Strings

Clase #10

IIC1103 – Introducción a la Programación

Marcos Sepúlveda (marcos@ing.puc.cl)

Veremos hoy …

- Código ASCII
- Formato de strings
- Ejercicios

Código ASCII

- Es un estándar de codificación de caracteres
  - Utiliza 7 bits para representar los caracteres

- Define códigos para 32 caracteres no imprimibles, de los cuales la mayoría son caracteres de control obsoletos que tienen efecto sobre cómo se procesa el texto, más otros 95 caracteres imprimibles que les siguen en la numeración (empezando por el caracter espacio).

- A menudo se llama incorrectamente ASCII a otros códigos de caracteres de 8 bits, como el estándar ISO-8859-1, que es una extensión que utiliza 8 bits para proporcionar caracteres adicionales usados en idiomas distintos al inglés, como el español.

Fuente: http://es.wikipedia.org/wiki/ASCII

Código ASCII

Ctrl	Dec	Hex	Char	Code
^@	0	00		NUL
^A	1	01		SOH
^B	2	02		STX
^C	3	03		ETX
^D	4	04		EOT
^E	5	05		ENQ
^F	6	06		ACK
^G	7	07		BEL
^H	8	08		BS
^I	9	09		HT
^J	10	0A		LF
^K	11	0B		VT
^L	12	0C		FF
^M	13	0D		CR
^N	14	0E		SO
^O	15	0F		SI
^P	16	10		DLE
^Q	17	11		DC1
^R	18	12		DC2
^S	19	13		DC3
^T	20	14		DC4
^U	21	15		NAK
^V	22	16		SYN
^W	23	17		ETB
^X	24	18		CAN
^Y	25	19		EM
^Z	26	1A		SUB
^[	27	1B		ESC
^\	28	1C		FS
^]	29	1D		GS
^^	30	1E	▲	RS
^-	31	1F	▼	US

Dec	Hex	Char
32	20	
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

Dec	Hex	Char
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Char
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	⌂

* ASCII code 127 has the code DEL. Under MS-DOS, this code has the same effect as ASCII 8 (BS). The DEL code can be generated by the CTRL + BKSP key.

Fuente: http://msdn.microsoft.com/en-US/library/60ecse8t(v=vs.80).aspx

Código ASCII

Dec	Hex	Char
128	80	Ç
129	81	ü
130	82	é
131	83	â
132	84	ä
133	85	à
134	86	å
135	87	ç
136	88	ê
137	89	ë
138	8A	è
139	8B	ï
140	8C	î
141	8D	ì
142	8E	Ä
143	8F	Å
144	90	É
145	91	æ
146	92	Æ
147	93	ô
148	94	ö
149	95	ò
150	96	û
151	97	ù
152	98	ÿ
153	99	Ö
154	9A	Ü
155	9B	¢
156	9C	£
157	9D	¥
158	9E	₧
159	9F	ƒ

Dec	Hex	Char
160	A0	á
161	A1	í
162	A2	ó
163	A3	ú
164	A4	ñ
165	A5	Ñ
166	A6	ª
167	A7	º
168	A8	¿
169	A9	⌐
170	AA	¬
171	AB	½
172	AC	¼
173	AD	¡
174	AE	«
175	AF	»
176	B0	░
177	B1	▒
178	B2	▓
179	B3	│
180	B4	┤
181	B5	╡
182	B6	╢
183	B7	╖
184	B8	╕
185	B9	╣
186	BA	║
187	BB	╗
188	BC	╝
189	BD	╜
190	BE	╛
191	BF	┐

Dec	Hex	Char
192	C0	└
193	C1	┴
194	C2	┬
195	C3	├
196	C4	─
197	C5	┼
198	C6	╞
199	C7	╟
200	C8	╚
201	C9	╔
202	CA	╩
203	CB	╦
204	CC	╠
205	CD	═
206	CE	╬
207	CF	╧
208	D0	╨
209	D1	╤
210	D2	╥
211	D3	╙
212	D4	╘
213	D5	╒
214	D6	╓
215	D7	╫
216	D8	╪
217	D9	┘
218	DA	┌
219	DB	█
220	DC	▄
221	DD	▌
222	DE	▐
223	DF	▀

Dec	Hex	Char
224	E0	α
225	E1	ß
226	E2	Γ
227	E3	π
228	E4	Σ
229	E5	σ
230	E6	µ
231	E7	τ
232	E8	Φ
233	E9	Θ
234	EA	Ω
235	EB	δ
236	EC	∞
237	ED	φ
238	EE	ε
239	EF	∩
240	F0	≡
241	F1	±
242	F2	≥
243	F3	≤
244	F4	⌠
245	F5	⌡
246	F6	÷
247	F7	≈
248	F8	°
249	F9	∙
250	FA	·
251	FB	√
252	FC	ⁿ
253	FD	²
254	FE	■
255	FF	

Fuente: http://msdn.microsoft.com/en-US/library/9hxt0028(v=vs.80).aspx

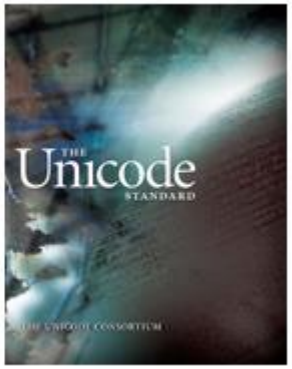
Código Unicode

- Es un estándar de codificación de caracteres diseñado para facilitar el tratamiento informático, transmisión y visualización de textos de múltiples lenguajes y disciplinas técnicas.

- Este estándar es mantenido por el *Unicode Consortium*.

- Ver detalle del código en:
  - http://www.unicode.org/charts

Unicode provides a unique number for every character,
no matter what the platform,
no matter what the program,
no matter what the language.



Fuente: http://www.unicode.org/standard/WhatIsUnicode.html

Fuente: http://es.wikipedia.org/wiki/Unicode

Funciones para trabajar con códigos ASCII (o Unicode)

- **`chr(i)`**
  - Retorna el string que representa un caracter cuyo código Unicode es el entero **`i`**.
  - Ejemplo: chr (97) retorna el string 'a'.
  - El rango válido para el argumento es de 0 a 1.114.111 (0x10ffff en base 16).

- **`ord(c)`**
  - Retorna el entero que representa el código Unicode del caracter **`c`**.
  - Ejemplo: ord ('a') retorna el entero 97.

Ejemplo

- ¿Qué muestra el siguiente código?

```
import sys

# chr, ord
c = 'a'
print(c, ord(c))

cod = 97
print(cod, chr(cod))

# códigos ASCII
print("Código ASCII")
for i in range(0,256):
    print(i, chr(i), end="\t")
    if i%8 == 0:
        print()
```

¿Cómo es mi string?

- **`s.isalnum()`**
  - **`True`** si hay al menos un caracter y todos los caracteres son alfanuméricos.
- **`s.isalpha()`**
  - **`True`** si hay al menos un caracter y todos los caracteres son alfabéticos.
- **`s.isdecimal()`**
  - **`True`** si sólo hay caracteres decimales (numéricos).
- **`s.islower()`**
  - **`True`** si hay al menos un caracter y todos los caracteres son letras minúsculas.
- **`s.isspace()`**
  - **`True`** si hay al menos un caracter y todos los caracteres pueden ser considerados espacios en blanco.
- **`s.istitle()`**
  - **`True`** si hay al menos un caracter y puede ser considerado un título: todas las palabras empiezan con mayúscula y los demás caracteres están en minúsculas.
- **`s.isupper()`**
  - **`True`** si hay al menos un caracter y todos los caracteres son letras mayúsculas.

¿Cómo es mi string?

```
def como_es_mi_string(s):
    print("--------")
    print(repr(s))
    print("isalnum():", s.isalnum())
    print("isalpha():", s.isalpha())
    print("isdecimal():", s.isdecimal())
    print("islower():", s.islower())
    print("isspace():", s.isspace())
    print("istitle():", s.istitle())
    print("isupper():", s.isupper())
    print("--------")

como_es_mi_string("Hola")
como_es_mi_string("CHAO")
como_es_mi_string("hola y chao")
como_es_mi_string("")
como_es_mi_string(" ")
como_es_mi_string(" \t\n  \n")
como_es_mi_string("1234")
como_es_mi_string("juan1234")
```

```
>>>
--------
'Hola'
isalnum(): True
isalpha(): True
isdecimal(): False
islower(): False
isspace(): False
istitle(): True
isupper(): False
--------
--------
'CHAO'
isalnum(): True
isalpha(): True
isdecimal(): False
islower(): False
isspace(): False
istitle(): False
isupper(): True
--------

...
```

Formato en print

- Se puede dar formato al texto que se despliega usando ***print***

- Para escribir números, se utilizan las siguientes secuencias:
  - **`%d`** – formato decimal
  - **`%o`** – formato octal (base 8)
  - **`%x`** – formato hexadecimal (base 16)
  - **`%E`** – formato en Notación Científica
  - **`%f`** – formato de punto flotante

Ejemplo

- Programa en Python que muestra los número del 0 al 20 en formato: decimal, octal y hexadecimal.

```
print("Decimal","\t Octal","\t Hexadecimal")
for i in range(0,21):
    print("%d" %i, "\t %o" %i, "\t %X" %i)
```

```
>>>
Decimal    Octal     Hexadecimal
0          0         0
1          1         1
2          2         2
3          3         3
4          4         4
5          5         5
6          6         6
7          7         7
8          10        8
9          11        9
10         12        A
11         13        B
12         14        C
13         15        D
14         16        E
15         17        F
16         20        10
17         21        11
18         22        12
19         23        13
20         24        14
```

Ejemplo

- Programa en Python que muestra el número **π** en distintas notaciones de punto flotante.

```
import math

print("Pi:", "%f" %math.pi)
print("Pi:", "%E" %math.pi)

print("1000*Pi:", "%.8f" %(1000*math.pi))
print("1000*Pi:", "%.8E" %(1000*math.pi))

print("1000*Pi:", "%20.8f" %(1000*math.pi))
print("1000*Pi:", "%20.8E" %(1000*math.pi))
```

```
>>>
Pi: 3.141593
Pi: 3.141593E+00
1000*Pi: 3141.59265359
1000*Pi: 3.14159265E+03
1000*Pi:         3141.59265359
1000*Pi:       3.14159265E+03
```

`str.format()`

- Los strings de formato contienen “campos de reemplazo” rodeadas por llaves `{}`.
- Todo lo que no está contenido entre las llaves se considera el texto literal, que se copia sin cambios a la salida.
- Si se requiere incluir un caracter de llave en el texto literal, basta con duplicar la llave correspondiente: `{{` o `}}`.

`str.format()` – ejemplos

```
print("Yo me llamo {} y tengo {} años".format("Juan","18"))
print("{0} con {1}".format("arroz","leche"))
print("{1} con {0}".format("arroz","leche"))
print("El curso {codigo} es muy {adjetivo}".format(
    codigo="IIC1103",
    adjetivo="entretenido"))
```

```
>>>
Yo me llamo Juan y tengo 18 años
arroz con leche
leche con arroz
El curso IIC1103 es muy entretenido
```

`str.format()`

- Para escribir números, se utilizan las siguientes secuencias:
  - `:d` – formato decimal
  - `:b` – formato binario (base 2)
  - `:o` – formato octal (base 8)
  - `:x` – formato hexadecimal (base 16, con letras minúsculas)
  - `:X` – formato hexadecimal (base 16, con letras mayúsculas)
  - `:e` – formato en Notación Científica (con e minúscula)
  - `:E` – formato en Notación Científica (con E mayúscula)
  - `:f` – formato de punto flotante

`str.format()` – ejemplo

- Programa en Python que muestra los número del 0 al 20 en formato: decimal, binario, octal y hexadecimal.

```
print("Decimal\tBinario\tOctal\tHexadecimal")
for i in range(0,21):
    print("{:d}\t{:b}\t{:o}\t{:X}".format(i, i, i, i))
```

```
>>>
Decimal  Binario  Octal    Hexadecimal
0        0        0        0
1        1        1        1
2        10       2        2
3        11       3        3
4        100      4        4
5        101      5        5
6        110      6        6
7        111      7        7
8        1000     10       8
9        1001     11       9
10       1010     12       A
11       1011     13       B
12       1100     14       C
13       1101     15       D
14       1110     16       E
15       1111     17       F
16       10000    20       10
17       10001    21       11
18       10010    22       12
19       10011    23       13
20       10100    24       14
```

`str.format()` – ejemplo

- Programa en Python que muestra el número **π** en distintas notaciones de punto flotante.

```
import math

print("Pi: {:f}".format(math.pi))
print("Pi: {:E}".format(math.pi))

print("1000*Pi: {:.8f}".format(1000*math.pi))
print("1000*Pi: {:.8E}".format(1000*math.pi))

print("1000*Pi: {:20.8f}".format(1000*math.pi))
print("1000*Pi: {:20.8E}".format(1000*math.pi))
```

```
>>>
Pi: 3.141593
Pi: 3.141593E+00
1000*Pi: 3141.59265359
1000*Pi: 3.14159265E+03
1000*Pi:         3141.59265359
1000*Pi:       3.14159265E+03
```

`str.format()` – más información

- https://docs.python.org/3/library/string.html#formatstrings
- https://docs.python.org/3/tutorial/inputoutput.html