Nota 0: realiza estos ejercicicisо solo si controlas perfectamente el temario de la UD1 en cualquier otro caso mejor atiende al profesor

Nota 1: salvo que se especifique lo contrario cada uno de los ejercicios se resuelve programando una función. No te limites a programar las líneas de código que resuelven el problema, encapsúlalas en una función.

Nota 2: es buena práctica que las funciones tengan nombres representativos de la función que realizan, sin embargo para facilitar la corrección de los ejercicios te pido que les pongas el nombre 'ejX()' dónde X es el número del ejercicio.

Nota 3: algunos ejercicios serán marcados con (¡!) que expresará que son de mayor dificultad. Enhorabuena si consigues resolverlos, será señal de que estás dominando la programación con Python.

Bloque 1. Flujo de control y listas.

Ejercicio 1. Crea una función que obtenga el máximo de una lista de números. No está permitido el uso de la función max(). (int[]: nums) => (int: máximo)

Ejercicio 2. Crea una función que obtenga la sumatoria de una lista de números. (int[]: nums) => (int: sumatoria)

Ejercicio 3. Crea una función que dada una distancia en millas calcule su correspondiente en kms. Los parámetros (int: distancia_millas) => (int: distancia_kms)

Ejercicio 4. Crea una función que determine si una cadena de texto es un palíndromo. (string: palabra) => (boolean: es_palindromo)

Ejercicio 5. Crea una función que encuentra la palabra de mayor longitud en una cadena texto y la longitud de la misma. (string: texto) => (string: palabra, int: longitud)

Ejercicio 6. Crea una función que cuenta cuantas veces aparece una letra en un texto. (string: texto, string: letra) => (int: n_veces)

Ejercicio 7. Crea una función que cuenta cuantas veces aparece una subcadena en un texto. (string: texto, string: subcadena) => (int: n_veces)

Ejercicio 8 (¡!). Crea una función que cuenta cuantas veces aparece una subcadena en un texto y el índice de inicio y fin de cada una de las instancias de la subcadena dentro del texto. (string: texto, string: subcadena) => (int: n_veces, [[int: inicio, int: fin]])

Ejercicio 9. Crea una función que determine que palabras de un texto tiene una longitud mayor a una dada. (string: texto, int: longitud) => (string[]: palabras)

Ejercicio 10. Crea una función que determina si una letra es una vocal. (string: letras) => (boolean: es_vocal)

Ejercicio 11. Crea una función que pone en mayúscula la primera letra de cada palabra de un texto. (string: texto) => (string: texto)

Ejercicio 12. Crea una función que cambie todas las ocurrencias de una letra dentro de un texto por otra letra. (string: texto, string: letra1, string: letra2) => (string: texto)

Ejercicio 13 (¡!). Crea una función que encuentre todas las palabras de un texto que empiecen por una determinada subcadena y el porcentaje, con respecto al total del texto, que representan dichas palabras. (string: texto, string: subcadena) => (string[]: palabras, int: porcentaje)

Ejercicio 14 (¡!). Crea una función que encuentre todas las palabras de un texto que empiecen por una subcadena y que NO termine por otra subcadena. (string: texto, string: subcadena1, string: subcadena2) => (string[]: palabras)

Ejercicio 15. Crea una función que dada la longitud de los tres lados de un triángulo determina el tipo de triangulo, i.e.: equilátero, isósceles o escaleno, que es. (int[]: longitudes) => (string: tipo)

Ejercicio 16. Crea una función que cuenta la cantidad de números pares de una lista de números. (int[]: lista_nums) => (int: n_pares)

Ejercicio 17. Crea una función que calcula el área de triángulo o cuadrado dados su base y altura o el de una circunferencia dado su radio. (string: figura, int[]: longitudes) => (int: área)

Ejercicio 18. Crea una función que calcula el número de múltiplos de un número en un intervalo [a, b). (int: multiplicador, int: inicio, int: fin) => (int[]: múltiplos)

Ejercicio 19(¡!). Crea una función que sume los dígitos de un número. Ejemplo: sumaDigitos(245) = 2 + 4 + 5 = 11. (int: n) => (int: suma)

Ejercicio 20(¡!). Crea una función que calcule el máximo común divisor de dos números naturales. (int: a, int: b) => (int: mcd)

Ejercicio 21. Crea una función que calcule si un número dado es primo o no. (int:n) => (boolean: es_primo)

Ejercicio 22. Crea una función que encuentra todos los números primos en un intervalo [a, b). (int: a, int: b) => (int[]: primos)

Ejercicio 23. Crea una función que calcule el término n-ésimo de la sucesión de Finbonacci. (int: i) => (int: n)

Ejercicio 24(¡!). Crea una función que determine si dos números son primos relativos. (int: a, int: b) => (int[]: primos)

Ejercicio 25. Crea una función que determine si un número dado es capicúa. (int: n) => (boolean: es_capicua)

Ejercicio 26. Crea una función que calcule todos los números capicúas de un intervalo [a, b). (int: a, int: b) => (int[]: nums)

Ejercicio 27. Crea una función que imprima el siguiente 'mosaico':

1
22
333
4444
55555
666666

(int:n) => void. Siendo n el número de líneas, en el ejemplo anterior n = 6.

Ejercicio 28(¡!). Crea una función que imprima un mosaico rombo de anchura variable. (int: anchura) => void. Ejemplo ej28(9) imprimiría:

Ejercicio 29. Crea una función que inserte un texto dentro de una etiqueta html. (string:text, string:etiqueta) => (string:código_html)

Por ejemplo: ej29("p", "Soy un párrafo HTML") devolvería:

Soy un párrafo HTML

Ejercicio 30(¡!). Crea una función que filtre las palabras de un texto. Devolverá aquellas palabras de longitud superior a una longitud mínima, de longitud menor a una longitud máxima, que NO empiecen por un conjunto de letras y que contengan una subcadena. (string: texto, int: min_longitud, int: max_longitud, string[]: no_empiece, string: subcadena) => (string[]: palabras_filtradas). Por ejemplo ej30(texto, 5, 10, ['c', 'd'], "eto") devolvería todas las palabras de texto que tenga más de 5 letras, menos de 10, no empiecen por c ni d y que contengan la subcadena "eto".

Ejercicio 31. Crea una función que dado un texto devuelva una lista con las palabras que se repiten al menos dos veces en el texto. (string:texto) => (string[]: palabras)

Ejercicio 32. Crea una función que dada una lista devuelva esa misma lista sin elementos repetidos. ([]: lista) => ([]:lista_sin_repes)

Ejercicio 33. Crea una función que dada una lista de números devuelva aquellos que sean números primos. La función creada deberá utilizar la función filter. (int[]:nums) => ([]:primos)

Ejercicio 34. Crea una función que dado un texto devuelva una lista que contenga las palabras del texto ordenadas de mayor a menor. (string:texto) => (string[]:palabras_ordenadas)

Ejercicio 35. Crea una función que dado un texto devuelva una lista que contenga las palabras del texto ordenadas de menor a mayor. (string:texto) => (string[]:palabras_ordenadas)

Ejercicio 36. Crea una función que dada una lista de listas de números devuelva dicha lista ordenada, de mayor a menor, según el valor máximo de sus sub-listas. (int[][]: lista) => (int[][]: lista)

Ejercicio 37. Crea una función que dada una lista de números ordene dicha lista de tal forma que los número primos vayan antes que los no primos. (int[]: nums) => (int[]: nums)

Ejercicio 38. Crea una función que dada una lista de número devuelva otra lista con el cuadrado de cada número. (int[]: nums) => (int[]: cuadrados)

Ejercicio 39. Crea una función que dada una lista de números devuelva esa misma lista pero eliminando los números consecutivos repetidos. (int[]: nums) => (int[]: nums)