Operaciones con Strings

Strings

- Los strings son cadenas de caracteres donde guardamos información textual.
- Son muy útiles por razones obvias: la información está en un formato que entendemos las personas.
- Los strings son tipos de datos inmutables, es decir, no puedes cambiar el valor de una cadena una vez asignado.

Strings como cadenas

- Nos referimos al tipo texto como cadenas porque están definidas como una sucesión de caracteres encadenados.
- Esto significa que podemos referirnos a los caracteres de forma individual:

```
Ej:  a = ‘Hola Mundo’
     print(a[0]) # imprime ‘H’
     print(a[-1]) # imprime la última letra ‘o’
     a[5] = ‘y’ # error, no puedes reasignar
```

Slicing

- También podemos coger coger trozos de cadena usando los dos puntos:

```
texto[inicio:final(:salto)]
```

Ej:

```
a = ‘Hola Mundo!’
print(a[5:]) # imprime Mundo!
print(a[:4]) # imprime Hola
print(a[2:7]) # imprime “la Mu”
print(a[::2]) # imprime “Hl ud!”
```

Operadores de strings

- + : concatena dos cadenas.

  Ej: “Hola” + “ Mundo!” → “Hola Mundo!”

- * : junto a un entero repite la cadena.

  Ej: “ja” * 4 → “jajajaja”

- in: determina si hay una subcadena dentro de otra.

  Ej: “melaza” in “patatas y melaza” → True

Ejercicio

- Ejecuta las siguientes expresiones:

```
a = ‘Python!’
print(a[-7])
print(a[:-1])
print(a[-2:])
print(a[7::-2])
print(‘p’ in a)
```

Métodos de strings

- A veces los tipos contienen unas funciones especiales que les permiten realizar operaciones sobre el dato que contienen. Llamamos a estas funciones **métodos.** Para acceder a los métodos hay que usar el operador de acceso '.'.

  Ej: "HOLA".lower() → "hola"

- En el caso de los strings los métodos nos permiten devolver copias modificadas de la propia cadena. Por ejemplo, podemos poner una cadena en mayúsculas o en minúsculas.

- Tenemos una lista de todos los métodos en la referencia oficial de Python. [Ver]

Métodos de strings

- Algunos métodos útiles son:
  - capitalize # Primera letra en mayúscula
  - lower # Letras a minúsculas
  - upper # Letras a mayúsculas
  - find # Encontrar substring
  - count # Numero de substring
  - replace # Reemplaza un substring por otro
  - startswith # Verdadero si coincide el primer substring
  - endswith # Verdadero si coincide el último substring
  - isalpha # Verdadero si solo hay letras en la cadena
  - isnumeric # Verdadero si solo hay numeros en la cadena

Strings con formato

- A veces es necesario introducir en una cadena algún dato con el que estemos trabajando. Para ello hay dos formas:
  - Usando el método format: con este modo podemos sustituir en una cadena posiciones y nombres por variables y valores.

```
Ej:  'Hola a {0}'.format("todos") → Hola a todos
   # Cambia posición 0 por todos
     '{0} + {1} = {2}'.format(1,3,4) → 1 + 3 = 4
     # Cambia las posiciones 0,1,2 por 1,3,4
     '{0} + {1} = {resul}'.format(2,2,resul=4) → 1
     + 2 = 4
     # Cambia 0,1 por 2 y 2 y resul por 4.
```

Strings con formato

- Usando las f-strings. Es más intuitivo porque simplemente evalúa el contenido de las llaves {}.

Ej: f‘1 + 2 = {1+2}’ → ‘1 + 2 = 3’

```
f‘El cuadrado de 56 es {56**2}’
a = 49
f‘7 x {a//7} = {a}’
```

len

- El operador len devuelve la longitud de la cadena de caracteres.
- Más adelante será muy útil.

  Ej: len(“Hola Mundo!”) → 11

Ejercicio

- Formulario.

  Rellena y muestra el siguiente formulario a través de la terminal:

  ```
  Nombre:
  Apellidos:
  NIF:
  Edad:
  Aficiones:
  ```

Ejercicios

- Extrae info.

  Recibimos cada pocos minutos mensaje desde una estación meteorológica con este formato:

  “temp:00X-hum:X-lluvia:[si|no]”

  Es decir, la temperatura en kelvin, la humedad en porcentaje relativo (0-99) y ‘si’ o ‘no’ cuando hay lluvia. Un ejemplo puede ser:

  “temp:290-hum:54-lluvia:si”

  Extrae la información y muestra por pantalla la misma información con mejor formato y la temperatura en grados centígrados.