📝 Apuntes de Expresiones Regulares en Python (Módulo `re`)

Las expresiones regulares (regex) son una herramienta esencial para el manejo de cadenas de texto. Permiten buscar, validar y manipular patrones complejos dentro de cualquier texto.

1. El Módulo `re` y las Cadenas Raw (`r""`)

Para usar expresiones regulares en Python, siempre debes importar el módulo `re`.

```
import re
```

La Cadena Raw (`r""`) - ¡Clave para Evitar Errores!

Concepto: En Python, el carácter barra invertida (`\`) se usa para secuencias de escape (ej: `\n` es un salto de línea). Las regex también usan `\` para metacaracteres (ej: `\d` para dígitos).

Solución: Para que Python no interprete el `\` como una secuencia de escape normal, siempre se debe anteponer una `r` a la cadena del patrón.

Ejemplo:

- **Incorrecto:** `patron = "\[6-8]"` (Python lo procesa mal)
- **Correcto:** `patron = r"\[6-8]"` (La `r` indica que es una "raw string", respetando el `\` para la regex).

2. Sintaxis del Patrón: Metacaracteres y Cuantificadores

A. Conjuntos de Caracteres (`[]`)

El corchete define un conjunto de caracteres válidos para una única posición.

Símbolo	Significado	Ejemplo del Código	Resultado
`[a-z]`	Rango de minúsculas	`[6-8]`	Coincide con el dígito 6, 7 u 8.
`[0-9]`	Rango de dígitos	`[0-9]{8}`	8 dígitos, del 0 al 9.
`[A-Za-z]`	Rango de letras	`[A-Za-zÑ...]`	Coincide con letras mayúsculas, minúsculas, incluyendo Ñ y acentos.
`[^abc]`	**Negación**	`[^579]`	Coincide con **cualquier carácter excepto** 5, 7, o 9.

B. Cuantificadores (Repetición)

Indican cuántas veces puede repetirse el carácter o grupo que le precede.

Símbolo	Significado	Ejemplo del Código
`{n}`	Exactamente `n` veces.	`[0-9]{8}` (Ocho dígitos)
`{n, m}`	Entre `n` y `m` veces (inclusive).	`[A-Z]{4,8}` (Entre 4 y 8 letras)

C. Operadores de Repetición (Modificadores de Ocurrencia)

Estos son abreviaturas comunes para definir la opcionalidad o el número de apariciones.

Símbolo	Significado	Equivalente en `{}`	Ejemplo del Código
`?`	**Opcional:** Cero o una vez.	`{0,1}`	`[\s
`*`	Cero o más veces.	`{0,}`	`a*` (Ninguna 'a' o varias 'a')
`+`	**Obligatorio:** Una o más veces.	`{1,}`	`a+` (Debe haber al menos una 'a')

3. Las Tres Funciones Clave: Match, Search y Fullmatch

Esta es la parte más importante para el examen, ya que define **dónde** en la cadena se buscará el patrón. Todas devuelven un objeto de coincidencia (Match Object) si tienen éxito, o `None` (Falso) si fallan.

Función	Propósito	Regla de Coincidencia	Uso Ideal (Validación)
`re.match(patron, cadena)`	**Desde el principio.**	Solo comprueba si el patrón coincide al **inicio** de la cadena. Ignora el resto.	No es ideal para validar cadenas completas.
`re.search(patron, cadena)`	**En cualquier parte.**	Busca la **primera** coincidencia del patrón en **toda** la cadena, sin importar dónde empiece o termine.	Para encontrar un fragmento de texto.
`re.fullmatch(patron, cadena)`	**Cadena completa.**	Solo coincide si el patrón cubre la **totalidad** de la cadena, de principio a fin.	**El mejor para la validación estricta** (números de teléfono, matrículas, etc.).

4. Análisis de los Patrones de tu Código

Ejemplo 1: Validación de Número de Teléfono (España)

Patrón: `r"[6-8][0-9]{8}"`

Desglose:

1. `r""` : Cadena raw.
2. `[6-8]` : El primer dígito debe ser 6, 7 u 8 (en España, los móviles y fijos suelen empezar con estos).

3. `[0-9]{8}` : Debe ir seguido de exactamente 8 dígitos (del 0 al 9).
4. **Total:** El patrón requiere exactamente 9 dígitos, comenzando con 6, 7 u 8.

¿Por qué `fullmatch` es el mejor?

- `re.match()` validaría "651112345ABC" (porque coincide al principio).
- `re.search()` validaría "ABC651112345XYZ" (porque encuentra la subcadena).
- `re.fullmatch()` SOLO valida "651112345" (la cadena COMPLETA).

Ejemplo 2: Validación de Matrícula (Formato simple)

Patrón: `r"[0-9]{4}[\s|-]?[B-DF-HJL-NPR-TV-z]{3}"`

Desglose:

1. `[0-9]{4}` : Cuatro dígitos obligatorios.
2. `[\s|-]?` : **Separador opcional.** Puede ser un espacio (`\s`) o un guion (`-`). El `?` lo hace opcional (cero o una vez).
3. `[B-DF-HJL-NPR-TV-z]{3}` : Tres letras **obligatorias**, usando rangos que excluyen vocales y ciertas letras.

Ejemplo 3: Validación de Caracteres Extendidos

Patrón: `r"[A-Za-zÑàèòùÀÈÌÒÙ]{4,8}"`

Desglose:

1. `[A-Za-z]` : Letras latinas básicas (mayúsculas y minúsculas).
2. `ÑàèòùÀÈÌÒÙ` : Se añaden explícitamente caracteres específicos (e.g., Ñ y vocales acentuadas) al conjunto.
3. `{4,8}` : La cadena completa debe tener una longitud de entre 4 y 8 caracteres.