ntic master
revolucionamos la comunicación

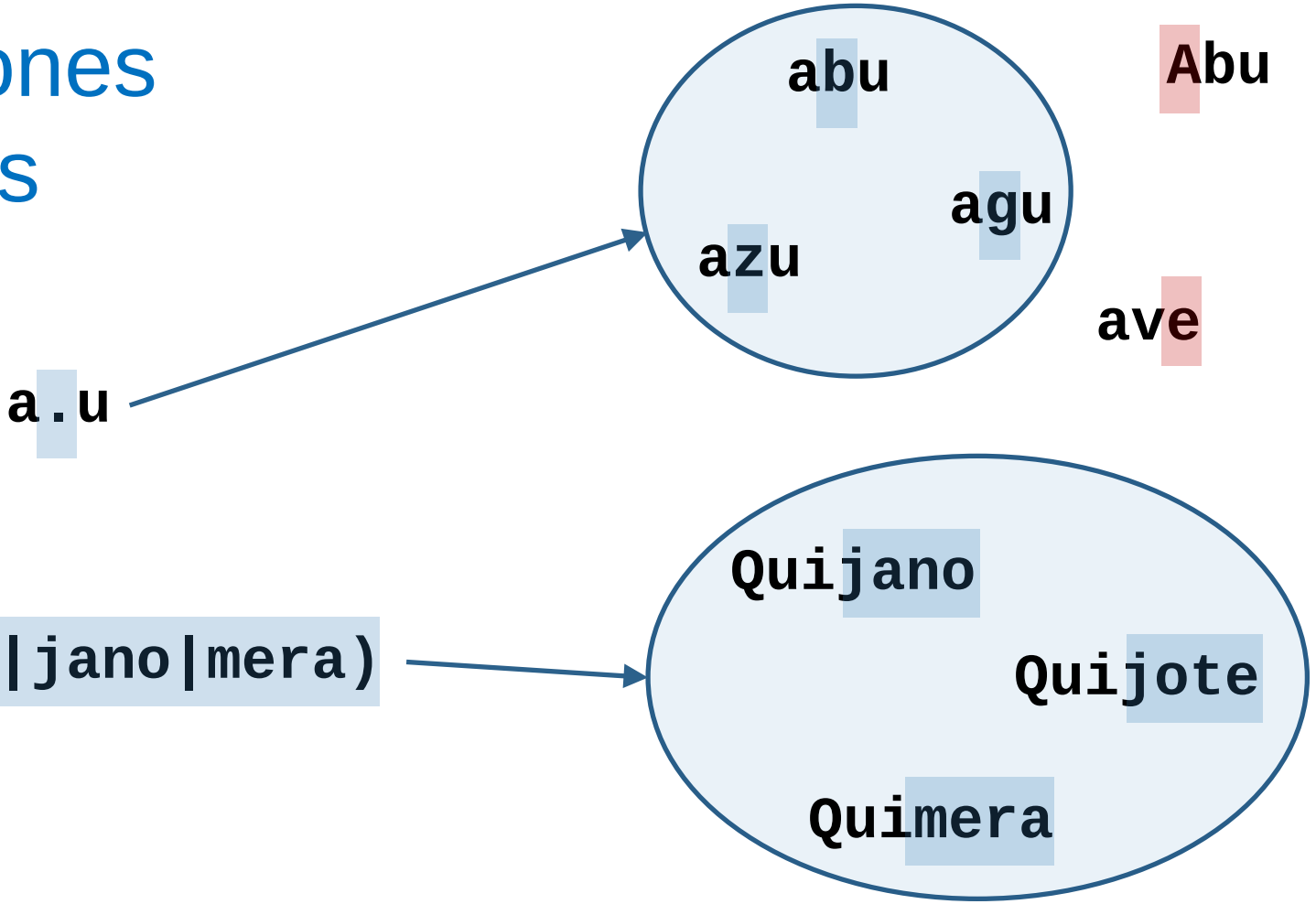
UNIVERSIDAD COMPLUTENSE DE MADRID

Programación. Python

Expresiones regulares

Cristóbal Pareja Flores

Expresiones regulares



Función match()

```python
import re

patron = re.compile("a.u")

cadena = "abuelo"
if re.match(patron, cadena):
    print("El patrón encaja en la cadena " + cadena)
else:
    print("El patrón NO encaja en la cadena " + cadena)

cadena = "Abuelo"
if re.match(patron, cadena):
    print("El patrón encaja en la cadena " + cadena)
else:
    print("El patrón NO encaja en la cadena " + cadena)

for cadena in ["abuelo", "Abuelo", "avuelo", "avutarda", "aguja", "aaaaa"]:
    if re.match(patron, cadena):
        print(cadena)
```

```
El patrón encaja en la cadena abuelo
El patrón NO encaja en la cadena Abuelo
abuelo
avuelo
avutarda
aguja
```

En forma de función

```
import re

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

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

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

```
def comprobar_encaje_varias_cadenas(patron_str, varias_cadenas):
    patron = re.compile(patron_str)
    for cadena in varias_cadenas:
        if re.match(patron, cadena):
            print("Ok: " + cadena)
        else:
            print("No: " + cadena)

cadenas = ["abuelo", "Abuelo", "avuelo", "avutarda", "aguja", "aaaaa"]
comprobar_encaje_varias_cadenas("a.u", cadenas)
```

```
Ok: abuelo
No: Abuelo
Ok: avuelo
Ok: avutarda
Ok: aguja
No: 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(.I..)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"]

comprobar_encaje_varias_cadenas("ab*o", cadenas)
```

```
No: abuelo
No: Abuelo
No: avuelo
No: avutarda
No: aguja
No: 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"

```
comprobar_encaje_varias_cadenas("a.*b+", cadenas)

comprobar_encaje_varias_cadenas("a(b|v|g).*", cadenas)

comprobar_encaje_varias_cadenas("^En.*Mancha$",
                                ["En un lugar de la Mancha", "En la Mancha..."])
```

```
No: abuelo
No: Abuelo
No: avuelo
No: avutarda
No: aguja
No: aaaaa
```

```
Ok: abuelo
No: Abuelo
Ok: avuelo
Ok: avutarda
Ok: aguja
No: aaaaa
```

```
Ok: En un lugar de la Mancha
No: En la Mancha...
```

Función 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 ac
print(encaje)
print(encaje.start())
print(encaje.end())
```

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

Función 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("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']
```

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("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! ¿Puedes decirme tu nombre?"

for p in patrones:
    print("Búsqueda con el patrón " + p)
    print(re.findall(p,frase))
```

```
Búsqueda con el patrón [^!.? ]+
['¡Qué', 'lindos', 'ojos', '¿Puedes', 'decirme', 'tu', 'nombre']
Búsqueda con el patrón [a-z]
['u', 'l', 'i', 'n', 'd', 'o', 's', 'o', 'j', 'o', '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', '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']
```

Función 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)
```

```
['En un lugar de la Man', 'de ', ' nombre no quiero a',
'arme, había un hidalgo, de los de lanza en astillero, r
o', 'fla', ' galgo ', 'edor...']
```

Función sub()

```
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----ar
me, había un hidalgo, de los de lanza en astillero, ro---
-fla---- galgo ----edor...
```

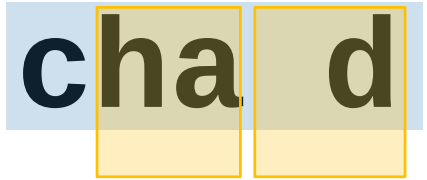
Función group()

```
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))
```

```
cha d
cha d
ha d
ha
 d
```



Función group()

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

patr_ent = re.compile("[^0-9]*([0-9]+)[^0-9].*")
cadena = "el número de orden es 17, el nombre es Calixto y su novia es Melibea."
encaje = re.search(patr_ent, cadena)
print(encaje.group(1))
```

```
17
```

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

patr_ent_np = re.compile("[^0-9]*([0-9]+)[^0-9][^A-Z]*([A-Z][a-z]*).*")
cadena = "el número de orden es 17, el nombre es Calixto y su novia es Melibea."
encaje = re.search(patr_ent_np, cadena)
print(encaje.group(1))
print(encaje.group(2))
```

```
17
Calixto
```

Secuencias de escape

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

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

for p in patrones:
    print("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',
'de', 'la', 'bestia!']
Búsqueda con el patrón \w+
['El', 'número', 'del', 'anticristo', 'es', '666', 'el', 'número',
'de', 'la', 'bestia']
Búsqueda con el patrón \W+
['¡', ' ', ' ', ' ', ' ', ' ', ', ', ' ', ' ', ' ', ' ', '!']
```

Ejemplos adicionales

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

patron_email = r"([a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+)"
frase = "Mi email es cpareja@ucm.es, no c.pareja@sip.ucm.es ni c_pareja@SIP.ucm.es.es.es"
print(re.findall(patron_email,frase))
```

```
['cpareja@ucm.es', 'c.pareja@sip.ucm.es', 'c_pareja@SIP.ucm.es.es.es']
```

```
# 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
```

Referencias

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