UTN AVELLANEDA
Tecnicatura Universitaria en Programación

Expresiones Regulares

Programación y Laboratorio I

Versión 22.1

UTN AVELLANEDA
Tecnicatura Universitaria en Programación

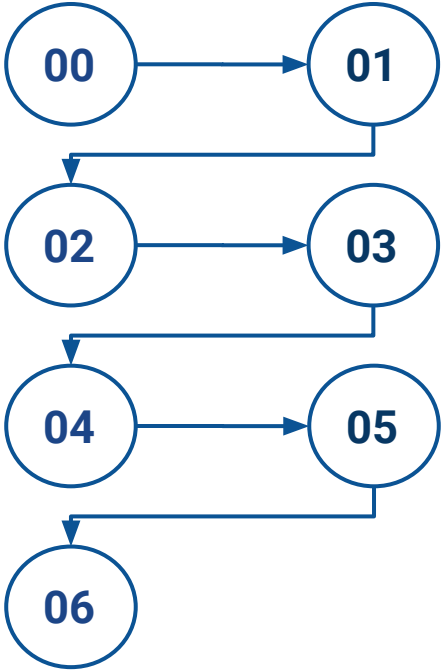
Expresiones regulares

¿Qué es una RegEx?

Expresiones

search()

sub()



Modulo re

split()

findall()

¿Qué es una RegEx?

El término de expresiones regulares nace a partir del término inglés regular expressions.

Las expresiones regulares **RegEx**, son una serie de símbolos que nos permitirán definir patrones de búsqueda en cadenas de texto.

Módulo re

Python tiene un paquete integrado llamado **re**, el cual se utiliza para trabajar con expresiones regulares.

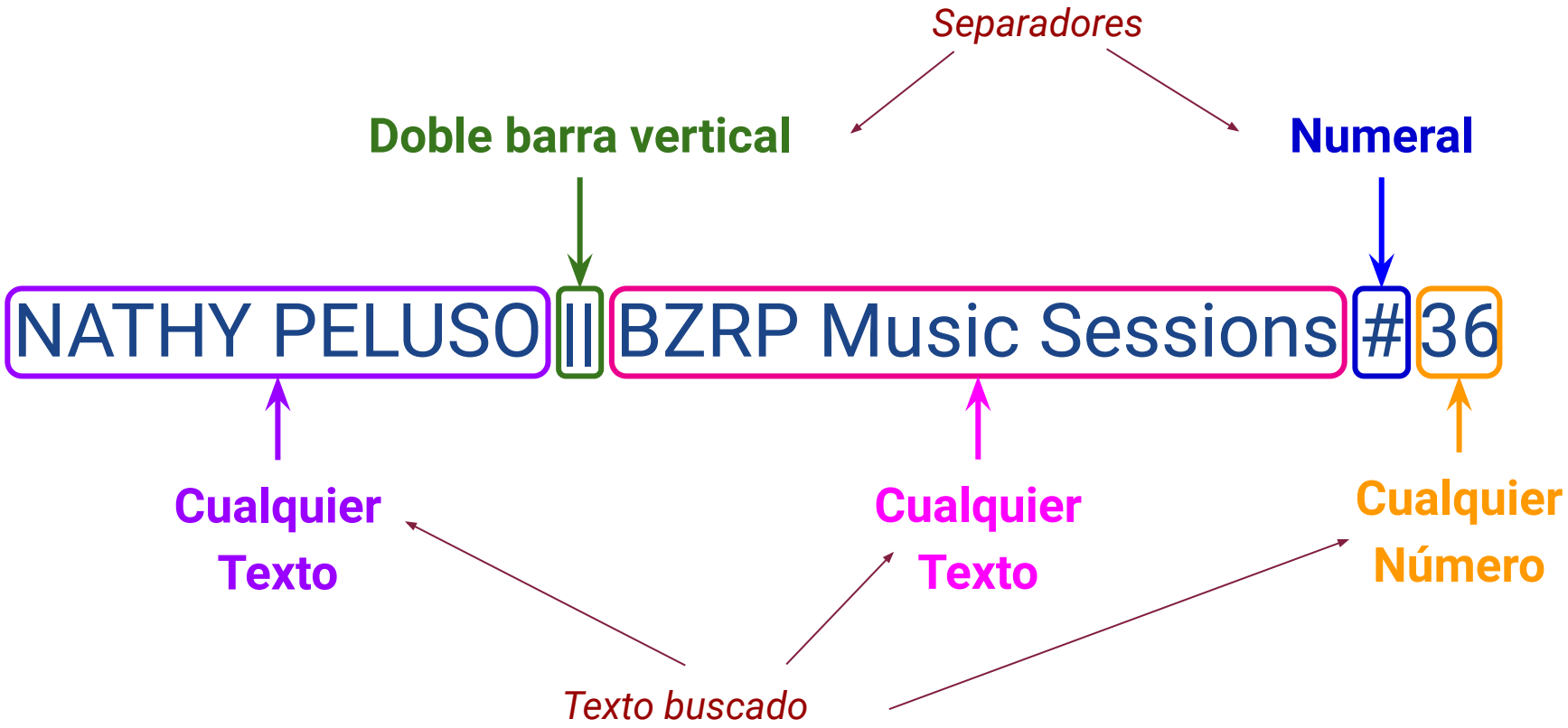
```
import re
```

Módulo re

El módulo **re** cuenta con un conjunto de métodos que permiten comprobar si una determinada cadena coincide con una expresión regular dada.

UTN AVELLANEDA
Tecnicatura Universitaria en Programación

Texto

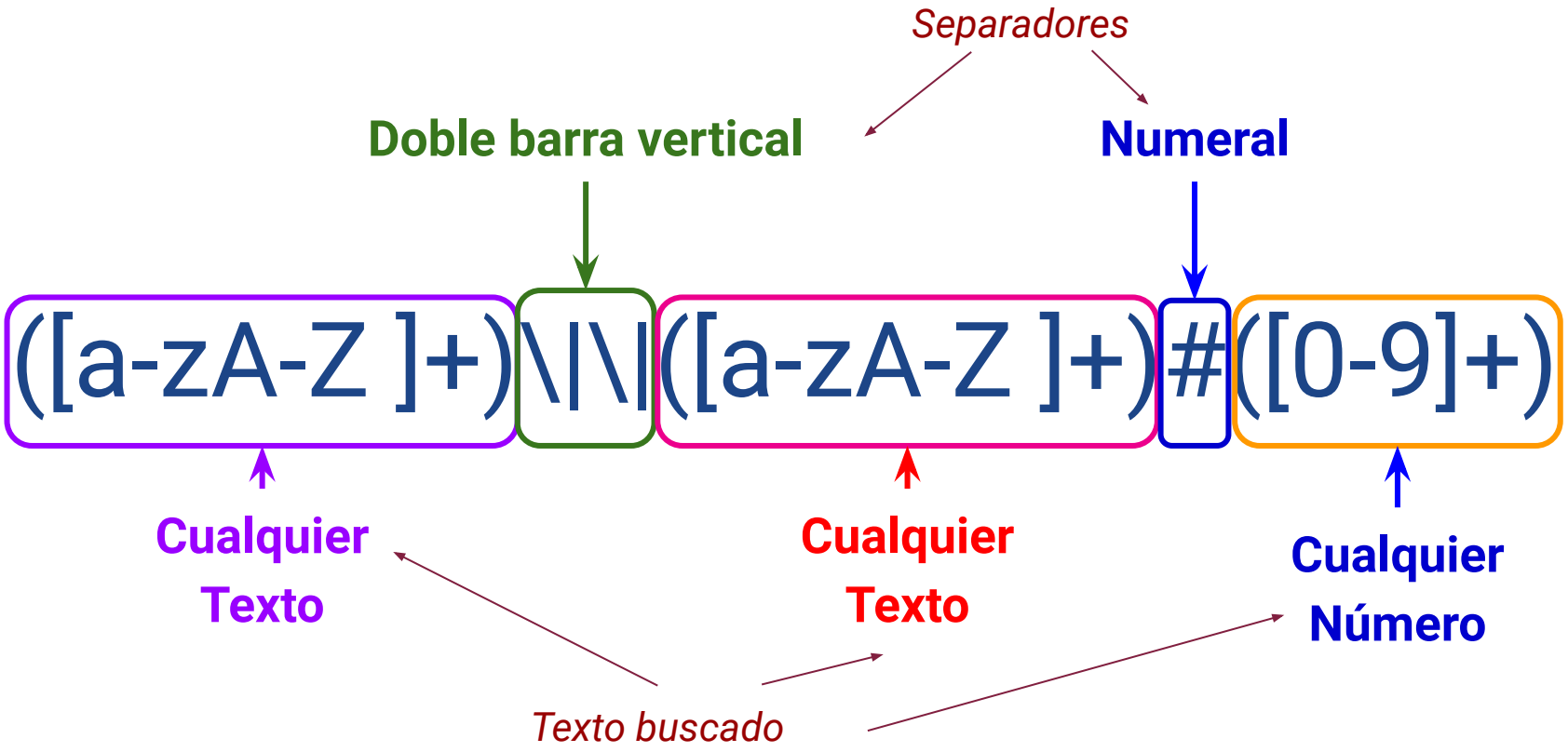


Expresion

```
([a-zA-Z ]+)\|\|([a-zA-Z ]+)#([0-9]+)
```

UTN AVELLANEDA
Tecnicatura Universitaria en Programación

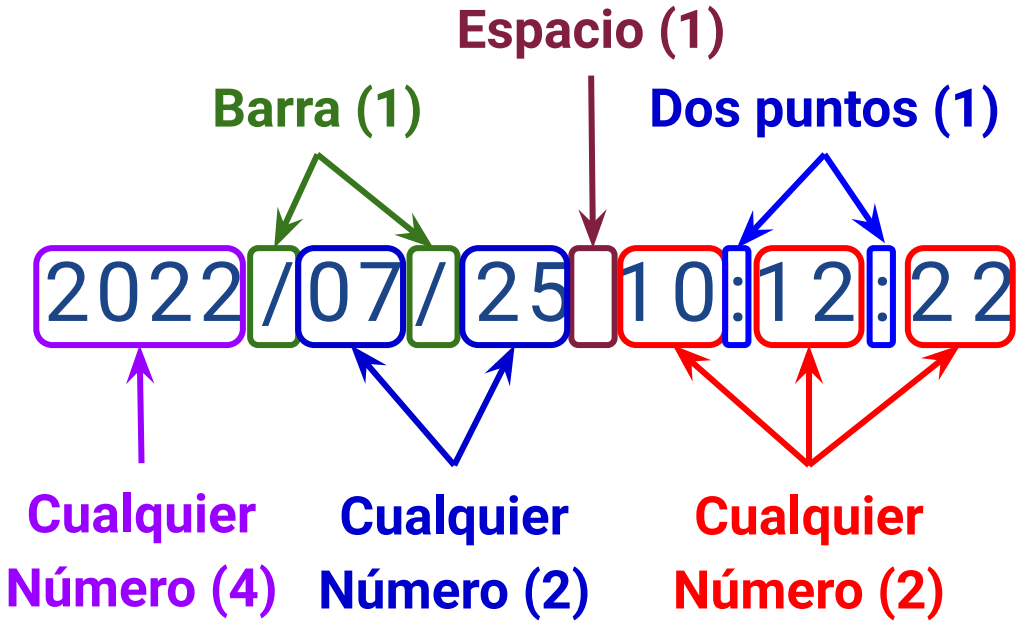
Expresion



Texto

["2023/11/99 19:22:00"] - ["2023/11/99","19","22","00"]

([0-9]{4}/[0-9]{2}/[0-9]{2}) ([0-9]{2}):([0-9]{2}):([0-9]{2})

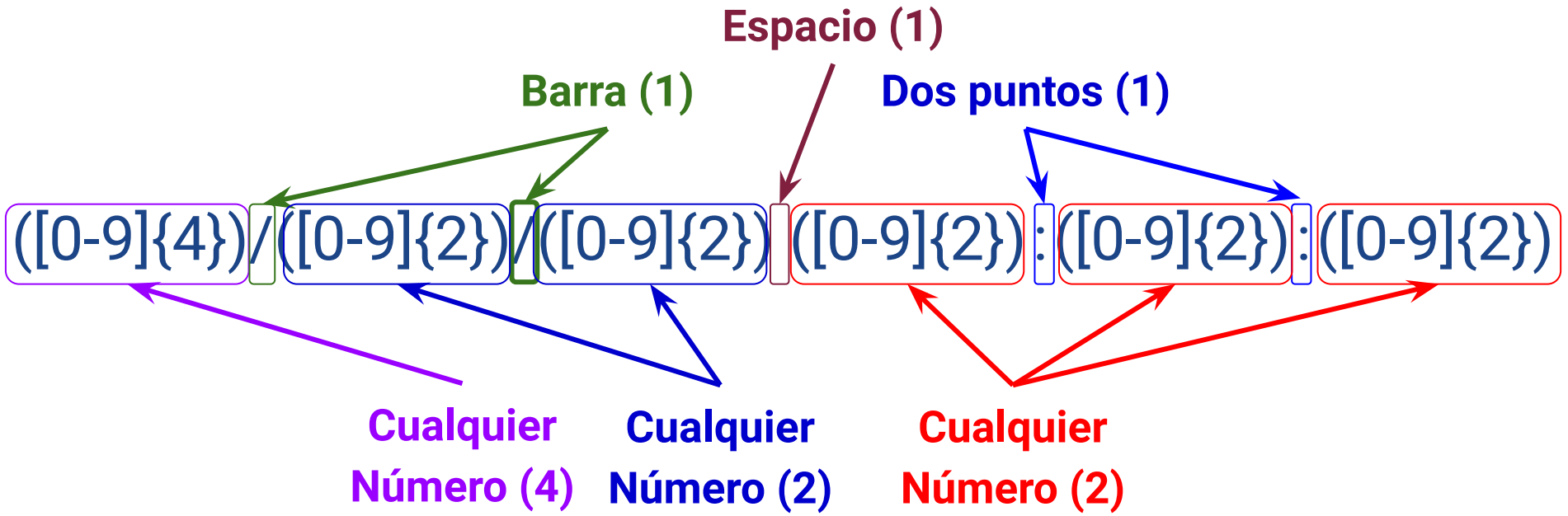


Expresion

([0-9]{4})/([0-9]{2})/([0-9]{2}) ([0-9]{2}):([0-9]{2}):([0-9]{2})

UTN AVELLANEDA
Tecnicatura Universitaria en Programación

Expresion



Sintaxis

[]	Conjunto de caracteres	"[a-z]"
\	Permite determinar secuencias especiales y escapar caracteres	"\d"
.	Hace referencia a cualquier caracter	"ho.a"
^	Empieza con	"^hola"
$	Termina con	"mundo$"

Sintaxis

*	Ninguna o más ocurrencias	"ho.*a"
+	Una o más ocurrencias	"ho.+a"
?	Cero o una ocurrencia	"ho.?a"
{}	Especifica el número de ocurrencias	"ho.{1}a"
|	Una o la otra	"hola|chau"
()	Permite seleccionar un grupo	

split()

Retorna una lista que contiene la cadena dividida por el número de ocurrencias del patrón

```
import re
texto = 'uno 1 dos 2 tres 3 cuatro'
print(re.split(' ', texto))
#['uno', '1', 'dos', '2', 'tres', '3', 'cuatro']
print(re.split('[0-9]+', texto))
#['uno ', ' dos ', ' tres ', ' cuatro']
print(re.split('[a-z ]+', texto))
#['', '1', '2', '3', '']
```

search()

Retorna **re.Match object** si contiene por lo menos una ocurrencia del patrón y **None** sino

```
import re
texto = ' uno 1 dos 2 tres 3 cuatro'
print(re.search(' ', texto))
#<re.Match object; span=(0, 1), match=' '>
print(re.search('[0-9]+', texto))
#<re.Match object; span=(5, 6), match='1'>
print(re.search('[a-z ]+', texto))
#<re.Match object; span=(0, 5), match=' uno '>
```

findall()

Retorna una lista que contiene todas las coincidencias del pattern («patrón»)

```
import re
texto = ' uno 1 dos 2 tres 3 cuatro'
print(re.findall(' ', texto))
#[' ', ' ', ' ', ' ', ' ', ' ', ' ']
print(re.findall('[0-9]+', texto))
#['1', '2', '3']
print(re.findall('[a-z ]+', texto))
#[' uno ', ' dos ', ' tres ', ' cuatro']
```

sub()

Reemplaza una o más coincidencias

```
# Borra abc
result = re.sub('abc',  '',    input)

# Repmpaza abc por xyz
result = re.sub('abc',  'xyz', input)

# Eliminalos espacios duplicados
result = re.sub(r'\s+', ' ',   input)
```