





# CP03.5-Listos para las listas (Pro version)

Curso 2025-2026

<b>Using for Loops</b>	
<b>Using while loops</b>	
<b>Using recursive if statements</b>	
<b>Just copying and pasting your code for each iteration</b>	

*Puede no parecerlo pero esta CP es más fácil que la anterior*

## 1. Filtrado Pro

Dado un entero  $k$  y una lista con  $n$  enteros, crea una nueva lista solamente con los valores que son menores que ese  $k$ . Para este ejercicio puede ser útil la función `append` de las listas.

## 2. Filtrado Pro Plus

Dado una lista  $A$  de  $n$  nombres y una lista  $B$  de  $n$  nombres, crea una nueva lista que guarde el índice donde se puede encontrar cada nombre de la lista  $B$  en la lista  $A$ . Para este ejercicio puede ser útil la función `index`. En caso de que el nombre no se encuentre se guarda el valor  $-1$ .

## 3. LShift Pro

Dada una lista de  $n$  elementos crea una nueva lista que consista en todos los elementos de la anterior desplazados en  $k$  lugar hacia la izquierda y con el primer valor ahora como último.

**Ejemplo**

```
A = [20,1,56,7,900,10,31,10]

# Here goes the magic

print(new_list) # new_list se refiere a la nueva lista creada, use cualquier nombre

# Se debe mostrar [1,56,7,900,10,31,10,20]
```

#### 4. Planificación de apagones

El día está dividido en  $n$  horarios diurnos principales y  $m$  horarios nocturnos. En cada horario hay una demanda de kW que se da en una lista de tamaño  $n + m$ . Dado un  $w$  que representa la disponibilidad máxima de kW y un  $b$  que representa la cantidad de bloques construya una lista donde diga la cantidad de bloques que deben apagarse para que la demanda no sobrepase la capacidad de generación. Para esto suponga que todos los bloques tienen la misma demanda.

#### 5. Lista Normalizada

Se tiene una lista  $A$  con  $n$  enteros, una lista  $B$  con  $g$  enteros y un entero  $k$  de forma que la suma de todos los valores de  $B$  es igual a  $n$  y  $k$  es mayor o igual que el mayor elemento de  $B$ . Los valores de  $B$  delimitan cuantos elementos tienen cada uno de los  $g$  grupos en los que se puede separar  $A$ . **Cambia la lista en  $A$**  para que cada uno de los grupos divididos en las secciones delimitadas por  $B$  tengan tamaño  $k$  y la cantidad de elementos en  $A$  sea igual a  $g * k$ . Rellena con 0s los valores adicionales.

**Ejemplo:**

```
A = [1,3,3,3,2,2] # 6 valores con referencia al tamaño del grupo que pertenecen
B = [1,3,2]       # Cada grupo de 1 solo elemento viene seguido por un grupo de 2
k = 3             # Quiero que cada grupo tenga k elementos

# Here goes de magic

print(A)

# Se debe mostrar: [1,0,0,3,3,3,2,2,0]
# El primer grupo (de un solo elemento) se rellenó con 2 ceros
# El segundo grupo no hizo falta ya que habían ya 3 elementos
# El cuarto grupo se rellenó con un único cero
```

#### 6. Torneo Descompuesto

Se realizó un torneo entre  $2^k$  jugadores con cuartos de final, semifinal y final. El torneo listó los jugadores en la tupla `players` de tal forma que al primer jugador le tocaba contra el segundo en los cuartos de final, al tercero contra el cuarto y así sucesivamente. Por ejemplo, si `players=["Bolzano", "Gauss", "Cauchy", "Markov", "Fourier", "Dijkstra", "Bool", "Yosvany"]` entonces sabemos que los primeros juegos se realizaron de esta forma:

- Bolzano vs. Gauss
- Cauchy vs Markov
- Fourier vs Dijkstra
- Boole vs Yosvany

Posterior a esto el ganador del primer encuentro jugó contra el ganador del segundo encuentro y de forma análoga se dió fin al torneo. El problema es que los resultados de los juegos se guardaron erróneamente en la tupla `games` únicamente con el nombre de la persona que perdió el juego y es necesario saber quien ganó el torneo.

**Ejemplo:**

```
# Con los players definidos arriba

games = [
    # Cuartos de Final
    "Gauss", # Ganó Bolzano
    "Cauchy", # Ganó Markov
    "Fourier", # Ganó Dijkstra
    "Boole", # Ganó Yosvany
    # Semifinal
    "Cauchy", # Ganó Markov
    "Dijkstra", # Ganó Yosvany
    # Final
    "Markov", # Ganó Yosvany
]

# Here goes the magic

print(winner)
# Debe mostrar: Yosvany
```

## 7. Rumpelstiltskin

Se quiere reconstruir un nombre desconocido del cual solo se tienen en una lista  $n$  instrucciones para componerlo y un nombre inicial. Las instrucciones son una tupla con alguno de los siguientes formatos:

- `("put", text, i, r)`: Una instrucción de esta forma indica que se debe insertar el string `text` en la posición `i` (entera positiva) mientras el nombre original se encuentra en reversa si `r` es `True` o sino en el sentido normal.
- `("erase", i, j, r)`: Esta indica que se debe eliminar desde el índice `i` hasta el `j` (sin incluir `j`) y, al igual que “put”, si `r` es `True` se considera la palabra en sentido inverso sino, en el sentido normal.

**Ejemplo:**

```
name = "r"

instructions = [
    ("put", "umpel", 1, False), # Inserta umpel al final
    # Va quedando "rumpel"
```

```

    ("put","nikw",0,True), # Inserta el texto al inicio del inverso("lepmur")
    # Va quedando "rumpelwkin" porque se devuelve el sentido
    ("erase", 6, 7, False), # Elimina la "w" que sobra
    # Va quedando "rumpelkin"
    ("put", "stiltskin", 6, False),
    # Va quedando "rumpelstiltskinkin"
    ("erase", 3, 6, True),
    # Elimina los caracteres que están en los índices 3, 4 y 5 del nombre invertido
]

# Here goes the magic

print(name)

# Debe mostrar: "rumpelstiltskin"

```