28.05.2021

Prosjektoppgave

APROG200 Anvendt Python-programmering (våren 2021)

Jon Bjarne Bø

Innhold

Av: Jon Bjarne Bø

Innledning

I denne prosjektoppgaven vil jeg ta for meg 2 ulike undervisningsopplegg. Det første opplegget vil rettes inn mot naturfag for påbyggselever når den nye læreplanen trer i kraft. Det andre opplegget er rettet mot matematikk for dagens påbyggselever (2P-Y). Anslått tidsbruk er skrevet i parentes under hver underoverskrift.

Temaene fra kurset som jeg har valgt å sette søkelys på er simulering/modellering i naturfagsdelen, og sannsynlighet i matematikkdelen.

Naturfag

I de nye kompetansemålene for naturfag påbygg som trer i kraft skoleåret 2022/2023 står følgende kompetansemål oppført:

«Mål for opplæringen er at eleven skal kunne

- vurdere og lage programmer som modellerer naturfaglige fenomener
- utforske og beskrive elektromagnetisk og ioniserende stråling og vurdere informasjon om stråling og helseeffekter av ulike strålingstyper» (https://www.udir.no/lk20/nat01-04/kompetansemaal-og-vurdering/kv68)

I det første undervisningsopplegget er derfor fokuset halveringstid, og hvordan vi kan jobbe med å forstå halveringstid og lage en simulering av dette ved å bruke programmering.

Selv om disse målene ikke er gjeldende enda, er de gamle målene her ganske like, og opplegget kan brukes på dagens påbyggselever også.

Matematikk

I den andre undervisningsopplegget tar jeg utgangspunkt i nåværende læreplan for påbygg i matematikk (2P-Y) og kompetansemålet som har fokus her er:

«Mål for opplæringen er at eleven skal kunne

- lage eksempler og simuleringer av tilfeldige hendelser og gjøre rede for begrepet sannsynlighet» (https://www.udir.no/kl06/MAT6-03/Hele/Kompetansemaal/kompetansemal-etter-2p-y)

I dette andre undervisningsopplegget er altså søkelyset på sannsynlighetsregning.

Jeg har valgt å ikke ta med de matematiske utregningene som elevene selv skal jobbe med, med mindre det er relevant for programmeringen. I opplegget om sannsynlighetsregning er det mange av oppgavene som legger til rette for at elevene både skal regne den teoretiske sannsynligheten, men bruke programmering for å simulere den empiriske sannsynligheten.

Av: Jon Bjarne Bø

Undervisningsopplegg 1 - Halveringstid

For å gjennomføre dette opplegget må elevene ha litt erfaring med Python fra før av. De bør kunne det aller mest grunnleggende, da en skal jobbe litt med numpy og matplotlib.pyplot.

Telle opp gunstige utfall

(Tidsbruk: omtrent 45 minutter med visning fra lærer)

I oppstarten av økten introduserer lærer np.random.choice metoden ved å bruke denne i en løsning på én av oppgavene under. Elevene jobber med resten av oppgavene selv for å bli kjent med metoden.

Oppgaver:

1. *Du har en 6-sidet terning som du skal kaste 100 ganger. Lag et program som teller antall seksere du har etter 100 kast.*
2. *Du har en 20-sidet terning som du skal kaste 10000 ganger. Lag et program som teller antall énere du har etter 10000 kast.*
3. *Lag et program som trekker et tilfeldig tall blant de 20 første partallene 100 ganger. Programmet skal lage en liste med alle de 100 partallene og i tillegg regne ut summen av alle tallene som er i lista.*
4. *Emma, Per, Tore og Jan skriver på navnet sitt på hver sin Post-It-lapp som de legger i en boks. De trekker opp en lapp av boksen, skriver ned navnet og legger lappen tilbake i boksen igjen. Dette skal de gjøre 10 ganger. Når de er ferdig, teller de opp hvor mange ganger de ulike navnene er blitt trukket.*

   Lag et program som trekker navn slik som i tilfellet over, men trekker navnene 1000 ganger. Programmet skal også skrive ut en oversikt over hvor mange ganger de ulike navnene ble trukket blant de 1000 trekkene.

Oppgavene over kan også løses med å bruke en for-løkke, så her kan lærer trekke frem fordelene ved å bruke random.choice, men legge til rette for at elevene også kan bruke en for-løkke om de ønsker det.

Etter at elevene har fått litt erfaringer med å trekke tilfeldig av et utvalg skal de jobbe med en simulering av terningkast for å sette seg inn i begrepet halveringstid.

Halveringstid og terningkast

(Tidsbruk: 20-45 min avhengig av om elevene får tilgang til koden under eller ikke)

Programmet som elevene skal lage nå skal være grunnlag for utforsking av halveringstid og fungere som en inngangsport til å lage simuleringer for halveringstid av radioaktive isotoper. Programmet skal simulere terningkast med 5000 terninger hvor alle 6-erne som trilles skal tas bort for hvert kast. Når dette er gjort skal de kaste de resterende terningene på nytt og gjenta dette 8 ganger. Antall terninger skal plottes slik at elevene kan bruke dette til å approksimere halveringstiden til terningene.

For mange av elevene kan dette bli et litt omfattende program å lage helt fra bunn, så om oppgaven er for vanskelig kan man differensiere ved å gi elevene koden som står under oppgaveteksten (men ikke ha med denne koden til de andre elevene som kan få til å bygge hele programmet selv). De elevene som får tilgang til koden, skal kopiere denne og kun gjøre endringer i for-løkken, slik at programmet fungerer som det skal. Om man ønsker at elevene skal skrive koden som plotter kan man kutte nederste del av koden bort. Oppgaven, slik elevene får den, står beskrevet på neste side.

Av: Jon Bjarne Bø

Oppgave om halveringstid av terninger

Du skal lage et program som skal simulere terningkast av 5000 terninger. Terningene skal kastes 8 ganger, men for hvert kast skal alle terningene som viser 6 skal tas bort.

Programmet skal skrive ut antall terninger som er igjen etter hvert kast (etter at 6-erne er tatt bort), og plotte resultatene. Plottet skal være utgangspunkt for å finne et estimat for halveringstiden (hvor mange kast som trengs for at antall terninger er blitt halvert).

For å komme i gang kan det være lurt å starte med å skrive en pseudokode først slik at du kan strukturen til programmet, men dette er ikke nødvendig.

Om du synes det er vanskelig å komme i gang med programmet kan du skrive av koden under. Da blir oppgaven å gjøre endringer i for-løkken slik at programmet under fungerer slik det er beskrevet i oppgaveteksten over.

```
# Simulering av halveringstid ved å bruke 6-kantet terning

import numpy as np
import matplotlib.pyplot as plt

# Definerer variabler
terning_verdier = [1, 2, 3, 4, 5, 6]
antall_terninger = 5000
halvering = antall_terninger/2
kast_array = [0, 1, 2, 3, 4, 5, 6, 7, 8]
terning_array = np.zeros(9) # Array som skal inneholde y-verdiene (antall terninger igjen)
terning_array[0] = antall_terninger

for i in range(8):
    # Utvid programmet ved å skrive inn kode i denne for-løkken slik at
    # variabelen terning_array inneholder antall terninger om man tar bort alle
    # sekserne man får på hvert kast.

# Plotter en graf med de utregnede punktene
fig = plt.figure(1, figsize=(12, 9))
plt.subplot(1, 1, 1)
plt.plot(kast_array, terning_array, 'r.-')
plt.axhline(y=halvering)
plt.xlabel('Antall kast')
plt.ylabel('Antall terninger')
plt.title('Halveringstid av terninger')
plt.show()
```

Av: Jon Bjarne Bø

Halveringstid og radioaktive isotoper

(Tidsbruk: 60-120 minutter avhengig av hvor mange av ekstraoppgavene man krever)

Etter arbeidet med terningkastene i forrige oppgave er elevene klare for å jobbe med å lage en simulering av halveringer med isotoper. Elevene skal ta utgangspunkt i halveringstid til noen isotoper, og lage et program som regner ut antall kjerner som er igjen etter et gitt antall år. Oppsettet for denne oppgaven er litt annerledes da poenget er at elevene skal bruke vekstfaktor og ikke halveringstiden direkte. Derfor får elevene ut et ark med den sentrale delen av koden som et «Parsons problem» (kode-puslespill). Jeg ble inspirert til å teste ut denne metoden da jeg kom over den i et tidsskrift om matematikk i skolen kalt «tangenten». Artikkelen jeg har brukt som inspirasjon ligger her: http://www.caspar.no/tangenten/2021/tangenten%201%202021%20Fl%C3%B8.pdf

De to neste sidene inneholder selve opplegget og skrives ut dobbeltsidig til elevene (evt. kan det skrives ut på et A3-ark). Side 2 av opplegget er ekstraoppgaver for de elevene som klarer å gjennomføre side 1 raskt, eller som trenger noen ekstra utfordringer. Opplegget er estimert til å vare i 60-90 minutter, avhengig av hvor mye elevene har programmert tidligere og hvor gode de er på utregninger med potensregler og omgjøring av formler. Om man velger å gi alle elevene alle oppgavene kan dette også være et opplegg for en 3-timersøkt (3 x 45 minutter) som skal leveres inn og vurderes.

I forkant av denne økten kan det være lurt å ha gått gjennom litt om radioaktivitet på forhånd. Elevene kan eventuelt lese seg litt opp ved å bruke NDLA: https://ndla.no/subject:3d9454e8-460e-42c7-8f28-71663fbbf6e6/topic:dbc23651-7216-4610-bc38-dde58f013724/topic:1b4bd84e-b60f-40cb-83a8-23ad8926298a/resource:e766c14e-26b7-4cb5-a558-d0919fcf3097?filters=urn:filter:f18b0daa-6507-4025-8998-b8a11c8ccc70

Vedlagt ligger kun programmet til selve hovedoppgaven, og ikke ekstraoppgavene, da disse er valgfrie og kun små modereringer av selve hovedoppgaven.

Av: Jon Bjarne Bø

Halveringstid for radioaktive isotoper

Oppgave
Du skal lage et program som simulerer halvering av karbon-14. Programmet skal ta utgangspunkt i at du har 10 000 karbon-14 isotoper og med utgangspunkt i halveringstiden for karbon-14 plotte mengden av karbon-14 hvert år de neste 30 000 årene. Du skal deretter bruke denne simuleringen til å anslå mengden av karbon-14 som er igjen etter 30 000 år.

Informasjon

En radioaktiv isotop er en ustabil isotop av et grunnstoff som nedbrytes spontant, desintegrerer, og sender ut radioaktiv stråling. Nedbrytningshastigheten varierer fra isotop til isotop, og denne hastigheten kaller vi for halveringstid. Halveringstiden er den tiden det tar før halvparten av massen (antall atomer) av et radioaktivt stoff er omdannet til et annet grunnstoff. Halveringstidene innebærer også noe usikkerhet som vi ikke forholder oss til i denne oppgaven.

Til høyre er en tabell med noen ulike isotoper og halveringstiden til disse isotopene.

Isotop	Halveringstid
Uran-238	$4{,}468 \cdot 10^9$ år
Uran-235	$7{,}038 \cdot 10^8$ år
Plutonium-239	24 110 år
Karbon-14	5730 år
Cesium-137	30 år
Thorium-234	24,1 døgn
Radon-222	3,8 døgn
Astat-211	7,2 timer
Polonium-214	$1{,}643 \cdot 10^{-4}$ sekunder

Fase 1: Ta utgangspunkt i halveringstiden til radon-222 og finn ut hvor mange prosent radon-222 minker med hvert døgn. Finn vekstfaktoren og bruk denne til å kontrollere at du har regnet riktig (Husk: $mengde \cdot vekstfaktor^{døgn} = ny\ mengde$)

Fase 2: Bruk formelen fra fase 1 til å regne deg frem til en generell formel for vekstfaktor om man har halveringstid. Her må dere bruke kunnskaper fra potensregningen (Husk: $(a^m)^{\frac{1}{m}} = a$)

Fase 3: I bildet under ser du 12 linjer med kode. Lag et program ved å bruke disse linjene (du kan også programmet ditt fra bunn selv om du heller ønsker det). Programmet skal lage to lister; én array med x-verdier som skal inneholde hvert år de neste 30000 årene, og en array med tilhørende y-verdier (mengden igjen av karbon-14 det gitte døgnet).

Fase 4: Utvid programmet du laget i forrige fase slik at det plotter verdiene og lager en oversiktlig graf. Du skal også få programmet til å skrive ut hvor mye som er igjen av karbon-14 etter 30 000 år.

```
antall_verdier = maks_x + 1
                                y_verdier[i] = antall_atomkjerner * vekstfaktor ** x_verdier[i]
 maks_x = 30000
                                    x_verdier = np.linspace(0, maks_x, antall_verdier)
enhet_tid = "år"
                                  import numpy as np
antall_atomkjerner = 10000
                                    y_verdier = np.zeros(antall_verdier)
isotop = "karbon-14"
                               for i in range(antall_verdier):
 halveringstid = 5730
     vekstfaktor = ((antall_atomkjerner/2) / antall_atomkjerner) ** (1/halveringstid)
```

Av: Jon Bjarne Bø

Ekstraoppgaver:

1. Gjør endringer i programmet slik at du modellerer en situasjon hvor du har 1 000 000 astat-211 isotoper. Du bestemmer selv hvor mange timer du vil lage simuleringen for, men reflekter rundt dette og begrunn valget av x-verdier.
2. Hvis du har 6000 radon-222 isotoper. Hvor mange isotoper har du igjen etter 16 døgn?
3. Hvis du har en mengde polonium-214 isotoper, hvor lang tid tar det før det er mindre enn 1% igjen?
4. Du gjør målinger og måler at det er omtrent 15 000 thorium-234 isotoper et sted. Hvor mange isotoper var det der for 1 år siden?

Refleksjonsoppgave:

I en utgave av illustrert vitenskap står det:

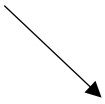
Karbon 14-metoden har siden femtitallet hjulpet arkeologene med å aldersbestemme funn som er opp til 50 000 år gamle.

Hvorfor kan vi ikke aldersbestemme funn som er eldre enn 50 000 år gamle med karbon-14-metoden?

Utvidelsesoppgave 1:

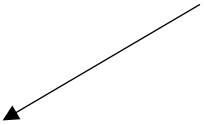
Gjør endringer i programmet slik at du sammenlikner plutonium-239, karbon-14 og cesium-137 ved å alle de tre grafene i samme koordinatsystem.

Ta som utgangspunkt at du starter med like mange atomkjerner av alle isotopene.



Utvidelsesoppgave 2:

Gjør endringer i programmet slik at du definerer en funksjon til å regne ut y-verdiene som du bruker i programmet.



Utvidelsesoppgave 3:

Gjør endringer i programmet slik at programmet også modellerer for verdier i negativ x-retning (altså negativ tid).

Av: Jon Bjarne Bø

Undervisningsopplegg 2 – Sannsynlighetsregning

For å gjennomføre dette opplegget må elevene ha litt erfaring med Python fra før av. De bør kunne det aller mest grunnleggende, da en skal jobbe litt med numpy og matplotlib.pyplot.

Terningkast

(Tidsbruk: 90 minutter)

Lærer starter med å gi elevene koden under. Elevene skal deretter, sammen med sidemannen, forklare koden linje for linje (om det er noen linjer de ikke forstår skal de søke seg frem på internett) i tillegg til å lage en siste linje som skriver ut sannsynligheten for å få 1, 2 eller 3 i denne simuleringen.

```
import numpy as np

terning_verdier = [1, 2, 3, 4, 5, 6]
antall_kast = 200
antall = 0

for i in range(antall_kast):
    terningkast = np.random.choice(terning_verdier)
    if terningkast < 4:
        antall += 1
```

Lærer forklarer deretter programmet raskt i plenum og kommer med forslag til en linje som skriver ut sannsynligheten for å få 1, 2 eller 3. Lærer forklarer da også at om man øker antall kast vil man komme nærmere og nærmere den teoretiske sannsynligheten som man kan regne på.

Med bakgrunn i denne introduksjonen skal elevene få noen oppgaver som er litt vanskeligere og som de skal lage et program som kan løse. Alle oppgavene skal også løses teoretisk ved utregning og elevene skal sammenlikne det teoretiske svaret med den sannsynligheten de finner ved å bruke Python. Jeg velger å ikke presentere de teoretiske svarene i oppgaven, men programkoder til alle oppgavene ligger vedlagt.

Oppgave 1
Lag et program som simulerer et kast med to sekskantede terninger og finner sannsynligheten for at summen av terningene blir 9.

Oppgave 2
Lag et program som simulerer et kast med to ulike terninger. Én terning som har seks kanter og en terning som har 20 kanter. La programmet regne ut sannsynligheten for at summen skal bli et partall.

Oppgave 3
Lag et program som simulerer et kast med to ulike terninger. Én terning som har seks kanter og en terning som har 20 kanter. Programmet skal finne sannsynligheten for å få en sum som **ikke** er mellom 12 og 15 (medregnet tallene selv).

Oppgave 4
Summen av to sekskantede terninger må være et tall mellom 2 og 12. Utvid programmet fra oppgave 1 til å finne de ulike sannsynlighetene for alle summene og legge dem i en liste. Programmet skal plotte resultatet i et stolpediagram.

Av: Jon Bjarne Bø

Oppgavene på forrige side kan by på noen utfordringer, men de fleste av disse utfordringene går på logikk, som også er veldig viktig når man regner teoretisk på sannsynlighetene. Når elevene har gjennomført oppgavene og opplegget knyttet til terningkast er de klare for å fortsette over på å finne sannsynligheter ved ulike trekninger.

Trekninger

(Tidsbruk: 120 minutter)

I den første delen av arbeid med trekninger vil lærer gå gjennom et enkelt eksempel på trekning av tre elever:

```
import numpy as np

elever = ['Emma', 'Jone', 'Niclas']

trekning = np.random.choice(elever)

print(trekning)
```

Lærer skriver programmet samtidig som hen forklarer hva som skjer. Her bør også lærer repetere at random.choice også kan lage en liste om man ønsker å trekke flere ganger og ha dataene lagret i samme variabel. Det kan også være lurt å vise hvordan man kan telle antall tilfeller i en liste eller array:

```
import numpy as np

elever = ['Emma', 'Jone', 'Niclas']

trekning = np.random.choice(elever, 12)

antall_emma = np.count_nonzero(trekning=='Emma')

print(trekning)
print(antall_emma)
```

Det er også lurt å vise at det er mulig å trekke både med og uten tilbakelegg med random.choice:

```
trekning = np.random.choice(elever, 2)
trekning = np.random.choice(elever, 2, replace=False)
```

Når lærer har gått gjennom disse ulike metodene er det på tide å levere ut et oppgaveark til elevene. Arket som elevene får utlevert, står på neste side og skal skrives ut dobbeltsidig. Her må de reflektere rundt et ferdig program og gjøre endringer. De får også en utfordring, på slutten av den første siden, hvor de skal lage et lotteriprogram (dette kan ta litt tid, så det kan droppes om tiden blir knapp).

Når de er ferdig å jobbe med side 1 av arket kan de gå over til de mer tradisjonelle oppgavene på baksiden. Oppgavene som elevene skal jobbe med skal også løses teoretisk slik at svarene kan sammenliknes og brukes til å få en bekreftelse/avkreftelse på at man har fått programmet til å fungere riktig.

Av: Jon Bjarne Bø

Trekning av kuler

```python
import numpy as np

kuler = ['Rød', 'Rød', 'Rød', 'Rød', 'Blå', 'Blå', 'Blå', 'Gul']

antall_trekninger = 120
trekninger = np.random.choice(kuler, antall_trekninger)

red = np.count_nonzero(trekninger=='Rød')
blue = np.count_nonzero(trekninger=='Blå')
yellow = antall_trekninger - (blue + red)

prob_yellow = yellow / antall_trekninger

print(f'{prob_yellow:.2%}')
```

1. Se på programmet til venstre skriv en kort forklaring om hva programmet gjør.

2. Skriv av programmet i Spyder og kjør koden. Hva skjer?

3. Forklar koden

Kodelinje	Forklaring
`import numpy as np`	
`kuler = ['Rød', 'Rød', 'Rød', 'Rød', 'Blå', 'Blå', 'Blå', 'Gul']`	
`antall_trekninger = 120`	
`trekninger = np.random.choice(kuler, antall_trekninger)`	
`red = np.count_nonzero(trekninger=='Rød')` `blue = np.count_nonzero(trekninger=='Blå')`	
`yellow = antall_trekninger - (blue + red)`	
`prob_yellow = yellow / antall_trekninger`	
`print(f'{prob_yellow:.2%}')`	

4. Endre koden

a) Gjør endringer i programmet slik at det egner seg bedre til å estimere den faktiske sannsynligheten for å trekke gul.

b) Gjør endringer i programmet slik at det regner ut og skriver ut sannsynligheten for å trekke rød.

c) Lag programmet slik at det skriver ut sannsynlighetene for alle fargene.

d) Kan du fjerne en linje i programmet og fortsatt få samme resultat? Må du endre noe mer i tillegg?

e) Gjør endringer slik at programmet finner sannsynligheten for at du først trekker en gul kule, og deretter trekker en rød kule.

5. Utfordring

I ett litt forenklet lotto trekkes det 7 tall av 34 mulige (1-34). Spilleren skal på forhånd velge seg 7 tall som hen vil satse på at blir trukket. Lag et program med en liste som inneholder dine 7 tall. Deretter skal programmet trekke 7 tilfeldige tall av de 34 mulige og gjenta dette helt til den har trukket de samme tallene som er i din liste. Hvor mange ganger måtte programmet trekke, og hva ble sannsynligheten for at du vant?

Av: Jon Bjarne Bø

Flere oppgaver med trekning

(Disse oppgavene skal du både løse ved regning i boka og ved å programmere)

Oppgave 1

I en boks ligger det 6 grønne, 4 røde og 2 lilla kuler. Du trekker to kuler (uten å legge noen tilbake).

a) Hva er sannsynligheten for at begge er lilla?

b) Hva er sannsynligheten for at den første kula du trekker er rød og den andre kula er lilla?

c) Hva er sannsynligheten for at du trekker én rød og en lilla kule?

Oppgave 2

Du ser et lykkehjul. $\frac{1}{6}$ av hjulet er blått, $\frac{1}{3}$ av hjulet er rødt, og $\frac{1}{2}$ av hjulet er grønt. Du spinner hjulet to ganger.

a) Hva er sannsynligheten for at lykkehjulet stopper på rødt begge gangene?

Sammenlikn svaret i programmet og det du har regnet ut for hånd.

b) Hvor sto forskjell er det mellom programmet og den eksakte verdien?

c) Hvor mange forsøk må du gjennomføre før feilen blir mindre enn 0,5 prosentpoeng?

Oppgave 3

Du kaster en tikrone 3 ganger.

a) Hva er sannsynligheten for at du får nøyaktig to mynt?

b) Hva er sannsynligheten for at du får minst to mynt?

Eksamensoppgave 2P-Y, V2018 – Oppgave 6 del 2:

Maria vil legge to røde og to blå kuler i en kopp.

- Hun vil trekke to kuler fra koppen tilfeldig.
- Hun vil ikke legge tilbake den første kula før hun trekker den neste.

a) Tegn et valgtre, gjør beregninger, og avgjør hvilken av påstandene nedenfor som er riktig.

Påstand 1: Det er mest sannsynlig at hun kommer til å trekke to kuler med samme farge.

Påstand 2: Det er mest sannsynlig at hun kommer til å trekke to kuler med ulik farge.

Påstand 3: Sannsynligheten for at hun kommer til å trekke to kuler med samme farge, er like stor som sannsynligheten for at hun kommer til å trekke to kuler med ulik farge.

b) Gjør beregninger, og avgjør hvilken av påstandene ovenfor som er riktig dersom Maria i stedet legger tre røde og én blå kule i koppen og trekker to kuler på samme måte som beskrevet ovenfor.

Av: Jon Bjarne Bø