Clase 5

Repaso

Expresiones Regulares

re El módulo que contiene las funciones para manejar expresiones regulares en Python.

r"" Cuando utilicen expresiones regulares, recuerden añadir una 'r' antes de sus textos. De esta manera, Python sabe que no debe esperar caracteres especiales, más que de expresiones regulares.

compile() Esta es una función del módulo `re`, y le indica a Python que el texto que recibe como parámetro ES una expresión regular y debe interpretarse como tal. Esta función *regresa* la expresión regular para que se guarde en una variable y se puedan utilizar las funciones propias de las expresiones regulares.

Repaso

Expresiones Regulares

search() La primera función que vimos que pertenece a las variables de expresiones regulares (aquellas que se obtienen con la función `compile()` del módulo `re`). Esta función recibe como parámetro el texto en el que se debe buscar la expresión regular, y *regresa* la primera coincidencia que encuentra. Deja la variable vacía si es que no encuentra nada.

finditer() Esta función de las expresiones regulares hace una búsqueda de la expresión regular sobre el texto que recibe como parámetro, pero *regresa* un iterador sobre TODAS las apariciones que encuentra.

group(0) Las expresiones regulares se pueden agrupar, lo veremos más adelante. Por ahora, recuerden que la esta función *regresa* el contenido del grupo que recibe como parámetro, por ahora, estamos usando el primer grupo, es decir, el 0.

Repaso

Expresiones Regulares

- `.` El punto es un símbolo especial de las expresiones regulares, coincide con cualquier caracter, con excepción del salto de línea.
- `*` El asterisco es un repetidor. Afecta al caracter que se encuentra inmediatamente antes, y va a coincidir con 0 o más repeticiones de dicho caracter.
- `+` Muy similar al asterisco. El símbolo más coincide con 1 o más repeticiones del caracter inmediatamente anterior.
- `?` El signo de interrogación también afecta al caracter anterior. Coincide cuando hay 0 o 1 aparición de dicho caracter, es decir, lo hace opcional.

Repaso

Expresiones Regulares

- ^ Este símbolo coincide con el inicio del texto. NO coincide con el inicio de la línea, es solo el inicio del texto. Dicho esto, si ustedes dividen su texto, por ejemplo en líneas, cada parte tendrá un inicio de texto.
- $ Este símbolo coincide con el final del texto. Es la contraparte del símbolo anterior y la misma nota aplica.

Ejercicio 9

- Encuentren todas las "palabras" de 3 o 4 cuatro letras de uno de sus documentos.
- Vamos a entender por "palabra" cualquier cosa que esté entre espacios.
- USEN EXPRESIONES REGULARES.

Y por costumbre, el segundo ejercicio:

- A partir de una de SUS carpetas (Descargas, Escritorio, Documentos). Encuentren con una expresión regular todos sus archivos DOC y DOCX.

Expresiones regulares

- El signo de interrogación (?) tiene además una función importante.
- Se puede usar en conjunto con los símbolos que buscan muchas letras (como el '+' y el '*').
- Normalmente la expresión regular busca el texto mas largo que coincida, pero si se usa el signo de interrogación eso cambiará y buscará el mas corto.

```
expresion_larga=re.compile(r"\(.*\)")
expresion_corta=re.compile(r"\(.*?\)")
resultados_largos=expresion_larga.finditer(texto)
resultados_cortos=expresion_corta.finditer(texto)

for resultado in resultados_largos:
        print(resultado.group(0))
print()
for resultado in resultados_cortos:
        print(resultado.group(0))
```

Ejercicio 10

Confío en que todos conocemos el abecedario.

- Recorran una lista de las letras mayúsculas.
- Obtengan todas las oraciones de su texto.
- Por oraciones, vamos a entender todo lo que hay de mayúscula a punto

Expresiones regulares

- Seguramente han notado en los ejemplos anteriores que los símbolos afectan únicamente a la letra que los precede.
- Es posible también afectar a un grupo de letras, para eso, es necesario agruparlas, y ese es precisamente la función de los paréntesis: "()".

```
expresion_regular=re.compile(r"(el)?(los)? artículos?")
resultados_busqueda=expresion_regular.finditer(texto)

for resultado in resultados_busqueda:
        print(resultado.group(0))
```

Expresiones regulares

- En el ejemplo anterior se logra un efecto similar a capturar una de las dos expresiones "el" o "los", seguido de "artículo" o "artículos".
- Las expresiones regulares tienen una instrucción propia para especificar que se quiere una cosa o la otra, y es con el símbolo: '|'.

```
expresion_regular=re.compile(r"(el|los) artículos?")
resultados_busqueda=expresion_regular.finditer(texto)

for resultado in resultados_busqueda:
        print(resultado.group(0))
```

Expresiones regulares

- Otro símbolo especial, los corchetes: "[]".
- Lo podemos ver como un atajo para poner diferentes opciones, aunque las opciones sólo pueden ser de una letra, aunque se pueden poner muchas letras dentro del corchete. Al momento de la búsqueda el programa intentará coincidir con alguna de ellas.

```
expresion_regular=re.compile(r"M[eéa][xr]ic?[ao](nos)?")
resultados_busqueda=expresion_regular.finditer(texto)

for resultado in resultados_busqueda:
        print(resultado.group(0))
```

Expresiones regulares

- Además de eso, los corchetes también aceptan rangos de valores. Para eso se utilizan los extremos del rango con un símbolo de menos (-) entre ellos.
- Los usos del corchete se pueden combinar.

```
expresion_regular=re.compile(r"M[a-záéíóú][a-z]ic?[a-z](nos)?")
resultados_busqueda=expresion_regular.finditer(texto)

for resultado in resultados_busqueda:
        print(resultado.group(0))
```

Ejercicio 11

- Repitan el ejercicio anterior SIN hacer un recorrido en lista.

Expresiones regulares

- Y todavía un uso más del corchete. Podemos usarlo para indicar lo que queremos que NO coincida en nuestra expresión regular.
- Para lograr eso, usamos el símbolo '^' justo al inicio del corchete, antes del conjunto de letras que queremos excluir.

```
expresion_regular=re.compile(r"M[^a-z][^0-9]ic?[a-z](nos)?")
resultados_busqueda=expresion_regular.finditer(texto)

for resultado in resultados_busqueda:
        print(resultado.group(0))
```

Expresiones regulares

- Con los corchetes podemos hacer recorridos de letras o números para manejar grupos.
- Pero las expresiones regulares tienen predefinidos ciertos grupos de letras, números y símbolos que facilitan las cosas aún más.
- Comencemos con el grupo: `\d` , que incluye a todos los dígitos. Sería equivalente a: `[0-9]`

```
expresion_regular=re.compile(r"\d+(,\d+)*(\.\d+)?")
resultados_busqueda=expresion_regular.finditer(texto)

for resultado in resultados_busqueda:
        print(resultado.group(0))
```

Expresiones regulares

- Otro grupo predefinido es el de los espacios: **\s**
- Este conjunto incluye al espacio, al tabulador, y a otros símbolos que se usan para poner espacios en blanco en un texto.

```
expresion_regular=re.compile(r"[^\s]+")
resultados_busqueda=expresion_regular.finditer(texto)

for resultado in resultados_busqueda:
        print(resultado.group(0))
```

Expresiones regulares

- Un grupo muy importante también: `\w`
- Este conjunto incluye todos los caracteres que se esperaría pudieran aparecer en una palabra. Esto incluye a todas las letras, todos los dígitos y el guión bajo ('_')
- Ojo, este grupo tiene la ventaja de que SÍ incluye a los acentos.

```
expresion_regular=re.compile(r"\w+")
resultados_busqueda=expresion_regular.finditer(texto)

for resultado in resultados_busqueda:
        print(resultado.group(0))
```

Expresiones regulares

- Los grupos en mayúsculas, es decir: `\D` , `\S` , y `\W` . Coinciden con los opuestos de `\d` , `\s` y `\w` respectivamente.
- Es decir, es equivalente a ponerlos dentro de corchetes con '^' inicial.

Ejercicio 12

Obtengan con una expresión regular los artículos constitucionales que se mencionan en el texto. Tomen en cuenta que:

- Los artículos van precedidos de la palabra: "artículo" o "artículos".
- Los artículos son números.
- Cuando se habla de varios artículos, éstos pueden estar separados por comas, o por la palabra "y"