INFO132 oppgavesett Lister – løsningsforslag
(Noen oppgaver kan være basert på AI-generert tekst, og kvalitetssikret av en lærer)

Enkle oppgaver

1. Lag en funksjon som tar inn en liste av tall, og fjerner alle elementer som er mindre enn et spesifisert tall. Den nye listen skal returneres

```
>>> fjern_mindre_tall([2, 4, 6, 8, 10], 6)
[6, 8, 10]
```

2. Lag en funksjon `multipliserFaktor(`*tall-liste*`,`*faktor*`)` som returnerer en ny liste der alle tallene i tall-listen er multiplisert med faktoren

```
>>> multipliserFaktor([1, 2, 3, 4, 5], 2)
[2, 4, 6, 8, 10]
```

3. Tenk deg at du representerer en kø, vha en liste. Lag en funksjoner som administrerer køen slik at nestemann er den som står først i køen, mens nye legges til bakerst i køen.
Hint: `.append, .pop`

```
>>> Kø=[]   # tom kø
>>> ny(Kø, 'Kari') # Kari stiller seg i køen
>>> ny(Kø, 'Liv')
>>> ny(Kø, 'Ole')
>>> Kø
['Kari', 'Liv', 'Ole']
>>> neste(Kø)
'Kari'
>>> Kø
['Liv', 'Ole']
>>>
```

4. Lag tilsvarende funksjoner som i 3) men for å administrere en *stabel*. Dvs at nye elementer legges først, og tas ut før tidligere elementer

```
>>> Stabel=[] #tom stabel
>>> ny(Stabel, 1)  #setter inn tallet 1 i stabelen
>>> ny(Stabel, 2)
>>> ny(Stabel, 3)
>>> Stabel
[3, 2, 1]
>>> neste(Stabel)
3
>>> Stabel
[2, 1]
>>> neste(Stabel)
2
>>> Stabel
[1]
>>> ny(Stabel, 5)
>>> Stabel
[5, 1]
>>>
```

Middels vanskelige oppgaver

5. Liste-metoden *L*.`remove(`*e*`)` fjerner kun 1 forekomst av *e* fra listen *L*.
Lag en funksjon `fjernAlle(`*Liste, e*`)` som fjerner alle forekomster av *e* slik at
a) funksjonen endrer listen permanent.
b) funksjonen returnerer en resultat-liste mens originalen forblir uendret.

```
a)
>>> L=[1,2,3,4,5,6,5,4,3,2,1]
>>> fjernAlle(L,4)
[1, 2, 3, 5, 6, 5, 3, 2, 1]
>>> L
[1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1]

b)
>>> fjernAlle(L,4)
>>> L
[1, 2, 3, 5, 6, 5, 3, 2, 1]
```

6. Lag en funksjon som tar inn to lister som argumenter. Den skal returnere en ny liste som inneholder elementene som er felles for begge listene, uten duplikater.

```
>>> felles_elementer([1, 2, 3, 4, 5], [4, 5, 6, 7, 8])
[4, 5]
```

7.

```
>>> lst=[[1],[2],[3]]
>>> kopi=lst[:]
>>> kopi.append('Hei')
>>> kopi
[[1], [2], [3], 'Hei']
>>> lst
[[1], [2], [3]]
>>> kopi[0][0]=99
>>> kopi
[[99], [2], [3], 'Hei']
>>> lst
[[99], [2], [3]]
>>>
```

Forklar hvorfor `kopi.append('Hei')` ikke har bi-virkning på `lst`, mens `kopi[0][0]=99` har bi-virkning

Vanskelige oppgaver

8. L_1 er en *sub-liste* av L_2 dersom L_1 er et segment i L_2. Dvs at elementene fra L_1 forekommer i L_2, uavbrutt og i samme rekkefølge.

a) Lag en funksjon som sjekker om en liste er subliste av en annen liste

b) Lag en funksjon som finner alle posisjonene til en subliste i en større liste

Eksempler

```
>>> subliste([3,4,5], [1,2,3,4,5,6,7])
True
>>> subliste([3,4,5], [7,6,5,4,3,2,1])
False
>>>
>>> sublistePos([4,5], [1,2,3,4,5,6,5,4,3,4,5,4,5,6,3,4,5,6,7])
[3, 9, 11, 15]
>>>
```

9. Lag funksjoner for å håndtere 2-dimensjonale tabeller med tall, representert som lister

Tabell:

1	2	3
4	5	6
7	8	9
10	11	12

Liste-representasjon: [[1,2,3], [4,5,6], [7,8,9], [10,11,12]]

a) Funksjonen `lagTabell`(antallKolonner, antallRader) skal opprette en tabell med de gitte dimensjonene. Initielt kan alle celler inneholde verdien 0.

b) Funksjonen `skrivTabell`(tabell) skriver ut tabellen på en passende måte

c) Funksjonen `fyllTabell`(tabell, verdiListe) skal legge inn verdiene i liste fra venstre, mot høyre og ovenfra og ned. For tabellen over ville vi brukt verdi-listen `[1,2,3,4,5,6,7,8,9,10,11,12]`

d) Funksjonen `hentVerdi`(tabell, rad, kolonne) skal returnere verdien som ligger i det gitte cellen.

e) Funksjonen `settVerdi`(tabell, rad, kolonne, verdi) skal legge inn den gitte verdien i cellen.

Eksempler:

```
>>> t=lagTabell(3,4)
>>> fyllTabell(t, [11,12,13,21,22,23,31,32,33,41,42,43])
>>> skrivTabell(t)
 11 12 13
 21 22 23
 31 32 33
 41 42 43
>>> hentVerdi(t,2,3)
32
>>> settVerdi(t,1,4,0)
>>> skrivTabell(t)
 11 12 13
 21 22 23
 31 32 33
  0 42 43
```

Ekstra oppgaver

Oppgave 9. fortsatt

f) Funksjonen `leggTilRad`(tabell, verdi-liste, n) skal legge til en ny rad i posisjon n, med de gitte verdiene. Antallet rader skal øke med 1, dvs at den gamle raden med posisjon n skal nå ha posisjon n+1, og så videre

g) Funksjonen `leggTilKolonne`(tabell, verdi-liste, n) tilsvarer `leggTilRad`

Eksempler

```
>>> skrivTabell(t)
 11 12 13
 21 22 23
 31 32 33
  0 42 43
>>> leggTilRad(t, [1,1,1], 3)
>>> skrivTabell(t)
 11 12 13
 21 22 23
  1  1  1
 31 32 33
  0 42 43
>>> leggTilKolonne(t, [9,9,9,9,9], 3)
>>> skrivTabell(t)
 11 12  9 13
 21 22  9 23
  1  1  9  1
 31 32  9 33
  0 42  9 43
```

løsning:

```
def leggTilRad(tabell, verdier, posisjon=None):
   antallRader=len(tabell)
   antallKolonner=len(tabell[0])
   if antallKolonner!=len(verdier):
      print('Feil antall verdier i raden')
   elif posisjon==None: tabell.append(verdier)
   elif posisjon<1 or posisjon>antallRader+1:
      print('Ugyldig posisjon')
   else: tabell.insert(posisjon-1,verdier)


def leggTilKolonne(tabell, verdier, posisjon=None):
   antallRader=len(tabell)
   antallKolonner=len(tabell[0])
   if posisjon==None: posisjon=antallKolonner+1  #ny kolonne bakerst
   if posisjon<1 or posisjon>antallKolonner+1:
      print('Ugyldig posisjon')
   elif len(verdier)!=antallRader:
      print('Feil antall verdier i kolonnen')
   else:
      i=0
      while i<antallRader:
         tabell[i].insert(posisjon-1,verdier[i])
         i=i+1
```

10. (vanskelig)
Lag en funksjon som tar inn en liste av strenger og returnerer en ny liste der strengene er sortert alfabetisk, *uten å bruke en innebygget Python-funksjon (som sort() eller sorted()).*

```
>>> strenger = ["Python", "Java", "C++", "Ruby"]
>>> sorter_strenger(strenger)
["C++", "Java", "Python", "Ruby"]
```

løsning

```
def sorter_strenger(liste):
    sortert_liste = []
    lengde=0
    for s in liste:
        i=0
        while i<lengde:
            if s<=sortert_liste[i]: break
            i=i+1
        sortert_liste.insert(i,s)
        lengde=lengde+1
    return sortert_liste
```