ntic master
revolucionamos la comunicación

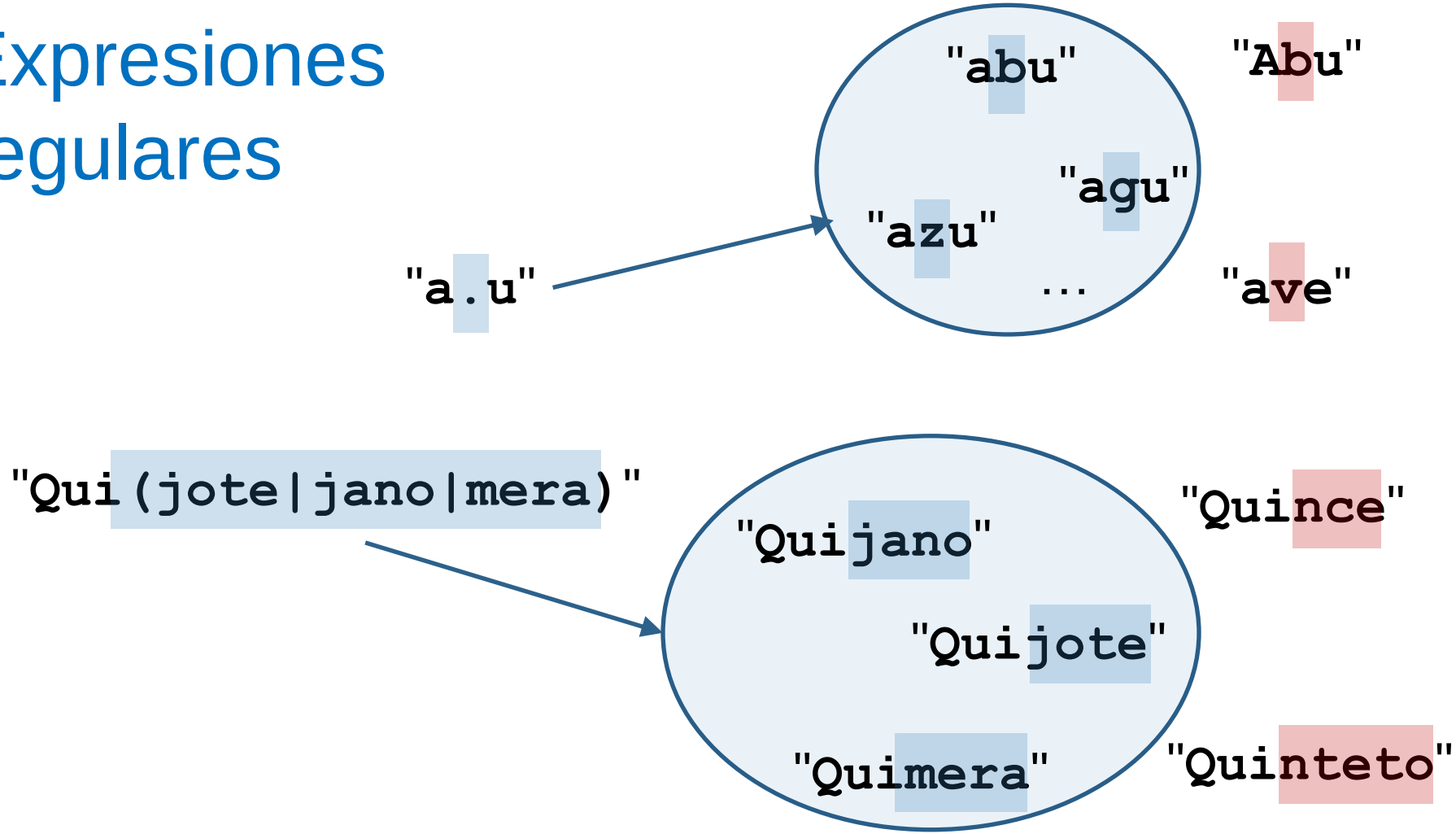
UNIVERSIDAD COMPLUTENSE DE MADRID

Programación. Python

Expresiones regulares

Cristóbal Pareja Flores

Expresiones regulares



Operación match()

```python
import re

def comprobar_encaje(patron_str, cadena):
    patron = re.compile(patron_str)
    if re.match(patron, cadena):
        print(f"El patrón '{patron_str}' SÍ encaja en la cadena '{cadena}'")
    else:
        print(f"El patrón '{patron_str}' NO encaja en la cadena '{cadena}'")

comprobar_encaje("a.u", "abuelo")
comprobar_encaje("a.u", "Abuelo")
```

```
El patrón 'a.u' SÍ encaja en la cadena 'abuelo'
El patrón 'a.u' NO encaja en la cadena 'Abuelo'
```

```python
for cad in ["abuelo", "Abuelo", "avuelo", "avutarda", "aguja", "aaaaa"]:
    comprobar_encaje("a.u", cad)
```

```
El patrón 'a.u' SÍ encaja en la cadena 'abuelo'
El patrón 'a.u' NO encaja en la cadena 'Abuelo'
El patrón 'a.u' SÍ encaja en la cadena 'avuelo'
El patrón 'a.u' SÍ encaja en la cadena 'avutarda'
El patrón 'a.u' SÍ encaja en la cadena 'aguja'
El patrón 'a.u' NO encaja en la cadena 'aaaaa'
```

Algunos patrones básicos

Patrón	Significado
"a*"	El carácter a, ninguna o más veces
"a+"	El carácter a, una o más veces
"[a-z]"	Una letra de la "a" a la "z"
"[1-9]+"	Un dígito entre uno y nueve, una o más veces
"mi(.l..)o"	Empieza por "mi", luego uno o dos caracteres y luego una "o": mito, mico, mirlo, miedo
"^The.*Spain$"	Empieza por "The" y termina con "Spain"

```
cadenas = ["abuelo", "Abuelo", "abono", "abbbbonar",
           "abuela", "abuelitos", "aaaaa", "ao"]

for cad in cadenas:
        comprobar_encaje("a.*b+", cad)

print(".....................................")

for cad in cadenas:
        comprobar_encaje("a(b|v|g).*", cad)

print(".....................................")

for cad in ["En un lugar de la Mancha", "En la Mancha..."]:
        comprobar_encaje("^En.*Mancha$", cad)
```

```
'a.*b+' === 'abuelo'
'a.*b+' =/= 'Abuelo'
'a.*b+' === 'abono'
'a.*b+' === 'abbbbonar'
'a.*b+' === 'abuela'
'a.*b+' === 'abuelitos'
'a.*b+' =/= 'aaaaa'
'a.*b+' =/= 'ao'
.....................................
'a(b|v|g).*' === 'abuelo'
'a(b|v|g).*' =/= 'Abuelo'
'a(b|v|g).*' === 'abono'
'a(b|v|g).*' === 'abbbbonar'
'a(b|v|g).*' === 'abuela'
'a(b|v|g).*' === 'abuelitos'
'a(b|v|g).*' =/= 'aaaaa'
'a(b|v|g).*' =/= 'ao'
.....................................
'^En.*Mancha$' === 'En un lugar de la Mancha'
'^En.*Mancha$' =/= 'En la Mancha...'
```

Método fullmatch()

```
for cad in ["En un lugar de la Mancha", "En la Mancha..."]:
        print(re.fullmatch("En.*Mancha", cad))
```

```
<re.Match object; span=(0, 24), match='En un lugar de la Mancha'>
None
```

Método search()

```
patron = re.compile("c...")

encaje = re.search(patron, "En un lugar...")
print(encaje)

encaje = re.search(patron, "... de la Mancha, de cuyo nombre no quiero acordarme...")
print(encaje)
print(encaje.start())
print(encaje.end())
```

```
None
<re.Match object; span=(13, 17), match='cha,'>
13
17
```

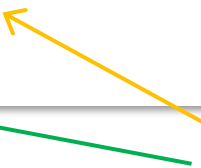
Método findall()

```
cadena = """En un lugar de la Mancha de cuyo nombre \
no quiero acordarme, había un hidalgo, de los de \
lanza en astillero, rocín flaco y galgo corredor..."""

print(re.findall("c...", cadena))

print(re.findall(". l..", cadena))
```

```
['cha ', 'cuyo', 'cord', 'cín ', 'co y', 'corr']
['n lug', 'e la ', 'e los', 'e lan']
```



Algunos patrones más

```
patrones = ['.ab*',       # un carácter, el carácter "a" seguido por cero o más caracteres "b".
            '.ab+',       # un carácter, el carácter "a" seguido por uno o más caracteres "b".
            '.ab?',       # un carácter, el carácter "a" seguido por cero o un carácter "b".
            '.ab{2}',     # un carácter, el carácter "a" seguido por dos caracteres "b".
            '.ab{2,4}',   # un carácter, el carácter "a" seguido por 2, 3 o 4 caracteres "b".
            '.[ab].',     # "[ab]" es el carácter "a" o el carácter "b".
            '.[ab]+',     # un carácter seguido de uno o más caracteres "a" o "b"
           ]

frase = "0a 1b 2ab 3aba 4abc 5aaaaaaa 6abbab 7abbbbb 6abababababab"

for p in patrones:
    print(f"Búsqueda con el patrón '{p}'")
    print(re.findall(p, frase))
```

```
Búsqueda con el patrón '.ab*'
['0a', '2ab', '3ab', '4ab', '5a', 'aa', 'aa', 'aa', '6abb', '7abbbbb', '6ab', 'bab', 'bab']
Búsqueda con el patrón '.ab+'
['2ab', '3ab', '4ab', '6abb', '7abbbbb', '6ab', 'bab', 'bab']
Búsqueda con el patrón '.ab?'
['0a', '2ab', '3ab', '4ab', '5a', 'aa', 'aa', 'aa', '6ab', 'bab', '7ab', '6ab', 'bab', 'bab']
Búsqueda con el patrón '.ab{2}'
['6abb', '7abb']
Búsqueda con el patrón '.ab{2,4}'
['6abb', '7abbbb']
Búsqueda con el patrón '.[ab].'
['0a ', '1b ', '2ab', '3ab', '4ab', '5aa', 'aaa', 'aa ', '6ab', 'bab', '7ab', 'bbb', '6ab', 'aba',
'bab', 'aba']
Búsqueda con el patrón '.[ab]+'
['0a', '1b', '2ab', '3aba', '4ab', '5aaaaaaa', '6abbab', '7abbbbb', '6abababababab']
```

Más patrones

```
patrones = ["[^!.? ]+",    # Un carácter que no es "!", ni ".", ni "?" ni " ".
            "[a-z]",       # rango de caracteres
            "[a-zA-Z]",    # una minúscula o una mayúscula
            "[A-Z][a-z]", # una minúscula seguida de una mayúscula
           ]

frase = "¡Qué lindos ojos tienes! ¿Puedes decirme tu nombre?"

for p in patrones:
    print(f"Búsqueda con el patrón '{p}':")
    print(re.findall(p,frase))
```

```
Búsqueda con el patrón '[^!.? ]+':
['¡Qué', 'lindos', 'ojos', 'tienes', '¿Puedes', 'decirme', 'tu', 'nombre']
Búsqueda con el patrón '[a-z]':
['u', 'l', 'i', 'n', 'd', 'o', 's', 'o', 'j', 'o', 's', 't', 'i', 'e', 'n', 'e', 's', 'u', 'e',
'd', 'e', 's', 'd', 'e', 'c', 'i', 'r', 'm', 'e', 't', 'u', 'n', 'o', 'm', 'b', 'r', 'e']
Búsqueda con el patrón '[a-zA-Z]':
['Q', 'u', 'l', 'i', 'n', 'd', 'o', 's', 'o', 'j', 'o', 's', 't', 'i', 'e', 'n', 'e', 's', 'P',
'u', 'e', 'd', 'e', 's', 'd', 'e', 'c', 'i', 'r', 'm', 'e', 't', 'u', 'n', 'o', 'm', 'b', 'r',
'e']
Búsqueda con el patrón '[A-Z][a-z]':
['Qu', 'Pu']
```

Un ejemplo

```
# Ejemplo: un número entero, con signo o sin él:

entero = re.compile(r'[\+\-]?[0-9]+')
cantidades = re.findall(entero, "Tengo 123 euros, en el banco, -150€ y gano +347€")

print(cantidades)
print(sum([int(c) for c in cantidades]))
```

```
['123', '-150', '+347']
320
```

El método split()

```
cadena = """En un lugar de la Mancha de cuyo nombre \
no quiero acordarme, había un hidalgo, de los de \
lanza en astillero, rocín flaco y galgo corredor..."""

separada = re.split("c...", cadena)
print(separada)

print("....................................")

separadores_de_frase = "[.;:] " # uno de esos caracteres + un espacio en blanco
frases = re.split(separadores_de_frase,
                  "Estoy de acuerdo. Pero no del todo; otro día lo discutimos: ahora no puedo.")
print(frases)
```

```
['En un lugar de la Man', 'de ', ' nombre no quiero a', 'arme, había un hidalgo, de los de lanza e
n astillero, ro', 'fla', ' galgo ', 'edor...']
....................................
['Estoy de acuerdo', 'Pero no del todo', 'otro día lo discutimos', 'ahora no puedo.']
```

Método sub()

sustituir

```
cadena = """En un lugar de la Mancha de cuyo nombre \
no quiero acordarme, había un hidalgo, de los de \
lanza en astillero, rocín flaco y galgo corredor..."""

separada = re.sub("c...", "----", cadena)
print(separada)
```

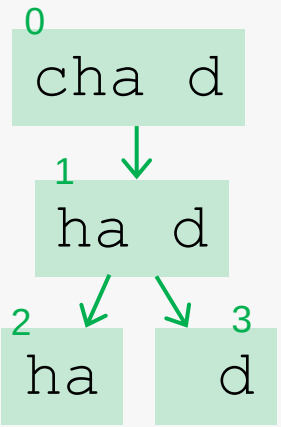
```
En un lugar de la Man----de ---- nombre no quiero a----arme, había un
hidalgo, de los de lanza en astillero, ro----fla---- galgo ----edor...
```

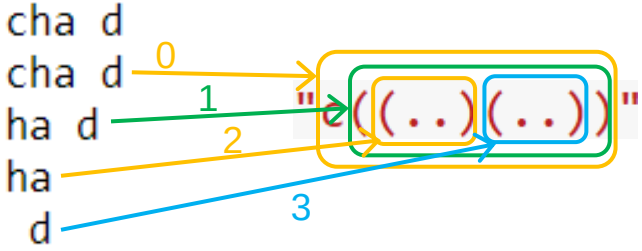
Método group()

```
cadena = """En un lugar de la Mancha de cuyo nombre\
no quiero acordarme, había un hidalgo, de los de\
lanza en astillero, rocín flaco y galgo corredor..."""

patron = re.compile("c....")
encaje = re.search(patron, cadena)
print(encaje.group())

patron = re.compile("c((..)(..))")
encaje = re.search(patron, cadena)
print(encaje.group(0))
print(encaje.group(1))
print(encaje.group(2))
print(encaje.group(3))
```





Dos ejemplos de group() y findall()

```
# Buscamos un número entero en una cadena:

patr_ent = re.compile("[^0-9]*([0-9]+)[^0-9]")
cadena = """[...] ejemplares únicos en la \
Hispanic Society (Fadrique de Basilea, Burgos, 1499 pero, \
en realidad, 1500-1502), la Sociedad Bodmeriana \
(Pedro Hagenbach, Toledo, 1500) y en la Biblioteca Nacional de Francia \
(Estanislao Polono, Sevilla, 1501) [...]"""

encaje = re.search(patr_ent, cadena)
print(encaje.group(0))
print(encaje.group(1))

# Buscamos ahora todos los enteros:

print(re.findall(patr_ent, cadena))
```

```
[...] ejemplares únicos en la Hispanic Society (Fadrique de Basilea, Burgos, 1499
1499
['1499', '1500', '1502', '1500', '1501']
```

Método finditer(), como group() iterando

```
cadena = """[...] ejemplares únicos en la \
Hispanic Society (Fadrique de Basilea, Burgos, 1499 pero, \
en realidad, 1500-1502), la Sociedad Bodmeriana \
(Pedro Hagenbach, Toledo, 1500) y en la Biblioteca Nacional de Francia \
(Estanislao Polono, Sevilla, 1501) [...]"""
```

```
# Buscamos un número entero y luego un nombre propio en una cadena:

patr_ent_np = re.compile("[^0-9]*([0-9]+)[^0-9][^A-Z]*([A-Z][a-z]*)")
encajes = re.finditer(patr_ent_np, cadena)
for enc in encajes:
    print("-> ", enc.group(0))
    print("-> ", enc.group(1))
    print("-> ", enc.group(2))
    print()
```

```
->  [...] ejemplares únicos en la Hispanic Society (Fadrique de Basile
a, Burgos, 1499 pero, en realidad, 1500-1502), la Sociedad
->  1499
->  Sociedad

->   Bodmeriana (Pedro Hagenbach, Toledo, 1500) y en la Biblioteca
->  1500
->  Biblioteca
```

Secuencias de escape

```
patrones = ["\d+", # Secuencia de uno o más dígitos
            "\D+", # Secuencia de uno o más no dígitos
            "\s+", # Secuencia de uno o más espacios en blanco
            "\S+", # Secuencia de uno o más no espacios en blanco
            "\w+", # Secuencia de uno o más caracteres alfanuméricos
            "\W+", # Secuencia de uno o más no caracteres alfanuméricos
           ]

frase = "¡El número del anticristo es 666, el número de la bestia!"

for p in patrones:
    print(f"'Búsqueda con el patrón '{p}'")
    print(re.findall(p, frase))
```

```
'Búsqueda con el patrón '\d+'
['666']
'Búsqueda con el patrón '\D+'
['¡El número del anticristo es ', ', el número de la bestia!']
'Búsqueda con el patrón '\s+'
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
'Búsqueda con el patrón '\S+'
['¡El', 'número', 'del', 'anticristo', 'es', '666,', 'el', 'número', 'd
e', 'la', 'bestia!']
'Búsqueda con el patrón '\w+'
['El', 'número', 'del', 'anticristo', 'es', '666', 'el', 'número', 'd
e', 'la', 'bestia']
'Búsqueda con el patrón '\W+'
['¡', ' ', ' ', ' ', ' ', ' ', ', ', ' ', ' ', ' ', ' ', '!']
```

Ejemplos adicionales

```
# Patrón para identificar una fecha:

fecha_re = re.compile('\d{2}/\d{2}/\d{4}')

linea = '[26/11/1962 00:01:35] <font color="#00ff00"¡Sorpresa!</font>>'
encaje = fecha_re.search(linea)
print(encaje)
print(encaje.group(0))

linea_sin_fecha = "En tiempos de Ahrun al Rashid..."
encaje = fecha_re.search(linea_sin_fecha)
print(encaje)
```

```
<re.Match object; span=(1, 11), match='26/11/1962'>
26/11/1962
None
```

```
# Patrón para identificar una dirección de email:

dir_email = r'[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)+'
dir_email = r'[\w_.+-]+@[\w_.+-]+(?:\.[\w_.+-]+)+'

trozo = "[\w_.+-]+"
dir_email = f'{trozo}@{trozo}(?:\.{trozo})+'

frase = "Mi email: cpareja@ucm.es. No c.par@sip.ucm.es ni c-pa+reja@SIP.ucm.es.es.es"
print(re.findall(patron_email, frase1 + frase2))
```

Agrupamiento sin captura

equivalentes

```
['cpareja@ucm.es', 'c.par@sip.ucm.es', 'c-pa+reja@SIP.ucm.es.es.es', 'c_pareja@ucm.es']
```

Ejercicio

```
# Patrón para identificar una fecha y una hora:

linea = '[26/11/1962 00:01:35] <font color="#00ff00"¡Sorpresa!</font>>'

# Ahora definimos dos grupos, con los paréntesis:
fecha_con_hora = re.compile('                                    ')

encaje = fecha_con_hora.search(linea)
print(encaje)
print(encaje.group(0))
print(encaje.group(1))
print(encaje.group(2))
```

Completa tú esto

```
<re.Match object; span=(1, 20), match='26/11/1962 00:01:35'>
26/11/1962 00:01:35
26/11/1962
00:01:35
```

Solución

```
# Patrón para identificar una fecha y una hora:

linea = '[26/11/1962 00:01:35] <font color="#00ff00"¡Sorpresa!</font>>'

# Ahora definimos dos grupos, con los paréntesis:
fecha_con_hora = re.compile('(\d{2}/\d{2}/\d{4}) (\d{2}:\d{2}:\d{2})')

encaje = fecha_con_hora.search(linea)
print(encaje)
print(encaje.group(0))
print(encaje.group(1))
print(encaje.group(2))
```

¡Bien hecho!

```
<re.Match object; span=(1, 20), match='26/11/1962 00:01:35'>
26/11/1962 00:01:35
26/11/1962
00:01:35
```

Referencias

- https://www.w3schools.com/python/python_regex.asp
- https://docs.python.org/3/howto/regex.html
- https://regex101.com/

ntic master
revolucionamos la comunicación

UNIVERSIDAD COMPLUTENSE DE MADRID

Programación. Python

Expresiones regulares

Cristóbal Pareja Flores