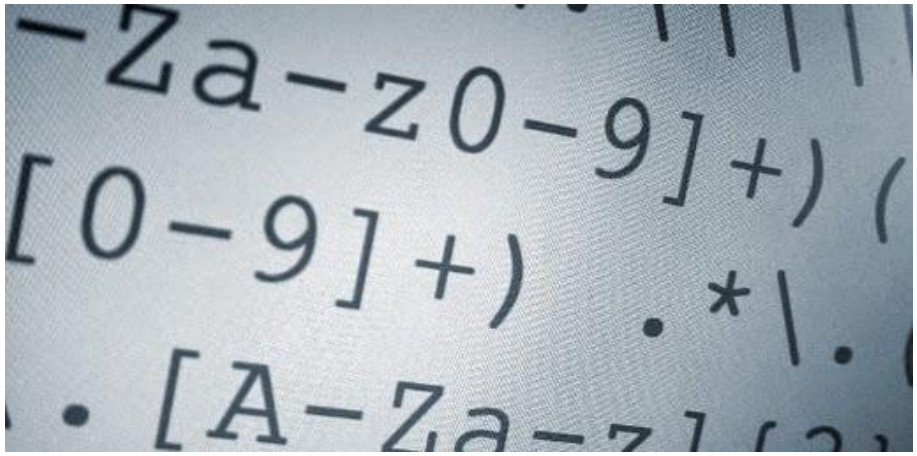
TI 3001 C

Analítica de datos y herramientas de inteligencia artificial

Módulo re

Tecnológico de Monterrey



¿Cómo usar las expresiones regulares?

En la librería estándar de Python podemos encontrar el **módulo re**, el cual nos proporciona todas las operaciones necesarias para trabajar con las expresiones regulares.

Importar el módulo **re** de Python.

```
# importando el modulo de regex de python
import re
```

¿Cómo usar las expresiones regulares?

Buscando coincidencias

Algunos métodos para usar el módulo **re** de Python:

- **match()**: Determina si la expresión regular tiene coincidencias en el comienzo del texto.
- **search()**: Escanea todo el texto buscando cualquier ubicación donde coincida la expresión regular. Devuelve un objeto math.
- **findall()**: Encuentra todos los subtextos donde coincide la expresión regular y devuelve estas coincidencias como una lista.

Todos estos métodos reciben dos parámetros: la **expresión a evaluar** y el **texto**.

Método search

- **search()**: Escanea todo el texto buscando cualquier ubicación donde coincida la expresión regular. Devuelve un objeto Match.

Ejemplo 1: Busca la palabra “Hola” en un texto.

```
1 import re
2 texto = input("Introduce un mensaje: ")
3 print(re.search("Hola", texto)) # (Expresión regular, texto)
```

Shell

```
>>> %Run ER_Search.py
Introduce un mensaje: Hola a todos
<re.Match object; span=(0, 4), match='Hola'>
```

La forma más simple de una expresión regular es una palabra

El método **search** regresa un objeto de tipo match, el span nos dice en que parte del texto se encontró la coincidencia “Hola” (0, 4), desde la posición 0 hasta una posición antes de la 4.

Un **objeto match** es un objeto en Python que nos da información sobre la coincidencia.

Método search

Ejemplo 2: Busca la palabra Adiós en un texto.

```
1 import re
2 texto = input("Introduce un mensaje: ")
3 print(re.search("Adiós", texto)) # (Expresión regular, texto)
```

Shell

```
>>> %Run ER_Search.py
Introduce un mensaje: Hola a todos
None
```

En caso de que no encuentre coincidencia, regresa **None**.

Método search

Cuantificador (+) Una o más veces

Ejemplo 3: Busca la palabra Python y que la admiración se repita 1 o más veces

```
1  import re
2  texto = """Hola Mundo.
3  Me gusta Python!!!!
4  Mi numero telefonico de casa es 442-130-45-12
5  Mi numero telefonico de celular es 442-241-37-46
6  Mi numero telefonico de oficina es 442-380-14-22
7  """
8  # Busca la primer coincidencia
9  print(re.search("Python!+", texto)) # (Expresión regular, texto)
10 # Regresa un objeto match
```

Shell

```
>>> %Run ER_SearchCuantificador.py
<re.Match object; span=(21, 31), match='Python!!!!'>
```

Método match

- **match()**: Determina si la expresión regular tiene coincidencias en el comienzo del texto.

Ejemplo 1:

```
1 import re
2 texto = input("Introduce un mensaje: ")
3 print(re.match("Hola", texto)) # (Expresión regular, texto)
```

Shell

```
>>> %Run ER_Match.py
Introduce un mensaje: Hola a todos
<re.Match object; span=(0, 4), match='Hola'>
```

Método match

Ejemplo 2:

```
1 import re
2 texto = input("Introduce un mensaje: ")
3 print(re.match("Adiós", texto)) # (Expresión regular, texto)
```

Shell

```
>>> %Run ER_Match.py
Introduce un mensaje: Hola y Adiós
None
```

Método findall

- **findall()**: Encuentra todos los subtextos donde coincide la expresión regular y devuelve estas coincidencias como una lista.

Ejemplo 1:

```
1 import re
2 texto = input("Introduce un mensaje: ")
3 print(re.findall("is", texto)) # (Expresión regular, texto)
```

Shell

```
>>> %Run ER_Match.py
Introduce un mensaje: She is my sister Lis
['is', 'is', 'is']
```

Método findall

Ejemplo 2:

```
1 import re
2 texto = input("Introduce un mensaje: ")
3 print(re.findall("is", texto)) # (Expresión regular, texto)
```

Shell

```
>>> %Run ER_Match.py
Introduce un mensaje: Ella es mi hermana Luz
[]
```

¿Cómo usar las expresiones regulares?

Modificando el texto de entrada

- **split():** El cual divide el texto en una lista, realizando las divisiones del texto en cada lugar donde se cumple con la expresión regular.
- **sub():** El cual encuentra todos los subtextos donde existe una coincidencia con la expresión regular y luego los reemplaza con un nuevo texto.
- **subn():** El cual es similar al anterior pero además de devolver el nuevo texto, también devuelve el numero de reemplazos que realizó.

Método split

- **split():** El cual divide el texto en una lista, realizando las divisiones del texto en cada lugar donde se cumple con la expresión regular.

Ejemplo 1: Divide el texto mientras no encuentre un carácter alfanumérico.

```python
import re

texto = """Podrá nublarse el sol eternamente;
Podrá secarse en un instante el mar;
Podrá romperse el eje de la tierra
como un débil cristal.
¡todo sucederá! Podrá la muerte
cubrirme con su fúnebre crespón;
Pero jamás en mí podrá apagarse
la llama de tu amor."""

# Dividir el texto por caracteres que no son alfanuméricos

lista = re.split("\W+", texto)
print(lista)
```

Shell

```
>>> %Run Expresiones_split.py
['Podrá', 'nublarse', 'el', 'sol', 'eternamente', 'Podrá', 'secarse', 'en', 'un
', 'instante', 'el', 'mar', 'Podrá', 'romperse', 'el', 'eje', 'de', 'la', 'tier
ra', 'como', 'un', 'débil', 'cristal', 'todo', 'sucederá', 'Podrá', 'la', 'muer
te', 'cubrirme', 'con', 'su', 'fúnebre', 'crespón', 'Pero', 'jamás', 'en', 'mí'
, 'podrá', 'apagarse', 'la', 'llama', 'de', 'tu', 'amor', '']
```

Método split

Ejemplo 2: Divide por línea.

```
1  import re
2
3  texto = """Podrá nublarse el sol eternamente;
4  Podrá secarse en un instante el mar;
5  Podrá romperse el eje de la tierra
6  como un débil cristal.
7  ¡todo sucederá! Podrá la muerte
8  cubrirme con su fúnebre crespón;
9  Pero jamás en mí podrá apagarse
10 la llama de tu amor."""
11
12 # Dividir el texto por caracteres que no son alfanuméricos
13
14 lista = re.split("\n", texto)
15 print(lista)
```

Shell

```
>>> %Run Expresiones_split2.py
['Podrá nublarse el sol eternamente; ', 'Podrá secarse en un instante el mar; '
, 'Podrá romperse el eje de la tierra ', 'como un débil cristal. ', '¡todo suce
derá! Podrá la muerte ', 'cubrirme con su fúnebre crespón; ', 'Pero jamás en mí
podrá apagarse ', 'la llama de tu amor.']
```

Método sub

- **sub():** El cual encuentra todos los subtextos donde existe una coincidencia con la expresión regular y luego los reemplaza con un nuevo texto.

Ejemplo 1: Sustituir "Podrá" por "Puede" en el texto.

```
3  texto = """Podrá nublarse el sol eternamente;
4  Podrá secarse en un instante el mar;
5  Podrá romperse el eje de la tierra
6  como un débil cristal.
7  ¡todo sucederá! Podrá la muerte
8  cubrirme con su fúnebre crespón;
9  Pero jamás en mí podrá apagarse
10 la llama de tu amor."""
11
12 resultado = re.sub("Podrá", "Puede", texto)
13 print(resultado)
```

Shell

```
>>> %Run Expresiones_sub.py
Puede nublarse el sol eternamente;
Puede secarse en un instante el mar;
Puede romperse el eje de la tierra
como un débil cristal.
¡todo sucederá! Puede la muerte
cubrirme con su fúnebre crespón;
Pero jamás en mí podrá apagarse
la llama de tu amor.
```

Método sub

- **sub():** El cual encuentra todos los subtextos donde existe una coincidencia con la expresión regular y luego los reemplaza con un nuevo texto.

Ejemplo 1: Sustituir "Podrá" por "Puede" en el texto.

```
3  texto = """Podrá nublarse el sol eternamente;
4  Podrá secarse en un instante el mar;
5  Podrá romperse el eje de la tierra
6  como un débil cristal.
7  ¡todo sucederá! Podrá la muerte
8  cubrirme con su fúnebre crespón;
9  Pero jamás en mí podrá apagarse
10 la llama de tu amor."""
11
12 resultado = re.sub("(P|p)odrá", "Puede", texto)
13 print(resultado)
```

Shell

```
>>> %Run Expresiones_sub.py
 Puede nublarse el sol eternamente;
 Puede secarse en un instante el mar;
 Puede romperse el eje de la tierra
 como un débil cristal.
 ¡todo sucederá! Puede la muerte
 cubrirme con su fúnebre crespón;
 Pero jamás en mí Puede apagarse
 la llama de tu amor.
```

Método sub

Ejemplo 2: Convertir un número (442)-223-78-90 por 4422237890.

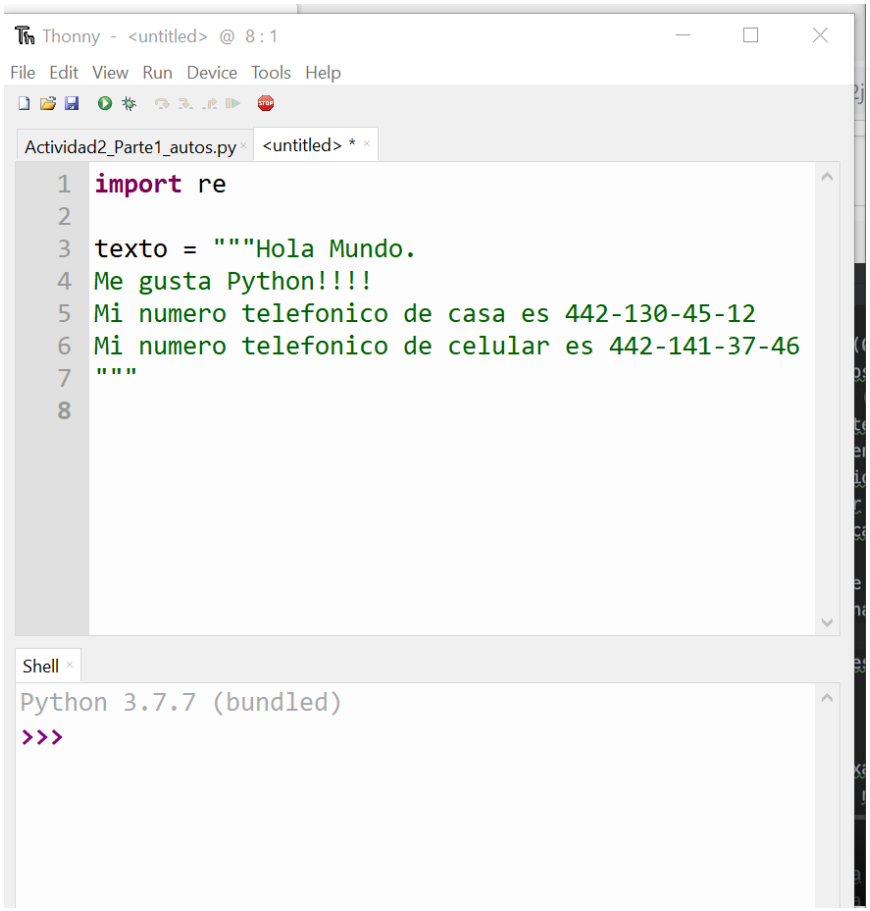
```
1  import re
2
3  texto = "(442)-442-153-32-42"
4
5  resultado = re.sub("\D", "", texto)
6  print(resultado)
```

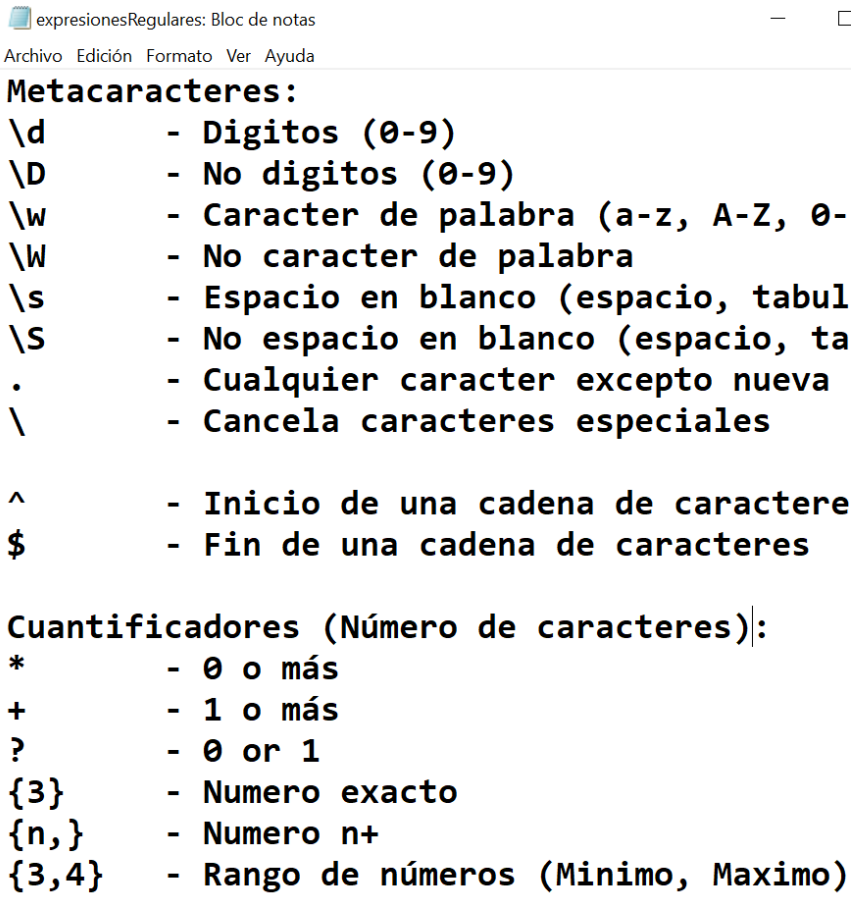
Shell

```
>>> %Run Expresiones_sub2.py
 4424421533242
```

Método re y expresiones regulares

- A la izquierda tenemos un texto en el cual vamos a identificar expresiones regulares.
- A la derecha tenemos una lista de caracteres especiales y diferentes caracteres usados en expresiones regulares.





Expresiones regulares: \d

Ejemplo: Busca el primer dígito en el texto.

r se utiliza en Python para anular caracteres o palabras especiales de Python.

re.search(r"Expresión regular", texto)

```
1  import re
2  texto = """Hola Mundo.
3  Me gusta Python!!!!
4  Mi numero telefonico de casa es 442-130-45-12
5  Mi numero telefonico de celular es 442-241-37-46
6  Mi numero telefonico de oficina es 442-380-14-22
7  """
8  # Busca la primer coincidencia
9  print(re.search(r"\d", texto)) # (Expresión regular, texto)
```

Shell

```
>>> %Run ER_SearchCuantificador2.py
<re.Match object; span=(64, 65), match='4'>
```

Expresiones regulares: \d

https://regexr.com/

20 coincidencias

Metacaracteres

Son caracteres con un significado especial.

- En los metacaracteres está el poder de las Expresiones regulares. El verdadero valor de las expresiones regulares son los metacaracteres.

MC	Descripción	Ejemplo
[]	Indica un conjunto de caracteres. Se usa '-' para indicar un rango. Algunos caracteres especiales pierden su significado.	[a-z]
.	Cualquier carácter excepto un salto de línea.	….ar
^	Coincide con el comienzo de la cadena.	^[A-Z]
$	Coincide con el final de la cadena.	@gmail.com$
|	Una 'or'.	[a-z] | [A-Z]

Metacaracteres

[] Indica un conjunto de caracteres

- Se usa '-' para indicar un rango.
- **findall()**: Encuentra todos los subtextos donde coincide la expresión regular y devuelve estas coincidencias como una lista.

Ejemplo 1: Busca todos los caracteres en minúsculas.

```
1 import re
2 texto = input("Introduce un mensaje: ")
3 print(re.findall("[a-z]", texto))
4 # Caracteres de la 'a' a la 'z' en minúsculas.
```

Shell

```
>>> %Run MC_Corchetes.py
Introduce un mensaje: Mundial 1986
['u', 'n', 'd', 'i', 'a', 'l']
```

Metacaracteres

[] Indica un conjunto de caracteres

Ejemplo 2: Busca todos los números.

```
1 import re
2 texto = input("Introduce un mensaje: ")
3 print(re.findall("[0-9]", texto))
4 # Caracteres de la 'a' a la 'z' en minúsculas.
```

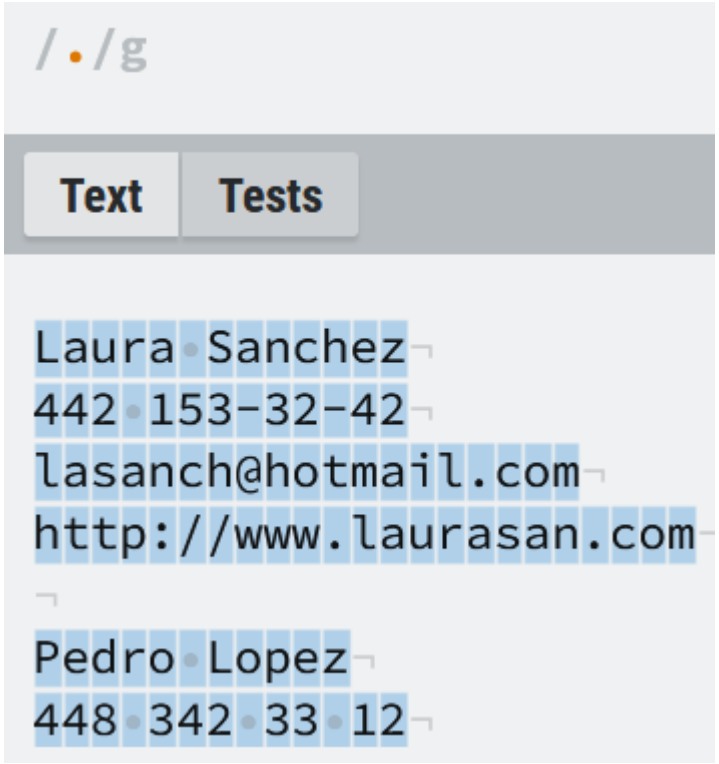
Shell

```
>>> %Run MC_Corchetes.py
Introduce un mensaje: Mundial 1986
['1', '9', '8', '6']
```

Metacaracteres

Punto(.) Cualquier carácter excepto el salto de línea

- Encuentra todos los caracteres.



Metacaracteres

Punto(.) Cualquier carácter excepto el salto de línea

Ejemplo 3: Texto con 4 caracteres y que termine en ar.

```
1 import re
2 texto = input("Introduce un mensaje: ")
3 print(re.findall("....Ar", texto))
```

Shell

```
>>> %Run MC_Punto.py
Introduce un mensaje: Saludos Armando!!
['dos Ar']
```

Metacaracteres

Punto(.) Cualquier carácter excepto el salto de línea

Ejemplo 3: Texto con 4 caracteres y que termine en ar.

```
1 import re
2 texto = input("Introduce un mensaje: ")
3 print(re.findall("....ar", texto))
```

Shell

```
>>> %Run MC_Punto2.py
 Introduce un mensaje: cantar, bailar, reir, soñar
 ['cantar', 'bailar', ' soñar']
```

Metacaracteres

^ Coincide con el comienzo de la cadena

Ejemplo 4: Letra que comience en mayúsculas.

```
1 import re
2 texto = input("Introduce un mensaje: ")
3 print(re.findall("^[A-Z]", texto))
```

Shell

```
Python 3.7.7 (bundled)
>>> %Run MC_Gorrito.py
 Introduce un mensaje: Hola Juan y Pedro!!
 ['H']
```

Metacaracteres

^ Coincide con el comienzo de la cadena

Ejemplo 4: Letra que comience en mayúsculas.

```
1 import re
2 texto = input("Introduce un mensaje: ")
3 print(re.findall("^[A-Z]", texto))
```

Shell

```
>>> %Run MC_Gorrito.py
Introduce un mensaje: adios Juan y Pedro!!
[]
```

Metacaracteres

$ Coincide con el final de la cadena

Ejemplo 5: El correo termine en @tec.mx y se encuentre al final del texto.

```
1 import re
2 texto = input("Introduce un mensaje: ")
3 print(re.findall("@tec.mx$", texto))
```

Shell

```
Python 3.7.7 (bundled)
>>> %Run MC_Gorrito.py
Introduce un mensaje: juanpa@tec.mx, peterjua@tec.mx
['@tec.mx']
```

Metacaracteres

$ Coincide con el final de la cadena

Ejemplo 5: El correo termine en @tec.mx y se encuentre al final del texto.

```
1 import re
2 texto = input("Introduce un mensaje: ")
3 print(re.findall("@tec.mx$", texto))
```

Shell

```
>>> %Run MC_Pesos.py
Introduce un mensaje: janpe@tec.mx, patysal@gmail.com
[]
```

Metacaracteres

($) Fin de una cadena de caracteres

Ejemplo: Si quiero que una línea termine con “Mundo.”

```
/Mundo.$/g
```

Text | Tests

```
Hola Mundo.
Me gusta Python!!!!
Mi numero telefonico de casa es 442-130-45-12
Mi numero telefonico de celular es 442-241-37-46
Mi numero telefonico de oficina es 442-380-14-22
```

Metacaracteres

($) Coincide con el final de una cadena de caracteres

- Si quiero que una línea termine con “Mundo.”

```python
import re

texto = """Hola Mundo.
Me gusta Python!!!!
Mi numero telefonico de casa es 442-130-45-12
Mi numero telefonico de celular es 442-141-37-46
"""

# Busca la primer coincidencia
print(re.search(r"Mundo.$", texto))
```

Shell

```
>>> %Run expresionesregulares2.py
None
```

Metacaracteres

($) Coincide con el final de una cadena de caracteres

- Si quiero que una línea termine con “Mundo.”
- Agregar flag **re.M** para tomar texto multilínea. Que lea cada línea por separado.

```python
import re

texto = """Hola Mundo.
Me gusta Python!!!!
Mi numero telefonico de casa es 442-130-45-12
Mi numero telefonico de celular es 442-141-37-46
"""

# Busca la primer coincidencia
print(re.search(r"Mundo.$", texto, flags=re.M))
```

Shell

```
>>> %Run expresionesregulares2.py
<re.Match object; span=(5, 11), match='Mundo.'>
```

Metacaracteres

| (or)

Ejemplo 6: Letras mayúsculas y minúsculas.

```
1  import re
2  texto = input("Introduce un mensaje: ")
3  print(re.findall("[a-z]|[A-Z]", texto))
4
```

Shell

```
>>> %Run MC_Or.py
Introduce un mensaje: Mundial 1986
['M', 'u', 'n', 'd', 'i', 'a', 'l']
```

Metacaracteres

Son caracteres con un significado especial.

MC	Descripción	Ejemplo
*	0 o más repeticiones de la expresión regular precedente.	.*er
+	1 o más repeticiones de la expresión regular precedente. salto de línea.	.+er
{}	Número exacto de repeticiones. Puede tener número máximo y mínimo.	.{5}er
?	0 o 1 repetición de la expresión regular precedente. Prioriza el mínimo.	<.*?>
()	Capturar y agrupar.	

Metacaracteres

+ Definen el número de repeticiones de la expresión regular precedente

Ejemplo 1:

Una palabra que tenga cualquier carácter repetido una o más veces y que luego le siga un **er**.

```
1 import re
2 texto = input("Introduce un mensaje: ")
3 print(re.findall(".+er", texto))
```

Shell

```
>>> %Run MC_MasPor.py
Introduce un mensaje: Comer y Correr
['Comer y Correr']
```

Metacaracteres

+ Definen el número de repeticiones de la expresión regular precedente

Ejemplo 2:

Una palabra que tenga cualquier tipo de caracteres repetido una o más veces y que luego le siga un **er**.

```
1 import re
2 texto = input("Introduce un mensaje: ")
3 print(re.findall(".+er", texto))
```

Shell

```
Python 3.7.7 (bundled)
>>> %Run MC_MasPor.py
Introduce un mensaje: Comer y bailar
['Comer']
```

Metacaracteres

+ Definen el número de repeticiones de la expresión regular precedente

Ejemplo 3:

Palabras que tengan caracteres de la ‘a’ a la ‘z’ repetidos una o más veces y que luego le siga un **er**.

```
1 import re
2 texto = input("Introduce un mensaje: ")
3 print(re.findall("[a-z]+er", texto))
```

Shell

```
>>> %Run MC_MasPor.py
Introduce un mensaje: correr, barrer, morder
['correr', 'barrer', 'morder']
```

Metacaracteres

{} Definen el número exacto de repeticiones de la expresión regular precedente

Ejemplo 4:

Palabras que tengan cuatro caracteres de la ‘a’ a la ‘z’ repetidos una o más veces y que luego le siga un **er**.

```
1 import re
2 texto = input("Introduce un mensaje: ")
3 print(re.findall("[a-z]{4}er", texto))
```

Shell

```
>>> %Run MC_MasPor.py
Introduce un mensaje: barrer, comer, correr
['barrer', 'correr']
```

Metacaracteres

{} Definen el número exacto de repeticiones de la expresión regular precedente

Ejemplo 5:

Palabras que tengan cinco caracteres de la ‘a’ a la ‘z’ repetidos una o más veces y que luego le siga un **er**.

```
1 import re
2 texto = input("Introduce un mensaje: ")
3 print(re.findall("[a-z]{5}er", texto))
```

Shell

```
>>> %Run MC_Llaves.py
Introduce un mensaje: barrer, comer, correr
['barrer', 'comer']
```

Metacaracteres

{} Definen el número exacto de repeticiones de la expresión regular precedente

Ejemplo 6:

Obtenemos una lista de tuplas.

```
1 import re
2 texto = input("Introduce un mensaje: ")
3 print(re.findall("([a-z]{4})(er)", texto))
```

Shell

```
Python 3.7.7 (bundled)
>>> %Run MC_Llaves.py
 Introduce un mensaje: correr, barrer, morder
 [('corr', 'er'), ('barr', 'er'), ('mord', 'er')]
```

Secuencias especiales

\d Coincide con cualquier dígito decimal. Equivalente a [0-9].

\D Coincide con cualquier carácter que **no** sea un dígito. Equivalente a [^0-9].

\s Coincide con un espacio en blanco.

\S Coincide con cualquier carácter que **no** sea un espacio en blanco. Equivalente a [^ \t\n\r\f\v].

\w Coincide con cualquier carácter alfanumérico e incluye vocales con acentos. Equivalente a [a-zA-Z0-9_].

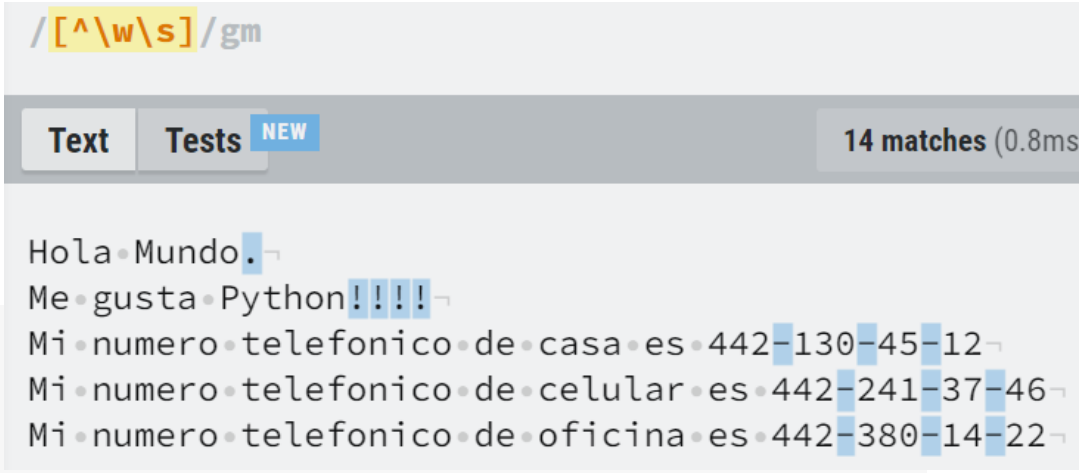
\W Coincide con cualquier carácter **no** alfanumérico. Equivalente a [^a-zA-Z0-9_].

Ejercicio

Encontrar puntuaciones

\w Caracter o guion bajo.

\s Espacio, tab, salto de línea




```python
import re

texto = """Hola Mundo.
Me gusta Python!!!!
Mi numero telefonico de casa es 442-130-45-12
Mi numero telefonico de celular es 442-241-37-46
Mi numero telefonico de oficina es 442-380-14-22"""

print(re.findall("[^\w\s?]", texto))
```

```
['.', '!', '!', '!', '!', '-', '-', '-', '-', '-', '-', '-', '-', '-']
```

Flags

re.M Multilínea

- Si quiero que una línea termine con “Mundo.”
- Agregar flag **re.M** para tomar texto multilínea. Que lea cada línea por separado.

```python
import re

texto = """Hola Mundo.
Me gusta Python!!!!
Mi numero telefonico de casa es 442-130-45-12
Mi numero telefonico de celular es 442-241-37-46
"""

# Busca la primer coincidencia
print(re.search(r"Mundo.$", texto, flags=re.M))
```

Shell

```
>>> %Run expresionesregulares3suma.py
<re.Match object; span=(5, 11), match='Mundo.'>
```

Flags

re.I (Ignore mayúsculas y minúsculas)

- Si quiero buscar una palabra como sin importar que este escrita en mayúsculas o minúsculas.

```python
import re

texto = """Hola Mundo.
Me gusta Python!!!!
Mi numero telefonico de casa es 442-130-45-12
Mi numero telefonico de celular es 442-241-37-46
"""

# Busca la primer coincidencia
print(re.search("^hola", texto, flags=re.I))
```

```
hell ×
>>> %Run expresionesregulares3suma.py
<re.Match object; span=(0, 4), match='Hola'>
```

Flags

re.I (Ignore mayúsculas y minúsculas)

- Si quiero buscar una palabra como sin importar que este escrita en mayúsculas o minúsculas.

```python
import re

texto = """Hola Mundo.
Me gusta Python!!!!
Mi numero telefonico de casa es 442-130-45-12
Mi numero telefonico de celular es 442-241-37-46
"""

# Busca la primer coincidencia
print(re.search("^hOLA", texto, flags=re.I))
```

Shell

```
>>> %Run expresionesregulares3suma.py
<re.Match object; span=(0, 4), match='Hola'>
```

Gracias