

CursoPython (/github/LlanosTobarra/CursoPython/tree/master)
/ Sesión 2 (/github/LlanosTobarra/CursoPython/tree/master/Sesión 2)

INTRODUCCIÓN A PYTHON. SESIÓN 2

Trabajo con módulos

Dentro de un módulo tendremos un conjunto de funciones y objetos que podremos invocar desde otro programa poniendo al principio:

```
import nombreModulo
```

El módulo tiene que estar en el mismo directorio que el programa o en el path del sistema

Para llamar a una función del módulo, ponemos primero el nombre del módulo seguido de punto y la invocación a la función. Ej:

```
matrices.suma(m1,m2)
```

```
In [88]: #importamos el módulo
import matrices

m1 = [[1,2],[3,4]]
m2 = [[1,2],[3,4]]

matrizSuma = matrices.suma(m1,m2)
print matrizSuma

[[2, 4], [6, 8]]
```

Podemos importar solo una función de un módulo y entonces no hace falta poner el nombre del módulo al invocar la función Usamos la fórmula:

```
from módulo import función
```

```
In [90]: from random import shuffle

sorteo = [1,2,3,4,5,6,7,8,9]
shuffle(sorteo)
print sorteo

[7, 1, 4, 6, 9, 3, 8, 5, 2]
```

Expresiones regulares

Haremos uso del paquete `re`

Podemos usar el método `search(patrón,texto)` para encontrar la primera coincidencia del patrón en la cadena

```
In [91]: import re

cadena = """En un lugar de la Mancha, de cuyo nombre no quiero acordarme,
de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor
salpicón las más noches, duelos y quebrantos los sábados, lentejas los vi
domingos, consumían las tres partes de su hacienda. """

#buscamos la primera palabra de cinco o más letras
patronCincoLetras = r"\s\w\w\w\w\w\s"

palabraCincoLetras = re.search(patronCincoLetras,cadena)

print palabraCincoLetras.group()

lugar
```

```
In [92]: patronCincoLetrasEmpiezaVocal = r"\s[aeiou]\w\w\w\w+\s"

palabraCincoLetrasEmpiezaVocal = re.search(patronCincoLetrasEmpiezaVocal,

print palabraCincoLetrasEmpiezaVocal.group()

adarga
```

Si queremos obtener varias partes dentro de un mismo patrón, podemos especificarlo mediante grupos. Cada grupo se indica encerrando la parte correspondiente del patrón entre paréntesis

```
In [94]: #patrón que nos captura el espacio inicial y la coma final
patronCincoLetrasEmpiezaVocal2 = r"\s[aeiou]\w\w\w\w+\W"

#veríamos la coma que captura \W
palabraCincoLetrasEmpiezaVocal = re.search(patronCincoLetrasEmpiezaVocal2
print palabraCincoLetrasEmpiezaVocal.group()

#para evitar la coma, agrupamos cada parte del patrón
patronCincoLetrasEmpiezaVocal3 = r"\s([aeiou]\w\w\w\w+)\W"

palabraCincoLetrasEmpiezaVocal = re.search(patronCincoLetrasEmpiezaVocal3

print palabraCincoLetrasEmpiezaVocal.group(1)

acordarme,
acordarme
```

Con la función `findall` podemos obtener todas las repeticiones del patrón

```
In [95]: cadenaEmails = 'escribeme a alice@google.com o contacta con mi compañero

correos = re.findall(r'([\w\.-]+)@([\w\.-]+)', cadenaEmails)
for correo in correos:
    print correo

('alice', 'google.com')
('bob', 'abc.com')
```

Para buscar a lo largo de varias líneas, usamos la opción DOTALL

```
In [96]: coincidencia = re.search(r'hidalgo.*astillero', cadena)

if coincidencia is not None:
    print coincidencia.group(0)
else:
    print "No se encuentra el patrón"

No se encuentra el patrón
```

```
In [98]: coincidencia = re.search(r'hidalgo.*astillero', cadena, re.DOTALL)

if coincidencia is not None:
    print coincidencia.group()
else:
    print "No se encuentra el patrón"

hidalgo
de los de lanza en astillero
```

Si queremos buscar el inicio o final de una línea, usamos la opción MULTILINE

```
In [99]: coincidencia = re.search(r'^de los.*', cadena)

if coincidencia is not None:
    print coincidencia.group()
else:
    print "No se encuentra el patrón"

No se encuentra el patrón
```

```
In [100]: coincidencia = re.search(r'^de los.*', cadena, re.MULTILINE)

if coincidencia is not None:
    print coincidencia.group()
else:
    print "No se encuentra el patrón"

de los de lanza en astillero, adarga antigua, rocín flaco y galgo corre
```

Podemos usar la opción IGNORECASE para que se traten por igual mayúsculas y minúsculas en un patrón

```
In [101]: coincidencia = re.findall(r'[\s\W]m\S+[\s\W]', cadena)
if coincidencia is not None:
    print coincidencia
    for c in coincidencia:
        print c
else:
    print "No se encuentra el patrón"
```

```
[' mucho ', ' m\xc3\xals ', ' m\xc3\xals ']
mucho
más
más
```

```
In [102]: coincidencia = re.findall(r'[\s\W]m\S+[\s\W]', cadena, re.IGNORECASE)
if coincidencia is not None:
    print coincidencia
    for c in coincidencia:
        print c
else:
    print "No se encuentra el patrón"
```

```
[' Mancha, ', ' mucho ', ' m\xc3\xals ', ' m\xc3\xals ']
Mancha,
mucho
más
más
```

Podemos realizar reemplazos de partes de una cadena por otra con la función `sub(patrón, reemplazo, cadena)`

```
In [104]: #cambiamos las comas por <stop>
nuevaCadena = re.sub(',', ' <stop>' , cadena)
print nuevaCadena
```

```
En un lugar de la Mancha <stop> de cuyo nombre no quiero acordarme <stop>
de los de lanza en astillero <stop> adarga antigua <stop> rocín flaco y
salpicón las más noches <stop> duelos y quebrantos los sábados <stop> l
domingos <stop> consumían las tres partes de su hacienda.
```

La función `sub` nos permite hacer referencia a distintos grupos dentro del patrón para cambiarlos de sitio o mantenerlos en la cadena sustituida. Para hacer referencia a estos grupos en el reemplazo, usamos la barra `\` y el número de grupo comenzando en 1.

```
In [106]: cadenaMarcado = """<h1>Esto es un primer encabezado</h1>
<h2>Esto es un segundo encabezado</h2>"""

cadenaNueva = re.sub(r'<h(\d)>(.*?)</h\d>' , r'<h3>\2 que antes era de tip
print cadenaNueva
```

```
<h3>Esto es un primer encabezado que antes era de tipo 1</h3>
<h3>Esto es un segundo encabezado que antes era de t:
```

