Expresiones Regulares (RegEx)

Las expresiones regulares, también conocidas como regex (del inglés "regular expressions"), son patrones de caracteres que se utilizan para buscar y manipular texto en una variedad de aplicaciones informáticas, incluyendo editores de texto, procesadores de texto, bases de datos y lenguajes de programación.

En términos simples, una expresión regular es una secuencia de caracteres que define un patrón de búsqueda. Por ejemplo, una expresión regular podría ser utilizada para buscar todas las direcciones de correo electrónico en un archivo de texto o para validar una entrada de usuario como una dirección de correo electrónico o un número de teléfono.

Las expresiones regulares son extremadamente poderosas y flexibles, y pueden ser utilizadas en una amplia variedad de aplicaciones.

En Python para trabajar con expresiones regulares utilizamos el módulo ‘re’. Es un módulo integrado que proporciona soporte para trabajar con expresiones regulares.

Para utilizar el módulo *re* en Python, primero debemos importarlo. Podemos hacerlo mediante:

```
import re
```

Una vez importado el módulo, podemos utilizar sus funciones para trabajar con expresiones regulares. Algunas de las funciones más comunes son:

- re.search(): esta función se utiliza para buscar un patrón de expresión regular en una cadena de texto. Devuelve un objeto Match si encuentra una coincidencia, o None si no encuentra ninguna.

- re.match(): esta función es similar a re.search(), pero sólo busca coincidencias al principio de la cadena de texto. Si no hay coincidencias al principio de la cadena, devuelve None.

- re.findall(): esta función busca todas las coincidencias de un patrón de expresión regular en una cadena de texto y devuelve una lista con todas las coincidencias encontradas.

- re.sub(): esta función se utiliza para buscar un patrón de expresión regular en una cadena de texto y reemplazarlo por otra cadena. Devuelve una nueva cadena de texto con todas las coincidencias reemplazadas.

- re.split(): esta función se utiliza para dividir una cadena de texto en una lista de subcadenas, utilizando un patrón de expresión regular como delimitador.

GUIA EJERCICIOS BASICOS CON EXPRESIONES REGULARES

1. Crear una función llamada *es_mayuscula* que reciba un string y devuelva *True* en caso de que todas las letras sean mayúsculas o *False* en el caso contrario

2. Crear una función llamada *es_minuscula* que reciba un string y devuelva *True* en caso de que todas las letras sean mayúsculas o *False* en el caso contrario

3. Crear una función llamada *es_entero* que reciba un string y devuelva *True* en caso de que se trate de un número entero y *False* en caso contrario.

4. Crear una función llamada *es_solo_texto* que reciba un string y devuelva *True* en caso de que trate solo de caracteres alfabéticos y el espacio y *False* en el caso contrario

5. Crear una función llamada *es_decimal* que reciba dos strings: el primero representa la expresión a evaluar y el segundo el separador decimal (puede ser punto . o coma ,). Debe devolver *True* en caso que se trate de un número decimal y *False* en el caso contrario.

6. Crear una función llamada *es_alfanumerico* que devuelva *True* en caso de tratarse de solo letras y números y *False* en el caso contrario (es decir que contenga caracteres especiales)

7. Crear una función llamada *es_binario* que devuelva *True* en caso de un número binario válido (solo ceros y unos) o *False* en el caso contrario

8. Crear una función que reciba una lista de palabras y devuelva otra lista con las palabras que comienzan con la letra 'U'

9. Crear una función que reciba un texto y devuelva una lista con las palabras que contienen entre 3 y 6 caracteres de largo

10. Crear una función que reciba un texto y devuelva una lista de todas las palabras que terminan con '*ción*'. Ejemplo de texto: https://onlinegdb.com/swyremkF6

11. Crear una función que reciba un texto y devuelva una la lista de palabras encuentra que comienzan con una vocal

12. Crear una función llamada *obtener_oraciones* que reciba como parámetro un string y que devuelva una lista con las oraciones (delimitadores, '.', '!', '?'). Ejemplo de texto: https://onlinegdb.com/UMdr3hl3G

13. Crear una función que reciba un texto como parámetro y que corrija los errores de ortografía que no cumplan con la regla ortográfica que indica que antes de V se escribe N y que antes de B se escribe M.
    Por ejemplo, si el texto es: *"Mi comvicción me hace sentir que es el momento de imvertir tiempo para finalmente enbarcar en esta nueva aventura."* La función debería devolver:
    "*Mi convicción me hace sentir que es el momento de invertir tiempo para finalmente embarcar en esta nueva aventura."*

14. Crear la función *es_fecha* que reciba dos string, el primero representa la expresión a evaluar y el segundo el separador de la fecha (puede ser la barra / o el guión -) y que devuelva *True* en caso de tratarse de una fecha válida y *False* en el caso contrario. Los formatos admitidos son: 'dd/mm/aaaa' o 'dd-mm-aaaa'

15. Crear la función es_hora que reciba un string y que devuelva *True* en caso de tratarse de una hora válida y *False* en el caso contrario. El formato admitido es 'hh:mm:ss'

16. Crear la función *validar_codigo_postal* que reciba un string como parámetro y devuelva *True* en caso de tratarse de código postal válido o *False* en caso contrario.

17. Crear la función *validar_patente* que reciba un string como parámetro y devuelva *True* en caso de tratarse de un número de patente válido o *False* en caso contrario. Debe poder admitir estos dos tipos:

18. Crear una función que se llame *obtener_direcciones_email* que reciba un texto y devuelva una lista con todas las direcciones de email válidas que encuentren en el mismo. Acá dejamos un texto de ejemplo: https://onlinegdb.com/Ln0jhatKeI

19. Crear una función llamada *validar_password* que reciba un string y verifique si se trata de una contraseña cumple con los requisitos mínimos de seguridad:

    - Al menos 8 caracteres
    - Al menos una letra mayúscula y una letra minúscula
    - Un número
    - Un carácter especial

    En caso de no cumplir con alguno de los requisitos, imprimir un mensaje informando cual no se cumplio

20. Crear una función llamada *validar_ip* que reciba un string como parámetro y verifique si se trata de una dirección IP v4 válida, en caso de serlo retornar *True* de lo contrario retornar *False*.
    Se considera una dirección IP válida si tiene el formato xxx.xxx.xxx.xxx, donde xxx es un número entero entre 0 y 255.