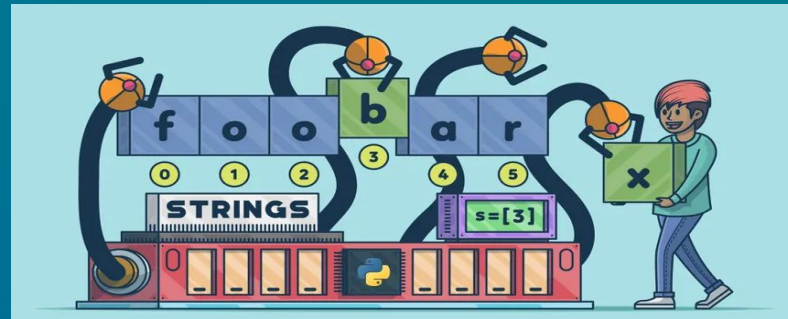


# Cadenas de caracteres



# ¿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"
```

# 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
```

### 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	Ł	224	Ó
129	ü	161	í	193	ł	225	ô
130	é	162	ó	194	Ł	226	ö
131	â	163	ú	195	ł	227	õ
132	ä	164	ñ	196	Ł	228	ö
133	à	165	Ñ	197	ł	229	õ
134	â	166	°	198	Ł	230	μ
135	ç	167	°	199	ł	231	þ
136	ê	168	¿	200	Ł	232	þ
137	ë	169	®	201	ł	233	Û
138	è	170	¬	202	Ł	234	Ü
139	ï	171	½	203	ł	235	Ü
140	î	172	¼	204	Ł	236	ý
141	ì	173	⅓	205	ł	237	Ý
142	À	174	«	206	Ł	238	ˆ
143	Á	175	»	207	ł	239	ˆ
144	Â	176	¸	208	Ł	240	≡
145	æ	177	¸	209	ł	241	±
146	Æ	178	¸	210	Ł	242	≡
147	ø	179	¸	211	ł	243	¼
148	ø	180	¸	212	Ł	244	½
149	ø	181	¸	213	ł	245	¾
150	ù	182	À	214	Ł	246	÷
151	ù	183	Á	215	ł	247	÷
152	ÿ	184	©	216	Ł	248	÷
153	Ø	185	¸	217	ł	249	÷
154	Ø	186	¸	218	Ł	250	÷
155	ø	187	¸	219	ł	251	÷
156	£	188	¸	220	Ł	252	÷
157	Ø	189	¸	221	ł	253	÷
158	×	190	¥	222	Ł	254	÷
159	f	191	¸	223	ł	255	nbps

## 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