Manejo de archivos

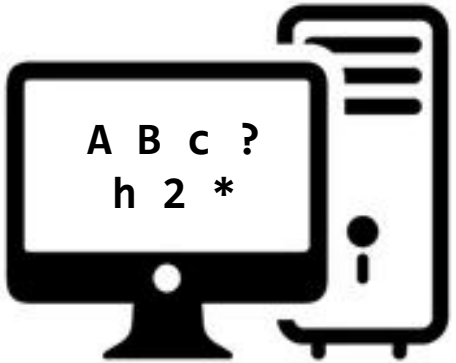
- Código ASCII
- Archivos de texto
- Apertura y cierre
- Lectura y escritura

Código ASCII

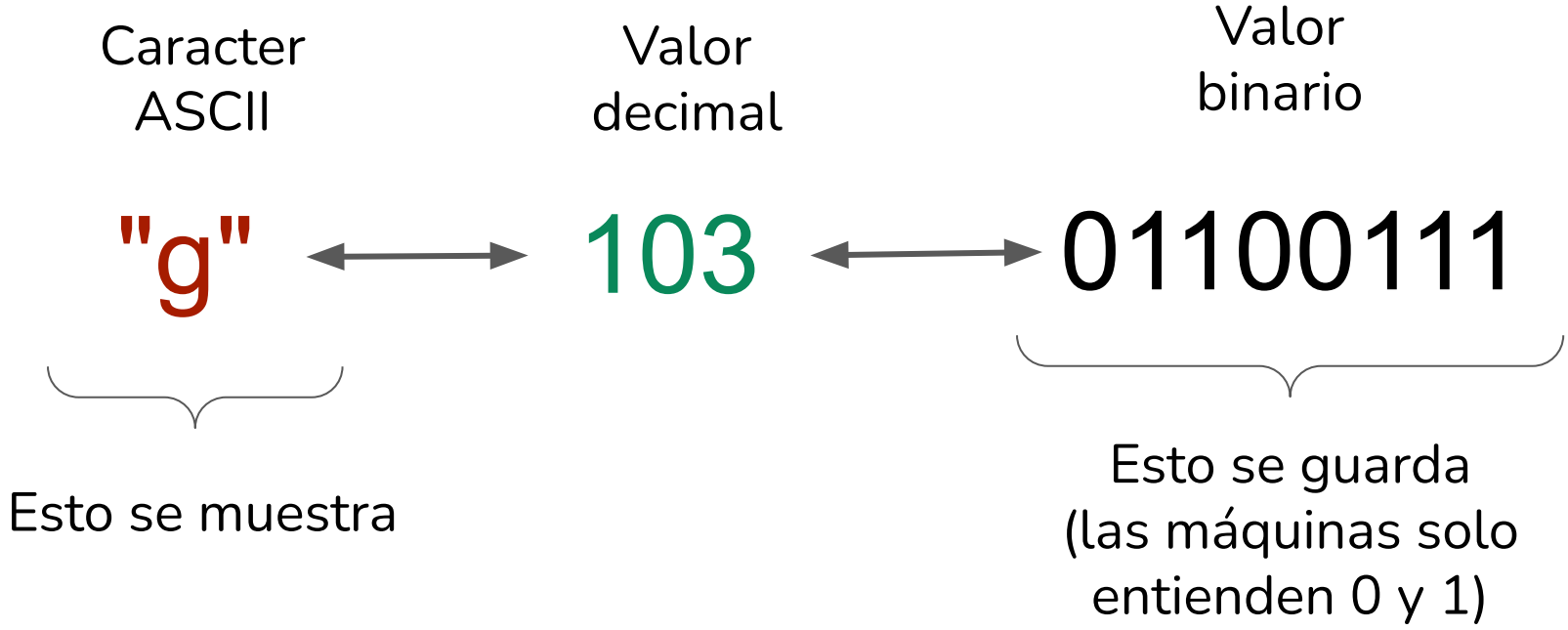
Código ASCII

ASCII: **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange

Es un código mediante el cual se le asigna un "número" a símbolos y caracteres alfanuméricos (ya que las máquinas solo entienden números)



Código ASCII



Código ASCII

Universidad de San Andrés

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							
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

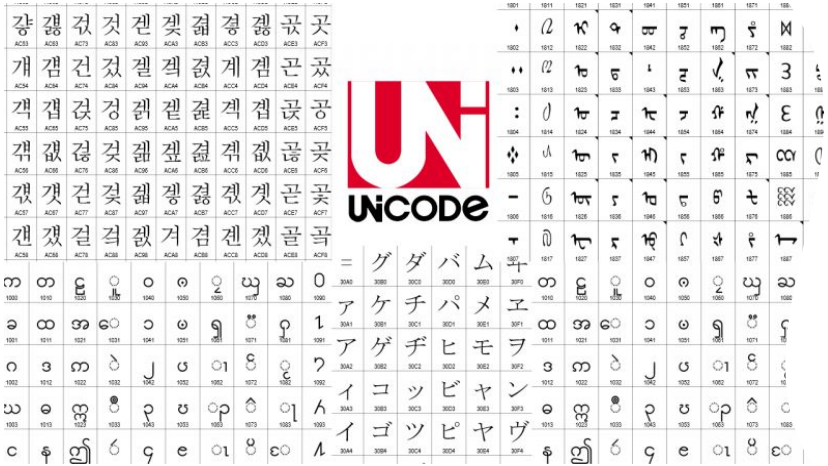
Unicode

- Unicode es el estándar que se utiliza para codificar, representar y manejar texto en las computadoras, dispositivos de telecomunicaciones y otros equipos.

- Para compatibilidad, Unicode incluye la misma representación de los caracteres ASCII equivalentes, pero agrega muchos caracteres más.

UNICODE

ASCII



Conversión entre caracter y decimal: chr() y ord()

Universidad de SanAndrés

chr(): retorna un str con un caracter y recibe un número entero con el valor *Unicode* que lo representa. Ejemplo:

```
ordinal = 946
caracter = chr(ordinal)
print(ordinal,caracter)
```

```
946 β
```

ord(): retorna un valor *Unicode* y recibe un str con el caracter que se representa con ese valor. Ejemplo:

```
caracter = '?'
ordinal = ord(caracter)
print(ordinal,caracter)
```

```
63 ?
```

Archivos

¿Qué es un archivo?

- Secuencia de datos **almacenados en memoria** (disco rígido, USB, etc)
- Se identifica por un **nombre** a nivel de usuario (la extensión es parte del nombre).
- El sistema operativo sabe dónde empieza y termina el archivo. También nos da funciones que nos permiten usarlo.
- Existen 2 tipos de archivos: archivos de **texto** y archivos **binarios**.

```
01011011110100010110001
101100100101011110100
01101010111010010100101
10110100101010101101110
10101010101011010 1010
10100101000110101001010
```

¿Qué es un archivo de binario?

- Es un archivo cuyos datos se almacenan en un modo que **no** es **legible** por una persona. La información se **interpreta** a nivel de **bit o byte** en lugar de caracteres ASCII o Unicode
- Un ejemplo de este tipo de archivos es un documento de Microsoft Word, que requiere de dicho programa para ser *interpretado* y mostrado en pantalla.
- Otro ejemplo típico es un archivo ejecutable.

program.exe

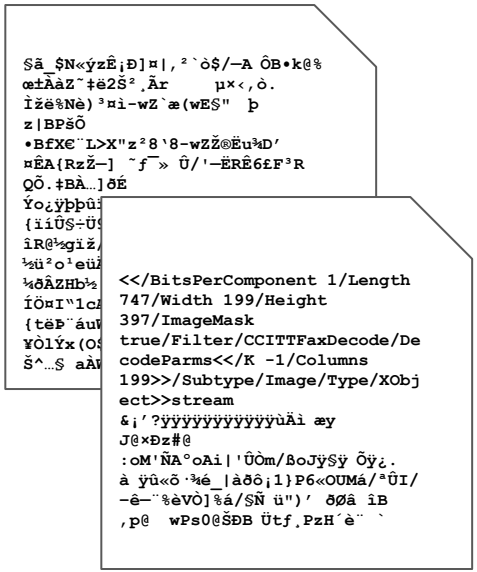
tesis.docx

¿Qué es un archivo de binario?

- Es un archivo cuyos datos se almacenan en un modo que **no** es **legible** por una persona. La información se **interpreta** a nivel de **bit o byte** en lugar de caracteres ASCII o Unicode
- Un ejemplo de este tipo de archivos es un documento de Microsoft Word, que requiere de dicho programa para s… …o en pantalla.
- Otro ejemplo típico es …

Si intentamos abrir un archivo binario con un editor de texto, la información se convierte a caracteres ASCII sin ningún sentido

program.exe



tesis.docx

¿Qué es un archivo de texto plano?

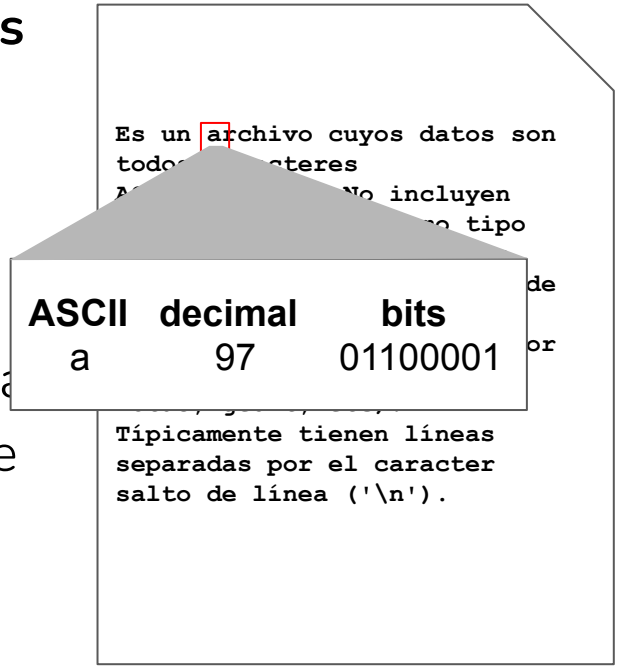
Universidad de SanAndrés

- Es un archivo cuyos datos son todos **caracteres** ASCII/Unicode.
- **No incluyen** información extra como tipo de letra, cursiva, negrita, viñetas, etc.
- No requieren de un software especializado para abrirlos, sólo un **editor de texto plano** (bloc de notas, gedit, etc.)
- Típicamente tienen líneas separadas por el caracter salto de línea (`'\n'`).

```
Es un archivo cuyos datos son
todos caracteres
ASCII/Unicode. No incluyen
información extra como tipo
de letra, cursiva, negrita,
viñetas, etc. No requieren de
un software especializado
para abrirlos, sólo un editor
de texto plano (bloc de
notas, gedit, etc).
Típicamente tienen líneas
separadas por el caracter
salto de línea ('\n').
```

¿Qué es un archivo de texto plano?

- Es un archivo cuyos datos son todos **caracteres** ASCII/Unicode.
- **No incluyen** información extra como tipo de letra, cursiva, negrita, viñetas, etc.
- No requieren de un software especializado para abrirlos, sólo un **editor de texto plano** (bloc de notas, gedit, etc.)
- Típicamente tienen líneas separadas por el caracter salto de línea (`'\n'`).



¿Qué es un archivo de texto plano?

Universidad de SanAndrés

- Es un archivo cuyos datos son todos **caracteres** ASCII/Unicode.
- **No incluyen** información extra como tipo de letra, cursiva, negrita, viñetas, etc.
- No requieren de un software especializado para abrirlos, sólo un **editor de te** notas, gedit, etc.)
- Típicamente tienen líneas separa por el caracter salto de línea (`'\n'`).

```
Es un archivo cuyos datos son
todos caracteres
ASCII/Unicode. No incluyen
información extra como tipo
de letra, cursiva, negrita,
viñetas, etc. No requieren de
un software especializado
para abrirlos, sólo un editor
de texto plano (bloc de
                dit, etc).
                e tienen líneas
                por el caracter
                ínea ('\n').
```

Dependiendo del sistema operativo, ese caracter se almacena como '\n'–unix, linux, gnu/linux, macOS– o como '\r\n'–Windows–.

¿Qué es un archivo de texto plano?

Un ejemplo de este tipo de archivos es el código fuente de un programa escrito en **python**, los archivos **.py** (***usamos la extensión para identificarlos, por comodidad únicamente***).

programa.py

```
def sum_abs(a, b):
  return ans(a) + abs(b)

def sum_squares(a, b):
  return a**2 + b**2

a = 2
b = 3
if input('sum_abs? (si/no)')=='si':
    c = sum_abs(a, b)
else:
    c = sum_squares(a, b)

print(f'c={c}')
```

¿Qué es un archivo de texto plano?

Universidad de SanAndrés

También existen archivos de texto sin formato de uso general, como es el caso de los archivos con extensión .**txt**.

info.txt

```
Es un archivo cuyos datos son todos
caracteres ASCII/Unicode. No incluyen
información extra como tipo de letra,
cursiva, negrita, viñetas, etc. No
requieren de un software especializado
para abrirlos, sólo un editor de texto
plano (bloc de notas, gedit, etc.).
Típicamente tienen líneas separadas por
el caracter salto de línea ('\n').
```

¿Qué es un archivo de texto plano?

Universidad de San Andrés

Otro ejemplo son los archivos CSV (Comma Separated Values) que permiten agrupar datos en forma de tablas separando las columnas con “comas” y las filas con “saltos de línea". La extensión es **.csv**.

	A	B	C	D	E
1	Departamento	ID	Nombre	Edad	Tarea
2	Producción	1	José	30	Mecánico
3	Producción	2	Julieta	35	Electrónica
4	Producción	3	Mario	40	Soldador
5	Administrativo	4	Nancy	30	RRHH
6	Administrativo	5	Miguel	45	Contador

tabla.csv

```
Departamento,ID,Nombre,Edad,Tarea
Producción,1,José,30,Mecánico
Producción,2,Julieta,35,Electrónica
Producción,3,Mario,40,Soldador
Administrativo,4,Nancy,30,RRHH
Administrativo,5,Miguel,45,Contador
```

¿Qué es un archivo de texto plano?

Universidad de San Andrés

Otro ejemplo son los archivos CSV (Comma Separated Values) que permiten agrupar datos en forma de tablas separando las columnas con “comas” y las filas con “saltos de línea". La extensión es **.csv**.

	A	B	C	D	E
1	Departamento	ID	Nombre	Edad	Tarea
2	Producción	1	José	30	Mecánico
3	Producción	2	Julieta	35	Electrónica
4	Producción	3	Mario	40	Soldador
5	Administrativo	4	Nancy	30	RRHH
6	Administrativo	5	Miguel	45	Contador

tabla.csv

```
Departamento,ID,Nombre,Edad,Tarea
Producción,1,José,30,Mecánico
Producción,2,Julieta,35,Electrónica
Producción,3,Mario,40,Soldador
Administrativo,4,Pedro,30,RRHH
Administrativo,5,Miguel,45,Contador
```

Este formato es apropiado para trabajar con una **hoja de cálculo** desde herramientas como excel, google sheet, OpenOffice Calc, etc.

Apertura de archivos

Apertura de archivos en python

```
f = open("file.txt", mode='r')
```

Objeto para manipular el archivo en python

Ruta/Nombre del archivo

Por ejemplo:

`/home/user/Lisa/week.csv`

`C:/users/Moe/beer.txt`

`../../sources/calc.py`

(default) r, w, x, a, **b, t**, +

- r → read
- w → write
- x → create
- a → append
- b → binario
- t → texto
- \+ → r+w

Apertura de archivos en python

Universidad de San Andrés

Modo	Descripción	Leer	Escribir	no existe archivo	sí existe archivo
`'r'`	Solo lectura	✅	❌	Error	Desde el inicio
`'w'`	Solo escritura	❌	✅	Lo crea	Trunca (borra)
`'a'`	Añadidura	❌	✅	Lo crea	Añade al final
`'x'`	Solo escritura	❌	✅	Lo crea	Error
`'r+'`	lectura + escritura	✅	✅	Error	Desde el inicio
`'w+'`	lectura + escritura	✅	✅	Lo crea	Trunca (borra)
`'a+'`	lectura + añadidura	✅	✅	Lo crea	Añade al final
`'x+'`	lectura + escritura	✅	✅	Lo crea	Error

Cierre de archivos

Cierre de archivos

```
f = open("ejemplo.txt")
# manipular archivo
# ...
f.close()
```

Una vez procesado el archivo, debe cerrarse aplicando el método **.close()** al objeto **f**

Es necesario cerrar el archivo cuando se deja de utilizar.

1. Hay sistemas operativos que no soportan abrir varias veces **un archivo al mismo tiempo**
2. Cambios que **no se guardan** hasta que el archivo **se cierra**.
3. **Límites** en la cantidad de **archivos abiertos**, entre otras.

Lectura de archivos de texto

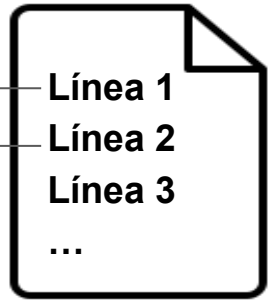
Lectura de archivos de texto – .readline()

```
f = open("ejemplo.txt")

linea = f.readline()
```

Lee una línea. Cada invocación lee una línea diferente. Devuelve la línea leída incluyendo \n.

```
linea1 = f.readline()
linea2 = f.readline()
```



Cada vez que se hace una lectura, devuelve un **str** con los caracteres hasta el primer **\n** que encuentra.
El **puntero** del archivo se ubica en la siguiente posición. Cuando no hay mas líneas por leer, devuleve un str vacío **""**.

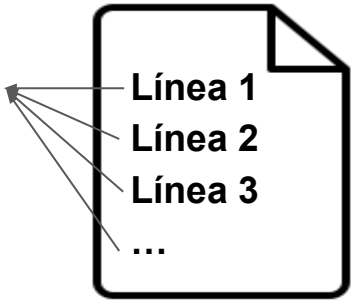
Lectura de archivos de texto – .readlines()

```
f = open("ejemplo.txt")

lineas = f.readlines()
```

Lee todo el contenido del archivo a una lista de cadenas (`list[str]`)

```
lineas = f.readlines()
```



En una sola llamada devuelve una lista de strings, donde cada elemento es una línea del archivo:

```
lineas = ["Linea 1\n",
"Linea 2\n", "Linea 3\n",
"Linea 4\n", ...]
```

Lectura de archivos de texto – Ejemplo de uso

Universidad de SanAndrés

```
archivo = open('nombre.txt', 'rt')

linea = archivo.readline()
print(f"Primera línea del archivo: '{linea}'")
while linea != '':
    # hacer algo con la línea
    linea = archivo.readline()
    print(f"Otra línea: '{linea}'")
archivo.close()
```

Lectura de archivos de texto – Ejemplo de uso

Universidad de SanAndrés

```
archivo = open('nombre.txt')

linea = archivo.readline()
print(f"Primera línea del archivo: '{linea}'")
while linea != '':
    # hacer algo con la línea
    linea = archivo.readline()
    print(f"Otra línea: '{linea}'")
archivo.close()
```

Lectura de archivos de texto – Ejemplo de uso

Puede leerse una determinada **cantidad de caracteres** por línea especificando el argumento de .readline().

```
archivo = open('nombre.txt')
n = 5 # largo máximo de la línea
linea = archivo.readline(n)
print(f"Primera línea del archivo: '
while linea != '':
    linea = archivo.readline(n)
    print(f"Línea con {n} caracteres máximo: '{linea}'")
archivo.close()
```

Lee hasta n caracteres o hasta el primer salto de línea.

Lectura de archivos de texto – Ejemplo de uso

Universidad de San Andrés

Los archivos en python soportan la **iteración** directamente, por lo que se puede escribir

```
archivo = open('nombre.txt', 'rt')

for linea in archivo:
    # hacer algo con la línea
    print(f"Línea: '{linea}'")
archivo.close()
```

Lectura de archivos de texto – Ejemplo de uso

Universidad de San Andrés

Usando .readlines() para obtener una lista con todas las líneas

```
archivo = open('nombre.txt', 'rt')

lista_de_lineas = archivo.readlines()

for linea in lista_de_lineas:
    # hacer algo con la línea
    print(f"Línea: '{linea}'")
archivo.close()
```

Métodos
strip(), rstrip(), lstrip()

strip(), rstrip(), lstrip() – Ejemplos

Universidad de SanAndrés

```
<cadena1>.strip(<caracteres>) → <cadena2>
<cadena1>.rstrip(<caracteres>) → <cadena2>
<cadena1>.lstrip(<caracteres>) → <cadena2>
```

Elimina en ambos extremos

Elimina solo desde la derecha

Elimina solo desde la izquierda

- Elimina de los extremos de la cadena los caracteres especificados
- Por defecto elimina los caracteres no imprimibles **\n**, **\t**, **\r** y **espacios** en blanco.

strip(), rstrip(), lstrip() – Ejemplos

```
cadena1 = "¿Qué está ocurriendo?"
print(cadena1.strip("¿?"))
print(cadena1.rstrip("¿?"))
print(cadena1.lstrip("¿?"))

cadena2 = "1) qué sigue? *+-##-!!"
print(cadena2.strip("1*+#-!) "))
print(cadena2.rstrip("1*+#-!) "))
print(cadena2.lstrip("1*+#-!) "))
```

```
Qué está ocurriendo
¿Qué está ocurriendo
Qué está ocurriendo?
```

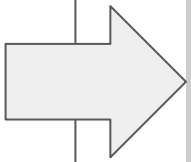
```
qué sigue?
1) qué sigue?
qué sigue? *+-##-!!
```

strip(), rstrip(), lstrip() – Ejemplos

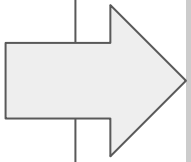
Universidad de San Andrés

```
cadena1 = "   Nueva línea\n"
print(f"-{cadena1.strip()}-")
print(f"-{cadena1.rstrip()}-")
print(f"-{cadena1.lstrip()}-")

cadena2 = "  \n  Otra cadena  "
print(f"-{cadena2.strip()}-")
print(f"-{cadena2.rstrip()}-")
print(f"-{cadena2.lstrip()}-")
```



```
-Nueva línea-
-   Nueva línea-
-Nueva línea
-
```



```
-Otra cadena-
-
  Otra cadena-
-Otra cadena  -
```

Operador de contexto

Operador de contexto

Universidad de San Andrés

Para evitar tener que abrir y cerrar archivos, una mejor práctica es utilizar el operador de contexto **with**:

```
with open("nombre.txt", "rt") as archivo:
    # hacer algo con el archivo

# y acá está automáticamente cerrado el archivo
```

Ejemplo de procesamiento

Universidad de San Andrés

Imprimir números de línea

```
archivo = open('nombre.txt', 'rt')
i = 1
for linea in archivo:
    linea = linea.rstrip('\n')
    print(f"{i}: {linea}")
    i += 1
archivo.close()
```

Ejemplo de procesamiento

Universidad de San Andrés

Mejores prácticas

```
with open('nombre.txt', 'rt') as archivo:
    i = 1
    for linea in archivo:
        linea = linea.rstrip('\n')
        print(f"{i}: {linea}")
        i += 1
```

Ejemplo de procesamiento

Universidad de San Andrés

Y todavía mejor . . .

```
with open('nombre.txt', 'rt') as archivo:
    for i, linea in enumerate(archivo, 1):
        print(f"{i}: {linea}", end="")
```

Escritura de archivos de texto

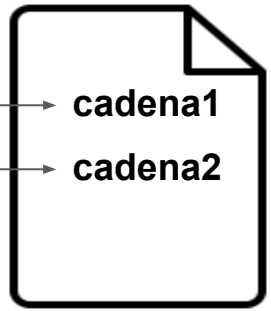
Escritura de archivos de texto

```
f = open("ejemplo.txt","w")

f.write(<str>)
```

Escribe la línea de texto en la posición actual del puntero

```
f.write("cadena1\n")
f.write("cadena2\n")
```



Escribe en el archivo la cadena pasada como argumento (no agrega '\n' al final, hay que agregarlo a mano).

Escritura de archivos de texto

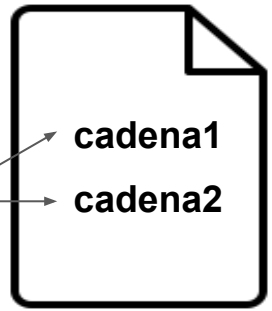
```
f = open("ejemplo.txt","w")

f.writelines(<list[str]>)
```

Escribe las líneas de texto desde la posición actual del puntero

```
lineas=["cadena1\n",
        "cadena2\n"]

f.writelines(lineas)
```



Escribe en el archivo las cadenas pasadas como lista de strings (hay que agregar a mano '\n' al final). Es similar a invocar .write() múltiples veces

Escritura de archivos de texto – Ejemplos

```
archivo = open("ejemplo1.txt","w")
archivo.write("Esta es una línea\n")
archivo.write("Esta es otra línea\n")
archivo.close()
```

ejemplo1.txt

```
Esta es una línea
Esta es otra línea
```

```
archivo = open("ejemplo2.txt","w")
lista_str = ["Esta es la línea 1\n",
             "Esta es la línea 2\n"]
archivo.writelines(lista_str)
archivo.close()
```

ejemplo2.txt

```
Esta es la línea 1
Esta es la línea 2
```

Escritura de archivos de texto – Ejemplo

Universidad de SanAndrés

generar.py

```
with open('numerar.py', 'wt') as dest:
    dest.write("with open('numerar.py', 'rt') as archivo:\n")
    dest.write("    for i, linea in enumerate(archivo, 1):\n")
    dest.write("        print(f'{i}: {linea}', end='')\n")
```

Mirar y ejecutar el contenido del archivo **"numerar.py"**.

Añadidura: un ejemplo para “*loguear*”

Universidad de San Andrés

- *Loguear* es el proceso mediante el cual vamos **registrando** en un archivo de texto (llamado log, o bitácora), lo que va haciendo el programa.
- Es similar a usar print para depurar, pero almacenando lo que se imprime en un archivo de texto.
- Vamos a crear un módulo propio, llamado log (`log.py`), que nos dará la posibilidad de loggear de una manera sencilla.

Módulo: log.py

Universidad de SanAndrés

```python
import datetime

def guardar(archivo, msj: str):
    tiempo_actual = datetime.datetime.now()
    archivo.write(f"{tiempo_actual} {msj}\n")

def abrir(nombre: str):
    archivo = open(nombre, "at")
    guardar(archivo, "INFO Inicia nueva sesión de registro de evento")
    return archivo

def cerrar(archivo):
    guardar(archivo, "INFO Fin de la sesión")
    archivo.close()
```

Ejemplo de uso

Universidad de SanAndrés

```
import log

archivo_log = log.abrir("mi_app.log")
# ...
# hacer cosas que puedan dar error

if error:
    log.guardar(archivo_log, f"ERROR {error}")
# ...

# Cerrar el archivo al finalizar
log.cerrar(archivo_log)
```