese texto.