Python y expresiones regulares

Lenguaje de programación **multi-paradigma**, **interpretado** y con **tipado dinámico**, creado por Guido van Rossum a finales de los 80.

Ampliamente usado

- Scripting
- ML y PLN
- Deep Learning
- Desarrollo web
- Etc.

Guido van Rossum en 2006, de https://es.wikipedia.org/wiki/Python

Versión y ejecución

- Usaremos la versión **python 3**

- **Verificar la versión actual**

  python --version

- **Ejecutar un programa**

  python programa.py *param1 param2 ...*

Módulos

- Cada archivo **.py** define un módulo
- Para importar:
  - **import** m
  - **from** m **import** x, y, …
- En un módulo pueden haber **funciones**, **clases**, **variables, etc.**
- **Módulo de expresiones regulares:**

  import re

Tipos de datos

- **Bool** – True, False
- **Int** – 17, 8, 0, -1
- **Float** – 1.25, -7.49
- **Str** – ‘teoria de lenguajes’, “hola” + ‘!’
- **List** – [1,2,3], [‘a’,1], lista[0:5:1]
- **Tuple** (listas inmutables) – (1,2,3), tuple([1,2])
- **Dict** – {‘a’: 0, ‘b’: 7}, dict(a=0, b=7), diccionario[‘a’]
- **Set** – set(…) – soporta unión, intersección, etc.

Ejemplo

```python
# Calcula el n-ésimo número de la secuencia de fibonacci
def fibonacci_recursivo(n):
    if n == 0 or n == 1:
        return n
    else:
        return fibonacci_recursivo(n-1) + fibonacci_recursivo(n-2)

def fibonacci_iterativo(n):
    x0 = 0
    x1 = 1
    for i in range(0, n):
        temp = x0 + x1
        x0 = x1
        x1 = temp
    return x0


print(fibonacci_recursivo(10))
print(fibonacci_iterativo(10))
```

Expresiones regulares en Python

En python (en el **módulo re**) una expresión regular se denota como **r'...'**.

Ejemplos:

- r'ab*'
- r'(ab)*'
- r'a*|b*'

Funciones:

- re.search
- re.findall
- re.sub

Funciones del módulo re

- **re.search(pattern, string, flags=0)**
  - Busca una ocurrencia del patrón en la string
  - Retorna el **match**, o *None* si el patrón no ocurre en la string
  - En caso de ocurrencia, con **group** se accede a los grupos del match

```
m.group(0)  # 0 es el grupo de la er completa
```

  - El método **groups** retorna todos los grupos de la er

Funciones del módulo re

- **re.findall(pattern, string, flags=0)**
  - Retorna la lista de ocurrencias no solapadas del patrón en la string
  - Si hay grupos definidos en la er, retorna una lista de tuplas

- **re.sub(pattern, repl, string, count=0, flags=0)**
  - Retorna la string resultado de reemplazar las ocurrencias del patrón por *repl*
  - Si *count* es distinto de 0, reemplaza un máximo de *count* ocurrencias

Metacaracteres

. – cualquier caracter menos fin de línea

***** – 0 o más ocurrencias

{n,m} – de n a m ocurrencias

? – 0 o 1 ocurrencias

+ – 1 o más ocurrencias

| – unión

**** – caracter de escape

^ y **$** – principio y fin de la entrada

(...) – agrupamiento

Expresiones regulares

- **Clases de caracteres**

  […] – cualquiera de los caracteres

  Admite rangos, ej. [1-5]

- **Abreviaciones de clases**
  - **\d** es un dígito [0-9]
  - **\w** es un caracter alfanumérico [a-zA-Z0-9]
  - **\s** es un espacio, **\b** el comienzo de una palabra
  - Cualquiera de ellos, en mayúscula, refiere al complemento

    Ej. **\D** es un no-dígito

Expresiones regulares

- **Flags**
  - **re.I –** ignora mayúsculas y minúsculas
  - **re.MULTILINE –** cambia el comportamiento de ^ y $
  - **re.DOTALL –** . matchea fines de línea

- **Operadores greedy y non-greedy**
  - **?** Indica a los operadores de repetición que abarquen lo menos posible
  - Uso: *?, +?, ??

Ejemplo de entrada

```
<doc id="841352" title="Nucifraga" nonfiltered="1" processed="1"
dbindex="230002">

El género Nucifraga perteneciente a la familia Corvidae incluye
a dos especies de cascanueces: el cascanueces del Viejo Mundo y
el cascanueces norteamericano o de Clark.




ENDOFARTICLE.
</doc>
```

Ejemplos

- **Eliminar etiquetas de abrir**
  - re.sub(r"<doc[^>]*>", "", texto)
  - re.sub(r"<doc.*?>", "", texto)
- **Obtener palabras (delimitadas por espacios)**
  - re.findall(r"(\S+)", texto)
- **Cantidad de palabras**
  - len(re.findall(r"(\S+)", texto))