



El futuro digital  
es de todos

MinTIC



# Estructuras de Datos I

## Cadenas

Research Group on Artificial Life  
Grupo de investigación en vida artificial (Alife)  
Computer and System Department  
Engineering School  
Universidad Nacional de Colombia

Jonatan Gomez Perdomo, Ph. D.  
[jgomezpe@unal.edu.co](mailto:jgomezpe@unal.edu.co)

Arles Rodríguez, Ph.D.  
[aerodriguezp@unal.edu.co](mailto:aerodriguezp@unal.edu.co)

Camilo Cubides, Ph.D. (c)  
[eccubidesg@unal.edu.co](mailto:eccubidesg@unal.edu.co)

Carlos Andrés Sierra, M.Sc.  
[casierrav@unal.edu.co](mailto:casierrav@unal.edu.co)



UNIVERSIDAD  
NACIONAL  
DE COLOMBIA

# Agenda

- 1 Introducción
- 2 Operadores
- 3 Cadenas y estructuras de control
- 4 Métodos



# Carácter

Un **carácter** es el elemento mínimo de información usado para representar, controlar, transmitir y visualizar datos. Al conjunto de caracteres usados con este fin se le llama **Esquema de codificación**. Los esquemas de codificación en general usan un número de bits o bytes fijos.



# Esquemas de Codificación - ASCII

Código Estadounidense Estándar para el Intercambio de Información  
(*American Standard Code for Information Interchange*)

- En su versión original usa 7 bits, definiendo 128 caracteres.
- En la versión extendida usa 8 bits (esto es 1 byte), definiendo 256 caracteres.
- Es la base de los archivos de texto plano (o sin formato).
- Es el esquema base para la escritura de programas en casi todos los lenguajes de programación (incluido **Python**).



# Esquemas de Codificación - ASCII

Caracteres ASCII de control			Caracteres ASCII imprimibles			ASCII extendido (Página de código 437)										
00	NULL	(carácter nulo)	32	espacio	64	@	96	`	128	Ç	160	á	192	Ł	224	Ó
01	SOH	(inicio encabezado)	33	!	65	A	97	a	129	ü	161	í	193	ł	225	ô
02	STX	(inicio texto)	34	"	66	B	98	b	130	é	162	ó	194	Ł	226	ö
03	ETX	(fin de texto)	35	#	67	C	99	c	131	â	163	ú	195	ł	227	õ
04	EOT	(fin transmisión)	36	\$	68	D	100	d	132	ä	164	ñ	196	Ł	228	ö
05	ENQ	(consulta)	37	%	69	E	101	e	133	à	165	Ñ	197	ł	229	ó
06	ACK	(reconocimiento)	38	&	70	F	102	f	134	á	166	ª	198	Ł	230	µ
07	BEL	(timbre)	39	'	71	G	103	g	135	ç	167	º	199	Ł	231	þ
08	BS	(retroceso)	40	(	72	H	104	h	136	ê	168	¿	200	Ł	232	þ
09	HT	(tab horizontal)	41	)	73	I	105	i	137	ë	169	®	201	Ł	233	Û
10	LF	(nueva línea)	42	*	74	J	106	j	138	è	170	¬	202	Ł	234	Ü
11	VT	(tab vertical)	43	+	75	K	107	k	139	ï	171	½	203	Ł	235	Ù
12	FF	(nueva página)	44	,	76	L	108	l	140	î	172	¾	204	Ł	236	Ý
13	CR	(retorno de carro)	45	-	77	M	109	m	141	í	173	¿	205	Ł	237	Ÿ
14	SO	(desplaza afuera)	46	.	78	N	110	n	142	Ä	174	«	206	Ł	238	—
15	SI	(desplaza adentro)	47	/	79	O	111	o	143	Å	175	»	207	Ł	239	·
16	DLE	(esc.vínculo datos)	48	0	80	P	112	p	144	Ê	176	◊	208	Ł	240	≡
17	DC1	(control disp. 1)	49	1	81	Q	113	q	145	æ	177	◊	209	Ł	241	±
18	DC2	(control disp. 2)	50	2	82	R	114	r	146	Æ	178	◊	210	Ł	242	—
19	DC3	(control disp. 3)	51	3	83	S	115	s	147	ó	179	◊	211	Ł	243	¼
20	DC4	(control disp. 4)	52	4	84	T	116	t	148	ô	180	◊	212	Ł	244	¶
21	NAK	(conf. negativa)	53	5	85	U	117	u	149	õ	181	◊	213	Ł	245	§
22	SYN	(inactividad sinc)	54	6	86	V	118	v	150	û	182	◊	214	Ł	246	÷
23	ETB	(fin bloque trans)	55	7	87	W	119	w	151	ü	183	◊	215	Ł	247	°
24	CAN	(cancelar)	56	8	88	X	120	x	152	ÿ	184	◊	216	Ł	248	°
25	EM	(fin del medio)	57	9	89	Y	121	y	153	ÿ	185	◊	217	Ł	249	°
26	SUB	(sustitución)	58	:	90	Z	122	z	154	ÿ	186	◊	218	Ł	250	°
27	ESC	(escape)	59	;	91	[	123	{	155	ÿ	187	◊	219	Ł	251	°
28	FS	(sep. archivos)	60	<	92	\	124		156	ÿ	188	◊	220	Ł	252	°
29	GS	(sep. grupos)	61	=	93	]	125	}	157	ÿ	189	◊	221	Ł	253	°
30	RS	(sep. registros)	62	>	94	^	126	~	158	ÿ	190	◊	222	Ł	254	°
31	US	(sep. unidades)	63	?	95	_			159	f	191	◊	223	Ł	255	nbsp
127	DEL	(suprimir)														

Figure: Imagen tomada de <https://elcodigoascii.com.ar/>

# Esquemas de Codificación - Unicode

Esquema de codificación cuyo objetivo es dar a cada carácter usado por cada uno de los lenguajes humanos su propio código, es decir, permitir la “internacionalización” de la computación.

**UTF – 8** : Definido por ocho (8) bits (un byte). Toma como base el ASCII, ANSI de Windows y el ISO – 8859 – 1. Muy usado en HTML.

**UTF – 16** : Definido por 16 bits (2 bytes). Usa una representación de longitud variable que permite su optimización en procesos de codificación a texto (usando un subconjunto de ASCII o UTF – 8).

**UTF – 32** : Definido por 32 bits (4 bytes). Es el más simple pues usa una representación de longitud fija.



# Esquemas de Codificación - Unicode

The image shows the homepage of the Unicode Consortium. On the left is a navigation menu with links: 'Adopt a Character', 'Emoji', 'Basic Info', 'News', 'Events', 'Connect', 'Membership', and 'Press'. The main content area features a grid of characters for adoption, each with a name and a Unicode code point (e.g., U+17B5, U+1F3A2, U+4E41, etc.). A central text block reads: 'Everyone in the world should be able to use their own language on phones and computers.' Below this is a link to 'LEARN MORE ABOUT UNICODE'. On the right, there is a circular gold seal for 'OFFICIAL GOLD SPONSOR' and a button that says 'ADOPT A CHARACTER'.

Figure: Captura de la página <https://home.unicode.org/>



# Usando caracteres en un programa

Dado que Python usa `ASCII` para la escritura de sus programas, se cuenta con un esquema de representación para indicar que se usarán los mismos. El carácter a usar se delimita por el carácter `'` o por el carácter `"` (llamado *escape*) de caracteres tanto de control o Unicode.

`'A'` : Se refiere al carácter A

`"3"` : Se refiere al carácter 3

`'"'` : Se refiere al carácter "

`"'"` : Se refiere al carácter '





# Usando caracteres en un programa

Cuando se requieren caracteres especiales, de control o de Unicode, se puede utilizar la secuencia de *escape* apropiada.

`\n` : Una nueva línea

`\t` : Una tabulación

`\"` : Una comilla doble

`\'` : Una comilla simple

`\\` : El carácter de diagonal invertida (*backslash*)

`\u0105` : El carácter ą

`\u01F4` : El carácter Ĝ



# Cadenas de caracteres (str)

Una cadena de caracteres `str` es una secuencia de cero o más caracteres. Una cadena de caracteres se delimita por el carácter `'` o por el carácter `"`. Una cadena de caracteres es una estructura de datos inmutable, esto significa que no puede ser cambiada.

- `'ejemplo de cadena'`
- `"Cadena con un tabulado \t y una nueva \n línea"`
- `'Cadena con un carácter unicode \u01F4 y una comilla doble'"'`
- `"Cadena con una comilla simple \', una comilla doble \" y una diagonal invertida \\""`
- La cadena vacía `""` o `''`



# Ejemplo 1

## Ejemplo

Para el programa

```
str1 = "ejemplo de cadena"  
print(str1)
```

La salida obtenida es:

```
ejemplo de cadena
```



## Ejemplo 2

### Ejemplo

Para el programa

```
cadena = "Cadena con un tabulado \t, y una nueva \n línea"  
print(cadena)
```

La salida obtenida es:

```
Cadena con un tabulado      , y una nueva  
línea
```



# Agenda

- 1 Introducción
- 2 Operadores
- 3 Cadenas y estructuras de control
- 4 Métodos



# Concatenar +

Concatena (pega) dos cadenas. Para el programa

```
nombre = "Minch Yoda"  
trabajo = "Stars War"  
print(nombre + " el maestro")  
print(nombre + trabajo)  
print(trabajo + " " + nombre)
```

La salida obtenida es:

```
Minch Yoda el maestro  
Minch YodaStars War  
Stars War Minch Yoda
```



# Comparar

Se usan los operadores convencionales (<, <=, >, >=, ==, !=) para comparar cadenas usando el orden **lexicográfico**. En el orden lexicográfico, se comparan de izquierda a derecha uno a uno los caracteres, mientras sean iguales. En el caso que no sean iguales, si el carácter de la primera cadena es menor que el de la segunda a la primer cadena se le considera menor, pero si es mayor, a la primer cadena se le considera mayor. Si todos los caracteres son iguales, las cadenas son iguales.



```
print("Rojas" < "Rosas")  
print("Rojas" == "rosas")
```

Para el anterior programa, la salida obtenida es:

```
True  
False
```



# Subíndice [ ]

Accede los elementos de una cadena, el primer índice de la cadena es cero (0). Para el programa

```
nombre = "Minch Yoda"  
print(nombre[0]) # imprime M  
print(nombre[6]) # imprime Y  
print(nombre[4]) # imprime h
```



La salida obtenida es:

M  
Y  
h





# Agenda

- 1 Introducción
- 2 Operadores
- 3 Cadenas y estructuras de control
- 4 Métodos



# Consultando una cadena

Es posible determinar si una subcadena se encuentra en una cadena de caracteres. Para el programa

```
text = "cien años de soledad"  
if "años" in text:  
    print("yes")  
else:  
    print("no")
```



La salida obtenida es:

yes



# Iterando una cadena

Es posible iterar una cadena de caracteres usando el ciclo for. Para el programa

```
s = "hola amigos mios"
for letra in s: # se puede iterar cada letra de la cadena
    print(letra, end = ", ")
```



La salida obtenida es:

```
h, o, l, a,  , a, m, i, g, o, s,  , m, i, o, s,
```



# Agenda

- 1 Introducción
- 2 Operadores
- 3 Cadenas y estructuras de control
- 4 **Métodos**



# Longitud (len)

la función `len` determina la longitud de una cadena. Para el programa

```
nombre = "Minch Yoda"  
trabajo = "Stars War"  
planeta = "Tatoon \t cinco"  
vacía = ""  
print(len(nombre))  
print(len(trabajo))  
print(len(planeta))  
print(len(vacía))
```



La salida obtenida es:

```
10  
9  
14  
0
```



# Subcadenas (slice)

La función `slice` obtiene una porción (subcadena) de una cadena. La notación es similar a la función `range`, `[inicio:fin:incremento]`. Para el programa

```
nombre = "Minch Yoda"  
print(nombre[:5])  
print(nombre[0:7])  
print(nombre[6:10])  
print(nombre[::-1])
```



La salida obtenida es:

```
Minch  
Minch Y  
Yoda  
adoY hcnIM
```



# Contando (count)

El método `count` obtiene las veces que una subcadena se encuentra en una cadena (o en una parte de ella). La notación es `count(subcadena, inicio, fin)`. Para el programa

```
str1 = "The avengers"  
print(str1.count("e"))  
print(str1.count("e", 0, 3))  
print(str1.count("e", 4, len(str1)))  
cad = "abcabcabcabcabc"  
print(cad.count("abc"))
```



La salida obtenida es:

3  
1  
2  
5



# Buscando (find, rfind)

Los métodos `find` y `rfind` obtienen la primera y última ocurrencia de una subcadena en una cadena (o en una parte de ella), respectivamente. La notación es `find/rfind(subcadena, inicio, fin)`. Para el programa

```
str2 = "It is not despair, for despair is " \
       "only for those who see the end " \
       "beyond all doubt. We do not."
print("first:", str2.find("despair"))
print("last:", str2.rfind("despair"))
```



La salida obtenida es:

```
first: 10
last: 23
```





## Mayúsculas/Minúsculas

Son varios métodos que operan de acuerdo a mayúsculas y minúsculas.

```
s = "cien años de soledad en Macondo"
print(s.lower()) # Muestra la cadena en minúsculas
print(s.upper()) # Muestra la cadena en mayúsculas
print(s.capitalize()) # Primer letra a mayúscula
print(s.title()) # Primer letra cada palabra a mayúscula
print(s.swapcase()) # Mayúsculas <-> minúsculas
```



Para el anterior programa, la salida obtenida es:

cien años de soledad en macondo  
CIEN AÑOS DE SOLEDAD EN MACONDO  
Cien años de soledad en macondo  
Cien Años De Soledad En Macondo  
CIEN AÑOS DE SOLEDAD EN mACONDO



# Removiendo caracteres (strip, lstrip, rstrip)

El método `strip/lstrip/rstrip` remueve los caracteres deseados a los dos lados/izquierda/derecha de una cadena. La notación es `strip/lstrip/rstrip(caracteres)`. Si no se dan caracteres como argumento, elimina espacios en blanco (espacios y tabulaciones).

```
s = "---++---cien años de soledad en Macondo---++---"  
print(s.strip("-+"))  
print(s.lstrip("-+"))  
print(s.rstrip("-+"))
```



Para el anterior programa, la salida obtenida es:

```
cien años de soledad en Macondo  
cien años de soledad en Macondo---++---  
---++---cien años de soledad en Macondo
```

Hechos  
QUE CONECTAN



# Dividiendo cadenas (split)

El método `split` divide una cadena de acuerdo a una subcadena que sirve como delimitador, dejando las partes separadas en una lista. La notación es `split(delimitador)`. Para el programa

```
sdate = "01-06-2021"  
sp = sdate.split("-")  
print(sp)  
print("día:", sp[0], "- mes:", sp[1], "- año:", sp[2])
```



La salida obtenida es:

```
['01', '06', '2021']  
día: 01 - mes: 06 - año: 2021
```



# Justificación de cadenas I

Existen cuatro métodos para justificar cadenas:

`ljust()` : Justificar una cadena a la izquierda

`rjust()` : Justificar una cadena a la derecha

`center()` : Centrar una cadena

`zfill()` : Llenar una cadena con ceros



# Justificando cadenas II

Para el programa

```
str1 = "Bogotá"  
print(str1.ljust(15, "#"))  
print(str1.rjust(15, "#"))  
print(str1.center(15, "#"))  
account = "123456789"  
print(account.zfill(15))
```



La salida obtenida es:

```
Bogotá#####  
#####Bogotá  
#####Bogotá####  
000000123456789
```



# Reemplazando replace I

El método `replace` reemplazar una subcadena en una cadena por otra. la notación es `replace(anterior, nueva)`. Para el programa

```
str1 = "cien años de soledad"
print(str1)
rep = str1.replace("cien", "setenta")
print(rep)
rep = rep.replace("años", "días")
print(rep)
rep = rep.replace("soledad", "clases sincrónicas!")
print(rep)
```



# Reemplazando replace II

La salida obtenida es:

```
cien años de soledad  
setenta años de soledad  
setenta días de soledad  
setenta días de clases sincrónicas!
```



# ... más métodos

`endswith` : Determinar si una cadena termina con.

`startswith` : Determinar si una cadena empieza con.

`isalpha` : Determinar si una cadena contiene letras únicamente.

`isalnum` : Determinar si una cadena contiene números y letras únicamente (alfanumérico).

`isdigit` : Determinar si una cadena contiene sólo dígitos.

`isspace` : Determinar si una cadena contiene sólo espacios.

`istitle` : Determinar si una cadena es un título.

`islower` : Determinar si una cadena contiene todos sus caracteres en minúsculas.

`isupper` : Determinar si una cadena contiene todos sus caracteres en mayúscula.





# Sugerencia

Se sugiere consultar un manual de Python o de sus librerías para determinar si ya existe un método para lo que se quiera realizar con una cadena de caracteres.

