

Strings

Clase #09

IIC1103 – Introducción a la Programación

Marcos Sepúlveda (marcos@ing.puc.cl)

Veremos hoy ...

- ▶ String
- ▶ Funciones y operadores básicos de String

¿Qué es un string?

- ▶ Un string es una secuencia de caracteres (un texto).
- ▶ ¿Para qué sirve?
 - Para hacer programas que manipulen texto
- ▶ Ejemplo: "Esto es un string"

Ejemplos de creación de strings

- Inicialización de un string

```
s = "Esto es un string"
```

- Crear un string a partir de otros; operador +:

```
s1 = "Hola, "
```

```
s2 = "¿qué tal?"
```

```
s3 = s1 + s2
```

- Crear un string a partir de otro; operador *:

```
s4 = "ABC-"*5
```

```
s5 = 5*"ABC-"
```

- Inicialización de un string con caracteres especiales

```
s6 = "[\tMensaje: \"Hola\"\n]"
```

Algunas funciones básicas

- Longitud de un string

```
len(s)
```

- Obtener carácter que se encuentra en una posición cualquiera (los índices van desde 0 a `len(s) - 1`)

```
s[i] # 0 <= i <= len(s) - 1
```

- Recorrer un string

```
s = "Hola, ¿qué tal?"
```

```
i = 0
```

```
while i < len(s):
```

```
    print("[", i, "] = ", s[i], sep='')
```

```
    i += 1
```

- Comparación de strings

```
s = "Hola"
```

```
r = "hola"
```

```
print(s == r) # False
```

```
print(s < r) # True; todas las mayúsculas vienen antes de las minúsculas
```

```
print(s > r) # False
```

Sobre índices en un string

- ▶ Los índices van desde 0 a `len(s) - 1`
`s[i]` # `0 <= i <= len(s) - 1`
- ▶ Un índice negativo, indica un índice desde el final del string. En dicho caso, los índices van desde `-1` a `-len(s)`
`s[i]` # `-len(s) <= i <= -1`
- ▶ En base a ello, se puede recorrer un string en dirección opuesta:

```
s = "Hola, ¿qué tal?"  
i = -1  
while i >= -len(s):  
    print("[", i, "] = ", s[i], sep='')  
    i -= 1
```

	H	o	l	a	,		¿	q	u	é		t	a	l	?
Índice positivo	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Índice negativo	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Ejercicio 1 – contar letras

- ¿Cuántas veces aparece la letra 'a' en la palabra "banana"?

```
s = "banana"
i = 0
cuentaA = 0
while i < len(s):
    if (s[i] == 'a'):
        cuentaA += 1
    i += 1
print("Hay", cuentaA, "letras 'a' en", s)
```

```
>>>
```

```
Hay 3 letras 'a' en banana
```

Ejercicio 2 – contar palabras

- ▶ ¿Cómo podemos contar las palabras correspondiente a una frase?
- ▶ Ejemplo: ¿Cuántas palabras tiene la siguiente frase?
 - “El 18 será espectacular; 4 días para estudiar Python”

```
s = "El 18 será espectacular; 4 días para estudiar Python"
i = 0
nPalabras = 0
while i < len(s):
    if (s[i] == ' '):
        nPalabras += 1
    i += 1
nPalabras += 1
print("Hay " + str(nPalabras) + " palabras en \"" + s + "\"")
```

```
>>>
```

```
Hay 9 palabras en "El 18 será espectacular; 4 días para estudiar Python"
```


Ejercicio 3 – mostrar palabras

- ▶ ¿Cómo podemos desplegar las palabras correspondiente a una frase?
- ▶ Ejemplo: ¿Cuáles son las palabras de la siguiente frase?
 - “El 18 será espectacular; 4 días para estudiar Python”

```
s = "El 18 será espectacular; 4 días para estudiar Python"
```

```
i = 0
nPalabras = 0
palabra = ""

while i < len(s):
    if (s[i] != ' '):
        palabra += s[i]
    else:
        nPalabras += 1
        print(nPalabras, "-", palabra)
        palabra = ""
    i += 1

nPalabras += 1
print(nPalabras, "-", palabra)
```

```
>>>
1 - El
2 - 18
3 - será
4 - espectacular;
5 - 4
6 - días
7 - para
8 - estudiar
9 - Python
```

Substring

- ▶ Un substring es un segmento de un string
- ▶ Para obtener un substring se utiliza:

```
s = "Hola, ¿qué tal?"  
s1 = s[:4]   # "Hola"  
s2 = s[4:6]  # ", "  
s3 = s[6:]   # "¿qué tal?"
```
- ▶ El operador [n:m] devuelve un substring desde el carácter “n-ésimo” hasta el carácter “m-ésimo”, incluyendo el primer carácter y excluyendo el último.
- ▶ Si se omite el primer índice, el substring se inicia en el comienzo del string.
- ▶ Si se omite el segundo índice, el substring llega hasta el final del string.

	H	o	l	a	,		¿	q	u	é		t	a	l	?
Índice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Los strings son “inmutable”

- ▶ No se puede modificar su contenido de manera directa con el operador []

- ▶ No podemos hacer:

```
>>> s = "Hola, ¿qué tal?"
```

```
>>> s[0] = 'h'
```

```
TypeError: 'str' object does not support item assignment
```

¡ no funciona !

- ▶ La alternativa es crear un nuevo string:

```
s = "Hola, ¿qué tal?"
```

```
s = 'h'+s[1:] # "hola, ¿qué tal?"
```



Búsqueda

- Escribir una función que busque si un caracter (ej: 'a') está contenido en un string. Retorna el índice donde se encuentra la primera aparición o -1 si no lo encontró.

```
def buscar_c(s, c):  
    i = 0  
    while i < len(s):  
        if s[i] == c:  
            return i  
        i += 1  
    return -1
```

```
s = "Hola, ¿qué tal?"  
print(s)
```

```
c = ','  
pos = buscar_c(s, c)  
print("'" + c + "' está en la posición " + str(pos))
```

```
c = 'z'  
pos = buscar_c(s, c)  
print("'" + c + "' está en la posición " + str(pos))
```

```
>>>  
Hola, ¿qué tal?  
' ,' está en la posición 4  
'z' está en la posición -1
```

Métodos de string

- ▶ Un **método** es similar a una función (recibe parámetros y retorna un valor), pero la sintaxis es diferente. Por ejemplo, el método **upper** recibe un string y retorna un nuevo string en que todas las letras del original están en mayúsculas.
- ▶ En lugar de la sintaxis de una función: **upper(s)**, se utiliza la sintaxis de un método: **s.upper()**.

```
s = "Hola, ¿qué tal?"  
print("s = " + s)  
t = s.upper()  
print("t = " + t)
```

```
>>>  
s = Hola, ¿qué tal?  
t = HOLA, ¿QUÉ TAL?
```

Métodos de string

- Podemos usar métodos para otros fines. Por ejemplo, podemos buscar un carácter en el string usando el método **find()**.

```
s = "Hola, ¿qué tal?"
```

```
c = ','
```

```
pos = s.find(c)
```

```
print("'" + c + "' está en la posición " + str(pos))
```

```
c = 'z'
```

```
pos = s.find(c)
```

```
print("'" + c + "' está en la posición " + str(pos))
```

```
>>>  
Hola, ¿qué tal?  
, está en la posición 4  
z está en la posición -1
```

Más métodos

► Ver referencia de Python:

- <https://docs.python.org/3/library/stdtypes.html#string-methods>

► Ejemplos:

```
s = "hola, ¿Qué tal?"  
print( s )  
print( s.capitalize() )  
print( s.lower() )  
print( s.upper() )  
print( str(s.find("Qué")) )  
print( s.replace("Qué tal", "Cómo te va") )
```

```
t = "    mejor sacar los espacios    "  
print( t.strip() )
```

```
>>>  
hola, ¿Qué tal?  
Hola, ¿qué tal?  
hola, ¿qué tal?  
HOLA, ¿QUÉ TAL?  
7  
hola, ¿Cómo te va?  
mejor sacar los espacios
```

Operador in

- ▶ **in** es un operador booleano que toma dos string y devuelve **True** si el primero aparece como substring en el segundo.

- ▶ Ejemplo:

```
s = "hola, ¿Qué tal?"
```

```
print("Qué" in s)
```

```
print("qué" in s)
```

```
>>>  
True  
False
```


Recorrer string con *while* y *for*

- ▶ Ya vimos como recorrer un string usando **while**:

```
s = "Hola, ¿qué tal?"
```

```
i = 0
```

```
while i < len(s):  
    print(s[i])  
    i += 1
```

- ▶ Otra forma es usando un ciclo **for**:

```
s = "Hola, ¿qué tal?"
```

```
for c in s:  
    print(c)
```

```
>>>  
H  
o  
l  
a  
,  
  
¿  
q  
u  
é  
  
t  
a  
l  
?
```

Ejemplo *for*

- Combinar prefijos con un sufijo dado:

```
prefijos = "CLMNPT"  
sufijo = "apa"  
  
for c in prefijos:  
    print(c + sufijo)
```

```
>>>  
Capa  
Lapa  
Mapa  
Napa  
Papa  
Tapa
```

- Ejercicio:

- Cambiar códigos anteriores usando **for**
- ¿Quedan más compactos?

Ejercicios string

- ▶ Escriba una función en Python de nombre **hay_mayores** que reciba como parámetro una frase (un string) y un número entero k, y que retorne True o False dependiendo si hay alguna palabra en la frase tiene longitud mayor a k.
- ▶ Escriba una función en Python de nombre **cuenta_numeros** que reciba como parámetro una frase (un string) y retorne cuántos números tiene esa frase. Por ejemplo, si la función recibe como parámetro la frase: “acá hay un 1, un 2, más 205, y nada más”, debe retornar 3.
- ▶ Escriba una función de nombre **pali** que reciba como parámetro un string y retorne True o False dependiendo si el string es o no un palíndromo. Un string se dice que es un palíndromo si se lee igual de izquierda a derecha, que de derecha a izquierda.
 - Ejemplo: “amo roma”
 - Se lee igual de izquierda a derecha que de derecha a izquierda (omitimos la lectura de los espacios en blanco).