

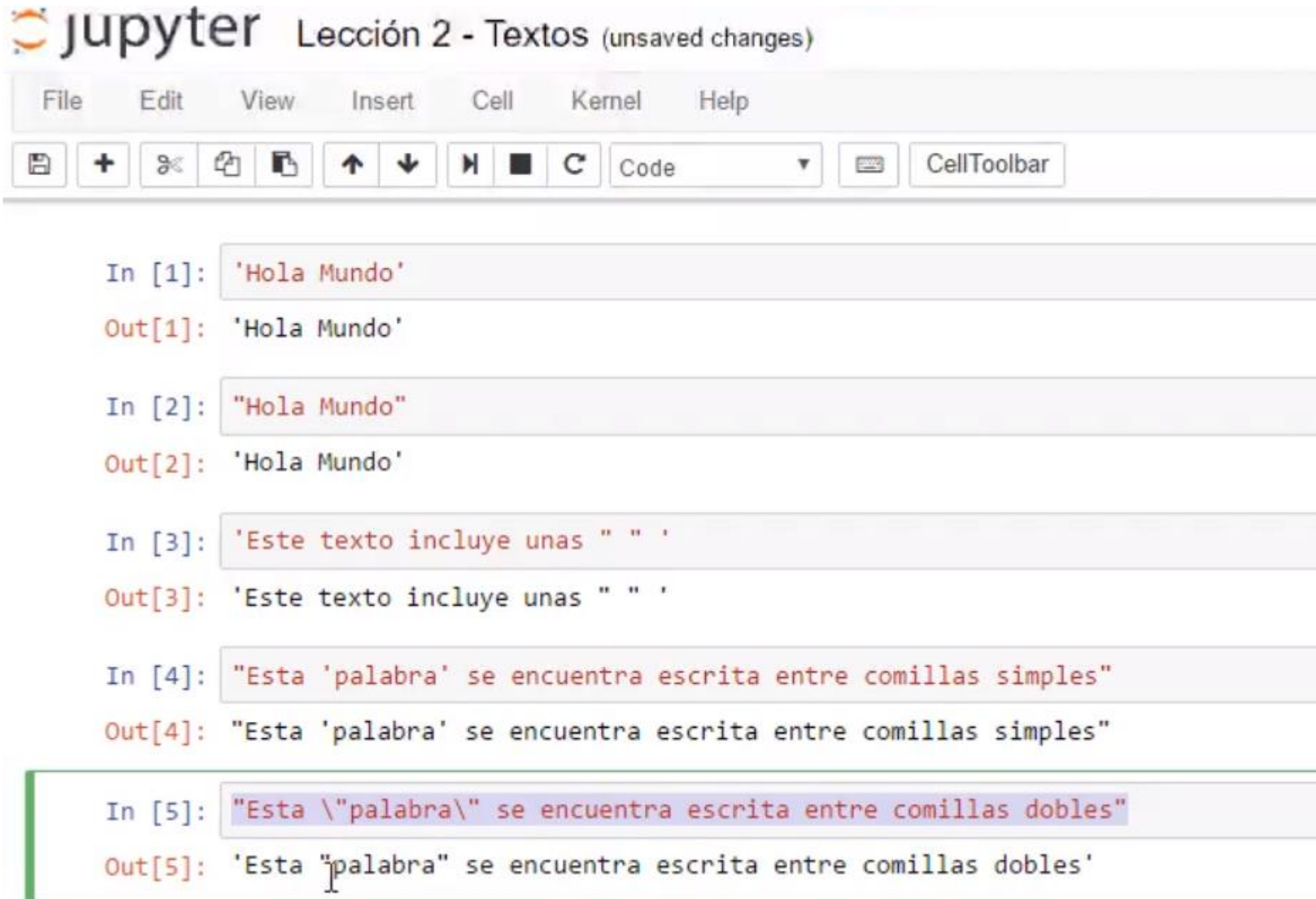
Text: Cadenes

1- Tractament de cadenes

- 1.1. Variables de text
- 1.2. Operacions amb cadenes
- 1.3. Índex o posició d'una cadena
 - Pràctiques P01, P02, P03
- 1.4. Mètodes o funcions de les cadenes
- 1.5. Algunes funcions de cadenes
 - Pràctiques P04, P05, P06



1. Tractament de cadenes



The image shows a Jupyter Notebook interface with the title "Lección 2 - Textos (unsaved changes)". The interface includes a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", and "Help". Below the menu bar is a toolbar with icons for saving, adding cells, undo, redo, and other functions. The notebook contains five input-output pairs demonstrating string handling in Python.

```
In [1]: 'Hola Mundo'
Out[1]: 'Hola Mundo'

In [2]: "Hola Mundo"
Out[2]: 'Hola Mundo'

In [3]: 'Este texto incluye unas " " '
Out[3]: 'Este texto incluye unas " " '

In [4]: "Esta 'palabra' se encuentra escrita entre comillas simples"
Out[4]: "Esta 'palabra' se encuentra escrita entre comillas simples"

In [5]: "Esta \"palabra\" se encuentra escrita entre comillas dobles"
Out[5]: 'Esta "palabra" se encuentra escrita entre comillas dobles'
```

jupyter Lección 2 - Textos (unsaved changes)

File Edit View Insert Cell Kernel Help

          Code   CellToolbar

```
In [6]: 'Esta \'palabra\' se encuentra escrita entre comillas dobles'
```

```
Out[6]: "Esta 'palabra' se encuentra escrita entre comillas dobles"
```

```
In [7]: "Una cadena"
        'otra cadena'
        'otra cadena más'
```

```
Out[7]: 'otra cadena más'
```

```
In [8]: print("Una cadena")
        print('otra cadena')
        print('otra cadena más')
```

```
Una cadena
otra cadena
otra cadena más
```

```
In [9]: print("Un texto\tuna tabulación")
```

```
Un texto      una tabulación
```

File Edit View Insert Cell Kernel Help

Save Add Split Cell Move Up Down Run Stop Restart Code CellToolbar

```
In [10]: print("Un texto\nuna nueva línea")
```

```
Un texto
una nueva línea
```

```
In [11]: print("C:\nombre\directorío")
```

```
C:
ombre\directorío
```

```
In [12]: print(r"C:\nombre\directorío")
```

```
C:\nombre\directorío
```

Per evitar que els caràcters especials, es transformin en comandes, hem d'indicar que una cadena es crua (raw)

```
In [13]: print("""Una línea
otra línea
otra línea\tuna tabulación""")
```

```
Una línea
otra línea
otra línea      una tabulación
```

```
print("Hola", "Adios")
```

Hola Adios

Cuando se trata de dos cadenas seguidas, se puede no escribir comas entre ellas, pero las cadenas se escribirán seguidas, sin espacio en blanco entre ellas:

```
print("Hola" "Adios")
```

HolaAdios

Al final de cada `print()`, Python añade automáticamente un salto de línea:

```
print("Hola")  
print("Adios")
```

Hola
Adios

Para generar una línea en blanco, se puede escribir una orden `print()` sin argumentos.

```
print("Hola")  
print()  
print("Adios")
```

Hola

Adios

Si no se quiere que Python añada un salto de línea al final de un `print()`, se debe añadir al final el argumento `end=""`:

```
print("Hola", end="")  
print("Adios")
```

HolaAdios

1.1. Variables de text

```
In [15]: c = "Esto es una cadena\ncon dos líneas"
```

```
In [16]: c
```

```
Out[16]: 'Esto es una cadena\ncon dos líneas'
```

```
In [17]: print(c)
```

```
Esto es una cadena
con dos líneas
```

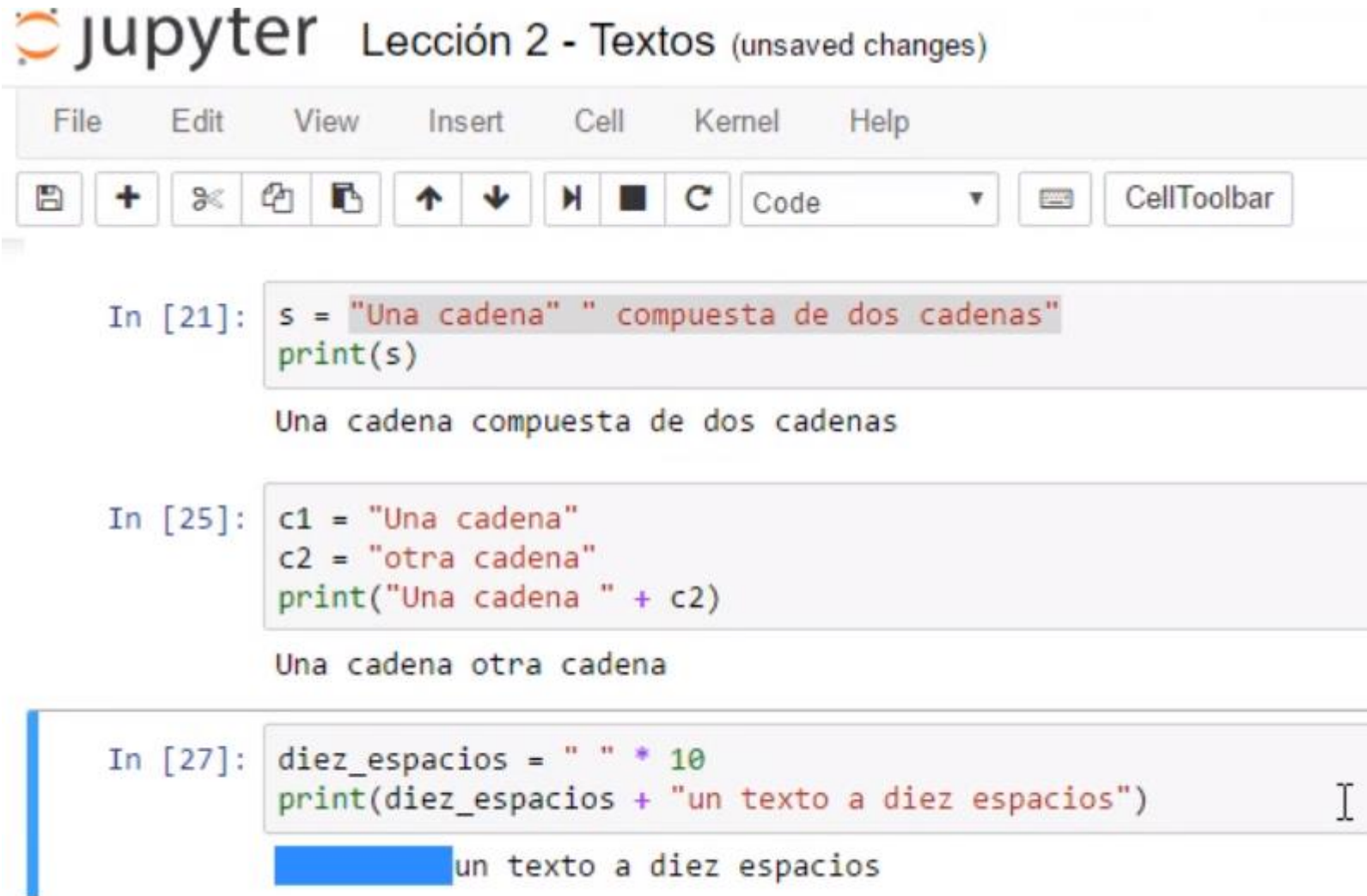
```
In [18]: c + c
```

```
Out[18]: 'Esto es una cadena\ncon dos líneasEsto es una cadena\ncon dos líneas'
```

```
In [19]: print(c+c)
```

```
Esto es una cadena
con dos líneasEsto es una cadena
con dos líneas
```


1.2. Operacions amb cadenes



The image shows a Jupyter Notebook interface with the title "Lección 2 - Textos (unsaved changes)". The interface includes a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", and "Help". Below the menu is a toolbar with icons for saving, adding, deleting, copying, pasting, undo, redo, and a dropdown menu currently set to "Code". A "CellToolbar" button is also visible.

The notebook contains three code cells:

```
In [21]: s = "Una cadena " "compuesta de dos cadenas"
print(s)

Una cadena compuesta de dos cadenas
```

```
In [25]: c1 = "Una cadena"
c2 = "otra cadena"
print("Una cadena " + c2)

Una cadena otra cadena
```

```
In [27]: diez_espacios = " " * 10
print(diez_espacios + "un texto a diez espacios")

un texto a diez espacios
```

1.3. Índex o posició d'una cadena

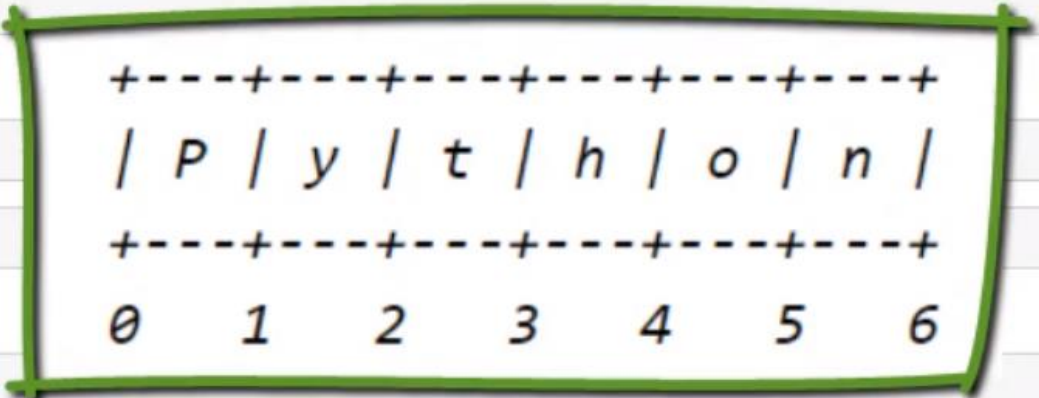
```
In [28]: palabra = "Python"
```

```
In [29]: palabra[0] # caràcter en la posició 0
```

```
Out[29]: 'p'
```

```
In [30]: palabra[3]
```

```
Out[30]: 'h'
```



	+	-	-	+	-	-	+	-	-	+	-	-	+	-	-	+
		P		y		t		h		o		n				
	+	-	-	+	-	-	+	-	-	+	-	-	+	-	-	+
	0		1		2		3		4		5		6			


```
In [47]: palabra[99]
```

```
-----  
IndexError                                Traceback (most recent call last)  
<ipython-input-47-b31ddef6ab27> in <module>()  
----> 1 palabra[99]  
  
IndexError: string index out of range
```

```
In [48]: palabra[:99]
```

```
Out[48]: 'Python'
```

```
+---+---+---+---+---+  
| P | y | t | h | o | n |
```

```
+---+---+---+---+---+
```

```
0    1    2    3    4    5
```

```
-6   -5   -4   -3   -2   -1
```

```
In [50]: palabra[99:]
```

```
Out[50]: ''
```

```
In [51]: palabra[0] = "N"
```

```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-51-c87a9e773639> in <module>()  
----> 1 palabra[0] = "N"  
  
TypeError: 'str' object does not support item assignment
```

```
In [52]: palabra = "N" + palabra[1:]  
palabra
```

```
Out[52]: 'Nython'
```

```
In [31]: palabra[-0]
```

```
Out[31]: 'p'
```

```
In [32]: palabra[-1]
```

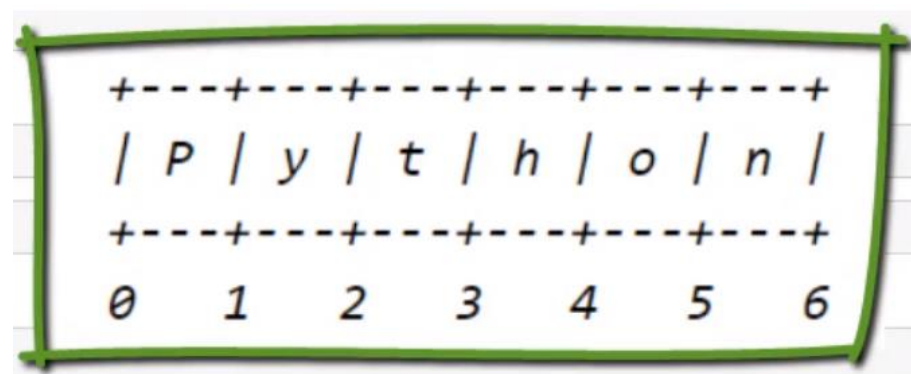
```
Out[32]: 'n'
```

```
In [33]: palabra[-2]
```

```
Out[33]: 'o'
```

```
In [35]: palabra[-6]
```

```
Out[35]: 'p'
```



+---+	+---+	+---+	+---+	+---+	+---+	+---+
/ P /	y /	t /	h /	o /	n /	
+---+	+---+	+---+	+---+	+---+	+---+	+---+
0	1	2	3	4	5	6

```
In [37]: palabra = "Python"
```

```
In [38]: palabra[0:2]
```

```
Out[38]: 'Py'
```

```
In [42]: palabra[2:]
```

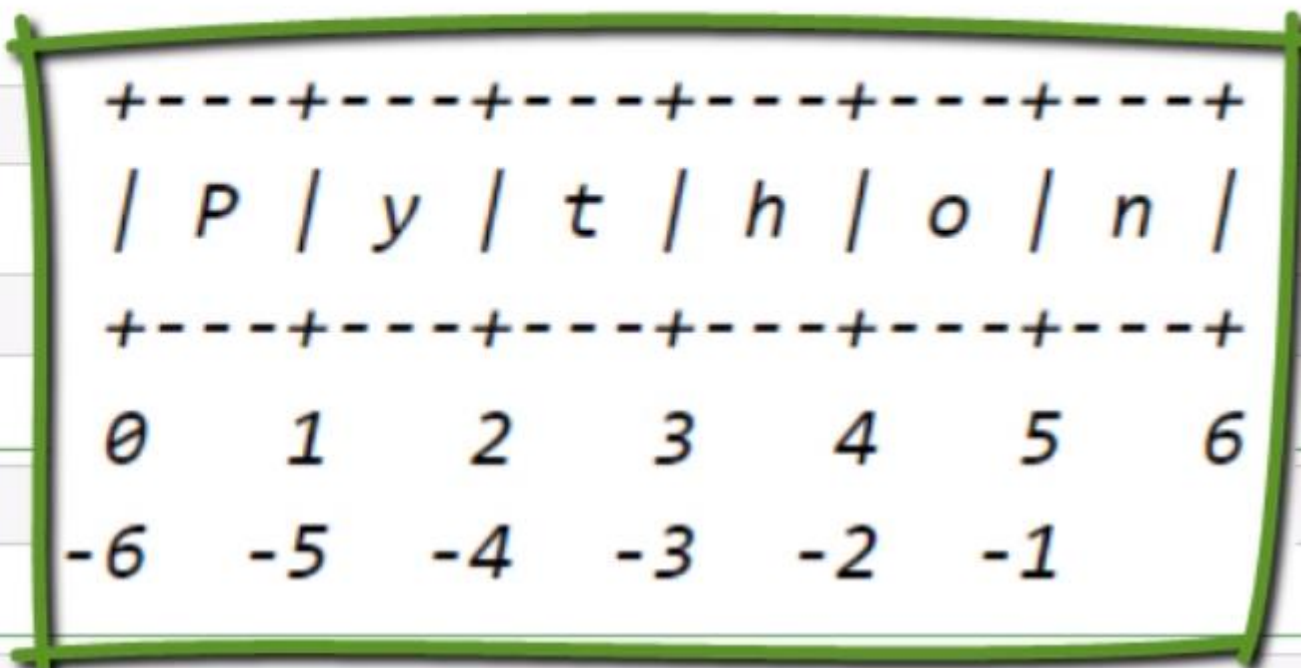
```
Out[42]: 'thon'
```

```
In [43]: palabra[:2]
```

```
Out[43]: 'Py'
```

```
In [44]: palabra[:]
```

```
Out[44]: 'Python'
```



P01. Pràctica de cadenes

Tenim aquestes cadenes : `str1="son"` `str2="ma"` `str3="ca"`
`str4="rro"`, forma les següents paraules, operant
aquestes cadenes i sense afegir cap lletra més :

1. maca
2. carro
3. macarro

4. roca
5. caso
6. moco

P02. Més Slicing : Tenint la variable `text = '<CADENA>'` calcula

```
text[1:4]  
text[:5]  
text[-4:]  
text[2]  
text[:]  
text[::-1]
```

P03. Avançat : Utilitza el slicing amb aquesta/es cadena/es per generar una llista vertical de noms femenins

L'advocació de Maria de la Mercè està lligada a la tasca carismàtica de l'ordre religiosa mercedària, nascuda a Barcelona amb sant Pere Nolasc, el seu fundador, mercader original, naturalitzat ciutadà barceloní. Va saber descobrir els dolors dels cristians captius en poder de musulmans, i es va obstinar, amb un grup de companys, primer amb el patrimoni personal, i després recollint almoines, en exercir el ministeri de la caritat eximia: ajudar, visitar i redimir captius. Quan faltaven diners per comprar la seva llibertat, rescatant de les masmorres africanes, s'obligaven tots a quedar en ostatges, esperant que arribessin els seus companys amb els diners previst. Si això no succeïa en el moment oportú, estaven disposats a lliurar la seva pròpia vida.

Pista1 : Copia el text al Notepad++ per saber les posicions de les paraules.

Pista2 : Crea dins del teu programa una variable de text, imagina que es diu c0.

Agafa la primera línia del text i la còpies com a valor de c0.

c0 = "L'advocació de Maria de la Mercè està lligada a la tasca carismàtica de l'ordre religiosa "

Ara ja pots extreure de c0, les posicions que t'interessi imprimir.

Solucions possibles

```
text = "<CADENA>"
```

```
#P02: calcula text[1:4], text[:5], text[-4:], text[2], text[:],  
#          text[::-1]
```

```
print(text[1:4])  
print(text[:5])  
print(text[-4:])  
print(text[2])  
print(text[:])  
print(text[::-1])
```

```
CAD  
<CADE  
E  
A  
<CADENA>  
>ANEDAC<
```

#P03 – Extreu del text una llista de noms femenins

```
c0 = "L'advocació de Maria de la Mercè està lligada a la tasca carismàtica de l'ordre religiosa"  
c1 = "mercedària, nascuda a Barcelona amb sant Pere Nolasc, el seu fundador, mercader "  
c2 = "original, naturalitzat ciutadà barceloní. Va saber descobrir els dolors dels cristians "  
c3 = "captius en poder de musulmans, i es va obstinar, amb un grup de companys, primer "  
c4 = "amb el patrimoni personal, i després recollint almoines, en exercir el ministeri de la "  
c5 = "caritat eximia: ajudar, visitar i redimir captius. Quan faltaven diners per comprar la seva "  
c6 = "llibertat, rescatant de les masmorres africanes, s'obligaven tots a quedar en ostatges, "  
c7 = "esperant que arribessin els seus companys amb els diners previst. Si això no succeïa "  
c8 = "en el moment oportú, estaven disposats a lliurar la seva pròpia vida."
```

```
print (c0[15:20])
print (c0[27:32])
print (c0[42:45])
print (c1[28:31])
print (c2[65:71])
print (c2[77:83]+c2[84]+c2[83])
```

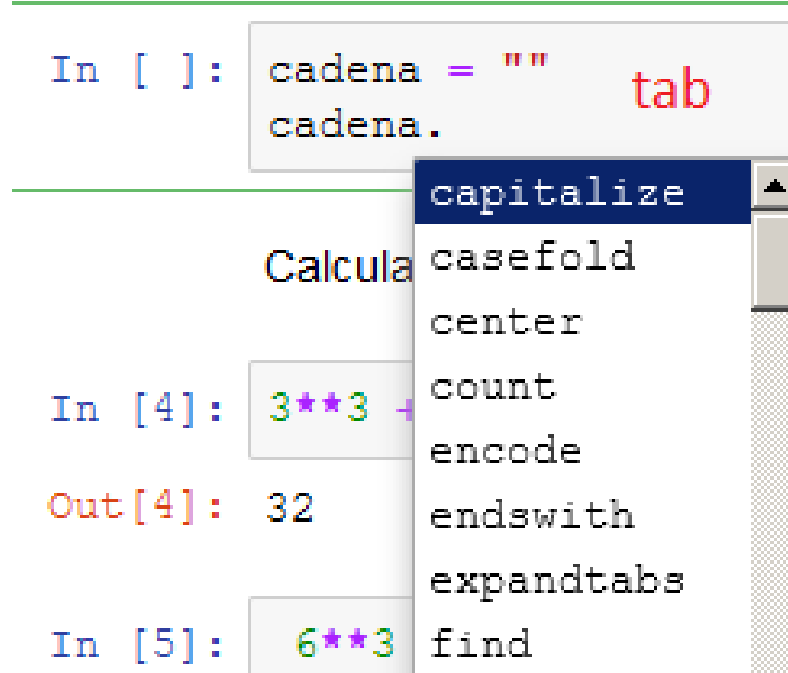
Maria
Mercè
ada
ona
dolors
cristina

caritat
visitacio
remei
llibertat
africa
pia

1.4. Mètodes o funcions de les cadenes

Quan tenim una variable de tipus **cadena**, tenim tot un seguit de funcions o mètodes que podem usar.

Els podem veure al Jupyter si creen una variable, i després posem el nom de la variable seguida d'un punt i prement la tecla **tabulador**.



```
In [ ]: cadena = ""
cadena.
```

Calcular

- capitalize
- casefold
- center
- count
- encode
- endswith
- expandtabs
- find

```
In [4]: 3**3
Out[4]: 32

In [5]: 6**3
```

1.5. Algunes funcions de cadenes

<code>upper()</code>	Passa una cadena a majúscules
<code>lower()</code>	Passa una cadena a minúscules
<code>capitalize()</code>	Passa la primera lletra del text a Majúscules.
<code>title()</code>	Posa la primera lletra de cada paraula en majúscules

`center ()`, `ljust ()` i `rjust ()` alineen una cadena al centre, a l'esquerra o la dreta respectivament

<code>isupper()</code>	Returns True if all characters in the string are upper case
<code>islower()</code>	“ “ “ all characters in the string are lower case
<code>istitle()</code>	“ “ if each word starts with uppercase

<code>isnumeric()</code>	Returns True if all characters in the string are numeric
<code>isdigit()</code>	Returns True if all characters in the string are digits

P04. Entra a la web de recursos Python, per comprendre la sintaxi de les funcions i practica-les al teu quadern Jupyter

<https://recursospython.com/guias-y-manuales/30-metodos-de-las-cadenas/>

P05. Entra a la web de w3schools I prova 5 operacions de cadena en el teu quadern.

https://www.w3schools.com/python/python_ref_string.asp

P06. Quin és el resultat d'aquestes operacions:

a) Calcula

```
txt = "Castaña, Asunción, María, José"  
x = txt.upper()  
print(x)
```

b) Calcula

```
txt = "CATALÁN DE CANCIÓN TULÚN!";  
mytable = txt.maketrans("ÁÉÍÓÚ", "AEIOU");  
print(txt.translate(mytable));
```

c) Estudia aquestes instruccions. Com es calcula una majúscula a partir d'una minúscula ?

```
print(ord("A"), ord("a"))  
print(ord("B"), ord("b"))  
print(ord("C"), ord("c"))  
print(ord("Ñ"), ord("ñ"))  
print(ord("Á"), ord("á"))  
print(ord("É"), ord("é"))
```

d) Calcula

```
txt = "excursiones, televisiones, macarrones"  
x = txt.replace("es", "s")  
print(x)
```


P07. Practica de demanar dades


Demana per separat les dades d'un client: `nom`, `primer_cognom`, `segon_cognom`.

Mostra el missatge : Benvingut `primer_cognom segon_cognom`, `nom`.

Algunes solucions

#P06

	Min	Maj
a. CASTAÑA, ASUNCIÓN, MARÍA, JOSÉ	65	97
b. CATALAN DE CANCION TULUN!	66	98
c. Sumant 32	67	99
d. excursions, televisions, macarrons	209	241
	193	225
	201	233



#P07.-----

#Exercici 1. Demana per separat les dades d'un client:

nom, primer_cognom, segon_cognom.

Mostra el missatge : Benvingut primer_cognom segon_cognom, nom.

```
nom = input("Nom: ")
cognom1 = input("Primer Cognom: ")
cognom2 = input("Segon Cognom: ")
print("Benvingut", cognom1, cognom2 + ", ", nom)
```