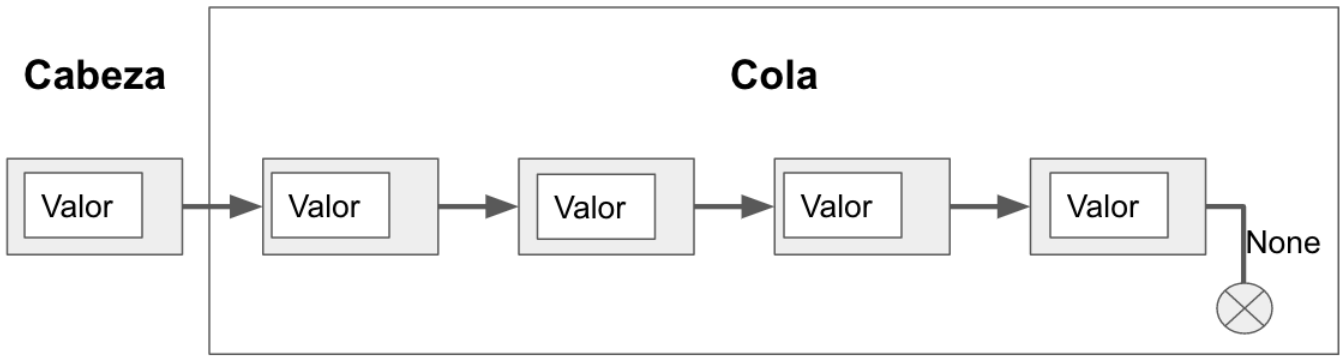
Mini-repaso

Listas



Ej:

```
# contiene : str lista(str) -> bool
# Determina si lista contiene el string s
# ejemplo contiene('auto', crearLista('auto', listaVacia)) retorna True
def contiene(s, unaLista):
    if vacia(unaLista):
        return False
    else:
        if cabeza(unaLista) == s:
            return True
        else:
            return contiene(s, cola(unaLista))
```

Abstracciones funcionales

1. Filtro: Filtrar algunos elementos de la lista y dejarlos en una nueva lista

Nota: en el modulo `abstraccion.py` hay dos versiones de filtro: 1) filtro con 2 parametros `filtro(operador, unaLista)`, y 2) filtro con 3 parametros `filtro(operador, unaLista,n)`. Ud. debe seleccionar cuál quiere usar (borrando la definición de filtro que no usará del modulo).

```
# filtro: lista(any) (any any->bool) any -> lista(any)
# Devuelve lista con valores de L para los que comparacion con x es True
# ej:filtro(lista(5,lista(4,None)),menorQue,5)->lista(4,None)
```

```
def filtro(operador,unaLista,n):
...

>>> filtro(menorQue, L, 3)
```

2. Mapa: Hacer algo a cada elemento de la lista y ponerlos en una nueva lista

```
# mapa : (X -> Y) lista(X) -> lista(Y)
# devuelve lista con funcion aplicada a todos sus elementos

def mapa(f, unaLista):

...

>>> mapa(aString,L) # crea una nueva lista con strings de fracciones
```

3. Reducir (Fold): Convertir los elementos de una lista en un solo valor

```
# fold: (X X -> X) X lista(X) -> X
# procesa la lista con la funcion f y devuelve un unico valor
# el valor init se usa como valor inicial para procesar el primer
# valor de la lista y como acumulador para los resultados
# parciales
# pre-condicion: la lista debe tener al menos un valor

def fold(f, init, unaLista):
...

>>> fold(multiplicar, 1, unaLista)
```