

L#9. Strings

Ago 2017



https://docs.python.org/2.5/lib/typesseq.html

Sequences for this course



```
import numpy as np
# Strings:
S='LEONARDO DA VINCI'
print(S)
# Lists:
L=[3,4.5,7+3j,'Leonardo',True,[1,2,3]]; LL=[1,2,3,4,5]
print(L);
# Tuples—same as Lists but inmutable
T=(3,4.5,7+3j,(1,2,3),'Leonardo',True,[1,2,3])
print(T)
# Arrays
                                   LEONARDO DA VINCI
                                   [3, 4.5, (7+3j), 'Leonardo', True, [1, 2, 3]]
A=np.array([1,2,3,4,5])
                                   (3, 4.5, (7+3j), (1, 2, 3), 'Leonardo', True, [1, 2, 3])
print(A)
```

Strings



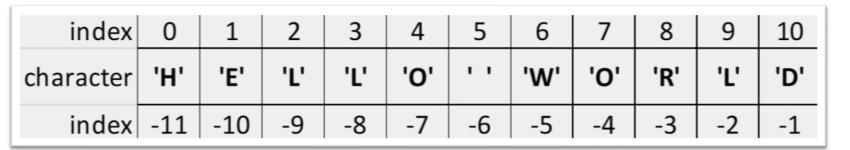
String literals are written in single or double quotes, e.g.: 'Hello World', "Hello World". The string objects are immutable (see optional slides at the end).

```
s='HELLO WORLD' # s=input("Enter Greeting"), user: HELLO WORLD
print(type(s)) # str
print(len(s)) # 11, len is short for length (longitud de la cadena de caracteres)
print(s) # HELLO WORLD
```

print(s[0]) # H
print(s[6]) # W
print(s[-1]) # D

index	0	1	2	3	4	5	6	7	8	9	10
character	'Η'	'E'	'L'	'L'	'0'	1 1	'W'	'0'	'R'	'L'	'D'
index	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Slicing





- Extract part of a string using the syntax [start:stop:step] which target characters from index **start** until index **stop** (exclusive) in increments of **steps**. The character at index **stop** is not included.
- Omitted index: if we omit **start**, it defaults to (first) index (i.e., zero). If we omit **stop**, it defaults to (last) index in the sequence. If we omit **step**, it defaults to (+1).

s="HELLO WORLD"

(1) print(s[0:5:1]) HELLO

(2) print(s[0:5]) HELLO # step defaults to 1, indexes: 0,1,2,3,4

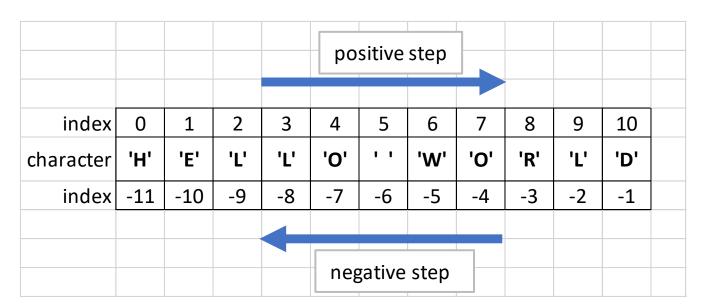
(3) print(s[::2]) HLOWRD # start=0, stop=11, step=2

(4) print(s[6:]) WORLD # start=6, stop=11, step=1









- One last word:
 - If step is positive, you go this direction: •
 - If step is negative, you go this direction:

```
s="HELLO WORLD"
```

print(s[0:5:1]) # HELLO

While slicing, we want to reverse the order as: OLLEH. Three solutions:

print(s[4:-12:-1]; print(s[-7:-12:-1], print(s[4::-1]) # OLLEH





Slicing

index	0	1	2	3	4	5	6	7	8	9	10
character	Ħ,	Ę'	ŗ.	ŗ.	,	-	'W'	,	'R'	'L'	'D'
index	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1



A bit more slicing:

s="HELLO WORLD"

print(s[:5:1]) HELLO

print(s[:5]) HELLO

print(s[:]) HELLO WORLD # which is the same as print(s)





Strings: Basic Operations



output

code [File: string03.py]

```
# String literals can span multiple lines: triple quotes
                                                           Cuando hay una tormenta
print(""" Cuando hay una tormenta
                                                           los pajaritos se esconde,
los pajaritos se esconde,
                                                           pero las aguilas vuelan mas alto.
pero las aguilas vuelan mas alto. """)
                                                           # Mahatma Gandhi
# A plus (+) concatenates strings
print("Sa" + "cu" + "dio")
                                                           Sacudio
# Iterate
for x in "Hola Mundo": print(x, end=")
                                                           Hola Mundo
# Multiply
print('HOLA'*5)
                                                            HOLAHOLAHOLAHOLA
# Replace a substring in a string with something else
s="cocodrile"; s2=s.replace("drile","nut")
print(s2)
                                                           coconut
```

Strings: Basic Operations



Operation	Result	Notes		
x in "Mexico"	True	a member		
x not in "San Juan"	True	not a member		
min("abcdefgh")	а	smallest item of "abcdefgh"		
max("abcdefgh")	h	largest item of "abcdefgh"		
print("PONCE"[0])	Р	first element		
print("MEXICO"[-1])	0	last element		
print("PONCE"[0:3])	PON	slice of first 3 elemnts		
s='MEXICO' print(s.replace('X',")) print(s)	MEICO MEXICO	removes X		



Strings objects are inmutable



File: strings02.py.

a = "co" # **a** now points to "**co**"

b = a # b points to the same "co" that a points to

a = a + a # a points to the new string "coco", but b still points to the old "co"

print(a) coco

print(b) co

The string objects themselves are immutable.

The variable, a, which points to the string, is mutable.

c= a + b + "drilo"

print(c) cococodrilo

c[0]="C" Error, bla, bla, bla





