DATA110 Laboppgaver 6

1. Gjensyn med dørvakten fra lab 5.
Dørvakten til et selskapslokale skal holde styr på hvor mange gjester som er sluppet inn og sikre at antallet ikke overstiger kapasiteten i lokalet. Lag en klasse `kapasitetVakt` for å holde telling med antall personer i et lokale, med metoder for å angi antall personer som kommer og går, samt for å angi ledig kapasitet. Vakten skal potensielt vokte flere lokaler, av varierende størrelse, samtidig.
Eksempel:

```
>>> lokale1=kapasitetVakt(10)  #plass til 10 personer i lokale 1
>>> lokale2=kapasitetVakt(15)  #og 15 i lokale 2
>>> lokale1.kommer(4)          #4 gjester kommer
>>> lokale2.kommer(5)
>>> lokale2.kommer(2)
>>> lokale1.ledig()            #det er plass til 6 nye gjester
6
>>> lokale2.ledig()
8
>>> lokale1.kommer(9)
For mange! Slipp inn  6
>>> lokale1.ledig()
0
>>> lokale1.går(5)             #5 gjester går
>>> lokale1.ledig()
5
```

2.
Fibonacci-serien er en tallrekke der de to første tallen i serien er 1'ere og deretter er det neste tallet summen av de to forrige: 1, 1, 2, 3, 5, 8, 13, 21, … osv.
En *fibonacci-generator* er et objekt `x` som holder et tall i fibonacci-serien og med en metode `x.neste()` som oppdaterer til det neste tallet i serien (og skriver det ut og/eller returnerer det)

Lag en klasse fibonacci for slike generator-objekter. Eksempel:

```
>>> g=fibonacci()
>>> g.neste()
1
>>> g.neste()
1
>>> g.neste()
2
>>> g.neste()
3
>>> g.neste()
5
>>> g.neste()
8
>>> g.neste()
```

3.

a) Ta utgangspunkt i klassen for kontoer som vi laget på forelesning
(Filen demo2.py i forelesningsnotatene)

```
class konto:
    antallKontoer=0  # for automatisk genererte kontonumre
    kontonummer=None
    innehaver=None
    saldo=None
    rentesats=None

    def __init__(self, innehaver,saldo=0, rentesats=1.5):
        konto.antallKontoer=konto.antallKontoer+1
        self.kontonummer=konto.antallKontoer
        self.innehaver=innehaver
        self.saldo=saldo
        self.rentesats=rentesats

    def __del__(self):  #varsler hvis kontoen blir slettet
        print('VARSEL! konto nr', self.kontonummer, 'blir slettet')
        self.skriv()

    def skriv(self):
        print(self.kontonummer,'('+str(self.rentesats)+'):',\
              self.saldo, ', Innehaver: '+self.innehaver)

    def innskudd(self,beløp):
        self.saldo=self.saldo+beløp

    def uttak(self,beløp):  #returnerer False dersom det ikke er dekning.
        if beløp>self.saldo:
            print('Ikke dekning')
            return False
        else:
            self.saldo=self.saldo-beløp
            return True
```

Legg til metoder for
- å endre rentesatsen til en konto
- renteoppgjør, dvs legg renteutbytte til saldoen
- overføring til en annen konto

b) Lag en klasser for banker, med bankens navn, standard innskuddsrente og liste over kontoer.
Klassen skal ha metoder for å
- endre standardrenten
- endre renten for individuelle kontoer
- opprette nye kontoer, men innehaver og start-saldo og standard innskuddsrente
- fjerne kontoer
- renteoppgjør for alle kontoer
- overføring mellom kontoer

Lag test-data med flere banker og kunder

4.
Lag et program som først skriver ut teksten 'fra NÅ!' og deretter venter et tilfeldig antall sekunder før det skriver ut 'til NÅ!' Så skal brukeren gjette på hvor mange sekunder som gikk. Skriv også ut hvor mange sekunder brukeren brukte for å gjette. Eksempel:

```
Gjett hvor lang tid det tar fra

       NÅ


... TIL...


      NÅ!

Gjett (sekunder): 8
Feil! Riktig svar er 6 sekunder
Du brukte 3.3 sekunder på å svare
```

Hint: metodene time og sleep i modulen time https://docs.python.org/3/library/time.html

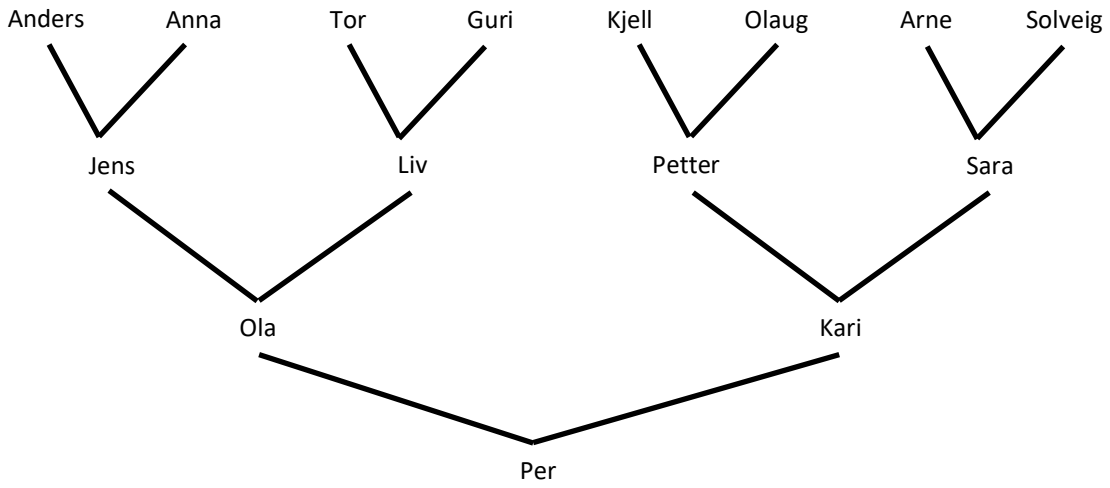
5.
Lag en klasse for å holde styr på køer. For eksempel en venteliste med personer

```
>>> venteliste=kø()
>>> venteliste.settInn('Per')   #setter inn personer i køen
>>> venteliste.settInn('Kari')
>>> venteliste.settInn('Liv')
>>> venteliste.skriv()
Per
Kari
Liv
>>> neste=venteliste.taUt()    #tar ut førstemann i køen
>>> neste.skriv()
Per
>>> venteliste.skriv()
Kari
Liv
>>> venteliste.taUt().skriv()  #tar ut førstemann i køen
Kari
>>> venteliste.taUt().skriv()  #tar ut førstemann i køen
Liv
>>> venteliste.skriv()
>>> venteliste.taUt()
Køen er tom
>>>
```

6.

Lag et program som kan skrive ut en kjede av forgjengere til en gitt person. (foreldre, besteforeldre, oldeforeldre, osv i vilkårlig mange generasjoner). Kjeden skal spesifiseres ved en sekvens som består av F'er og M'er, dvs en tekst som dette: 'FMMF'. I dette tilfellet skal programmet skrive ut personens far (F), så hans mor (M) etterfulgt av hennes mor (M) og til slutt hennes far (F). Hint: lag en person-klasse som registrerer navn samt far og mor - som også er personer!. På den måten kan du lage en struktur av nestede objekter som representerer stamtreet bakover fra en person.

Eksempel: Per sitt stamtre




```
>>> forgjengere(per,'FM') #per er en variabel med objektet for Per
Ola
Liv
>>> forgjengere(per,'MMF')
Kari
Sara
Arne
>>> forgjengere(per,'MFFF')
Kari
Petter
Kjell
kjeden er for lang
>>> forgjengere(liv,'M')
Guri
>>>
```