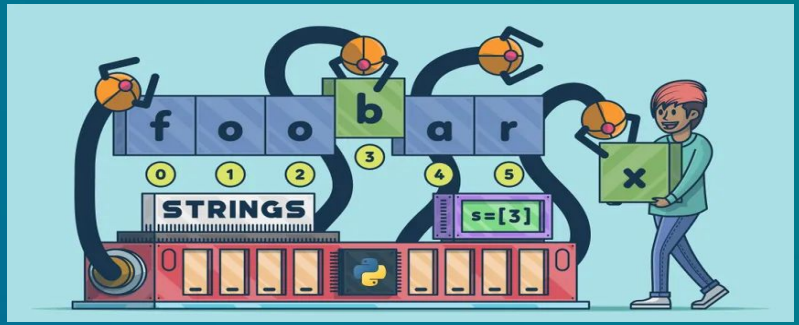
Cadenas de caracteres



¿Que es un carácter?

Un carácter es una unidad de información que corresponde a una letra, un número o un signo de puntuación.

¿Qué es una cadena de caracteres?

Una cadena de caracteres es una secuencia de símbolos alfanuméricos, signos de puntuación, espacios, etc. En Python se escriben entre comillas simples (‘ ’) o dobles (“ ”). Estas cadenas pueden pensarse como listas de caracteres, las cuales se pueden recorrer, separar, leer, etc.

```
cadena_uno = "Hello World"
cadena_dos = "Python es divertido!"
cadena_tres = "123 ABC abc"
```

Comillas

Puede darse el caso que dentro de una cadena de caracteres se requiera escribir con comillas, pero no es tan sencillo.

```
cadena = "Hello "World"" #❌SyntaxError
```

Ya que esto causará que la cadena se considere cerrada y, en este caso, la palabra “World” esté por fuera de la misma.

```
cadena = "Hello 'World'" #✔
cadena_dos = "Hello \"World\"" # ✔
```

Operaciones básicas

- Acceso a caracteres individuales -> Indexación [0]
- Slicing (subcadenas) -> [1:3] **(py)**
- Longitud de la cadena -> Función len()
- Concatenación -> Operador +
- Repetición de cadenas -> Operador * **(py)**
- Comparación de cadenas -> Operadores relacionales (<,>,!=,==,<=,>=)
- Recorrido y búsqueda -> For

Indexación

Las cadenas de caracteres tienen indexación con base 0, como una lista, lo que quiere decir que los caracteres de la cadena tienen un índice asignado comenzando desde el 0.

```
cadena = "Cadena de caracteres"
print(cadena[0]) # C
print(cadena[1]) # a
print(cadena[19])# s
```

Inmutabilidad

Las cadenas de caracteres son inmutables, esto quiere decir que no pueden ser modificadas de ninguna manera. Cualquier modificación realizada
dentro de una cadena de caracteres arrojará una excepción.

```
cadena = "Nueva"
cadena[0] = 'C' #Excepcion
```

```
    cadena[0] = 'C' #Excepcion
    ~~~~~~~^^^
TypeError: 'str' object does not support item assignment
```

Slicing

Existe una técnica llamada slicing la cual permite separar una cadena en partes.
La sintaxis es [Inicio : Fin], el inicio es incluido en el corte, mientras que el final no.

```
cadena = "Separar en partes"
print(cadena[0:7]) # Separar
print(cadena[8:10]) # en
print(cadena[11:17]) # partes
```

Función len

Para conocer el largo de una cadena podemos utilizar la función len.

```
cadena = "Aprendamos cadenas"
print(len(cadena)) # 18
```

Concatenación

Las cadenas son inmutables pero pueden ser concatenadas para formar nuevas cadenas. Es importante saber que al concatenar cadenas se está creando una nueva variable.

```
cadena = "Que tal"
print(id(cadena))
#ID: 1650653612336
#Cadena: Que tal

cadena += '?'
print(id(cadena))
#ID: 1650653612208
#Cadena: Que tal?
```

Repetición

Con el operador de multiplicación (*) podemos repetir una cadena tantas veces como queramos.

```
cadena = "Repetir "
print(cadena * 3)
```

```
Repetir Repetir Repetir
```

Comparación

Utilizando los comparadores lógicos (<, >, ==, etc) comparamos cadenas. Estas comparaciones trabajan lexicográficamente, es decir, se hacen basadas en los valores ASCII de los caracteres.

```
cadena = "Procesador"
print(cadena == "Procesador") #True
print(cadena != "Procesador") #False
print(cadena > "Mother") #True
print(cadena < "procesador") #True
```

Normalización

Las comparaciones entre cadenas son Case Sensitive, es decir, distinguirá entre mayúsculas y minúsculas.

```
cadena = "Caracteres"
print(cadena == "caracteres") #False
```

Por lo cual, si se busca ignorar esa diferencia, hay que aplicar una normalización, llevando ambas cadenas a comparar a un mismo formato.

Cadenas de caracteres

Código ASCII

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 sínc)
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

Recorrer cadenas

Las cadenas de caracteres pueden ser recorridas caracter por caracter, utilizando un bucle for, como si fueran una lista.

```
cadena = "A contar"
for i in range(len(cadena)):
    print(cadena[i])
```

```
A

c
o
n
t
a
r
```