1 Python - Expresiones regulares

Como en todos los lenguajes, en Python también podemos utilizar Expresiones Regulares para crear patrones que nos permitan filtrar texto.

Para ello, en Python tenemos el módulo re. Veamos una serie de ejemplos prácticos sencillos:

- **Comprobar si en un texto hay símbolos o espacios**

```
import re

texto = input("Introduce texto : ")

patronSimbol = re.compile('\W')

if patronSimbol.search(texto):
    print("Sí hay símbolos")
else:
    print("No hay símbolos")
```

  - ◊ **re.compile(pattern)**
    Compila un patrón de expresión regular en un objeto de expresión regular, que puede ser usado para las coincidencias usando **match()**, **search()** y otros métodos, descritos más adelante.
  - ◊ **Pattern.search(string[, pos[, endpos]])**
    Escanea a través de la *string* buscando la primera ubicación donde esta expresión regular produce una coincidencia, y retorna un objeto *match* correspondiente. Retorna *None* si ninguna posición en la cadena coincide con el patrón; notar que esto es diferente a encontrar una coincidencia de longitud cero en algún punto de la cadena.
    El segundo parámetro opcional *pos* proporciona un índice en la cadena donde la búsqueda debe comenzar; por defecto es 0.
    El parámetro opcional *endpos* limita hasta dónde se buscará la cadena; será como si la cadena fuera de *endpos* caracteres de largo, por lo que sólo se buscará una coincidencia entre los caracteres de *pos* a *endpos - 1*. Si *endpos* es menor que *pos*, no se encontrará ninguna coincidencia.

- **Buscar en un texto todas las IPsv4 y las MACs existentes :**

```
import re

texto = """
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s31f6: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN group default qlen 1000
    link/ether 88:28:fd:c3:6f:c2 brd ff:ff:ff:ff:ff:ff
3: wlp0s20f3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether ce:db:ac:da:84:28 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.15/24 brd 192.168.1.255 scope global dynamic noprefixroute wlp0s20f3
       valid_lft 85322sec preferred_lft 85322sec
    inet6 fe80::a8af:8041:769d:60f1/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
"""

patronIP = re.compile('(?:[0-9]{1,3}\.){3}[0-9]{1,3}')
patronMAC = re.compile('(?:[0-9A-F]{2}[:-]){5}[0-9A-F]{2}', re.IGNORECASE)

listaIPs = patronIP.findall(texto)
listaMACs = patronMAC.findall(texto)

print(listaIPs)
print(listaMACs)
```

  - ◊ **Pattern.findall(string[, pos[, endpos]])**
    Retorna todas las coincidencias no superpuestas de *pattern* en *string*, como una lista de *strings* o tuplas. El *string* se escanea de izquierda a derecha y las coincidencias se retornan en el orden en que se encuentran. Las coincidencias vacías se incluyen en el resultado.

--Volver