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

dt = 0.5 # mengde per steg
t = np.linspace(0,3,7)
u = np.zeros(len(t))
u[0] = 1
f = lambda x: x*((1/2) - x) # original differential funksjon

for n in range(len(t) - 1):
    u[n+1] = u[n] + dt * f(u[n])

plt.plot(t,u, label='forward')

xanalytisk = lambda t: np.exp(t/2) / (2*np.exp(t/2)+ - 1) # analytisk funksjon
t2 = np.linspace(0,3,1000)
plt.plot(t2, xanalytisk(t2), label='analytisk')

u_mid = np.zeros(len(t))
u_mid[0] = 1

for n in range(len(t) - 1):
    dt = t[n+1] - t[n]
    dt_2 = dt/2.0
    k_1 = f(u_mid[n])
    k_2 = f(u_mid[n] + dt_2*k_1)
    u_mid[n+1] = u_mid[n] + dt*k_2

plt.plot(t, u_mid, label='midpoint')
plt.legend()
plt.show()

"""
Løsningen x(t) er begrenset fordi for lave x verdier så blir uttrykket 1 / 2 - 1.
Det vil si at x(0) = 1. Etter hvert som t verdien øker så øker også e^t/2 verdiene kraftig.
Da blir -1 under brøkstreken ubetydelig og funksjonen konvergerer mot 1/2.
Dette fordi e^t/2 går mot uendelig, og uendelig / 2 * uendelig er lik 1/2.

Det er korrekt å anta at den samme begrensningen gjelder for eulers metode om vi øker antall steg.
Dersom ett ledd i steget vårt blir til en halv. Så blir utregningen for neste steg lik:
1/2 * (1/2 - 1/2), som blir 0.
Da ser vi fra utregningen til steg at neste ledd u+1 er lik u + 0.
Dvs. at vår tilnærming med eulers metode også konvergerer mot 1/2.
"""
```