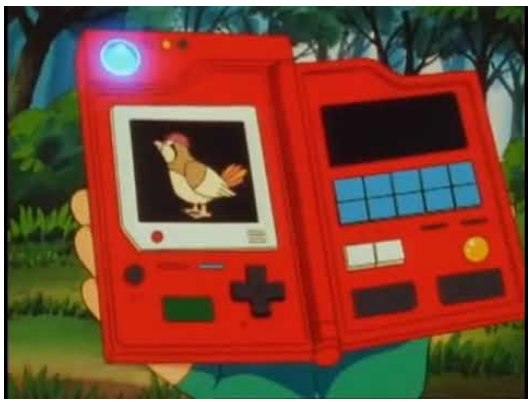
Oppgavesett dag 3
Avansert kurs ved Folkeuniversitetet

I denne seksjonen finner du oppgaver som hører til dag 3 av Folkeuniversitetet og Simula Learnings kurs Programmering i Python - Avansert. Tema for tredje dag er datastrukturer, fordypningsoppgaver og nettverk. Dersom du står fast er det bare å spørre. God koding!

Datastrukturer

Oppgave 1 *Pokemon*

Pokemon er en kjent TV-, film- og spillserie fra 90-tallet som fortsatt pågår idag. En viktig del av Pokemon-universet er Pokedexen - en oversikt over alle forskjellige Pokemon og informasjon om disse. I denne oppgaven har vi fått tilgang til en slik Pokedex.



a) Last ned filen *pokedex.json* og lag et nytt Python-program. Sørg for at

begge filene ligger i samme mappe. Ta en titt på json-filen ved å åpne den og se hvordan den ser ut.

Bruk følgende kodesnutt for å lese inn filen:

```
import json

with open("pokedex.json") as f:
    pokedex = json.load(f)
```

b) Bruk `print`-funksjonen til å undersøke `pokedex`. Hvordan er datastrukturen satt sammen? Hvordan brukes lister og ordlister? Forstå deg på hvordan data i `pokedex` er lagret.

c) Vi ønsker å finne hvor mange Pokemon har typen Flying". Bruk den innebygde funksjonen `filter` til å finne antallet. Du kan basere deg på koden under.

```
def find_flying_pokemon(pokemon):
    # Fyll inn.
    pass

flying_pokemon = list(filter(find_flying_pokemon,
   pokedex))
```

d) Baser deg på samme framgangsmåte og finn ut hvor mange pokemon som har et "Movesom heter Growl".

e) **BONUS:** Bruk lambda-funksjoner istedet for vanlige funksjoner i de to foregående oppgavene.

f) Vi ønsker å fjerne muligheten for Pokemon med typen Flying"og vil gjøre om alle tilfeller av dette til Ground". Bruk den innebygde funksjonen `map` for oppgaven. Du kan basere deg på koden under eller bruke en lambda-funksjon.

```
def ground_flying_pokemon(pokemon):
    # Fyll inn.
    pass

pokedex_with_no_flying = list(map(
   ground_flying_pokemon, pokedex))
```

g) Bruk samme framgangsmåte som i oppgave **c)** for å kontrollere at det ikke er pokemon med typen flying i pokedexen.

Løsning oppgave 1 *Pokemon*

a)

```
import json

with open("pokedex.json") as f:
    pokedex = json.load(f)
```

b)

```
print(type(pokedex))
print(type(pokedex[0]))
print(pokedex[0])
```

```
<class 'list'>
<class 'dict'>
{'Id': '001', 'Name': 'Bulbasaur', 'Type 1': '
   Grass', 'Type 2': 'Poison', 'Abilities': ['
   Overgrow', 'Chlorophyll'], 'Category': 'Seed
   Pok mon', 'Height (ft)': '2\'04"', 'Height (m)
   ': '0.7', 'Weight (lbs)': '15.2', 'Weight (kg)
   ': '6.9', 'Capture Rate': '45', 'Egg Steps':
   '5120', 'Exp Group': 'Medium Slow', 'Total':
   '318', 'HP': '45', 'Attack': '49', 'Defense':
   '49', 'Sp. Attack': '65', 'Sp. Defense': '65',
   'Speed': '45', 'Moves': {...}}
```

Vi ser fra `print` at pokedex er en liste over ordlister. Videre kan vi se at 'Abilities' er en liste og 'Moves' er en ordliste.

c)

```
def find_flying_pokemon(pokemon):
```

```
    return pokemon["Type 1"] == "Flying" or
        pokemon["Type 2"] == "Flying"

flying_pokemon = list(filter(find_flying_pokemon,
    pokedex))
print(len(flying_pokemon))
```

```
98
```

d)

```
def find_pokemon_with_growl(pokemon):
    return "Growl" in pokemon["Moves"]

pokemon_with_growl = list(filter(
    find_pokemon_with_growl, pokedex))
print(len(pokemon_with_growl))
```

```
139
```

e)

```
print(len(list(filter(lambda pokemon: pokemon["
    Type 1"] == "Flying" or pokemon["Type 2"] == "
    Flying", pokedex))))
print(len(list(filter(lambda pokemon: "Growl" in
    pokemon["Moves"], pokedex))))
```

```
98
139
```

f)

```
population_first = df.sort_values(by="Population",
    ascending=False)
population_first[["TotalCases", "TotalDeaths"]].
    head(1).plot(kind="bar")
```

g)

```python
def ground_flying_pokemon(pokemon):
    if pokemon['Type 1'] == 'Flying':
        pokemon['Type 1'] = 'Ground'
    elif pokemon['Type 2'] == 'Flying':
        pokemon['Type 2'] = 'Ground'

pokedex_with_no_flying = list(map(
   ground_flying_pokemon, pokedex))
```

h)

```python
flying_pokemon = list(filter(find_flying_pokemon,
   pokedex_with_no_flying))
print(len(flying_pokemon))
```

```
0
```

Nettverk

Oppgave 2 *Hente nettsider*

Vi skal bruke biblioteket `requests` for å hende innholdet av nettsider.

a) Hent innholdet fra nettsiden *http://www.webcode.me* med `requests.get("http://www.webcode.me")` og skriv ut innholdet til terminal.

b) Sjekk statuskoden til nettsiden *http://www.webcode.me* og nettsiden *http://www.webcode.me*.

c) Sjekk svaret du får når du henter

```
https://httpbin.org/get?name=Bjarne
```

d) Hent ut informasjon om universitetet i Oslo ved å bruke følgende API:

```
http://universities.hipolabs.com
http://universities.hipolabs.com/search?name=
   NAVNET
```

```
http://universities.hipolabs.com/search?name=
   NAVNET&country=LANDET
```

e) Bruk JSON sin funksjon loads til å gjøre det fra tekst til datastruktur.

f) Skriv ut alle nettsidene som kommer fra søket.

Løsning oppgave 2 *Hente nettsider*

a)

```
1  import requests as req
2
3  resp = req.get("http://www.webcode.me")
4
5  print(resp.text)
```

b)

```
1  resp = req.get("http://www.webcode.me")
2  print(resp.status_code)
3
4  resp = req.get("http://www.webcode.me/news")
5  print(resp.status_code)
```

c)

```
1  resp = req.get("https://httpbin.org/get?name=
      Bjarne")
2  print(resp.text)
```

```
{
  "args": {
    "name": "Bjarne"
  },
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate",
```

```
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.22.0",
    "X-Amzn-Trace-Id": "Root=1-635a2413-08
       a7da2279720f403e241dfe"
  },
  "origin": "81.191.235.97",
  "url": "https://httpbin.org/get?name=Bjarne"
}
```

d)

```
resp = req.get("http://universities.hipolabs.com/
   search?name=Oslo&country=norway")
print(resp.text)
```

e)

```
import json
resp = req.get("http://universities.hipolabs.com/
   search?name=Oslo&country=norway")
results = json.loads(resp.text)
```

f)

```
for university in results:
    print(f"{university['name']}: {university['
       web_pages']}")
```

Fordypningsoppgaver

Oppgave 3 *Sannsynlighet for å få en recessiv sykdom*

En person får en viss sykdom hvis man er homozygot for allel a. Dermed vil en syk person ha genotypen aa. Heterozygote med genotype Aa og homozygote med genotype AA er friske.

a) En familie har 7 barn, og begge foreldrene har genotype Aa. Lag et program velger et tilfeldig andel fra hver forelder og skriver ut den tenkte genotypen til de sju barna og om de er friske eller syke.

b) Tenk at et foreldrepar, begge med genotype Aa, har 1000 barn. Lag et program som genererer om hver av ungene er friske eller syke. Det eneste som skal printes ut som resultat er fraksjonen av alle barna som er friske og fraksjonen av alle barna som er syke.

Løsning oppgave 3 *Sannsynlighet for å få en recessiv sykdom*

a)

```
from matplotlib.pyplot import *
from random import *

forelder_1 = ["A", "a"]
forelder_2 = ["A", "a"]

antall_barn = range(5)

for barn in antall_barn:
    allel_1 = choice(forelder_1)
    allel_2 = choice(forelder_2)

    genotype = allel_1 + allel_2

    if "A" in genotype:
        fenotype = "frisk"
    else:
        fenotype = "syk"

    print(genotype, fenotype)
```

b)

```
forelder_1 = ["A", "a"]
forelder_2 = ["A", "a"]

antall_barn = range(1000)
```

```
frisk = 0
syk = 0

for barn in antall_barn:
    allel_1 = choice(forelder_1)
    allel_2 = choice(forelder_2)

    genotype = allel_1 + allel_2

    if "A" in genotype:
        frisk = frisk + 1
    else:
        syk = syk + 1



print("andel barn som er friske: ", frisk/len(
   antall_barn))
print("andel barn som er syke: ", syk/len(
   antall_barn))
```

Oppgave 4 *Vektvekst for selunger*

Grøndlandsseler dier ungene sine intensivt de 12 første levedagene. Melken er veldig rik på fett, og selungene legger på seg ca. 2.2 kilo om dagen denne perioden. Nyfødt veier selungene 11 kg.

a) Bruk en løkke til å finne vekten for selungen de 12 første levedagene og legge de i en liste. Lag også en liste med dagene, 1-12. Print listene.

b) Finn fra listen over vekt

- vekten til selen på dag 12
- vektene for dagene 5-7

c) Lag et plot med vekt mot dager. Gi aksene og plottet et navn.

Løsning oppgave 4 *Vektvekst for selunger*

a)

```
vekt = 11
dag = 1
vekt_liste = [11]
dag_liste = [1]
while dag < 12:
  ny_vekt = vekt + 2.2
  vekt_liste.append(ny_vekt)
  vekt = ny_vekt
  dag = dag + 1
  dag_liste.append(dag)
print(vekt_liste)
print(dag_liste)
```

b)

- Vekt på dag 12

```
    print(vekt_liste[-1])
```

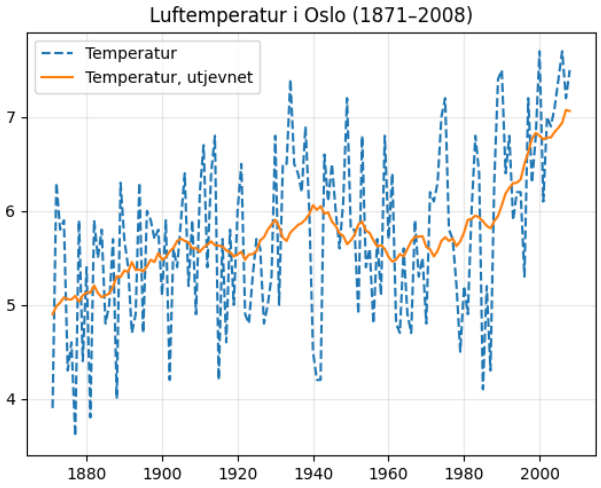
- Vektene for dagene 5-7

```
    print(vekt_liste[4:8])
```

c)

```
from matplotlib.pyplot import *

plot(dag_liste, vekt_liste)
xlabel("Antall dager etter fødsel")
ylabel("Vekt")
title("Vekt hos sel de 12 første dagene etter fø
   dsel")
show()
```

Oppgave 5 *Temperaturmålinger*



I denne oppgaven skal vi utforske hvordan man kan jevne ut data på en meningsfull måte. Vi skal se på et datasett over temperaturmålinger i Oslo,[a] plotte dette over tid og utforske ulike måter å jevne ut dataene på.

Vi tar utgangspunkt i en fungerende kode:

```
import numpy as np
import matplotlib.pyplot as plt

def glidende_gjennomsnitt(data, n):
    data_utjevnet = np.zeros_like(data)
    antall_elem = len(data)

    for i in range(n):
        indeks = n + i + 1
        data_utjevnet[i] = np.mean(data[:indeks])
        data_utjevnet[-i - 1] = np.mean(data[-indeks
            :])

    for i in range(n, antall_elem - n):
        data_utjevnet[i] = np.mean(data[i-n:i+n])

    return data_utjevnet

```

```
temperatur = np.array([
    3.9, 6.3, 5.8, 5.9, 4.3, 4.6, 3.6,
    5.9, 4.4, 5.4, 3.8, 5.9, 5.5, 5.8,
    4.8, 5.0, 5.7, 4.0, 6.3, 5.7, 5.4,
    4.7, 4.9, 6.3, 4.7, 6.0, 5.9, 5.7,
    5.8, 5.1, 5.9, 4.2, 5.6, 5.4, 5.9,
    6.4, 5.2, 5.9, 4.9, 6.2, 6.7, 5.4,
    6.4, 6.8, 4.2, 5.7, 4.6, 5.8, 5.0,
    6.0, 6.5, 4.9, 4.8, 5.3, 5.7, 5.6,
    4.8, 5.0, 5.3, 6.8, 5.0, 6.5, 6.5,
    7.4, 6.5, 6.4, 6.2, 6.9, 6.0, 4.5,
    4.2, 4.2, 6.6, 6.1, 6.5, 6.0, 5.6,
    6.1, 7.2, 5.8, 5.8, 4.9, 6.8, 5.4,
    5.6, 4.8, 5.7, 5.1, 6.8, 5.7, 6.4,
    4.8, 4.7, 5.6, 4.9, 4.7, 5.9, 5.3,
    5.5, 4.8, 6.2, 6.1, 6.3, 7.0, 7.2,
    5.8, 5.7, 5.2, 4.5, 5.2, 4.9, 6.0,
    6.8, 6.4, 4.1, 5.2, 4.3, 6.3, 7.4,
    7.5, 6.4, 6.8, 5.9, 6.2, 6.2, 5.3,
    7.2, 6.3, 6.8, 7.7, 6.1, 7.0, 6.9,
    7.1, 7.4, 7.7, 7.2, 7.5])

n = 10    # justerer gjennomsnittsintervallet
temperatur_utjevnet = glidende_gjennomsnitt(temperatur
    , n)
år = np.arange(1871, 2009, 1)

plt.grid(alpha=0.3)
plt.plot(år, temperatur, "--", label="Temperatur")
plt.plot(år, temperatur_utjevnet, label="Temperatur,
    utjevnet")
plt.legend()
plt.title("Luftemperatur i Oslo (1871 2008)")
plt.show()
```

a) Kopier/last ned koden over, og kjør den. Hva tror du koden over gjør, på et overordnet nivå? Prøv å forklare den til en annen person.

b) I koden over bruker vi variabelen `n` for å bestemme hvor stort intervall

vi skal ta et gjennomsnitt over, ved hjelp av noe som kalles for *glidende gjennomsnitt*. Prøv å justere `n` til andre verdier, og se hvordan det påvirker den utjevnede kurven.

c) Når vi bruker glidende gjennomsnitt tar vi ut et tidsintervall og regner ut gjennomsnittet over dette intervallet. Et annet alternativ hadde vært å bruke median over det samme intervallet. Opprett en funksjon `glidende_median` som gjør det samme som `glidende_gjennomsnitt`, men som bruker `np.median` istedenfor `np.mean` alle steder dette er aktuelt.

d) Utvid programmet slik at du både plotter de opprinnelige dataene, utjevnet temperatur ved gjennomsnitt samt utjevnet temperatur ved median.

e) Prøv deg igjen frem med ulike n-verdier. Er det noen forskjeller på glidende gjennomsnitt og glidende median? Hva tror du det kommer av?

f) Bruk plottet og de utjevnede dataene til å finne ut av hvor mye temperaturen har økt i løpet av de siste 30, 60 og 100 årene. Hvilken `n`-verdi brukte du her? Hva har valget av `n` å si i forhold til beregning av temperaturøkning?

g) Hvis vi antar at temperaturen de 30 årene etter 2008 vil øke like mye som den har økt de siste 30 årene før 2008, kan man si noe om forventet temperatur i 2038? Rapporter svaret for 3 ulike `n`-verdier, både for gjennomsnitt og median.

h) **Diskusjonsspørsmål** Hva kan beregningene over fortelle oss om forventede klimaendringer? Hva kan de ikke fortelle oss? Er en lineær økning basert på de siste 30 årene en rimelig modell? Eller hadde det gitt mer mening å gjøre det på en annen måte?

i) **Diskusjonsspørsmål** Vi ser jo at temperatur-dataene varierer mye fra år til år, samtidig som vi kan se en generell trend. Er det noen måte å legge inn forventet variasjon i modellen du utviklet over, slik at man kan si at forventet temperatur vil være innenfor et visst intervall istedenfor én tallverdi?

j) **Bonusoppgave** Erstatt temperaturmålingene med målinger fra noen andre byer, f.eks. Bergen og Trondheim (`https://www.ssb.no/a/histstat/`

tabeller/2-7.html). Se på oppgavene over med nye data. Er det noe som endrer seg?

[a]Kilde: Det norske meteorologiske institutt, Klimaavdelingen, https://www.ssb.no/a/histstat/tabeller/2-7.html

Løsning oppgave 5 *Temperaturmålinger*

a) Koden plotter målt lufttemperatur i Oslo for årene 1871–20, og en utjevnet versjon av de samme dataene regnet ut ved hjelp av et glidende gjennomsnitt.

b) Høyere n-verdi gir jevnere data, men også flatere kurve (jevner ut både lokal variasjon og global trend).

c)

```
def glidende_median(data, n):
    data_utjevnet = np.zeros_like(data)
    antall_elem = len(data)

    for i in range(n):
        indeks = n + i + 1
        data_utjevnet[i] = np.median(data[:indeks])
        data_utjevnet[-i - 1] = np.median(data[-indeks:])

    for i in range(n, antall_elem - n):
        data_utjevnet[i] = np.median(data[i-n:i+n])

    return data_utjevnet
```

d)

```
n = 10    # justerer gjennomsnittsintervallet
temperatur_gjennomsnitt = glidende_gjennomsnitt(temperatur, n)
```

```
temperatur_median = glidende_median(temperatur, n)
år = np.arange(1871, 2009, 1)

plt.grid(alpha=0.3)
plt.plot(år, temperatur, "--", label="Temperatur")
plt.plot(år, temperatur_gjennomsnitt, label="
   Temperatur, glidende gjennomsnitt")
plt.plot(år, temperatur_median, label="Temperatur,
    glidende median")
plt.legend()
plt.title("Luftemperatur i Oslo (1871 2008)")
plt.show()
```

e) Median ser ut til å ligge noe over gjennomsnittet, noe som indikerer at det er flere år med litt høyere temperatur enn gjennomsnittet, men at man av og til får år med mye lavere temperatur enn vanlig. Hvis man lar n være veldig stor, ser vi også at medianverdiene varierer mindre enn de tilsvarende gjennomsnittsverdiene.

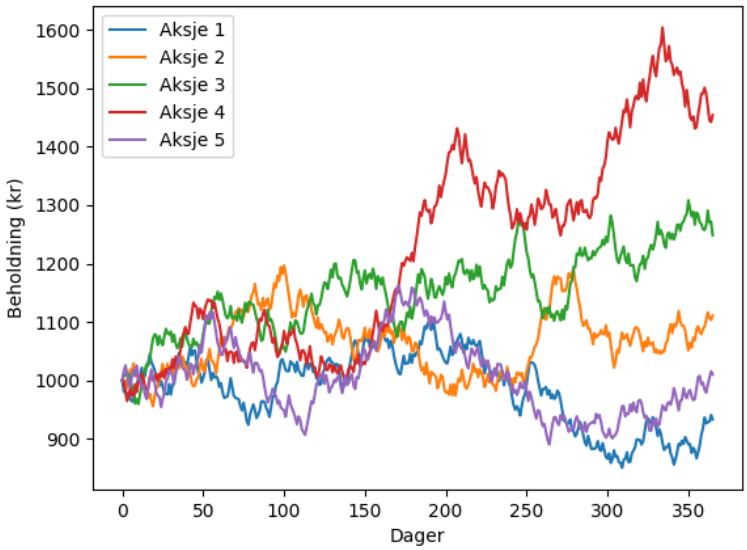
f)

```
for n in [5, 10, 20]:
  temperatur_gjennomsnitt = glidende_gjennomsnitt(
     temperatur, n)
  temperatur_median = glidende_median(temperatur,
     n)
  år = np.arange(1871, 2009, 1)

  forskjell_gjennomsnitt = temperatur_gjennomsnitt
     [-1] - temperatur_gjennomsnitt[-30]
  forskjell_median = temperatur_median[-1] -
     temperatur_median[-30]

  print(f"Forskjellen for n = {n}:")
  print(f" * {forskjell_gjennomsnitt:.2f} (snitt)"
     )
  print(f" * {forskjell_median:.2f} (median)")
```

g)

```python
for n in [5, 10, 20]:
    temperatur_gjennomsnitt = glidende_gjennomsnitt(temperatur, n)
    temperatur_median = glidende_median(temperatur, n)
    år = np.arange(1871, 2009, 1)

    forskjell_gjennomsnitt = temperatur_gjennomsnitt[-1] - temperatur_gjennomsnitt[-30]
    forskjell_median = temperatur_median[-1] - temperatur_median[-30]

    print(f"Forskjellen for n = {n}:")
    print(f" * {forskjell_gjennomsnitt:.2f} (snitt)")
    print(f" * {forskjell_median:.2f} (median)")

    temperatur_2008_snitt = temperatur_gjennomsnitt[-1]
    temperatur_2008_median = temperatur_median[-1]
    print(f"I 2008: ")
    print(f" * {temperatur_2008_snitt:.2f} (snitt)")
    print(f" * {temperatur_2008_median:.2f} (median)")

    økning_gjennomsnitt = temperatur_gjennomsnitt[-1] + forskjell_gjennomsnitt/30
    økning_median = temperatur_median[-1] + forskjell_median/30
    print(f"I 2038: ")
    print(f" * {økning_gjennomsnitt:.2f} (snitt)")
    print(f" * {økning_median:.2f} (median)")
```

Oppgave 6 *Aksjer*



I denne oppgaven skal vi simulere hva som *kan* skje med pengene dine dersom du velger å investere i én eller flere aksjer.

I mange tilfeller kan man få økt formue ved å investere i aksjemarkedet. Det er et samfunnsøkonomisk interessant spørsmål om hvorfor det slik, og om det faktisk er slik – men i denne oppgaven antar vi som et grunnleggende premiss at man i gjennomsnitt oppnår en økning. Imidlertid er det også en risiko for å tape penger, og det skal vi ta med i modellen vår ved hjelp av tilfeldighet.

a) Vi starter med å simulere hvordan én aksje utvikler seg over tid. Lag en variabel `beholdning` som inneholder et startbeløp, altså hvor mye vi vil starte investeringen vår med.

b) Vi vil legge inn en generell økning ved å la den vokse lineært, si med en daglig økning på 0.05 %. Lag en variabel `forventet_vekst` som inneholder forventet daglig vekst, altså 0.0005. For hver dag regner vi da med at beholdingen øker med `beholdning =(1 + forventet_vekst)* beholdning`. Lag en funksjon som regner ut dette. Lag en løkke som går over 365 dager, og skriv ut hvor mye beholdningen har endret seg etter ett år.

c) Lagre verdiene over tid, altså per dag, i en liste `over_tid` ved å bruke `append` etter hver gang du har regnet ut en ny verdi.

d) Plott verdiene over tid; her kan du ganske enkelt kalle på `plt.plot(over_tid)`. Her vil dagene være på x-aksen og pengebeholdningen på y-aksen. Legg inn merkelapper på aksene ved hjelp av `plt.xlabel("...")` og `plt.ylabel("...")` som viser dette.

e) Så langt er alt bestemt på forhånd av hvilke verdier vi har satt. Men aksjemarkedet medfører mye usikkerhet, og det vil vi også ta med! Lag en variabel `daglig_variasjon` som for hver iterasjon blir satt til et tilfeldig tall, si mellom -0.001 og 0.001. (**Hint**: Her kan du bruke `uniform` fra `random`-biblioteket.) Endre så måten beholdningen øker på ved å la `beholdning =(1 + forventet_vekst +daglig_variasjon) *beholdning`.

f) Kjør programmet noen ganger, og se på plottet som viser endring over tid. Ser det riktig ut? Endrer det seg mye fra kjøring til kjøring? Får du flest utfall der du tjener penger, eller flest der du taper penger?

g) Utvid programmet ditt til å jobbe med en liste av innskudd (f.eks. fem), der hvert innskudd representerer investering i én aksje. (**Hint:** Lag en løkke rundt nesten hele koden, men la `plt.show()` være utenfor løkken.) Plott utviklingen av de fem aksjene i samme plott, omtrent som vist i figuren i begynnelsen av oppgaven. Prøv igjen å kjøre programmet noen ganger. Hva kan du observere? Hva hadde du tenkt om det var dine ekte penger det representerte?

h) I denne oppgaven har vi regnet med en fast årlig vekst, og en gitt variasjon innenfor et visst intervall. Selv om vi bruker tilfeldighet til å simulere ulike mulige utfall, er det fortsatt en teoretisk modell. Tror du det er en realistisk modell? Hvorfor/hvorfor ikke? Hva kunne man gjort for å gjøre den mer realistisk?

Løsning oppgave 6 *Aksjer*

a)

```
1  beholdning = 1000
```

b)

```

def ny_beholdning(beholdning, forventet økning):
    return (1 + forventet_økning)*beholdning

beholdning = 1000
forventet_økning = 0.0005

for dag in range(365):
    beholdning = ny_beholdning(beholdning,
       forventet_økning)

print(f"Etter ett år har vi {beholdning:.2f} kr på
    konto.")
```

c)

```
def ny_beholdning(beholdning, forventet økning):
    return (1 + forventet_økning)*beholdning

beholdning = 1000
forventet_økning = 0.0005

over_tid = [beholdning]

for dag in range(365):
    beholdning = ny_beholdning(beholdning,
       forventet_økning)
    over_tid.append(beholdning)
```

```
import matplotlib.pyplot as plt

...

plt.plot(over_tid)
plt.xlabel("Dager")
plt.ylabel("Beholdning (kr)")
plt.show()
```

d)

```
from random import uniform
import matplotlib.pyplot as plt

def ny_beholdning(beholdning, forventet_økning,
   forventet_variasjon):
    daglig_variasjon = uniform(-
       forventet_variasjon, forventet_variasjon)
    return (1 + forventet_økning +
       daglig_variasjon)*beholdning

beholdning = 1000
forventet_økning = 0.0005
forventet_variasjon = 0.02

over_tid = [beholdning]

for dag in range(365):
    beholdning = ny_beholdning(beholdning,
       forventet_økning, forventet_variasjon)
    over_tid.append(beholdning)

plt.plot(over_tid)

plt.xlabel("Dager")
plt.ylabel("Beholdning (kr)")
plt.show()
```

e)

```

from random import uniform
import matplotlib.pyplot as plt

def ny_beholdning(beholdning, forventet_økning,
   forventet_variasjon):
    daglig_variasjon = uniform(-
       forventet_variasjon, forventet_variasjon)
    return (1 + forventet_økning +
```

```
        daglig_variasjon)*beholdning

aksjer = [1000, 1000, 1000, 1000, 1000]

for (i, beholdning) in enumerate(aksjer):
    forventet_økning = 0.0005
    forventet_variasjon = 0.02

    over_tid = [beholdning]

    for dag in range(365):
        beholdning = ny_beholdning(beholdning,
           forventet_økning, forventet_variasjon)
        over_tid.append(beholdning)

    plt.plot(over_tid, label=f"Aksje {i + 1}")

plt.xlabel("Dager")
plt.ylabel("Beholdning (kr)")
plt.legend()
plt.show()
~
```

Oppgave 7 *Hvor mye koster morgendusjen din?*

For å varme opp vann trenger vi energi, og i Norge kommer den energien vanligvis fra strøm. Enheten som brukes for å måle hvor mye strøm vi har brukt er gjerne kilowattimer (kWh), som representerer at du har brukt 1000 Watt i en time.

Til vanlig måler vi energiforbruk i Joule (J), så derfor kan det være nyttig å ha en funksjon som oversetter energi fra kWh til J og motsatt. Det kan vi gjøre med denne formelen

$$[\text{kWh}] = 3.6[\text{MJ}], \tag{1}$$

hvor MJ er megajoule, eller 10^6 J.

Når vi varmer opp vann, så bruker vi 4.2 J per liter (L) vann. For å gjøre

enhetsomregning lettere er det derfor vanlig å snakke om enheten *kalorier* (cal). En kalori representerer mengden energi som skal til for å varme opp en liter vann en grad Celcius (°C). Altså er en kalori gitt ved formelen

$$[\text{cal}] = 4.2[\text{kJ}], \tag{2}$$

hvor kJ er kilojoule, eller 1000 J.

a) Lag en funksjon `kWh_til_joule(energi_i_kWh)` som oversetter en energimengde oppgitt i kWh til en energimengde oppgitt i Joule.

b) Lag en funksjon `joule_til_kWh(energi_i_J)` som oversetter en energimengde oppgitt i Joule til en energimengde oppgitt i kWh.

c) Lag en funksjon `kalori_til_joule(energi_i_cal)` som oversetter en energimengde oppgitt i kalorier til en energimengde oppgitt i Joule.

d) Lag en funksjon `kalori_til_kWh(energi_i_cal)` som oversetter en energimengde oppgitt i kalorier til en energimengde oppgitt i kWh. (Tips: Bruk `joule_til_kWh`- og `kalori_til_joule`-funksjonene du allerede har laget).

e) Når vi dusjer bruker vi fort 60 L vann, og dette vannet blir gjerne varmet opp 35 grader. Bruk disse tallene for å estimere hvor mye energi (i kWh) du bruker hver gang du dusjer.

f) En tommelfingerregel er at hver kWh med energi vi kjøper koster ca 1 krone. Hvor mye penger bruker du da på å dusje hvis du dusjer hver dag i et år?

Løsning oppgave 7 *Hvor mye koster morgendusjen din?*

a)

```
1  def kWh_til_joule(energi_i_kWh):
2      return energi_i_kWh*1_000_000*3.6
```

b)

```python
def joule_til_kWh(energi_i_joule):
    return energi_i_joule/(1_000_000*3.6)
```

c)

```python
def kalori_til_joule(energi_i_cal):
    return energi_i_cal*4.2*1000
```

d)

```python
def kalori_til_kWh(energi_i_cal):
    energi_i_joule = kalori_til_joule(energi_i_cal
        )
    return joule_til_kWh(energi_i_joule)
```

e) Vi må bruke ca 2.5 kWh energi for å varme opp vannet til en dusj.

f) Et år med dusjing daglig koster ca 900 kroner.

Oppgave 8 *Hvor mye antibiotika i blodet*

Legen din har gitt deg 150 mg antibiotika, som du skal ta hver 24 time. Vi antar at all antibiotika går inn i blodet ditt, og at leveren fjerner 60 % av antibiotikanivået i blodet hver 24. time (dvs. vi tar vare på 40 %, angitt ved 0.4 nedenfor). Likningen for hvor mye antibiotika du har i blodet etter n doser:

$$x_n = 0.40x_{n-1} + 150 \tag{3}$$

Vi kan simulere dette på følgende måte:

```python
x = 0
dag = 0
antall_dager = 15
daglig_dose = 150

while dag < antall_dager:
  dag += 1
  x = 0.4*x + daglig_dose
```

```
9
10 print(f"Etter {antall_dager} har du {x} mg
      antibiotika i blodet.")
```

Lag et Python-program der du kopierer over eller skriver av dette programmet.Kjør programmet. Hvor mye antibiotika har du igjen etter 15 dager? Hva skjer på linje 6 (`dag += 1`)? Hva skjer om du tar bort denne linjen, tror du? Legen vurderer også en dosering med 10 dager med 160 mg hver dag. Endre på variablene `antall_dager` og `daglig_dose`. Fører dette til mer eller mindre antibiotika i blodet på siste dag? Lag en tom liste `antibiotikakonsentrasjon` før løkken, og bruk `append` til å legge til `x` i denne listen for hver iterasjon, slik at du ender opp med en liste over konsentrasjon per dag. Lag også en liste `dager` over antall dager på samme måte. Plott listen `antibiotikakonsentrasjon` over `dager` slik at du får et plott som viser hvor mye antibiotika du har i blodet pr dag. Legg på passende merkelapper på aksene ved hjelp av `xlabel` og `ylabel`.

d) Løsning oppgave 8 *Hvor mye antibiotika i blodet*

```
1  x = 0
2  dag = 0
3  antall_dager = 10
4  daglig_dose = 160
5
6  antibiotikakonsentrasjon = []
7  dager = []
8
9  while dag < antall_dager:
10   x = 0.4*x + daglig_dose
11   dag += 1
12   antibiotikakonsentrasjon.append(x)
13   dager.append(dag)
14
15 print(f"Etter {antall_dager} har du {x} mg antibiotika
       i blodet.")
16
17 plot(dager, antibiotikakonsentrasjon)
```

```
xlabel("Dag")
ylabel("Konsentrasjon (mg)")
show()
```

Oppgave 9 *Vekstfaktor*

I en petriskål lever en bakteriekultur som består av 1000 bakterier som formerer seg raskt nok til at mengden bakterier øker med 50% hver time. I denne opppgaven skal vi bruke plotting og python til å modellere og visualiserer veksten i bakteriekulturen over 10 timer

a) Opprett en variabel, `vekstfaktor`, som har vekstfaktoren tilhørende 50%

b) Bruk `arange` fra `pylab` til å opprette en array, `timer`, med tallene fra 1 til 10

Nå vil vi regne ut hvor mange bakterier som er i petri-skålen hver time. For å gjøre det, må vi først opprette en tom array og bruke en løkke til å "fylle" inn antall bakterie hver time. Men først, hva er egentlig en tom array? Vel, vi kan jo bruke en array hvor alle elementene er lik 0. Dette er det en funksjon for i `pylab` fra før av, og den heter `zeros`. Hvis vi skriver `ti_nuller =zeros(10)`, vil vi få en variabel `ti_nuller` som består av ti element som alle er lik null.

c) Bruk `zeros` fra `pylab` til å opprette en tom array, `bakteriemengde` med 10 elementer. Husk å importere `zeros` først!

Det neste steget er å endre verdiene til hvert element i arrayen vårt. Dette kan vi gjøre med *indeksering*. En array består av mange tall som er etter hverandre, og hvis du vil ha ut et tall fra en array bruker vi klammeparanteser. Hvis vi har en array-variabel, `x`, kan rett og slett skrive `x[0]` for å hente ut det første elementet i arrayen. Tilsvarende kan vi skrive `x[1]` for å hente ut det andre elementet i arrayen, osv. Under har vi et eksempel

```
from pylab import arange

tallrekke = arange(10)*2
```

```
første_tall = tallrekke[0]
femte_tall = tallrekke[4]

print(første_tall)
print(femte_tall)
```

```
0
8
```

Tilsvarende, kan vi starte med en tallrekke, og endre enkeltelement! Se eksempelet under.

```
from pylab import arange

tallrekke = arange(4)*2
print(tallrekke)

tallrekke[0] = -1
tallrekke[2] = 0

print(tallrekke)
```

```
[0 2 4 6]
[-1  2  0  6]
```

Vi ser her at vi bruker `tallrekke[i] =x` for å sette verdien til element nummer $i+1$ i arrayet lik x.

d) Bruk array-indeksering (klammeparanteser) til å sette det første elementet (element nummer 0) i `bakteriemengde` til `1000` som er startverdien.

e) Bruk en løkke `for time in range(1,10)` til å løkke igjennom alle timene fra 1 til 9

f) Bruk arrayindeksering til å sette bakteriemengden for tid `time` til å være lik vekstfart multiplisert med bakteriemengden for tid `time-1`. **Hint:** `bakteriemengde[time-1]`

g) Bruk `plot` og `show` funksjonene fra `pylab` til å plotte bakteriemengde på yaksen og timer på y-aksen

h) Bruk `xlabel` og `ylabel` funksjonene fra `pylab` til å legge merkelapper på x og y-aksen

i) Refleksjonsoppgave: Hva vil skje med bakteriene etterhvert som tiden går? Hva synes du om denne modellen? Er det noen svakheter ved en slik modell?

Løsning oppgave 9 *Vekstfaktor*

a)

```
1 vekstfaktor = 1.5
```

b)

```
1 from pylab import arange
2
3 timer = arange(10)
```

c)

```
1 from pylab import arange, zeros
2
3 bakteriemengder = zeros(10)
```

d)

```
1 bakteriemengder[0] = 1000
```

e)

```
1 for time in range(1, 10):
2     bakteriemengder[time] = bakteriemengder[time-1
         ]*vekstfaktor
```

f)

```
from pylab import plot, show

plot(timer, bakteriemengder)
show()
```

g)

```
from pylab import plot, show, xlabel, ylabel

plot(timer, bakteriemengder)
xlabel("Timer")
ylabel("Antall bakterier")
show()
```