Tipo de datos cadenas de caracteres

Como vimos en una unidad anterior, las cadenas de caracteres (`str`): Me permiten guardar secuencias de caracteres.

Además de las operaciones que ya hemos estudiado:

- Concatenación: `+`: El operador + me permite unir datos de tipos secuenciales, en este caso dos cadenas de caracteres.
- Repetición: `*`: El operador * me permite repetir un dato de un tipo secuencial, en este caso de cadenas de caracteres.
- Indexación: Puedo obtener el dato de una secuencia indicando la posición en la secuencia. En este caso puedo obtener el carácter de la cadena indicando la posición (**empezando por la posición 0**).
- Para obtener la longitud de una cadena (número de caracteres que tiene), utilizamos la función `len`.

Tenemos más operaciones que podemos realizar:

Otras operaciones con cadenas de caracteres

- Las cadenas de caracteres se pueden recorrer:

```
>>> cadena = "informática"
>>> for caracter in cadena:
...     print(caracter,end="")
...
informática
```

- Operadores de pertenencia: Se puede comprobar si un elemento (subcadena) pertenece o no a una cadena de caracteres con los operadores `in` y `not in`.

```
>>> "a" in cadena
True
>>> "b" in cadena
False
>>> "a" not in cadena
False
```

- Slice (rebanada): Puedo obtener una subcadena de la cadena de caracteres. Se indica el carácter inicial, y el carácter final, además podemos indicar opcionalmente un salto. Si no se indica el carácter inicial se supone que es desde el primero, sino se indica el carácter final se supone que es hasta el final. Por último podemos usar salto negativo para empezar a contar desde el final.

Como resumen de las distintas posibilidades podemos indicar:

- `cadena[start:end]` # Elementos desde la posición start hasta end-1
- `cadena[start:]` # Elementos desde la posición start hasta el final
- `cadena[:end]` # Elementos desde el principio hasta la posición end-1
- `cadena[:]` # Todos Los elementos

- `cadena[start:end:step]` # Igual que el anterior pero dando step saltos.

Veamos algunos ejemplos:

```
>>> cadena[2:5]
'for'
>>> cadena[2:7:2]
'frá'
>>> cadena[:5]
'infor'
>>> cadena[5:]
'mática'
>>> cadena[-1:-3]
''
>>> cadena[::-1]
'acitámrofni'
```

Conversión de tipos

Podemos convertir cualquier número en una cadena de caracteres utilizando la función `str`:

```
>>> cad = str(7.8)
>>> type(cad)
<class 'str'>
>>> print(cad)
7.8
```

Las cadenas de caracteres son inmutables

Cuando creamos una variable de tipo cadena de caracteres, estamos creando un **objeto** de la **clase** `str`. Una clase especifica lo que podemos guardar en un tipo de datos y las operaciones que pueden realizar, cada vez que creamos una variable de una determinada clase, creamos un objeto, que además de guardar información (en nuestro caso los caracteres de la cadena) puede realizar distintas operaciones que llamamos **métodos**.

Nosotros ya hemos usado un método de la clase `str`. El método `upper()` nos permite convertir la cadena a mayúsculas.

¿Qué significa que las cadenas de caracteres son inmutables?

No podemos cambiar los caracteres de una cadena de la siguiente forma:

```
>>> cadena = "informática"
>>> cadena[2]="g"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

Esto implica que al usar un método la cadena original no cambia, el método devuelve otra cadena modificada. Veamos un ejemplo:

```
>>> cadena = "informática"
>>> cadena.upper()
'INFORMÁTICA'
>>> cadena
'informática'
```

Si queremos cambiar la cadena debemos modificar su valor con el operador de asignación:

```
>>> cadena = cadena.upper()
>>> cadena
'INFORMÁTICA'
```

Métodos principales de cadenas

Aunque las cadenas de caracteres tiene muchos métodos definidos, vamos a estudiar los más importantes:

Métodos de formato

`capitalize()` nos permite devolver la cadena con el primer carácter en mayúsculas.

```
>>> cad = "hola, como estás?"
>>> print(cad.capitalize())
Hola, como estás?
```

`lower()` y `upper()` convierte la cadena de caracteres en minúsculas y mayúsculas respectivamente.

```
>>> cad = "Hola Mundo"
>>> print(cad.lower())
hola mundo

>>> cad = "hola mundo"
>>> print(cad.upper())
HOLA MUNDO
```

`swapcase()`: devuelve una cadena nueva con las minúsculas convertidas a mayúsculas y viceversa.

```
>>> cad = "Hola Mundo"
>>> print(cad.swapcase())
hOLA mUNDO
```

`title()`: Devuelve una cadena con los primeros caracteres en mayúsculas de cada palabra.

```
>>> cad = "hola mundo"
>>> print(cad.title())
Hola Mundo
```

Métodos de búsqueda

`count()`: Es un método al que indicamos como parámetro una subcadena y cuenta cuantas apariciones hay de esa subcadena en la cadena.

```
>>> cad = "bienvenido a mi aplicación"
>>> cad.count("a")
3
```

Además podemos indicar otro parámetro para indicar la posición desde la que queremos iniciar la búsqueda. Y otro parámetro optativo para indicar la posición final de búsqueda.

```
>>> cad.count("a",16)
2
>>> cad.count("a",10,16)
1
```

`find()` nos devuelve la posición de la subcadena que hemos indicado como parámetro. Sino se encuentra se devuelve -1.

```
>>> cad.find("mi")
13
>>> cad.find("hola")
-1
```

Métodos de validación

`startswith()` nos indica con un valor lógico si la cadena empieza por la subcadena que hemos indicado como parámetro. Podemos indicar también con otro parámetro la posición donde tiene que buscar.

```
>>> cad.startswith("b")
True
>>> cad.startswith("m")
False
>>> cad.startswith("m",13)
True
```

`endswith()` igual que la anterior pero indica si la cadena termina con la subcadena indicada. En este caso, se puede indicar la posición de inicio y final de búsqueda.

```
>>> cad.endswith("ción")
```

```
True
>>> cad.endswith("ción",0,10)
False
>>> cad.endswith("nido",0,10)
True
```

Otras funciones de validación: `isdigit()`, `islower()`, `isupper()`, `isspace()`, `istitle()`,...

Métodos de sustitución

`replace()`: Devuelve una cadena donde se ha sustituido las apariciones de la primera subcadena indicada por la segunda subcadena indicada como parámetro.

```
>>> buscar = "nombre apellido"
>>> reemplazar_por = "Juan Pérez"
>>> print ("Estimado Sr. nombre apellido:".replace(buscar, reemplazar_por))
Estimado Sr. Juan Pérez:
```

`strip()`: Devuelve una cadena donde se han quitado los espacios del principio y del final. Si indicamos una subcadena como parámetro quitará dicha subcadena del principio y del final.

```
>>> cadena = "   www.eugeniabahit.com   "
>>> print(cadena.strip())
www.eugeniabahit.com
>>> cadena="00000000123000000000"
>>> print(cadena.strip("0"))
123
```

Métodos de unión y división

Aunque todavía no lo hemos estudiado, el método `split()` nos permite convertir una cadena en una lista. Lo usaremos más adelante.

```
>>> hora = "12:23:12"
>>> print(hora.split(":"))
['12', '23', '12']
```

`splitlines()`: Nos permite separar las líneas que hay en una cadena (indicada con el carácter `\n`) en una lista.

```
>>> texto = "Linea 1\nLinea 2\nLinea 3"
>>> print(texto.splitlines())
['Linea 1', 'Linea 2', 'Linea 3']
```