

Auxiliar 9: Introducción a Listas indexadas

Todos los problemas deben ser resueltos en *Python*, utilizando estrictamente la Receta de Diseño entregada a lo largo del curso. Use nombres apropiados para funciones y variables, y testee cada vez que sea posible.

1. Listas

En esta sección, trabajaremos con los meses del año. Para ello:

- Defina una lista con la cantidad de días de cada mes (sin considerar años bisiestos). Ej: [31, 28, ..., 31]
- Escriba la instrucción para mostrar la cantidad de días del mes de julio
- Escriba una función que reciba la lista y calcule el total de días del año
- Escriba una función que retorne el índice del mes con menos días
- Escriba una función que retorne una nueva lista, con los índices de los meses con más días: debe determinar primero el máximo M y luego extraer los meses que tienen M días
- Defina ahora una lista con los nombres de los meses Ej: ["enero", "febrero", ..., "diciembre"]
- Escriba una función que reciba ambas listas (días y nombres de meses) e imprima en pantalla el nombre del mes y sus días (ordenados cronológicamente)
- Escriba una función que reciba las dos listas y devuelva el nombre del mes más corto
- Escriba una función que reciba las dos listas y devuelva una lista con los nombres de los meses que tienen n días (n es un parámetro)

2. Listas (ahora con historia :D)

Suponga que en el país más poderoso del mundo ha sido electo presidente un tipo misógino, racista, xenófobo y peligrosamente cercano al fascismo. Sin embargo, usted decide revisar los resultados de las elecciones, para ver si es que existe alguna posibilidad de que los cálculos estén mal (aunque honestamente, tampoco le gusta la candidata que quedó segunda, debido a sus ya conocidas políticas intervencionistas). Además, este país tiene un sistema de elección bastante particular: no gana el candidato con más votos, sino que usan un sistema de delegados: cada ciudad (estado más bien, pero detalles) tiene una cantidad de "delegados", y quien gana en esa ciudad se lleva todos los votos correspondientes (asuma que sólo hay dos candidatos).

Para llevar a cabo su noble tarea, usted necesita:

- Escribir una función `delegadosNecesarios(nDelegados): list ->int` que reciba una lista con la cantidad de delegados de cada ciudad, y luego calcule la cantidad que votos de delegados que necesita obtener un candidato para ganar (tiene que ser la mitad más uno)
- Escribir una función `ganadorVotoPopular(votosCandidato1, votosCandidato2): list, list ->int`, que determine qué candidato (1 o 2) ganó el voto popular, es decir, qué candidato obtuvo más votos totales (asuma que no hay empate)

- Escribir una función `ganador(delegados, votosC1, votosC2): list, list, list ->int`, que determine qué candidato (1 o 2) ganó la elección, de acuerdo a la regla de delegados explicada previamente (nuevamente, asuma que no hay empate)
- Ahora usted recibe una lista con los nombres de las ciudades. Escriba una función que retorne una lista con los nombres de la ciudades con más delegados.

Usted no quedó conforme con el resultado (confirmó, lamentablemente, que el candidato detestado ganó), así que ahora empieza a imaginar qué habría pasado si algunos estados no existieran. Por ello, usted se pone en los siguientes casos:

- Se pregunta "¿qué habría pasado si el estado con más delegados donde no existiera?". Para ello, escriba una función `sinEstadoConMasDelegados(delegados, v1, v2): list, list, list ->boolean`, que entregue `True` si el resultado cambia, `False` en caso contrario.
- Se pregunta "¿qué habría pasado si el último estado no existiera?", "¿qué habría pasado si los últimos dos estados no existieran?", etc, hasta encontrar una situación en que gana la otra candidata. Así, debe escribir una función `sacarEstados(delegados, votos1, votos2): list, list, list ->int`, que retorne cuántos estados de "al final" debería sacar para cambiar el resultado de la elección (en caso de que no se pueda con ninguno n , retorne -1)