Strings

Profesora Teresa Tapia Soto

Contenidos

- ¿Qué es un String?
- Usos
  - Imprimir
  - Sumar o concatenar
  - Multiplicar
  - Uso de Métodos

¿Qué es un String?

Definición: Un *string* una secuencia o cadena de caracteres (un texto).

Las cadenas de caracteres, son secuencias inmutables que contienen caracteres encerrado entre comillas.

Python soporta clases de cadenas de caracteres str y Unicode

- Son secuencias inmutables de cadenas de caracteres con soporte a caracteres ASCII.

- Son secuencias inmutables de cadenas de caracteres con soporte a caracteres Unicode.

Uso 1: Imprimir -1

Se pueden imprimir

a) Uso de comilla doble “” y simple ‘ ‘ es lo mismo

```
#Uso de "" en print
print("Hola Mundo") # Hola Mundo

#Uso de comilla simple ' en print
print('Hola Mundo') # Hola Mundo
```

Nota 1: La función print(), muestra por pantalla el contenido de la cadena, pero no las comillas delimitadoras de las cadenas.
Nota 2: Las cadenas se deben cerrar con las mismas comillas con las que se abrieron, de lo contrario estaremos cometiendo un error de sintaxis:

Uso 1: Imprimir - 2

Se pueden imprimir

b) Uso de comillas triples “” y triples simple ‘’’

```
>>> print(''' hola''')
 hola

>>> print(""" hola
como
estas""")
 hola
como
estas
>>>
```

Pero las comillas triples se utilizan sobre todo con una finalidad específica: la documentación de módulos, funciones, clases o métodos. Son las llamadas ***docstrings***. Son cadenas que se escriben al principio del elemento describiendo lo que hace el elemento. No producen ningún resultado en el programa, pero las herramientas de documentación de Python pueden extraerlas para generar documentación automáticamente.

Uso 1: Imprimir - 3

¿Qué pasa con caracteres de escape?

Los caracteres especiales empiezan por una contrabarra (\) o \(backslash) y son utilizados como caracteres de escape.

Secuencia Escape	Significado
`\newline`	Ignorado
`\\`	Backslash (\)
`\'`	Comillas simple (')
`\"`	Comillas doble (")
`\a`	Bell ASCII (BEL)
`\b`	Backspace ASCII (BS)
`\f`	Formfeed ASCII (FF)
`\n`	Linefeed ASCII (LF)
`\N{name}`	Carácter llamado *name* en base de datos Unicode (Solo Unicode)
`\r`	Carriage Return ASCII (CR)
`\t`	Tabulación Horizontal ASCII (TAB)
`\uxxxx`	Carácter con valor hex 16-bit *xxxx* (Solamente Unicode). Ver hex.
`\Uxxxxxxxx`	Carácter con valor hex 32-bit *xxxxxxxx* (Solamente Unicode). Ver hex.
`\v`	Tabulación Vertical ASCII (VT)
`\ooo`	Carácter con valor octal *ooo*. Ver octal.
`\xhh`	Carácter con valor hex *hh*. Ver hex.

Secuencia	Significado
\newline	No se incluye en la cadena (sirve para escribir literales de cadena que ocupen más de una línea).
\\	Barra invertida (*backslash*).
\'	Comilla simple.
\"	Comillas dobles.
\a	Campanilla (*bell*)
\b	Retroceso (*backspace*).
\f	Nueva página.
\n	Nueva línea.
\r	Retorno de carro.
\t	Tabulador horizontal.
\v	Tabulador vertical.
\ooo	Carácter ASCII con código octal *ooo*.
\xhh	Carácter ASCII con código hexadecimal *hh*.

Casi todos los lenguajes de programación tienen un carácter especial, conocido como "***carácter de escape***", que permiten interpretar caracteres con un significado especial.

Uso 1: Imprimir – 3

Ejemplos de caracteres de escape en el texto

a) Comilla doble “”””

```
>>> print("Es importante la \"palabra\" clave")
Es importante la "palabra" clave
```

b) Comilla simple ‘

```
>>> print("El caracter \'*\' indica multiplicación")
El caracter '*' indica multiplicación
```

c) Salto de línea: \n

```
>>> print("Es importante la clase\n de mañana")
Es importante la clase
 de mañana
```

d) Tabulador \t

```
>>> print("1\t2\t3")
1       2       3
```

Uso 1: Imprimir – 4

Ejemplos de cadenas de texto largas

De acuerdo con la guía de estilo oficial de Python, las líneas de código no deben contener más de 79 caracteres, para facilitar la legibilidad.
Si un programa contiene cadenas muy largas, las cadenas se pueden simplemente partir en varias cadenas.

```
>>> print("Esta línea está cortada  en dos líneas de menos de 79 caracteres"
      " partiendo la cadena en dos")
Esta línea está cortada  en dos líneas de menos de 79 caracteres partiendo l
a cadena en dos
>>>
```

También se puede escribir el carácter **contrabarra (\)** para partir una cadena en varias líneas.

```
>>> print(" Esto es una prueba \
para partir la cadena ")
 Esto es una prueba para partir la cadena
>>>
```

Uso 1: Imprimir – 5

Cadenas con %

Un operador muy utilizado con las cadenas, especialmente para formatear la salida del programa, es %. Como operando izquierdo recibe una cadena con indicaciones de formato similares a las de printf del lenguaje C.

Carácter	Significado
d, i	Entero en decimal.
o	Entero en octal.
x, X	Entero en hexadecimal.
e, E	Número en coma flotante con exponente.
f, F	Número en coma flotante sin exponente.
g, G	Número en coma flotante con o sin exponente, según la precisión y la talla del exponente.
s	Transforma el objeto en cadena usando str.

```
>>> print("El resultado es %d" % a)
El resultado es 10
```

Uso 1: Imprimir – 6

Cadenas f

En Python 3.6 se añadió una nueva notación para cadenas llamada cadenas "f", que simplifica la inserción de variables y expresiones en las cadenas. Una cadena "f" contiene variables y expresiones entre llaves ({}) que se sustituyen directamente por su valor. Las cadenas "f" se reconocen porque comienzan por una letra f antes de las comillas de apertura.

```
>>> nombre="Teresa"
>>> cargo="Docente"
>>> print(f"Me llamo {nombre} y soy {cargo}")
Me llamo Teresa y soy Docente
>>>
```

Nota Importante: Si no se escribe la letra f antes de la cadena, Python no sustituye los valores de las variables ni calcula las expresiones.

Uso 1: Imprimir – 5

Cadenas f – acepta operaciones

```
>>> semanas =4
>>> print(f"En {semanas} semanas hay {7*semanas} días.")
En 4 semanas hay 28 días.
>>>
```

Si se quieren escribir los caracteres dentro de las llaves { o }, se debe escribir la llaves duplicadas.

```
>>> nombre="pepe"
>>> print(f"si escribe {{nombre}} debe acompañarlo con {nombre}")
si escribe {nombre} debe acompañarlo con pepe
>>>
```

Nota Importante: Si no se escribe la letra f antes de la cadena, Python no sustituye los valores de las variables ni calcula las expresiones.

Uso 2: Sumar o concatenar

Ejemplos

```
#Uso de print con concatenacion
print('Hola' +' Mundo')
#uso de print con datos
val=2
va2=3
print('Los valores de val y va2 son',val,va2)
```

Salida

```
Hola Mundo
Los valores de val y va2 son 2 3
```

Uso 3: Multiplicar

Ejemplo

```
#uso print con multiplicacion
print('va*3'*3)
```

Salida

```
va*3va*3va*3
```

Uso 4: Comparaciones

Ejemplo

```
>>> 'a' == 'A'
False
>>> 'a' < 'A'
False
```

```
>>> "hola">"hola "
False
>>> "hola"=="hola"
True
```

Revisar el código ASCII

```
>>> ord('A')
65
>>> chr(65)
'A'
>>> ord('a')
97
>>> chr(97)
'a'
>>>
```

Caracteres ASCII de control

00	NULL	(carácter nulo)
01	SOH	(inicio encabezado)
02	STX	(inicio texto)
03	ETX	(fin de texto)
04	EOT	(fin transmisión)
05	ENQ	(consulta)
06	ACK	(reconocimiento)
07	BEL	(timbre)
08	BS	(retroceso)
09	HT	(tab horizontal)
10	LF	(nueva línea)
11	VT	(tab vertical)
12	FF	(nueva página)
13	CR	(retorno de carro)
14	SO	(desplaza afuera)
15	SI	(desplaza adentro)
16	DLE	(esc.vínculo datos)
17	DC1	(control disp. 1)
18	DC2	(control disp. 2)
19	DC3	(control disp. 3)
20	DC4	(control disp. 4)
21	NAK	(conf. negativa)
22	SYN	(inactividad sincr)
23	ETB	(fin bloque trans)
24	CAN	(cancelar)
25	EM	(fin del medio)
26	SUB	(sustitución)
27	ESC	(escape)
28	FS	(sep. archivos)
29	GS	(sep. grupos)
30	RS	(sep. registros)
31	US	(sep. unidades)
127	DEL	(suprimir)

Caracteres ASCII imprimibles

32	espacio	64	@	96	`		
33	!	65	A	97	a		
34	"	66	B	98	b		
35	#	67	C	99	c		
36	$	68	D	100	d		
37	%	69	E	101	e		
38	&	70	F	102	f		
39	'	71	G	103	g		
40	(	72	H	104	h		
41	)	73	I	105	i		
42	*	74	J	106	j		
43	+	75	K	107	k		
44	,	76	L	108	l		
45	-	77	M	109	m		
46	.	78	N	110	n		
47	/	79	O	111	o		
48	0	80	P	112	p		
49	1	81	Q	113	q		
50	2	82	R	114	r		
51	3	83	S	115	s		
52	4	84	T	116	t		
53	5	85	U	117	u		
54	6	86	V	118	v		
55	7	87	W	119	w		
56	8	88	X	120	x		
57	9	89	Y	121	y		
58	:	90	Z	122	z		
59	;	91	[	123	{		
60	<	92	\	124	|		
61	=	93	]	125	}		
62	>	94	^	126	~		
63	?	95	_				

ASCII extendido (Página de código 437)

128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

https://elcodigoascii.com.ar/

Estructura String

En una cadena cada carácter tiene una posición definida en él como se ilustra en la figura:

C	A	D	E	N	A
0	1	2	3	4	5
-6	-5	-4	-3	-2	-1

Uso: Dado un string, `s[i]` entrega el carácter ubicado en el índice `i`.

Función: La función **`len(s)`** recibe un texto **`s`** y retorna la cantidad de caracteres que tiene ese texto.

SubString

- **Definición:** Un *substring* es un segmento del string.
- **Para obtener un, substring** de la cadena "s" utilizar la siguiente estructura s[i : j], lo cual entrega un substring desde el i-ésimo carácter (inclusive) hasta el j-ésimo carácter (exclusive). Los parámetros son opcionales.

```
>>> f="Esto es un saludo mucho más largo"
>>> a=f[6:12]
>>> print(a)
s un s
>>> b=f[:6]
>>> print(b)
Esto e
>>> c=f[12:]
>>> print(c)
aludo mucho más largo
>>> d=f[:]
>>> print(d)
Esto es un saludo mucho más largo
>>>
```

Método

La clase string tiene varios métodos que son muy útiles para dar formato a las cadenas de texto.

- Revisaremos algunos métodos importantes. Para mayor información

https://docs.python.org/3/library/stdtypes.html#string-methods

- La sintaxis del uso de los métodos es generalmente `cadena.metodo()`

Métodos String

capitalize() devuelve una copia de la cadena con la primera letra en mayúscula.

center(n) devuelve una copia de la cadena centrada y con longitud n.

find(sub,[,desde[,hasta]]) devuelve la posición de la primera aparición de sub en la cadena; si se incluye desde, la búsqueda comienza en esa posición y termina en hasta, si se especifica.

isalnum() devuelve cierto si la cadena es no vacía y solo contiene letras y dígitos.

isalpha() devuelve cierto si la cadena es no vacía y solo contiene letras.

Métodos String

isdigit() devuelve cierto (True) si la cadena es no vacía y solo contiene dígitos (por lo menos hay un dígito en la cadena de largo 1).

islower() devuelve cierto (True) si todas las letras de la cadena son minúsculas y hay al menos una minúscula.

isspace() devuelve cierto si la cadena es no vacía y todos sus caracteres son espacios.

isupper() devuelve cierto si todas las letras de la cadena son mayúsculas y hay al menos una mayúscula.

lower() devuelve una copia de la cadena con las letras convertidas a minúsculas.

lstrip() devuelve una copia de la cadena con los blancos iniciales omitidos.

Métodos String

replace(v, n) devuelve una copia de la cadena donde se han sustituido todas las apariciones de la cadena v por n.

rstrip() devuelve una copia de la cadena con los blancos finales omitidos.

split([s]) devuelve una lista que contiene las palabras de la cadena. Si se incluye la cadena s, se utiliza como separador.

Strings

Profesora Teresa Tapia Soto


Inacap