```
# Dette script skal med tiden implementere en rÃ¦kke statistiske funktioner

import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as ss

# Denne funktion implementerer den funktionelle metode til at lave
# fejlpropagering. Funktionen tager en vilkÃ¥rlig fitte funktion samt dennes
# estimerede parametre samt covariansmatrice.
# Den beregner derefter fejlen for en given x-vÃ¦rdi

def propagation_function(x, f, popt, pcov):
    f_error = 0

# Standard afvigelser gemmes

    err =  list(np.sqrt(np.diagonal(pcov)))
    for i in range(len(err)):

# funktionen med standard afvigelsen pÃ¥ den i'te parameter konstrueres

        j = popt[:i] + [popt[i] + err[i]] + popt[i+1:]
        f_error += (f(x, *j)-f(x, *popt))**2
    return np.sqrt(f_error)

################################################
#
#   VÃ¦r venlig at ignorerer det der stÃ¥r herunder.
#
################################################

# f(x) = Ax + B
# Hvad er fejlen pÃ¥ f(x)?

# Lad z_A vÃ¦re fejlen pÃ¥ pÃ¥ A og z_B vÃ¦re fejlen pÃ¥ B.
# SÃ¥ er fejlen pÃ¥ f(x).

# z_f(x)**2 = (f((A+z_a)*x +B)-f(A*x+B))**2 + (f(A*x + (B+B_z) - f(A*x+B))**2

# fig, ax = plt.subplots()

# popt = [1, 2, 3, 1]
# pcov = [0.1, 0.2, 0.2, 0.3]

# def f(x, *p):
#     a = p[0]
#     b = p[1]
#     c = p[2]
#     d = p[3]
#     return a*x**2+b*x+c+d*np.sqrt(x)


# xs = np.linspace(0, 10, 100)

# error = propagation_function(xs, f, popt, pcov)

# ax.plot(xs, f(xs, *popt))
# ax.fill_between(xs, f(xs, *popt) + error,
#                 f(xs, *popt) - error, alpha = 0.3)

# plt.show()
```

```
# a = np.array([[1,2,3], [1,2,3], [1,2,3]])
# print(np.diagonal(a))

# a = [1,2,3,4]
# b = [1,2,3,4]

# c = [a[0]+b[0]] + a[1:]
# for i in [0,1,2,3]:
#     print(a[:i] + [a[i] + b[i]] + a[i+1:])
```