ProbabilityIntro_NO

January 26, 2026

[3]:
```
# to run from IN-STK1050 repository: formatting for presentation mode to␣
 ↪increase font
from IPython.display import HTML; display(HTML(f"<style>{open('../../
 ↪notebook_formatting.css').read()}</style>"))
```

```
<IPython.core.display.HTML object>
```

1 Introduksjon til sannsynlighet

1.1 Eksempelproblem

Et stort antall kort fra mange forskjellige kortstokker er spredt utover gaten. Hvor mange kort må man se på før alle 52 forskjellige kortene fra en kortstokk er sett?

1.2 Læringsmål

- kjenne til grunnleggende begreper som utfall, hendelse, sannsynlighet, sannsynlighetsfordeling, eksperiment og betinget sannsynlighet
- kunne nevne noen sannsynlighetsfordelinger som binomisk, uniform
- koble grunnleggende begreper til kode som simulerer tilfeldige prosesser
- utforske problemstillinger gjennom beregning av enkle sannsynligheter og sannsynlighetsfordelinger

1.3 Et raskt spørsmål før vi starter...

Lenke: https://www.menti.com/4ii97w7bfi

Kode for menti.com: 3269 1977

1.4 Eksempel: myntkast

[ ]:
```
import random
import numpy as np
from util import plott_hendelse_sannsynligheter, peng1, peng2
```

[ ]:
```
def peng():
    return random.sample(['kron', 'mynt'], 1)[0]

peng()
```

1.5 Eksempel: terningkast

```
[ ]: # live-kodet

def terning():
    return random.sample([1, 2, 3, 4, 5, 6], 1)[0]

terning()
```

1.6 Sannsynlighet

Sannsynligheten for å få kron når man kaster en peng:

hvis vi kaster en peng veldig mange ganger (uendelig mange), hvor ofte får vi kron

```
[ ]: antall_kast = 10000
kastene = [peng() for _ in range(antall_kast)]

print(f"Hvor ofte vi får kron: {round(kastene.count('kron') / antall_kast *␣
 ↪100)}% av tiden")
print(f"Hvor ofte vi får mynt: {round(kastene.count('mynt') / antall_kast *␣
 ↪100)}% av tiden")
print(f"Kast:\n{kastene}\n")
```

1.6.1 Sannsynlighet

```
[ ]: kast = [peng2() for _ in range(antall_kast)]

print(f"Hvor ofte vi får kron: {round(kast.count('kron') / antall_kast * 100)}%␣
 ↪av tiden")
print(f"Hvor ofte vi får mynt: {round(kast.count('mynt') / antall_kast * 100)}%␣
 ↪av tiden")
print(f"Kast:\n{kast}\n")
```

```
[ ]: ## Eksempel: vektet mynt

Hva skjedde med peng2?
```

```
[ ]: import inspect
print(inspect.getsource(peng2))
```

peng2 er slik at det er fire ganger så sannsynlig å få kron som det er å få mynt.

1.7 Alternativ implementering av peng-funksjonen

Tilbake til den originale pengen, den kunne ha blitt implementert på denne måten:

[ ]:
```python
def peng3():
    tilfeldig_verdi_mellom_0_og_1 = random.random()
    print(tilfeldig_verdi_mellom_0_og_1)
    return 'kron' if tilfeldig_verdi_mellom_0_og_1 < 0.5 else 'mynt'

peng3()
```

1.7.1 Øvelse: vektet mynt

Skriv en funksjon for å simulere en vektet peng med eksplisitt angitt sannsynlighet for å få kron.

[ ]:
```python
def vektet_peng(p_kron: float = 0.3):
    pass

vektet_peng()
```

1.8 Utfall og eksperimenter

Hvis vi kaster en peng, kan to ting skje:

- vi kan få kron
- vi kan få mynt

I denne sammenhengen er kron og mynt mulige **utfall** av å kaste en peng.

Å kaste en peng kalles da et **eksperiment**.

Mer formelt er et eksperiment en tilfeldig prosess som produserer nøyaktig ett mulig utfall.

1.9 Utfallsrom

Et sett med alle mulige utfall kalles et **utfallsrom**.

For pengekast: utfallsrom = {kron, mynt}

Elementene i utfallsrommet skal være distinkte og gjensidig utelukkende –> unikt utfall av et eksperiment

Utfallsrommet må være kollektivt uttømmende: uansett utfallet av et eksperiment, må utfallet være en del av utfallsrommet

For pengekast: både kron og mynt må være i utfallsrommet

1.10 Utfallsrom

Et sett med **alle mulige utfall** kalles et **utfallsrom**.

[ ]:
```python
def peng():
    utfallsrom = ['kron', 'mynt'] # utfallsrom med 2 mulige utfall
    utfall = random.sample(utfallsrom, 1)[0]
    return utfall
```

```
oppnådd_utfall = peng()

print(f"Utfall: {oppnådd_utfall}")
print(f"Gyldig utfall: {oppnådd_utfall in ['kron', 'mynt']}")
```

1.11 Eksempel: flere myntkast

Hvis vi kaster en mynt 3 ganger, hva er sannsynligheten for at vi får nøyaktig 2 kron?

Hva er de mulige utfallene her?

[ ]:
```
def kast_peng_n_ganger(n):
    # live-kodet:
    return tuple([peng() for _ in range(n)])

kast_peng_n_ganger(n=3)
```

Hvordan kan vi nå estimere sannsynlighetene for hendelser som å få nøyaktig 2 kron? Hvordan ville simuleringen se ut?

1.12 Hendelse

En hendelse A er et sett med mulige utfall.

Sannsynligheten for en hendelse A, betegnet P(A), beskriver kunnskapen eller troen om den kollektive "sannsynligheten" for elementene i A.

Eksempel med pengekast:

Utfall: K, M

Eksempler på hendelser: {K}, {M}

1.13 Hendelse

Kast en peng 3 ganger:

Utfall: KKK, KKM, KMK, KMM, MKK, MKM, MMK, MMM

Hendelser: A = {nøyaktig 2 kron forekommer} = {KKM, KMK, MKK}

[ ]:
```
def estimer_sannsynlighet_for_2_kron_fra_3_kast(antall_eksperimenter: int):

    # live-kodet

estimer_sannsynlighet_for_2_kron_fra_3_kast(antall_eksperimenter=1000)
```

1.13.1 Øvelse: partall ved terningkast

Skriv en funksjon som estimerer sannsynligheten for at etter å ha kastet en rettferdig 6-sidet terning, kommer det opp et partall. Hva er det tilfeldige eksperimentet her? Hva er de mulige utfallene? Hva er hendelsen det spørres om og hvilke utfall tilsvarer denne hendelsen?

[ ]:
```python
def estimer_partall_sannsynlighet() -> float:
    pass
```

1.14 Sannsynlighetsfordeling

En sannsynlighetsfordeling gir sannsynligheten for forekomsten av forskjellige mulige utfall:

- Hvis vi kaster en peng, hva er sannsynligheten for at vi får kron: P(kron), og hva er sannsynligheten for at vi får mynt: P(mynt)
- Hvis vi kaster en peng 3 ganger, hva er sannsynligheten for at vi får 0 kron: P(K0), 1 kron: P(K1), 2 kron: P(K2), eller 3 kron: P(K3)

Sannsynlighetsfordeling er en matematisk funksjon som kobler ikke-overlappende hendelser til sannsynligheter som summerer til 1.

NB: En diskret sannsynlighetsfordeling kan sees som en **ordbok** i Python, hvor de ikke-overlappende hendelsene er nøkler og sannsynlighetene som tilsvarer hendelsene er verdier.

[ ]:
```python
from itertools import product

def estimer_sannsynlighetsfordeling(antall_kast, antall_eksperimenter):

    # live-kodet:

sannsynlighetsfordeling = estimer_sannsynlighetsfordeling(antall_kast=3,␣
 ↪antall_eksperimenter=100)

print(f"Sannsynlighetsfordeling: {sannsynlighetsfordeling}")
```

Denne måten å få sannsynlighetsfordelinger gjennom simuleringer kalles **Monte Carlo-simulering**.

1.14.1 Vise resultatene av simuleringen grafisk

[ ]:
```python
n = 3
antall_eksperimenter = 100

sannsynlighetsfordeling = estimer_sannsynlighetsfordeling(n,␣
 ↪antall_eksperimenter)

plott_hendelse_sannsynligheter(sannsynlighetsfordeling)
```

1.14.2 Øvelse: sannsynligheten for å få minst 1 kron

Hva er sannsynligheten for å få minst 1 kron fra 3 kast? Skriv koden for å estimere den.

[ ]:
```python

```

```
def estimer_sannsynlighet_for_minst_1_kron_fra_3_kast(antall_eksperimenter:␣
 ↪int):
  pass

estimer_sannsynlighet_for_minst_1_kron_fra_3_kast(antall_eksperimenter=1000)
```

1.15 Eksempel: sannsynlighet for minst 1, 2 og 3 kron

Skriv nå koden for å estimere sannsynligheten for å få minst 1 kron, minst 2 kron, og minst 3 kron.

[ ]:
```
def estimer_sannsynlighet_for_minst_m_kron(antall_eksperimenter: int):
    antall_kast = 3
    fikk_minst_n_kron_tellinger = {i: 0 for i in range(antall_kast+1)}
    for _ in range(antall_eksperimenter):
        utfall = kast_peng_n_ganger(n=antall_kast)
        kron_antall = utfall.count('kron')
        for minst_n in fikk_minst_n_kron_tellinger.keys():
            if minst_n >= kron_antall:
                fikk_minst_n_kron_tellinger[minst_n] += 1

    sannsynligheter = {minst_n: telling / antall_eksperimenter for minst_n,␣
 ↪telling in fikk_minst_n_kron_tellinger.items()}
    return sannsynligheter

print(estimer_sannsynlighet_for_minst_m_kron(antall_eksperimenter=1000))
```

Et eksempel på output fra forrige funksjon: {0: 0.111, 1: 0.491, 2: 0.877, 3: 1.0} - **dette er ikke en gyldig sannsynlighetsfordeling!**

Hvorfor?

1.16 Utfall vs hendelse

"Minst n kron" er en **hendelse** - den inneholder flere utfall: (kron, mynt, kron) og (kron, mynt, mynt) er to forskjellige utfall, men samme hendelse "minst 1 kron".

"Minst n kron" **kan ikke være et utfall** - vi kan ikke ha en sannsynlighetsfordeling for 'minst n kron', for n=1,2,3 fordi disse ikke er distinkte og gjensid utelukkende - dersom vi får (kron, mynt, kron) tilhører dette både "minst 1" og "minst 2" kron-hendelsene.

1.17 Utfall vs hendelse

Utfall og hendelser er to abstraksjonsnivåer som brukes til å beskrive en tilfeldig prosess.

Utfall er elementære og lister opp alle de forskjellige mulighetene. Sannsynlighetene for alle utfall summerer til 1, ettersom ett utfall må forekomme som resultat av et eksperiment.

Hendelser er definert som sett av utfall som er av interesse. Sannsynligheter for hendelser er summen av sannsynlighetene for alle utfall som er en del av hendelsen.

1.18 Sannsynlighetsaksiomer

Sannsynlighetsaksiomer gjelder for enhver sannsynlighet per definisjon:

- Sannsynligheten for enhver hendelse A er alltid ikke-negativ: P(A) >= 0
- For to hendelser A og B som ikke overlapper (disjunkte hendelser), er sannsynligheten for at begge skjer summen av deres individuelle sannsynligheter: P(A U B) = P(A) + P(B)
- Sannsynligheten for hele utfallsrommet (av alle mulige utfall) er lik 1.

1.19 Tilbake til menti-eksempelet

Hvis vi kaster en rettferdig peng 10 ganger, hva er sannsynligheten for at vi får 5 kron og 5 mynt? Skriv koden for å simulere dette eksperimentet og empirisk estimere sannsynligheten.

Noen refleksjonspunkter: - Hva er hendelsen i dette tilfellet? - Hva er mulige utfall?

Skriv koden her og estimer sannsynligheten for å få nøyaktig 5 kron og 5 mynt i 10 myntkast:

[ ]:
```
hendelse_sannsynligheter = estimer_sannsynlighetsfordeling(antall_kast=10,␣
↪antall_eksperimenter=10000)

print("Sannsynligheter for forskjellige hendelser:")
print(hendelse_sannsynligheter)

print(hendelse_sannsynligheter[5])
```

1.20 Tilbake til Menti

Lenke: https://www.menti.com/4ii97w7bfi

Kode for menti.com: 3269 1977

1.21 Empiriske og analytiske løsninger

Gjennom simuleringer kan vi estimere sannsynligheten for en hendelse.

Vi kan også ofte matematisk beskrive sannsynlighetsfordelingen og beregne den ved hjelp av en formel.

Imidlertid, i noen tilfeller vet vi enten ikke formelen eller fordelingen ville vært for vanskelig å beregne, da går vi tilbake til (Monte Carlo) simuleringer.

1.22 Empiriske og analytiske løsninger

Sannsynlighetsmassefunksjon: en funksjon som gir sannsynligheten for et utfall eller en hendelse (f.eks. sannsynligheten for å få 2 kron i 3 myntkast).

Noen tilfeldige prosesser forekommer i mange forskjellige situasjoner. For eksempel kan vi kaste en mynt for å få kron/mynt, en student kan ta en eksamen hvor de består/stryker, eller en diagnostisk test som resulterer i en positiv/negativ diagnose - alle disse har en spesifikk “type tilfeldighet”.

1.23 Bernoulli-fordeling

Denne fordelingen modellerer ethvert ja/nei (1/0) spørsmål hvor ja (1) velges med sannsynlighet p.

Eksempel: et myntkast gir kron med sannsynlighet p og mynt med sannsynlighet 1-p.

[ ]:
```
from util import plott_bernoulli_eksempler

plott_bernoulli_eksempler()
```

1.24 Binomialfordeling

Binomialfordelingen beskriver antall suksesser k (f.eks. å få kron) i en sekvens av n forsøk (f.eks. i n myntkast) hvor sannsynligheten for suksess i hvert forsøk er p.

Sannsynligheten for å få k suksesser i n uavhengige forsøk hvis sannsynligheten for suksess er p kan beregnes ved hjelp av følgende sannsynlighetsmassefunksjon:

[ ]:
```
# Hva er sannsynligheten for å få nøyaktig én kron i tre kast av en rettferdig␣
 ↪mynt?

n = 3 # antall forsøk (kast)
k = 1 # antall suksesser (kron)
p = 0.5 # sannsynlighet for suksess i ett forsøk for en rettferdig mynt

# beregner formelen:

import math

sannsynlighet_en_kron_fra_formel = round(math.factorial(n) // (math.
 ↪factorial(k) * math.factorial(n-k)) * p**1 * (1-p)**2, 3)

from scipy.stats import binom

sannsynlighet_en_kron_fra_bibliotek = round(binom.pmf(k, n, p), 3)

print(f"Fra manuell beregning av formelen: {sannsynlighet_en_kron_fra_formel}")
print(f"Ved bruk av implementering fra scipy-biblioteket:␣
 ↪{sannsynlighet_en_kron_fra_bibliotek}")
```

[ ]:
```
from util import plott_binomialfordeling_eksempel

plott_binomialfordeling_eksempel()
```

2 Empiriske estimater vs analytiske sannsynligheter

Tidligere simulerte vi å kaste en mynt 3 ganger for å få sannsynligheten for å få forskjellige antall kron. Her er en eksempelimplementering som estimerer sannsynlighetene for disse 4 hendelsene.

2.0.1 Empiriske estimater

[ ]:
```
def kast_peng_n_ganger(p_kron: float, n: int):
    return ['K' if random.random() <= p_kron else 'M' for _ in range(n)]

def simuler_hendelse_sannsynligheter(p_kron: float, n: int,␣
 ↪antall_eksperimenter: int):
    hendelse_tellinger = {i: 0 for i in range(n+1)}

    for eksperiment in range(antall_eksperimenter):
        utfall = kast_peng_n_ganger(p_kron, n)
        hendelse_tellinger[utfall.count('K')] += 1

    sannsynlighetsfordeling = {f'K{hendelse}': telling / antall_eksperimenter␣
 ↪for hendelse, telling in hendelse_tellinger.items()}

    return sannsynlighetsfordeling

hendelse_sannsynligheter_simulering = simuler_hendelse_sannsynligheter(p_kron=0.
 ↪5, n=3, antall_eksperimenter=100)
print(hendelse_sannsynligheter_simulering)
```

2.0.2 Analytiske sannsynligheter

Dette kan også beregnes ved hjelp av en formel for binomialfordeling (sannsynligheten for å få k suksesser i n uavhengige forsøk når sannsynligheten for suksess er p):

[ ]:
```
import math

def kron_antall_sannsynlighet(p_kron, n, antall_kron):
    return math.factorial(n) / (math.factorial(antall_kron) * math.factorial(n␣
 ↪- antall_kron)) * (p_kron**antall_kron) * ((1-p_kron)**(n-antall_kron))

hendelse_sannsynligheter_fra_formel = {
    f'K{antall_kron}': kron_antall_sannsynlighet(p_kron=0.5, n=3,␣
 ↪antall_kron=antall_kron) for antall_kron in range(4)
}

print(hendelse_sannsynligheter_fra_formel)
```

2.0.3 Empirisk vs. analytisk sannsynlighet

Vi kan også vise de empirisk estimerte sannsynlighetene side om side med de vi beregnet analytisk:

[ ]:
```
from util import plott_sannsynlighet_sammenligning

plott_sannsynlighet_sammenligning(hendelse_sannsynligheter_simulering,␣
↪hendelse_sannsynligheter_fra_formel)
```

2.1 Kategoriske fordelinger

Når man kaster en terning én gang, er det 6 like sannsynlige mulige utfall: k=6, med sannsynligheter for hver p=1/6.

Kategorisk fordeling: generalisering av Bernoulli-fordelingen med k mulige kategorier, med sannsynlighet for hver kategori definert separat.

2.2 Multinomialfordeling

Multinomialfordelingen modellerer sannsynligheten for å få hver side av en terning når man kaster den n ganger.

I hvert av n forsøk får vi nøyaktig én av k kategorier med fast sannsynlighet for den kategorien.

Forsøkene er uavhengige.

2.3 Uniform fordeling

Hvis vi kaster en rettferdig terning, forventer vi at alle sidene kommer opp like ofte -> hver side har samme sannsynlighet.

I en uniform fordeling er alle utfall like sannsynlige.

2.3.1 Uniform fordeling

[ ]:
```
from util import plott_uniform_terning

plott_uniform_terning()
```

2.4 Hva skjer hvis vi kaster en mynt mange ganger: tilnærming til kontinuerlige fordelinger

[ ]:
```
p_kron, n, antall_eksperimenter = 0.5, 20, 1000
hendelse_sannsynligheter_sim = simuler_hendelse_sannsynligheter(p_kron, n,␣
↪antall_eksperimenter)
plott_hendelse_sannsynligheter(hendelse_sannsynligheter_sim)
```

2.4.1 Øvelse: Monopol

I spillet Monopol flytter man sin brikke rundt et brett med 40 felt, ved å kaste et par terninger hver tur. Hva er sannsynligheten for å fullføre din første runde på din femte tur?

Alternativt:

Hva er sannsynligheten for at summen av verdier passerer 40 etter å ha trukket et utvalg fra et par (2) randint(1,6) kall nøyaktig 5 ganger?

Skriv en kode for å simulere dette og estimere sannsynligheten.

[ ]:
```
def estimer_monopol_sannsynlighet(antall_eksperimenter: int = 1000) -> float:
  pass

estimer_monopol_sannsynlighet()
```

2.5 Generell prosedyre for simulering

1. *Problemspesifikk*: skriv kode for å simulere det ønskede eksperimentet
2. *Generisk*: for å simulere mange ganger, bruk Monte Carlo (MC) med en for-løkke
3. *Generisk*: Tell suksesser - hvor mange ganger hendelsen inntreffer (på tvers av MC-løkken)
4. *Generisk*: Beregn sannsynlighet som antall suksesser delt på antall MC-iterasjoner

For å i stedet beregne en full sannsynlighetsfordeling:

- Steg 3: tell i stedet hvor mange ganger hvert forskjellig utfall/hendelse inntreffer
- Steg 4: del antallet av hvert utfall/hendelse på antall MC-iterasjoner for å få sannsynligheten for hver hendelse

2.6 Betinget sannsynlighet

Betinget sannsynlighet: en måte å resonnere om utfallet av et eksperiment basert på delvis informasjon

Vi vet at utfallet er innenfor en hendelse B - vi ønsker å kvantifisere sannsynligheten for at utfallet også tilhører en annen hendelse A

2.7 Beregne sannsynligheten for en bokstav i en tekst

[ ]:
```
import re
```

```
original_tekst = """To be, or not to be, that is the question, Whether 'tis␣
 ↪nobler in the mind to suffer The slings and arrows of outrageous fortune,Or␣
 ↪to take arms against a sea of troubles,And by opposing end them? To die: to␣
 ↪sleep;No more; and by a sleep to say we endThe heart-ache and the thousand␣
 ↪natural shocksThat flesh is heir to, 'tis a consummationDevoutly to be␣
 ↪wish'd. To die, to sleep;To sleep: perchance to dream: ay, there's the rub;
 ↪For in that sleep of death what dreams may comeWhen we have shuffled off␣
 ↪this mortal coil,Must give us pause: there's the respectThat makes calamity␣
 ↪of so long life;For who would bear the whips and scorns of time,The␣
 ↪oppressor's wrong, the proud man's contumely,The pangs of despised love, the␣
 ↪law's delay,The insolence of office and the spurnsThat patient merit of the␣
 ↪unworthy takes,When he himself might his quietus makeWith a bare bodkin? who␣
 ↪would fardels bear,To grunt and sweat under a weary life,But that the dread␣
 ↪of something after death,The undiscover'd country from whose bournNo␣
 ↪traveller returns, puzzles the willAnd makes us rather bear those ills we␣
 ↪haveThan fly to others that we know not of?Thus conscience does make cowards␣
 ↪of us all;And thus the native hue of resolutionIs sicklied o'er with the␣
 ↪pale cast of thought,And enterprises of great pith and momentWith this␣
 ↪regard their currents turn awry,And lose the name of action.--Soft you now!
 ↪The fair Ophelia! Nymph, in thy orisonsBe all my sins remember'd."""
tekst = re.sub(r"[.,:\' ;\n\-?!]*", "", original_tekst).lower()

# print(original_tekst)
# print(tekst)

def estimer_marginal_sannsynlighet(bokstav, tekst):

    # live-kodet
    antall = tekst.count(bokstav)
    sannsynlighet = round(antall / len(tekst), 3)

    return sannsynlighet

def estimer_betinget_sannsynlighet(bokstav, forrige_bokstav, tekst):

    # live-kodet
    antall_forrige_bokstav = tekst.count(forrige_bokstav)
    antall_begge_bokstaver = tekst.count(forrige_bokstav + bokstav)

    betinget_sannsynlighet = round(antall_begge_bokstaver /␣
 ↪antall_forrige_bokstav, 3)

    return betinget_sannsynlighet

print(estimer_marginal_sannsynlighet("a", tekst))

print(estimer_betinget_sannsynlighet("a", "h", tekst))
```

2.7.1 Øvelse: Flere kron enn mynt i rettferdige myntkast

Vi kaster en rettferdig mynt 3 ganger. Hva er sannsynligheten for at flere kron enn mynt kommer opp hvis det første kastet er kron?

[ ]:
```
def estimer_flere_kron_sannsynlighet(antall_eksperimenter):
  pass

estimer_flere_kron_sannsynlighet(1000)
```

2.8 Problemet fra begynnelsen

"Et stort antall kort fra mange forskjellige kortstokker er spredt utover gaten. Hvor mange kort må man se på før man har sett 52 forskjellige kort (alle de ulike som finnes i en kortstokk)?"

- Hvordan ville du løst dette problemet nå?
- Hvordan formulere dette problemet i lys av sannsynlighetsteori?

[ ]:
```

```

2.8.1 Øvelse: sum av sider ved terningkast

Du kaster en terning tre ganger. Hva er sannsynligheten for at summen av sidene er mindre enn 12?

[ ]:
```
def estimer_sannsynlighet_for_sum(mål_sum, antall_eksperimenter):
    pass

estimer_sannsynlighet_for_sum(12, 100)
```

2.8.2 Øvelse: serie med kortspillkamper

Tre kortspillere spiller en serie med kamper. Sannsynligheten for at spiller 1 vinner et hvilket som helst spill er 30%, sannsynligheten for at spiller 2 vinner er 50% og sannsynligheten for at den tredje spilleren vinner er 20%. Hvis de spiller 6 spill, hva er sannsynligheten for at spiller 1 vinner minst 2 spill?

[ ]:
```
def estimer_vinner_sannsynlighet(spiller1_p, spiller2_p, spiller3_p,␣
↪antall_spill, antall_eksperimenter):
    pass

estimer_vinner_sannsynlighet(0.3, 0.5, 0.2, 6, 10000)
```

3 Løsninger for øvelsene

3.0.1 Øvelse: vektet mynt

[ ]:
```
def vektet_mynt(p_kron: float = 0.3) -> str:
    utfall = 'kron' if random.random() < p_kron else 'mynt'
    return utfall

vektet_mynt()
```

3.0.2 Øvelse: partall ved terningkast

[ ]:
```
def terning():
    return random.choice(list(range(1, 7)))

def estimer_partall_sannsynlighet(antall_eksperimenter: int = 1000) -> float:
    partall_skjedde = 0

    for _ in range(antall_eksperimenter):
        utfall = terning() # spesifikt tall som kom opp
        partall_skjedde += int(utfall % 2 == 0) # tell en hendelse til hvis␣
 ↪tallet er partall

    return partall_skjedde / antall_eksperimenter

estimer_partall_sannsynlighet()
```

3.0.3 Øvelse: sannsynligheten for å få minst 1 kron

[ ]:
```
def estimer_sannsynlighet_for_minst_1_kron_fra_3_kast(antall_eksperimenter:␣
 ↪int):
    fikk_minst_1_kron_antall = 0
    for _ in range(antall_eksperimenter):
        utfall = kast_peng_n_ganger(n=3, p_kron=0.5)
        fikk_minst_1_kron_antall += int(utfall.count('kron') >= 2)

    sannsynlighet_for_minst_1_kron = fikk_minst_1_kron_antall /␣
 ↪antall_eksperimenter
    return sannsynlighet_for_minst_1_kron

estimer_sannsynlighet_for_minst_1_kron_fra_3_kast(antall_eksperimenter=1000)
```

3.0.4 Øvelse: Monopol

[ ]:
```python
def estimer_monopol_sannsynlighet(antall_eksperimenter: int) -> float:
    suksesser = 0

    for eksperiment in range(antall_eksperimenter):
        utfall = [[terning() for _ in range(2)] for _ in range(5)]
        verdi = sum(sum(terninger) for terninger in utfall)
        if verdi > 40:
            suksesser += 1

    return suksesser / antall_eksperimenter

estimer_monopol_sannsynlighet(1000)
```

3.1 Øvelse: Flere kron enn mynt i rettferdige myntkast

[ ]:
```python
def estimer_flere_kron_sannsynlighet(antall_eksperimenter: int) -> float:

    flere_kron_totalt = 0
    kron_først = 0

    for eksperiment in range(antall_eksperimenter):
        utfall = [mynt() for _ in range(3)]
        if utfall[0] == 'kron':
            kron_først += 1
            if utfall.count('kron') > utfall.count('mynt'):
                flere_kron_totalt += 1
    return round(flere_kron_totalt / kron_først, 3)


estimer_flere_kron_sannsynlighet(1000)
```

3.2 Øvelse: sum av sider ved terningkast

[ ]:
```python
def estimer_sannsynlighet_for_sum(mål_sum, antall_eksperimenter):

    hendelser = ['mindre_enn_mål', 'større_lik_mål']
    sannsynlighetsfordeling = {hendelse: 0 for hendelse in hendelser}
    for eksperiment in range(antall_eksperimenter):
        utfall = [terning() for _ in range(3)]
        if sum(utfall) < mål_sum:
            sannsynlighetsfordeling['mindre_enn_mål'] += 1
        else:
            sannsynlighetsfordeling['større_lik_mål'] += 1
```

```
    sannsynlighetsfordeling = {hendelse: telling / antall_eksperimenter for␣
 ↪hendelse, telling in sannsynlighetsfordeling.items()}

    return sannsynlighetsfordeling

estimer_sannsynlighet_for_sum(12, 100)
```

3.3 Øvelse: serie med kortspillkamper

[ ]:
```
def estimer_vinner_sannsynlighet(spiller1_p, spiller2_p, spiller3_p,␣
 ↪antall_spill, antall_eksperimenter):

    hendelser = ['spiller1_vinner_2_pluss', 'spiller1_vinner_1_minus']
    sannsynlighetsfordeling = {hendelse: 0 for hendelse in hendelser}

    for eksperiment in range(antall_eksperimenter):
        spiller1_seire = 0
        for spill in range(antall_spill):
            spiller1_seire += random.random() < spiller1_p

        if spiller1_seire >= 2:
            sannsynlighetsfordeling['spiller1_vinner_2_pluss'] += 1
        else:
            sannsynlighetsfordeling['spiller1_vinner_1_minus'] += 1

    sannsynlighetsfordeling = {hendelse: telling / antall_eksperimenter for␣
 ↪hendelse, telling in sannsynlighetsfordeling.items()}

    return sannsynlighetsfordeling


estimer_vinner_sannsynlighet(0.3, 0.5, 0.2, 6, 10000)
```