Métodos de Strings

Clase #14

IIC1103 – Introducción a la Programación

El plan de hoy es...

- ¿Cómo van con el compilado?
  - ➔ Problemas: vherskov@ing.puc.cl
  - También peticiones de resolución de ejercicios
- ¿Cómo van con la tarea 1?

Menti

- x = ‘abracadabra’

a	b	r	a	c	a	d	a	b	r	a
0	1	2	3	4	5	6	7	8	9	10

- x[4]

a	b	r	a	c	a	d	a	b	r	a
0	1	2	3	4	5	6	7	8	9	10

- x[2:6]

a	b	r	a	c	a	d	a	b	r	a
0	1	2	3	4	5	6	7	8	9	10

- x[0:3] ó x[:3]

a	b	r	a	c	a	d	a	b	r	a
0	1	2	3	4	5	6	7	8	9	10

- x[5:11] ó x[5:]

a	b	r	a	c	a	d	a	b	r	a
0	1	2	3	4	5	6	7	8	9	10

x[1:9:2]

a	b	r	a	c	a	d	a	b	r	a
0	1	2	3	4	5	6	7	8	9	10

- x[:6:4]

a	b	r	a	c	a	d	a	b	r	a
0	1	2	3	4	5	6	7	8	9	10

- x[6::3]

a	b	r	a	c	a	d	a	b	r	a
0	1	2	3	4	5	6	7	8	9	10

- x[::5]

a	b	r	a	c	a	d	a	b	r	a
0	1	2	3	4	5	6	7	8	9	10

Problema #1

- Escribe una función que cuente cuántas veces aparece un caracter en un string

- `def contar(x,y): #cuenta apariciones de y en x`

- Ejemplos:
- `contar(“abracadabra”,”a”) entrega 5`
- `contar(“abracadabra”,”x”) entrega 0`

Solución

```
def contar(x,y):
    i = 0
    contador=0
    while i<len(x):
        if x[i]==y:
            contador+=1
        i+=1
    return contador

print(contar("abracadabra","a"))
print(contar("abracadabra","b"))
print(contar("abracadabra","x"))
```

for

- Itera sobre los ítemes de una secuencia (lista o **string**)

- `palabra=“supercalifragilisticoexpialidoso”`
- `for letra in palabra:`
- `    print(letra)`

Solución

```
def contar(x,y):
    contador=0
    for letra in x:
        if letra==y:
            contador+=1
    return contador
```

```
print(contar("abracadabra","a"))
print(contar("abracadabra","b"))
print(contar("abracadabra","x"))
```

Str: s.count(x)

- Entrega: el número de ocurrencias de x en s entre el **índice i, hasta j-1 (i y j son opcionales)**.

- `s="abracadabra"`
- `n1 = s.count("a")`
- `n2 = s.count("a",3)`
- `n3 = s.count("a",3,5)`

Problema #2

- Escribe una función que entregue la posición de un string dentro de otro

- `def buscar(x,y): #busca primera aparición de y en x`

- Ejemplos:
- `buscar(“abracadabra”,”a”) entrega 0`
- `buscar(“abracadabra”,”rac”) entrega 2`
- `buscar(“abracadabra”,”rb”) entrega -1`

Solución

```
def buscar(x,y):
    i=0
    while i<=len(x):
        if x[i:i+len(y)]==y:
            return i
        i+=1
    return -1

print(buscar("abracadabra","a"))
print(buscar("abracadabra","rac"))
print(buscar("abracadabra","rb"))
```

Str: s.find(x,i,j)

- Entrega: el menor índice en s donde el substring x está, **a partir del índice i, hasta j-1 (i,j opcionales)**.
- -1 si s no contiene x.

- `s="abracadabra"`
- `n1 = s.find("b")`
- `n2 = s.find("ra",4)`
- `n1 = s.find("a",1,3)`

Problema #3

- Escribe una función que ordene alfabéticamente 3 palabras. Ejemplo:
- `ordenar("casa", "barco", "beeee")` imprime: `"barco", "beeee", "casa".`

Solución

```
def ordenar(x,y,z):
    if x<y<z:
        print(x+","+y+","+z)
    elif x<z<y:
        print(x+","+z+","+y)
    elif y<x<z:
        print(y+","+x+","+z)
    elif y<z<x:
        print(y+","+z+","+x)
    elif x<y: #z debe ser la menor
        print(z+","+x+","+y)
    else:
        print(z+","+y+","+x)
```

Ojo!

- `a=input("palabra 1?") #Casa`
- `b=input("palabra 2?") #arbol`
- `c=input("palabra 3?") #Barco`
- `ordenar(a,b,c)`

A<…<Z<…<a…<z

- Hay que tener en cuenta que en el orden alfabético, están todas las mayúsculas antes de todas las minúsculas

¿Por qué?

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Conversiones

http://docs.python.org/3/library/functions.html

chr(*i*)

Return the string representing a character whose Unicode codepoint is the integer *i*. For example, `chr(97)` returns the string `'a'`. This is the inverse of `ord()`. The valid range for the argument is from 0 through 1,114,111 (0x10FFFF in base 16). `ValueError` will be raised if *i* is outside that range.

ord(*c*)

Given a string representing one Unicode character, return an integer representing the Unicode code point of that character. For example, `ord('a')` returns the integer `97` and `ord('\u2020')` returns `8224`. This is the inverse of `chr()`.

¿Y cómo lo soluciono?

- `a=input("palabra 1?")`
- `b=input("palabra 2?")`
- `c=input("palabra 3?")`
- **`a=a.lower()`**
- **`b=b.lower()`**
- **`c=c.lower()`**
- `ordenar(a,b,c)`

- `a=input("palabra 1?")`
- `b=input("palabra 2?")`
- `c=input("palabra 3?")`

- `ordenar(`**`a.lower(),b.lower(),c.lower()`**`)`

Str: s.lower()

- Entrega: string, con mayúsculas reemplazadas por minúsculas

- `s1=“Hola Como EstaS?”`
- `s2=s1.lower()`

Str: s.islower()

- Entrega: bool (True si el texto está en minúsculas, False si no)

- `s1="hola amigo!!!!"`
- `print(s1.islower())`

Str: s.upper()

- Entrega: string, con minúsculas reemplazadas por mayúsculas

- `s1="Hola Como EstaS?"`
- `s2=s1.upper()`

Str: s.isupper()

- Entrega: bool (True si el texto está en mayúsculas, False si no)

- `s1="HOLA COMO ESTAS?"`
- `print(s1.isupper())`

¿Qué pasa?

- palabra1? barco
- palabra2? casa
- palabra3? arbol

Str: s.strip([chars])

- Elimina los caracteres del principio y fin del string

- `s=“ abrac adabra ”`
- `s1 = s.strip()`
- `s2 = s.strip(“a “)`

x in s: ¿es un string parte de otro?

- ¿Es x un substring de s?
  - Entrega True o False
- Ejemplos
  - “m” in “manzana”
  - “i” in “manzana”
  - “za” in “manzana”
  - ‘az’ in ‘manzana’
  - ‘manzana’ in ‘manzana’

Métodos de string

Método	Explicación
`s.count(x,[i,j])`	Cuenta apariciones de x en s (opcional: entre i y j)
`s.endswith(x,[i,j])`	Entrega True si s termina con x (opcional: entre i y j)
`s.find(x,[i,j])`	Entrega posición de x dentro de s (opcional: entre i y j)
`s.upper()`	Entrega s en mayúsuculas
`s.lower()`	Entrega s en minúsculas
`s.replace(x,y,[count])`	Entrega nuevo string donde se reemplazan apariciones de x por y (opcional: cuantas)
`s.startswith(x.[i,j])`	Entrega True si s comienza por x (opcional: entre i y j)
`s.strip([chars])`	Borra los espacios y \n desde el comienzo y final del string (opcional: especificar cuales caracteres borrar)

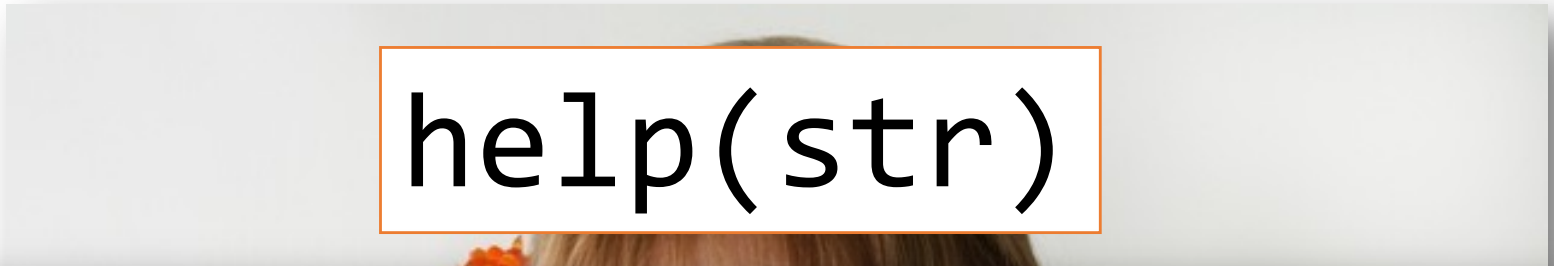
Métodos de string: ¿Cómo es el string?

Método	Explicación
`s.isalnum()`	True if all characters are alphanumeric and there is at least one character
`s.isalpha()`	True if all characters are alphabetic and there is at least one character
`s.isdecimal()`	True if there are only decimal characters
`s.islower()`	True if all lowercase and at least one cased character
`s.isspace()`	True if all characters are whitespace and there is at least one character
`s.istitle()`	True if titlecased string and at least one character
`s.isupper()`	True if all uppercase and at least one cased character

Ejemplos

s	isalnum	isalpha	isdecimal	islower	isspace	istitle	isupper
“”	False	False	False	False	False	False	False
“ “	False	False	False	False	True	False	False
“Hola”	True	True	False	False	False	True	False
“hola y chao”	False	False	False	True	False	False	False
“1234”	True	False	True	False	False	False	False
“JUANITO123”	True	False	False	False	False	False	True

¡Quiero más métodos de strings!



`help(str)`

```
capitalize(...)
    S.capitalize() -> str

    Return a capitalized version of S, i.e. make the first character
    have upper case and the rest lower case.

casefold(...)
    S.casefold() -> str

    Return a version of S suitable for caseless comparisons.

center(...)
    S.center(width[, fillchar]) -> str

    Return S centered in a string of length width. Padding is
    done using the specified fill character (default is a space)

count(...)
    S.count(sub[, start[, end]]) -> int

    Return the number of non-overlapping occurrences of substring sub in
    string S[start:end].  Optional arguments start and end are
    interpreted as in slice notation.
```