

# Fundamentos de programación

## Manejo de texto

# Strings

---

Fundamentos de Programación

- Un string es una secuencia de caracteres individuales.
- En Python se define usando comillas simples o dobles:

```
movie = "La vida es bella"
```

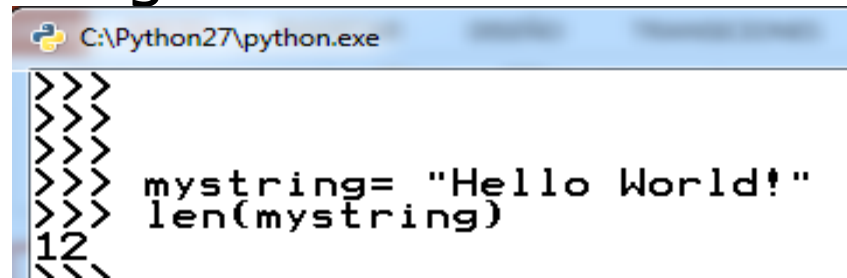
```
ciudad = 'Barcelona'
```

```
mystr = ''' Usando tres comillas  
           es posible crear texto en varias lineas'''
```

# Operaciones básicas

Fundamentos de Programación

- `len(str)`
  - Retorna la longitud de la variable de texto



A screenshot of a Python interpreter window titled 'C:\Python27\python.exe'. The window shows a series of prompt characters '>>>' followed by the code: `mystring= "Hello World!"` and `len(mystring)`. The output of the code is `12`.

- `str1 + str2`
  - Concatena 2 strings
- `str1 in str2`
  - Verifica si el texto de `str1` está contenido en `str2`

# Strings: Indices

- Los caracteres de un string se pueden acceder individualmente usando un índice entre corchetes.
- `variableName [ index ]`
- El primer índice comienza en cero. El último índice es la longitud del string - 1

```
>>> ciudad = "Manta"
>>> ciudad[0]
'M'
>>> ciudad[1]
'a'
>>> ciudad[2]
'n'
>>> ciudad[3]
't'
>>> ciudad[4]
'a'
>>>
```

índice	0	1	2	3	4
caracter	M	a	n	t	a

# Strings: Indices

Fundamentos de Programación

- Al indexar strings, se pueden especificar índices negativos para contar desde el fin.

```
>>>  
>>> ciudad = "Manta"  
>>> ciudad[-1]  
,a,  
>>> ciudad[-2]  
,t,  
>>> ciudad[-3]  
,n,  
>>> ciudad[-4]  
,a,  
>>> ciudad[-5]  
,M,  
>>>
```

índice	-5	-4	-3	-2	-1
caracter	M	a	n	t	a

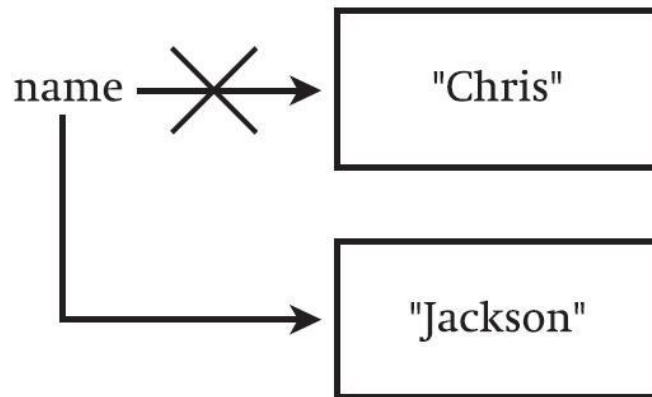
# String slicing

- `variableName[start:stop:step]`
  - `start`: índice del primer carácter a seleccionar
  - `stop`: índice del carácter posterior al último seleccionado
  - `step`: cuantos caracteres saltar. Si el `step` es negativo, se recorre el string de derecha a izquierda
- 
- `a = "abcdefghi"`
  - `a[:] = a[0:len(a):1] = 'abcbdefghi'` # a +1 step is the default
  - `a[::2] = a[0:len(a):2] = 'acegi'` # posiciones pares
  - `a[1::2] = 'bdfh'`
  - `a[::-1] = 'ihgfedcba'`

# String Immutability

Fundamentos de Programación

- Los strings no se pueden modificar una vez definidos.
- Cuando se concatenan dos strings, el sistema crea una variable nueva



# Crear nuevos strings a partir de los existentes

---

Fundamentos de Programación

```
nueva_frase += letra
```

- La concatenación da como resultado una variable nueva.
- De tal manera que, `nueva_frase` apunta al Nuevo string creado despues de la concatenación



# Funciones para strings en Python

---

Fundamentos de Programación

- `s.lower()`, `s.upper()`
  - Retorna el texto almacenado en `s` en minúsculas o mayúsculas respectivamente

```
name = "Espol @ Ecuador"  
length = len(name)  
nameUp = name.upper()  
print nameUp, "tiene", length, "caracteres"
```

**Salida:**

ESPOL @ ECUADOR tiene 15 caracteres

# Funciones para strings en Python

---

Fundamentos de Programación

- `s.strip()`
  - Retorna el texto almacenado en `s` con los espacios/fin de línea removidos al inicio y al final
- `s.isalpha()/s.isdigit()/s.isspace()`
  - Verifica si todos los caracteres de la variable `s` pertenecen a una clase determinada ( character, dígitos, puntuación)
- `s.startswith('other'), s.endswith('other')`
  - Verifica si el string comienza o termina con la cadena dada
- `s.find('other')`
  - Encuentra el índice de la primera ocurrencia del string "other". En caso de no ser encontrado, se retorna -1

# Funciones para strings en Python

---

Fundamentos de Programación

- `s.replace('old', 'new')`
  - Retorna un texto en el que todas las ocurrencias de 'old' han sido reemplazadas por 'new'
- `s.split('delim')`
  - retorna una lista de subcadenas separadas por el delimitador dado. Por ejemplo:
  - `'aaa,bbb,ccc'.split(',') -> ['aaa', 'bbb', 'ccc']`.
  - Si se usa sin argumento el delimitador son los "whitespace characters" ( enter, espacio en blanco )
- `s.join(list)`
  - opuesto a `split()`, une los elementos de una lista usando el delimitador dado.
  - `'---'.join(['aaa', 'bbb', 'ccc']) -> aaa---bbb---ccc`

# Caracteres de control

Fundamentos de Programación

- `\\` - Backslash
- `\'` - Single quote
- `\"` - Double quote
- `\a` - Bell
- `\b` - Backspace
- `\r` - Carriage return
- `\xhh` - Hex digits value *hh*
- `\0` - Null (binary zero bytes)

```
>>> print 'a\0m\0c' # 5 characters!
```

```
a m c
```

```
>>> print '\001 \002 \x03 \x04 \x05 \x06'
☺ ☹ ♥ ♦ ✚ ⬆
>>> print '\001\0\002\0\x03\0\x04\0\x05\0\x06'
☺ ☹ ♥ ♦ ✚ ⬆
```