Lenguaje de Programación Python

Strings

Dr. Mario Marcelo Berón

Universidad Nacional de San Luis
Departamento de Informática
Área de Programación y Metodologías de Desarrollo de Software

Secuencias

Una *Secuencia* es una secuencia de uno o más objetos que posee las siguientes características:

- Soporta el operador de membresia *in*, la función de longitud *len*, rodajas [], y es iterable.
- Soportan las operaciones de rodajas y zancadas.

Los *Strings* son secuencias inmutables.

Strings

- El texto es una forma de dato muy utilizada.
- Python posee muchos métodos útiles para procesar texto almacenados en strings.
- Se puede indexar, sacar rodajas, convertir los caracteres, etc.

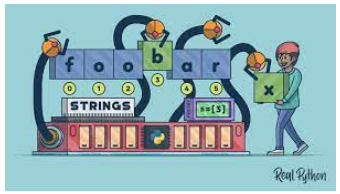
Strings: Características

Strings: secuencia de caracteres unicode

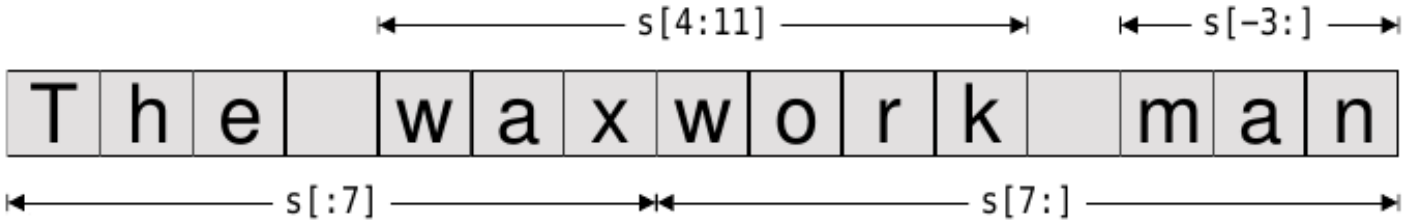
Características:

- El constructor de un string es *str()*.
- Los literales se escriben con comillas simples o comillas simples o dobles.
- Los literales que se necesitan escribir con diferentes líneas se escriben entre triple comillas doble. Este tipo de string son también utilizados para colocar comentarios multilínea.
- El caracter de escape es \. Es decir cada vez que se desea inhibir el comportamiento de un caracter especial el mismo debe ir precedido por una barra invertida \.
- Se puede colocar una *r* al principio del string con lo cual se logra que el mismo se imprima tal y como se escribió.

Strings: Indexación y Slicing



- Los strings usan índices y slices.
  - seq[start]
  - seq[start:end]
  - seq[start:end:step]
- Si en el acceso a un string se especifica un índice el resultado será el caracter que se encuentra en esa posición.
- Si se expresa un rango el resultado contendrá el caracter del índice de comienzo pero no el del índice final. Obviamente, el resultado contendrá los caracteres intermedios.
- Los operadores *in* y *not in* se pueden usar con los strings.



Ejemplo de Slice

```
>>> s = s[:12] + "wo" + s[12:]
>>> s
'The waxwork woman'
```

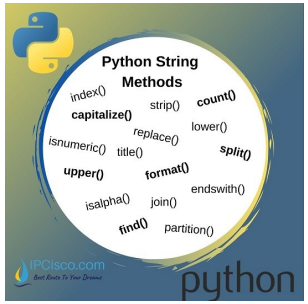
Strings: Indexación y Slicing

- Se puede usar interpolación para colocar strings dentro de otro strings en vez de concatenar.
- También se pueden usar los *f-strings* que es similar a la interpolación pero en vez de utilizar %s utiliza {}. Esto se logra colocando el prefijo *f* en el string.

Ejemplo

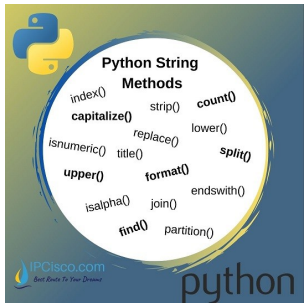
```
nombre = 'Carlos'
edad = 65
#Interpolacion
'Mi nombre es %s. Mi edad es %s.' %(name, age)
'Mi nombre es Carlos. Mi edad es 65.'
# f strings
f'Mi nombre es {nombre}. Cumplire {edad + 1}.'
'Mi nombre es Carlos. Cumplire 66.'
```

Strings: Métodos Útiles



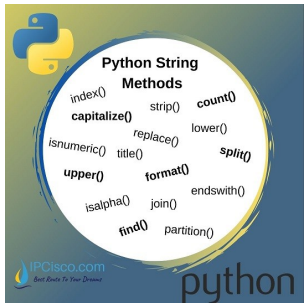
- *upper*, *lower*: retornan como resultado un string donde todas las letras del string receptor del mensaje han sido pasadas a mayúsculas o minúsculas según correspondan.
- *isupper*, *islower*: retornan como resultado *True* si el string receptor del mensaje tiene al menos una letra y todas las letras son mayúsculas o minúsculas según corresponda. En otro caso retornan *False*.

Strings: Métodos isX



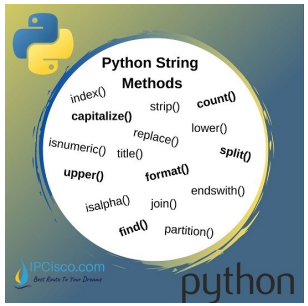
- *isalpha*: retorna *True* si el string receptor del mensaje consiste sólo de letras.
- *isalnum*: retorna *True* si el string receptor del mensaje consiste sólo de letras y números y no es blanco.
- *isdecimal*: retorna *True* si el string receptor del mensaje consiste sólo de caracteres numéricos y no es blanco.

Strings: Métodos isX



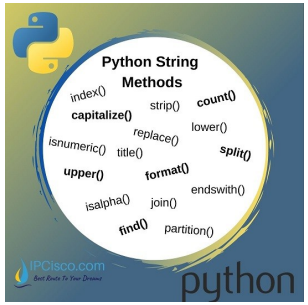
- *isspace*: retorna *True* si el string receptor del mensaje consiste de espacios en blanco, tabuladores, y enters.
- *istitle*: retorna *True* si el string receptor del mensaje consiste de palabras que comienzan con mayúsculas seguidos por letras en minúsculas.

Strings: Métodos startswith - endswith - join - split



- *startswith*: retorna *True* si el string receptor del mensaje comienza con el string que fue pasado como parámetro al método. *False* en otro caso.
- *endswith*: retorna *True* si el string receptor del mensaje finaliza con el string que fue pasado como parámetro al método. *False* en otro caso.

Strings: Métodos startswith - endswith - join - split



- *join*: retorna como resultado un string que consiste de los elementos de la lista recibida como parámetro separados con el string receptor del mensaje.
- *split*: realiza la función opuesta de *join*. Por defecto particiona al string receptor del mensaje teniendo en cuenta los caracteres espacios en blanco tales como tab, nueva línea, blanco, pero dicho caracter se puede cambiar pasándole como parámetro un string creado por el programador. Retorna como resultado una lista de strings.

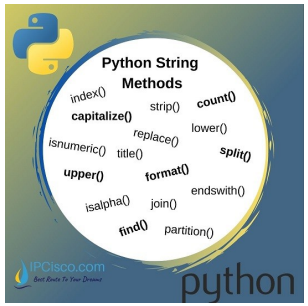
Strings: Método partition - Justificación de Texto

- *partition*: divide el string receptor del mensaje en el string previo y posterior a un string separador que se recibe como parámetro. Retorna una triupla donde la primer componente es el string previo al separador, la segunda componente es el separador y la tercera es el string posterior al separador.

Comentarios

- Si el separador se encuentra varias veces en el string receptor se toma la primera ocurrencia.
- Si el separador no se encuentra en el string receptor del mensaje se retorna una triupla con el receptor como primer comenente y el string blanco en las componentes restantes.

Strings: Método partition - Justificación de Texto



- *rjust*: retorna un string que consiste del receptor del mensaje con tantos espacios en blanco concatenados a la derecha como lo indica el número recibido como parámetro. Se le puede agregar un parámetro opcional con el caracter de relleno que por defecto es espacio.
- *ljust*: retorna un string que consiste del receptor del mensaje con tantos espacios en blanco concatenados a la izquierda como lo indica el número recibido como parámetro. Se le puede agregar un parámetro opcional con el caracter de relleno que por defecto es espacio.
- *center*: funciona como los métodos anteriores con la diferencia que coloca el receptor del mensaje en el centro del string resultante.

Strings: Otros Métodos Útiles

- *strip*: elimina caracteres en blanco del string receptor del mensaje. Se eliminan los caracteres al inicio y al final del receptor del mensaje.
- *lstrip*: elimina caracteres en blanco del string receptor del mensaje desde la izquierda.
- *rstrip*: elimina caracteres en blanco del string receptor del mensaje desde la derecha.

Comentarios

Opcionalmente se puede especificar los caracteres que se desean eliminar.

Ejemplo

```
spam = 'SpamSpamBaconSpamEggsSpamSpam'
spam.strip('ampS')
'BaconSpamEggs'
```