INFO132 Laboppgave 6
(Noen oppgaver kan være basert på AI-generert tekst, og kvalitetssikret av en lærer)

Enkle oppgaver

1. Lag en liste med navnene på dine fem favorittfrukter. Skriv deretter kode for følgende operasjoner:
    1. Skriv ut hele listen med favorittfrukter.
    2. Legg til en ny frukt i listen.
    3. Fjern den andre frukten i listen.
    4. Finn lengden (antall elementer) i listen.
    5. Sjekk om 'banan' er i listen.

2. Definer en klasse kalt `Student` som skal ha attributter for studentens navn, alder og studieretning, samt metoden `__init__` for å opprette `Student`-objekter.

```
>>>ola=Student('Ola Normann',23,'Geografi')
>>>liv=Student('Liv Lagesen',20,'Økonomi')
>>>
>>>ola.studieretning
'Geografi'
```

Legg til følgende metoder i klassen:

- `skrivNavn` skriver en setning på formen `'Studenten heter` *navn*'.
- `hentAlder` returnerer studentens alder.
- `skrivStudieretning` skriver en setning på formen '*navn* `studerer` *studieretning*'

```
>>>liv.skrivNavn()
Studenten heter Liv Lagesen
>>>ola.hentAlder()
23
>>>liv.skrivStudieretning()
Liv Lagesen studerer Økonomi
```

3.
a) Definer en klasse for emner med attributter for emnekoden, tittelen og antallet studiepoeng. Klassen skal ha en `__init__`-metode og en metode kalt `skriv` for å skrive ut opplysningene.

```
>>>exphil=Emne('EXPHIL','Examen philosophicum',10)
>>>exphil.skriv()
EXPHIL (10) Examen philosophicum
```

Lag en liste med `Emne`-objekter for favoritt-emnene dine og bruk en `for`-løkke til å traversere listen og skrive ut emnene på skjermen

```
>>>for  ...fyll ut det som mangler ...
EXPHIL (10) Examen philosophicum
ECON100 (10) Innføring i samfunnsøkonomi
DATA110 (10) Innføring i programmering
DATA150 (5) Datasett
INFO135 (10) Viderekommande programmering
STAT110 (10) Grunnkurs i statistikk
```

b) Lag en funksjon `finnEmne(`*emnekode*`)` som søker i `Emne`-listen og returnerer `Emne`-objektet for emnekoden, eller teksten 'ukjent emne' dersom emnekoden ikke finnes i listen.

```
>>>finnEmne('DATA110').skriv()
DATA110 (10) Innføring i programmering
>>>finnEmne('DATA220')
'ukjent emne'
```

4.
a) Definer en klasse `Eksamen` for eksamensresultater.
Klassen skal attributtene:

- `emne` som står for en emnekode, for eksempel `'DATA110'`
- `semester` med semesteret eksamen ble tatt, f.eks. `'h22'` for høstsemesteret 2022
- `karakter` med eksamensresultatet, dvs en bokstavkarakter A-F

og metodene:

- `__init__` for å opprette objekter
- `skriv` som skriver ut opplysningene på en linje
- `nyKarakter` og `nyttSemester` som kan brukes til å oppdatere opplysningene dersom studenten tar eksamen på nytt.

```
>>>data110Eks=Eksamen('DATA110','v22','E')
>>>data110Eks.skriv()
DATA110 v22 E
>>>
>>>data110Eks.nyttSemester('v23')
>>>data110Eks.nyKarakter('A')
>>>data110Eks.skriv()
DATA110 v23 A
```

b) Opprett en global liste `mineEksamener` med eksamensresultatene for dine favoritt-emner

```
>>>for e in mineEksamener:e.skriv()
EXPHIL h22 C
ECON100 h22 B
DATA110 v23 A
STAT110 v23 C
DATA150 h23 B
INFO135 h23 C
```

c) Lag en funksjon `skrivKarakterer` som skriver ut alle eksamensresultatene dine. På hver linje skal det stå emnekode, karakteren du har fått og emnets tittel i parentes.

```
>>>skrivKarakterer()
EXPHIL C (Examen philosophicum)
ECON100 B (Innføring i samfunnsøkonomi)
DATA110 A (Innføring i programmering)
STAT110 C (Grunnkurs i statistikk)
DATA150 B (Datasett)
INFO135 C (Viderekommande programmering)
```

Middels vanskelige oppgaver

5. Vi fortsetter med eksamensresultatene fra oppgave 4.
Lag en funksjon `oppdater` som gitt en emnekode, semester og karakter vil legge til emnet i `mineEksamener`-listen dersom emnet ikke finnes der fra før, ellers oppdatere med nye verdier for semester og karakter

```
>>>for e in mineEksamener:e.skriv()
EXPHIL h22 C
ECON100 h22 B
DATA110 v23 A
STAT110 v23 C
DATA150 h23 B
INFO135 h23 C
>>>
>>>oppdater('EXPHIL','h23','B')
>>>for e in mineEksamener:e.skriv()
EXPHIL h23 B
ECON100 h22 B
DATA110 v23 A
STAT110 v23 C
DATA150 h23 B
INFO135 h23 C
>>>
>>>oppdater('DATA120','h23','A')
>>>for e in mineEksamener:e.skriv()
EXPHIL h23 B
ECON100 h22 B
DATA110 v23 A
STAT110 v23 C
DATA150 h23 B
INFO135 h23 C
DATA120 h23 A
```

6. Definer en klasse kalt `Person` som har attributter for personens navn og alder.

```
>>>kari=Person('Kari Normann',35)
>>>per=Person('Per Spellmann',67)
>>>kari.navn
'Kari Normann'
>>>per.alder
67
```

a) Definer også en klasse `Bil` som har følgende attributter:

- `merke`, for eksempel `'Volvo'`
- `modell`, for eksempel `'EX90'`
- `eier` som står for et `Person`-objekt

samt en `__init__`-metode for å opprette `Bil`–ojekter, gitt et bilmerke, et modellnavn og et `Person`-objekt for eieren

Lag også `skriv`–metode som skriver ut bilens merke, modell samt eierens navn

```
>>>bil1=Bil('Volvo','EX90',kari)
>>>bil2=Bil('Toyota','Yaris',per)
>>>bil1.skriv()
Volvo EX90, eier:Kari Normann
```

b) Nå skal du endre `__init__`-metoden i `Bil`-klassen ved å legge til eierens navn og alder som parametere, i stedet for et forhåndslaget `Person`-objekt.

```
>>>bil3=Bil('Kia','E-Soul','Ola Normann',29)
>>>bil3.skriv()
Kia E-Soul, eier:Ola Normann
>>>bil3.eier.alder
29
```

7. 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 hente ledig kapasitet i lokalet. Vakten skal potensielt vokte flere lokaler, av varierende størrelse, samtidig.

```
>>> 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()            #lokale 1 har 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
```

Vanskelige oppgaver

8.

a) Ta utgangspunkt i følgende klasse for kontoer (jfr forelesning)
Programkoden er vedlagt i filen kontoklasse.py

```
class konto:
    kontonummer=None
    innehaver=None
    saldo=None
    rentesats=None

    def __init__(self,kontoNr, innehaver,saldo=0, rentesats=1.5):
        self.kontonummer=kontoNr
        self.innehaver=innehaver
        self.saldo=saldo
        self.rentesats=rentesats

    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) Definer en klasse 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

9. 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()           #skriver personene i køen
Per
Kari
Liv
>>> neste=venteliste.taUt()     #tar ut førstemann i køen
>>> neste.skriv()               # - vedkommende returneres
Per                             # - og fjernes fra køen
>>> 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
>>>
```

10. Fibonacci-serien er en tallrekke der de to første tallene 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 returnerer det

Definer 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()
```

Hint: bruk attributter for neste tall, forrige tall og tallet før forrige tall i rekken

Ekstraoppgaver

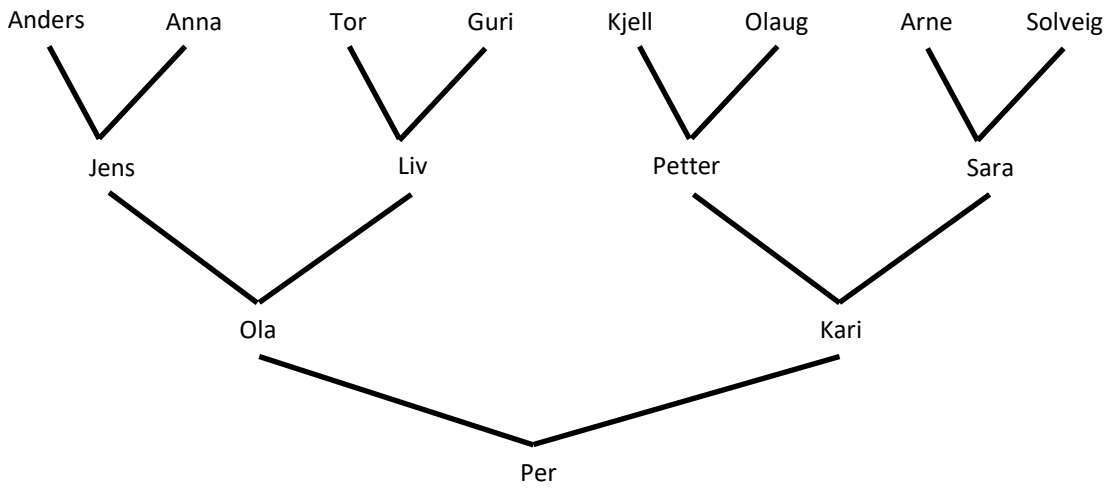
11. (Middels vanskelig)
Vi går tilbake til emner og eksamenskarakterer som i oppgave 3 og 4 men nå skal du droppe listen fra 3 a) der alle `Emne`-objektene var samlet. I stedet skal `Emne`-objektene erstatte emnekodene i `Eksamen`-objekter. Med andre ord: i `mineEksamener`-listen skal alle opplysningene om emner være nestet inn `Eksamens`-objektene.

Gjennomfør disse endringene og lag funksjonen `skrivKarakterer` på nytt.

```
>>>for e in mineEksamener:e.skriv()
EXPHIL h22 C
ECON100 h22 B
DATA110 v23 A
STAT110 v23 C
DATA150 h23 B
INFO135 h23 C
>>>
>>>skrivKarakterer()
EXPHIL C (Examen philosophicum)
ECON100 B (Innføring i samfunnsøkonomi)
DATA110 A (Innføring i programmering)
STAT110 C (Grunnkurs i statistikk)
DATA150 B (Datasett)
INFO135 C (Viderekommande programmering)
```

12 (Vanskelig)
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
>>>
```