Mappeoppgave 2

Informasjon om oppgaven

Når du besvarer oppgaven, husk:

- les oppgaveteksten nøye
- kommenter koden din
- sett navn på akser og lignende i figurene
- skriv hvor du har hentet kodesnutter fra, hvis du gjør det
- bruk engelske variabelnavn og vær konsistent med hvordan du bruker store og små bokstaver
- bruk mest mulig funksjoner for ting som kan repeteres
- En kort kode kan være en bra kode, så ikke gjør det mer komplisert enn det spørres om.

Du kan få full pott uten å svare på oppgaven som er markert "ekstrapoeng". Du blir likevel belønnet for denne (dvs. hvis du har noen feil og får 45 poeng totalt, så kan du få en høyere poengsum hvis du også har svart på "ekstrapoeng".

Innlevering av oppgavene

Du skal levere begge mappene samtidig (det vil si denne oppgaven og mappe 1). Innleveringsfristen er 6 desember kl 13:00. Begge oppgavene skal leveres i github (som jupyter-fil) og wiseflow (som PDF). Bruk navnet "SOK-1003-eksamen-2022-mappe2" på filene.

- For github: Husk å gi meg (brukernavn "okaars") tilgang til github-repositoriet deres. Hvis dere har satt repositoriet til public (anbefales ikke), må dere dele lenken til dette på ole.k.aars@uit.no
- For wiseflow: En person fra hver gruppe (for hver mappeoppgave), leverer inn. Ved innlevering kan du krysse av hvem som er på gruppen din

Se generell informasjon om hvordan man leverer oppgaven her.

NB!: En person fra gruppa må fylle ut dette skjemaet for å melde om hvem som er på gruppa. Dere vil i etterkant motta en epost om tidspunkt for presentasjon.

Presentasjon

Presentasjonen innebærer en kort gjennomgang av oppgaven (10-15 min) etterfulgt av kommentarer fra meg (10-15 min). Alle gruppemedlemmer skal bidra til presentasjonen. Det er anbefalt å laste opp besvarelsen på github forut for presentasjonen (helst to dager før) slik at jeg har mulighet til å lese gjennom. Dere vil ha mulighet til å endre besvarelsen etter presentasjonen, frem til endelig innlevering 6 desember.

Oppgave 1 (10 poeng)

a) Vi skal spille et spill der vi kaster en terning 6 ganger. Lag en funksjon med "for-løkke" som printer alle terningene som har blitt kastet. Du kan bruke `np.random.randint()` til å lage tilfeldige tall

In [ ]:
```
import numpy as np

for i in range(6):
    dice = np.random.randint(1,7)
    print(dice)
```

b) Juster den samme funksjonen slik at den lagrer tallene i en liste før den printer ut selve listen. Dere kan kalle denne listen for `lot_numbers`. Dere kan vurdere å bruke `append()` som del av funksjonen.

In [ ]:
```
lot_numbers = []

for i in range(6):
    dice = np.random.randint(1,7)
    lot_numbers.append(dice)

print(lot_numbers)
```

c) Juster den samme funksjonen slik at den har to argument. Disse argumentene er to terningverdier som du "tipper" blir kastet. Bruk `if`, `else` og `elif` til å generere vinnertall. Resultatet fra funksjonen skal printe ut ulike setninger avhengig av om man får 0, 1 eller 2 rette. Setningene velger du selv, men de skal inneholde tallene som du tippet, og tallene som ble trukket.

In [ ]:
```
lot_numbers = []

for i in range(6):
    dice = np.random.randint(1,7)
    lot_numbers.append(dice)

print(lot_numbers)

def guess(guess1, guess2):
    if guess1 in lot_numbers and guess2 in lot_numbers:
        print("Du gjettet begge riktig!")
    elif guess1 in lot_numbers or guess2 in lot_numbers:
        print("Du gjettet en riktig!")
    else:
        print("Du gjettet ingen riktige")

guess(1,2)
```

Oppgave 2 (10 poeng)

a) Du har nå begynt å spille lotto i stedet, og satser alt på ett vinnertall. Lag en while-løkke som printer ut tall helt til du har trukket riktig tall (som du definerer selv). For enkelthets skyld kan du begrense utfallsrommet av trekningene til mellom 0-30.

In [ ]:
```
dy,y=0,0

while y!=30:
    y=dy
    dy=np.random.randint(1,31)
    print(y)

#Gjetter tilfeldige tall mellom 1 og 30 frem til den gjetter 30.
```

b) Lag et plot av den while-løkken du nettopp lagde. Man blir belønnet om man;

- bruker `scatter`;
- lager plottet dynamisk (dvs at hver trekning vises hver for seg, og at x-aksen endrer seg etter en gitt verdi);
- viser hvor når siste trekningen blir gjort (dvs at den vises kun når du har trukket vinnertallet).

Avhengig av hvordan du lager figuren din kan du får bruk for å importere pakkene `Ellipse`, `display`, `clear_output`.

In [ ]:
```
import matplotlib.pyplot as plt
from matplotlib.patches import Ellipse
from IPython.display import display, clear_output
fig, ax = plt.subplots(figsize=(20, 10))
ax.set_ylim([0,30])
ax.set_xlim([0,100])

x,y,dy=0,0,0

ypath=[]
xpath=[]

while x!=30:
    x+=1
    y=dy
    dy=np.random.randint(1,31)

    xpath.append(x)
    ypath.append(y)


    ax.plot(xpath,ypath,label='Lotto')
    ax.scatter(xpath,ypath)
    ax.legend(loc='upper left',frameon=False,fontsize=30)


    c=Ellipse((x, y), 1,1, color='r')
    ax.add_patch(c)


    ax.text(x+0.1,y+3,np.round(y,1))


    display(fig)


    ax.cla()
    ax.set_ylim([0,30])
    ax.set_xlim([0,100])
    clear_output(wait = True)
```

c) Ekstrapoeng: gjør det samme som i (b), men lag et histogram som vises ved siden av. Dette histogrammet skal vise hvor mange ganger de ulike tallene ble trekt. Bruk `plt.hist` til dette. Husk at du må definere figur og akseobjekt først.

In [ ]:
```

```

Oppgave 3 (20 poeng)

En bedrift produserer biler. Produktfunksjonen til bedriften defineres slik $f(L,a,R) = 2RL^a$, hvor:

- `L` er arbeidskraft,
- `a` er produktiviteten til arbeiderne og
- `R` er antall robotmaskiner

a) Lag en formel for produktfunksjonen til bedriften og plot den grafisk med ulike verdier av `L` på x-aksen. Anta `a=0.6` og `R=2`

In [ ]:
```
def f(R,L,a):
    return 2*R*np.arange(0,L+1)**a
```

In [ ]:
```
produkt = f(2,200,0.6)
plt.plot(produkt)
plt.ylabel("Produksjon")
plt.xlabel("Arbeidere")
```

b) anta at profittfunksjonen til denne bedriften er $profit = f(L,a,R)p - wL - cR - K$, hvor

- `w` er månedslønnen til arbeiderne,
- `c` er kostnaden for robotmaskinene
- `K` er faste kostnader
- `p` er utsalgsprisen på bilene.

Anta `a=0.6`, `R=6`, `p=300 000`, `w=100 000`, `c= 1 000 000` og `K=90 000 000`. Plot profittfunksjonen figurativt for antall arbeidere (`L`) mellom 0 og 10 000. Vis profitten i millioner (dvs at du må dele på 1 000 000)

In [ ]:
```
def g(L, a, R, p, w, c, K):
    return (2*R*np.arange(0,L+1)**a)*p-w*np.arange(0,L+1)-c*R-K

g(10000, 0.6, 6, 300000, 100000, 1000000, 90000000)

profit1 = g(10000, 0.6, 6, 300000, 100000, 1000000, 90000000)/1000000
plt.plot(profit1)
plt.ylabel("Profitt")
plt.xlabel("Arbeidere")
```

c) Plot profittfunksjonen for antall robostmaskiner `R=[3, 6, 9]` i samme plot (dvs at tre profittfunksjoner vises sammen). Bruk av "for loops" for å gjøre dette belønnes

In [ ]:
```
r3 = g(10000, 0.6, 3, 300000, 100000, 1000000, 90000000)/1000000
r6 = g(10000, 0.6, 6, 300000, 100000, 1000000, 90000000)/1000000
r9 = g(10000, 0.6, 9, 300000, 100000, 1000000, 90000000)/1000000
plt.plot(r3, label="3 roboter")
plt.plot(r6, label="6 roboter")
plt.plot(r9, label="9 roboter")
plt.ylabel("Profitt")
plt.xlabel("Arbeidere")
plt.legend(loc="best")
```

d) finn profittmaksimum og optimal antall arbeidere ved hjelp av derivasjon med samme forutsetninger som i (1b). Bruk `sympy`-pakken til dette

In [ ]:
```
#pakker du kan få bruk for
import sympy as sp
from sympy.solvers import solve

L, a, R, p, w, c, K=sp.symbols("L a R p w c K")
```

In [ ]:
```
#Fikser funksjonen slik at den fungerer med sympy
def profitt(L, a, R, p, w, c, K):
    return (2*R*L**a)*p-w*L-c*R-K

#Finnder den deriverte til profittfunksjonen
d_profitt=sp.diff(profitt(L, a, R, p, w, c, K),L)

#Finner punktet der den deriverte er lik 0.
foc=sp.Eq(d_profitt,0)

#Løser for L som tilfredstiller likningen
L_max=solve(foc,L)[0]

#Fyller inn verdiene våres
num_dict={a:0.6, R:6, p:300000, w:100000, c:1000000, K:90000000}

#Finner optimal antall arbeidere
L_max.subs(num_dict)

#Oppdaterer listen med det optimale tallet
num_dict[L]=L_max.subs(num_dict)

#Finner profittmaksimum
profit_max_num=float(profitt(L, a, R, p, w, c, K).subs(num_dict))

print("Optimal antall arbeidere: " + str(L_max.subs(num_dict)))
print("Profittmaksimum: " + str(profit_max_num))
```

e) vis figurativt med bruk av `fill_between` arealet hvor man taper penger (i rødt) og hvor man tjener penger (i grønt). Marker også profittmaksimum og antall arbeidere i profittmaksimum - gjerne ved bruk av `vlines`. Bruk ellers samme forutsetninger for argumentene som i oppgave (1b)

In [ ]:
```

```

f) Plot nå to figurer sammen der du viser hva optimal antall arbeidere gir i profitt (slik som i (2e)) og produksjon av antall biler (som du får fra produktfunksjonen). Marker optimum med vlines. Ha grafen med profittfunksjonen over grafen med produktfunksjonen. Du kan bruke `fig, (ax1, ax2) = plt.subplots(2)` når du skal gjøre dette.

Hint: Du kan finne antall biler som blir produsert ved å bruke antall arbeidere i profittmaksimum, i produktfunksjonen.

In [ ]:
```

```

Oppgave 4 (10 poeng)

I denne oppgaven skal vi hente ut et datasett fra eurostat på investeringer i hosholdningen. Bruk koden under til å hente ut dataene.
NB!: Husk at dere må ha innstallert pakken `eurostat`. Dette gjør dere med å åpne "Terminal" og kjøre `pip install eurostat`.

In [ ]:
```
import eurostat

inv_data = eurostat.get_data_df('tec00098')
```

a) Bytt navn på kolonnen "geo\time" til "country" ved bruk av en av kodene under. Fjern så alle kolonner utenom "country" og alle årstallene.
NB!: Noen vil få en ekstra første kolonne som heter "freq" eller noe annet. Da må dere bruke versjon 2 av koden under.

In [ ]:
```
inv_data.columns = ['freq', 'unit',  'sector',  'na_item', 'country'] + list(range(2010, 2022)) #v2
inv_data = inv_data.drop(columns=['freq', 'unit', 'sector', 'na_item'])
```

b) fjern radene med nan verdi. Sett deretter indeksen til "country".
Hint: En metode er å bruke `isna()` og `any()` over radaksene (dvs. `axis=1`)

In [ ]:
```
inv_data = inv_data.dropna(axis=0)
inv_data = inv_data.set_index('country')

inv_data
```

c) Lag et nytt datasett hvor du kun har med de nordiske landene (dvs. "NO", "SE", "DK", "FI"). Det kan være nyttig å bruke `isin` til dette. Bytt så om på kolonner og rader ved hjelp av `transpose`.

In [ ]:
```
inv_nordic = inv_data.loc[inv_data.index.isin(["NO", "SE", "DK", "FI"])]
inv_nordic = inv_nordic.transpose()

inv_nordic
```

d) Lag en ny kolonne som du kaller "mean". Denne skal være gjennomsnittet av alle de nordiske landene for hvert av årene (dvs at du må ta gjennomsnittet over radene). Plot så dette og kall y-aksen for "investering"

In [ ]:
```
inv_nordic['mean'] = inv_nordic.mean(axis=1)

inv_nordic
```

In [ ]:
```
inv_nordic["mean"].plot()
plt.ylabel("Investering")
plt.xlabel("År")
plt.show()
```

In [ ]:
```

```