# Indexación de Strings

**Indexación de caracteres dentro de una cadena**
Cada uno de los caracteres de una cadena (incluidos los espacios) tiene asignado un índice. Este índice nos permite seleccionar su carácter asociado haciendo referencia a él entre corchetes ([]) en el nombre de la variable que almacena la cadena. Si consideremos el orden de izquierda a derecha, el índice empieza en 0 para el primer carácter, etc. También se puede considerar el orden de derecha a izquierda, en cuyo caso al último carácter le corresponde el índice -1, al penúltimo -2 y así sucesivamente. Este método es útil si por ejemplo queremos acceder a caracteres en las últimas posiciones de una cadena con muchos caracteres de la cual no conocemos su longitud.

| | | | | | | |
|---|---|---|---|---|---|---|
| Caracteres : | P | y | t | h | o | n |
| Índice : | 0 | 1 | 2 | 3 | 4 | 5 |
| Índice inverso : | -6 | -5 | -4 | -3 | -2 | -1 |

```
>>> cadena = "Programa en Python"
>>> cadena[0]
'P'
>>> cadena[-1]
'n'
```

# Rebanar Strings

Otra operación que podemos realizar a una cadena es seleccionar solamente una parte de ella. Para ello se usa la notación `[inicio:fin:paso]` también en el nombre de la variable que almacena la cadena, donde:

**Inicio**: es el índice del primer carácter de la porción de la cadena que queremos seleccionar.

**Fin**: es el índice del último carácter no incluido de la porción de la cadena que queremos seleccionar.

**Paso**: indica cada cuantos caracteres seleccionamos entre las posiciones de inicio y fin.

```
>>> cadena = "Programa en Python"
>>> cadena[0:8:1]
'Programa'
>>> cadena[12:18:1]
'Python'
```

Algunas consideraciones a tener en cuenta es que si omitimos las posiciones de inicio o fin, nos referimos respectivamente a la primera y última posición de la cadena. Además omitir el paso es equivalente a indicar que el paso es 1. De este modo las selecciones anteriores también se podrían realizar del siguiente modo.

```
>>> cadena = "Programa en Python"
>>> cadena[:8]
'Programa'
>>> cadena[12:]
'Python'
```

Tomando en cuenta las consideraciones anteriores podemos invertir una cadena mediante la notación `[::-1]`.

```
>>> cadena = "Programa en Python"
>>> cadena[::-1]
'nohtyP ne amargorP'
```

# Operaciones con Strings

Una operación que podemos realizar es la **concatenación** que consiste en unir distintas cadenas mediante el uso del signo más (+).

```
>>> nombre = "Luke"
>>> apellido = "Skywalker"
>>> nombre + " " + apellido
'Luke Skywalker'
```

También podemos concatenar con el signo de multiplicación (*), que en este caso significa adjuntarle un determinado número de copias a la cadena.

```
>>> cadena = 'Python.'
>>> cadena * 4
'Python.Python.Python.Python.'
```

Además, las cadenas presentan la propiedad de **inmutabilidad**. Esto significa que una vez han sido creadas no pueden modificarse. En efecto, si intentamos modificar una cadena el intérprete nos indica que a ésta no se pueden asignar elementos.

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

Sin embargo, aplicando algunos de los conceptos vistos sí que podemos llegar a cambiar el contenido de una cadena.

```
>>> cadena = "Pithon"
>>> cadena = cadena[0] + "y" + cadena[2:]
```

```
>>> cadena
'Python'
```

## Métodos de los Strings

Las cadenas, como objetos que son, nos proporcionan métodos que nos facilitan su manipulación. Tenemos por ejemplo distintos métodos que nos permiten variar la capitalización de los caracteres que forman una cadena, remover espacios en blanco o separar palabras.

**`x.upper()`**: este método devuelve una copia de la cadena convirtiendo las letras minúsculas a mayúsculas. El hecho de que devuelva una copia significa que la variable original no se ve afectada por la operación.

```
>>> x = "Programa en Python 3"
>>> x.upper()
'PROGRAMA EN PYTHON 3'
>>> x
'Programa en Python 3'
```

**`x.lower()`**: devuelve una copia de la cadena convirtiendo las letras mayúsculas a minúsculas.

```
>>> x = "PROGRAMA EN PYTHON 3"
>>> x.lower()
'programa en python 3'
```

**`x.title()`**: devuelve una copia de la cadena usando la notación de título.

```
>>> x = "programa en python 3"
>>> x.title()
'Programa En Python 3'
```

`x.replace(viejo, nuevo)`: devuelve una copia de la cadena a la cual se le ha cambiado la primera ocurrencia del carácter especificado en viejo por el especificado en nuevo.

```
>>> x = "Programa En Python 3"
>> x.replace("E", "e")
'Programa en Python 3'
```

`x.lstrip()`: devuelve una copia de la cadena a la cual se le han eliminado los espacios del principio.

`x.rstrip()`: se comporta igual que el anterior pero para los espacios del final.

```
>>> x = " Programa en Python "
>>> x.lstrip()
'Programa en Python '
>>> x.rstrip()
' Programa en Python'
```

`x.split(sep=None)`: devuelve una lista de las palabras de la cadena separadas acorde el parámetro sep. Si este parámetro no se especifica o es None, la cadena se separa teniendo en cuenta los espacios en blanco. Si usamos el símbolo arroba (@) como separador y nuestra cadena es una dirección de correo electrónico, podemos separar los nombres de usuario y servidor.

```
>>> x = "Programa en Python"
>>> x.split()
['Programa', 'en', 'Python']
>>> email = "nombre.apellido@ejemplo.tld"
>>> email.split('@')
['nombre.apellido', 'ejemplo.tld']
```