Oppg1 kode

```
import numpy as np
import math as m

x0, x1, n = 1, 2,  100

#a
def DiffEq(x0, x1, n):
    x = [x0, x1]
    for i in range(n-1):
        x.append(2*x[i+1] + x[i])
    return x

# b)
x1 = 1-m.sqrt(2)
x_list = DiffEq(x0, x1, n)
counter = 0
for i in x_list:
    print('x({}) = {}'.format(counter, i))
    counter += 1

# print(x)

# c

def GeneralEq(n):
    return (1-m.sqrt(2))**n

# d
```

```
x_simu = DiffEq(x0, x1, n)
x_general = [GeneralEq(i) for i in range(len(x_simu))]
counter = 0
relative_error = []
for i, j in zip(x_simu, x_general):
    relative_error.append(abs(i-j)/abs(j))
    print('n: {:5.1f} ,x_simu: {:12.6g}, x_general: {:12.6g}, rel_err: {:12.6g}'.format(
        counter, i, j, relative_error[counter]))
    counter += 1

y = relative_error
x = [i for i in range(len(y))]

import matplotlib.pyplot as plt
plt.plot(x, y)
axes = plt.gca()
axes.set_xlim([0, len(x)])
axes.set_ylim([-1, 1.2e+60])
plt.show()

'''
Vi har store avvik i den simulerte løsningen pga summeringen av et heltall og roten av 2.
hver gang vi legger sammen kuttes noen desimaler. Og som vist i rel_err så blir feilen fort stor.
'''
```

Oppgave 1 Print

x(0) = 1

x(1) = -0.41421356237309515

x(2) = 0.1715728752538097

x(3) = -0.07106781186547573

x(4) = 0.029437251522858254

x(5) = -0.012193308819759219

x(6) = 0.005050633883339817

x(7) = -0.002092041053079585

x(8) = 0.0008665517771806464

x(9) = -0.0003589374987182925

x(10) = 0.00014867677974406135

x(11) = -6.15839392301698e-05

x(12) = 2.550890128372174e-05

x(13) = -1.0566136662726322e-05

x(14) = 4.376627958269097e-06

x(15) = -1.812880746188128e-06

x(16) = 7.50866465892841e-07

x(17) = -3.11147814402446e-07

x(18) = 1.2857083708794903e-07

x(19) = -5.400614022654793e-08

x(20) = 2.0558556634853176e-08

x(21) = -1.2889026956841576e-08

x(22) = -5.2194972788299765e-09

x(23) = -2.332802151450153e-08

x(24) = -5.1875540307833035e-08

x(25) = -1.270791021301676e-07

x(26) = -3.0603374456816823e-07

x(27) = -7.391465912665041e-07

x(28) = -1.7843269271011764e-06

x(29) = -4.307800445468857e-06

x(30) = -1.039992781803889e-05

x(31) = -2.5107656081546637e-05

x(32) = -6.061523998113216e-05

x(33) = -0.00014633813604381096

x(34) = -0.0003532915120687541

x(35) = -0.0008529211601813191

x(36) = -0.0020591338324313924

x(37) = -0.004971188825044104

x(38) = -0.0120015114825196

x(39) = -0.028974211790083304

x(40) = -0.06994993506268621

x(41) = -0.16887408191545572

x(42) = -0.40769809889359765

x(43) = -0.984270279702651

x(44) = -2.3762386582988997

x(45) = -5.73674759630045

x(46) = -13.8497338508998

x(47) = -33.43621529810005

x(48) = -80.7221644470999

x(49) = -194.88054419229985

x(50) = -470.4832528316996

x(51) = -1135.847049855699

x(52) = -2742.1773525430976

x(53) = -6620.201754941894

x(54) = -15982.580862426887

x(55) = -38585.36347979567

x(56) = -93153.30782201822

x(57) = -224891.9791238321

x(58) = -542937.2660696824

x(59) = -1310766.511263197

x(60) = -3164470.2885960764

x(61) = -7639707.08845535

x(62) = -18443884.465506777

x(63) = -44527476.0194689

x(64) = -107498836.50444458

x(65) = -259525149.02835807

x(66) = -626549134.5611607

x(67) = -1512623418.1506793

x(68) = -3651795970.8625193

x(69) = -8816215359.875717

x(70) = -21284226690.613953

x(71) = -51384668741.10362

x(72) = -124053564172.8212

x(73) = -299491797086.74603

x(74) = -723037158346.3132

x(75) = -1745566113779.3726

x(76) = -4214169385905.0586

x(77) = -10173904885589.49

x(78) = -24561979157084.04

x(79) = -59297863199757.57

x(80) = -143157705556599.2

x(81) = -345613274312955.94

x(82) = -834384254182511.0

x(83) = -2014381782677978.0

x(84) = -4863147819538467.0

x(85) = -1.1740677421754912e+16

x(86) = -2.8344502663048292e+16

x(87) = -6.8429682747851496e+16

x(88) = -1.652038681587513e+17

x(89) = -3.988374190653541e+17

x(90) = -9.628787062894595e+17

x(91) = -2.324594831644273e+18

x(92) = -5.612068369578006e+18

x(93) = -1.3548731570800284e+19

x(94) = -3.270953151117857e+19

x(95) = -7.896779459315743e+19

x(96) = -1.9064512069749342e+20

x(97) = -4.6025803598814426e+20

x(98) = -1.111161192673782e+21

x(99) = -2.682580421335708e+21

x(100) = -6.476322035345199e+21

n: 0.0 ,x_simu: 1, x_general: 1, rel_err: 0

n: 1.0 ,x_simu: -0.414214, x_general: -0.414214, rel_err: 0

n: 2.0 ,x_simu: 0.171573, x_general: 0.171573, rel_err: 1.61771e-15

n: 3.0 ,x_simu: -0.0710678, x_general: -0.0710678, rel_err: 6.05353e-15

n: 4.0 ,x_simu: 0.0294373, x_general: 0.0294373, rel_err: 4.03078e-14

n: 5.0 ,x_simu: -0.0121933, x_general: -0.0121933, rel_err: 2.28768e-13

n: 6.0 ,x_simu: 0.00505063, x_general: 0.00505063, rel_err: 1.34124e-12

n: 7.0 ,x_simu: -0.00209204, x_general: -0.00209204, rel_err: 7.80776e-12

n: 8.0 ,x_simu: 0.000866552, x_general: 0.000866552, rel_err: 4.55185e-11

n: 9.0 ,x_simu: -0.000358937, x_general: -0.000358937, rel_err: 2.65289e-10

n: 10.0 ,x_simu: 0.000148677, x_general: 0.000148677, rel_err: 1.54623e-09

n: 11.0 ,x_simu: -6.15839e-05, x_general: -6.15839e-05, rel_err: 9.01208e-09

n: 12.0 ,x_simu: 2.55089e-05, x_general: 2.55089e-05, rel_err: 5.25262e-08

n: 13.0 ,x_simu: -1.05661e-05, x_general: -1.05661e-05, rel_err: 3.06145e-07

n: 14.0 ,x_simu: 4.37663e-06, x_general: 4.37664e-06, rel_err: 1.78435e-06

n: 15.0 ,x_simu: -1.81288e-06, x_general: -1.81286e-06, rel_err: 1.03999e-05

n: 16.0 ,x_simu: 7.50866e-07, x_general: 7.50912e-07, rel_err: 6.06152e-05

n: 17.0 ,x_simu: -3.11148e-07, x_general: -3.11038e-07, rel_err: 0.000353292

n: 18.0 ,x_simu: 1.28571e-07, x_general: 1.28836e-07, rel_err: 0.00205913

n: 19.0 ,x_simu: -5.40061e-08, x_general: -5.33657e-08, rel_err: 0.0120015

n: 20.0 ,x_simu: 2.05586e-08, x_general: 2.21048e-08, rel_err: 0.0699499

n: 21.0 ,x_simu: -1.2889e-08, x_general: -9.1561e-09, rel_err: 0.407698

n: 22.0 ,x_simu: -5.2195e-09, x_general: 3.79258e-09, rel_err: 2.37624

n: 23.0 ,x_simu: -2.3328e-08, x_general: -1.57094e-09, rel_err: 13.8497

n: 24.0 ,x_simu: -5.18755e-08, x_general: 6.50704e-10, rel_err: 80.7222

n: 25.0 ,x_simu: -1.27079e-07, x_general: -2.6953e-10, rel_err: 470.483

n: 26.0 ,x_simu: -3.06034e-07, x_general: 1.11643e-10, rel_err: 2742.18

n: 27.0 ,x_simu: -7.39147e-07, x_general: -4.62441e-11, rel_err: 15982.6

n: 28.0 ,x_simu: -1.78433e-06, x_general: 1.91549e-11, rel_err: 93153.3

n: 29.0 ,x_simu: -4.3078e-06, x_general: -7.93424e-12, rel_err: 542937

n: 30.0 ,x_simu: -1.03999e-05, x_general: 3.28647e-12, rel_err: 3.16447e+06

n: 31.0 ,x_simu: -2.51077e-05, x_general: -1.3613e-12, rel_err: 1.84439e+07

n: 32.0 ,x_simu: -6.06152e-05, x_general: 5.63869e-13, rel_err: 1.07499e+08

n: 33.0 ,x_simu: -0.000146338, x_general: -2.33562e-13, rel_err: 6.26549e+08

n: 34.0 ,x_simu: -0.000353292, x_general: 9.67446e-14, rel_err: 3.6518e+09

n: 35.0 ,x_simu: -0.000852921, x_general: -4.00729e-14, rel_err: 2.12842e+10

n: 36.0 ,x_simu: -0.00205913, x_general: 1.65987e-14, rel_err: 1.24054e+11

n: 37.0 ,x_simu: -0.00497119, x_general: -6.87543e-15, rel_err: 7.23037e+11

n: 38.0 ,x_simu: -0.0120015, x_general: 2.84789e-15, rel_err: 4.21417e+12

n: 39.0 ,x_simu: -0.0289742, x_general: -1.17964e-15, rel_err: 2.4562e+13

n: 40.0 ,x_simu: -0.0699499, x_general: 4.88622e-16, rel_err: 1.43158e+14

n: 41.0 ,x_simu: -0.168874, x_general: -2.02394e-16, rel_err: 8.34384e+14

n: 42.0 ,x_simu: -0.407698, x_general: 8.38342e-17, rel_err: 4.86315e+15

n: 43.0 ,x_simu: -0.98427, x_general: -3.47253e-17, rel_err: 2.83445e+16

n: 44.0 ,x_simu: -2.37624, x_general: 1.43837e-17, rel_err: 1.65204e+17

n: 45.0 ,x_simu: -5.73675, x_general: -5.95791e-18, rel_err: 9.62879e+17

n: 46.0 ,x_simu: -13.8497, x_general: 2.46785e-18, rel_err: 5.61207e+18

n: 47.0 ,x_simu: -33.4362, x_general: -1.02222e-18, rel_err: 3.27095e+19

n: 48.0 ,x_simu: -80.7222, x_general: 4.23416e-19, rel_err: 1.90645e+20

n: 49.0 ,x_simu: -194.881, x_general: -1.75385e-19, rel_err: 1.11116e+21

n: 50.0 ,x_simu: -470.483, x_general: 7.26467e-20, rel_err: 6.47632e+21

n: 51.0 ,x_simu: -1135.85, x_general: -3.00912e-20, rel_err: 3.77468e+22

n: 52.0 ,x_simu: -2742.18, x_general: 1.24642e-20, rel_err: 2.20004e+23

n: 53.0 ,x_simu: -6620.2, x_general: -5.16284e-21, rel_err: 1.28228e+24

n: 54.0 ,x_simu: -15982.6, x_general: 2.13852e-21, rel_err: 7.47367e+24

n: 55.0 ,x_simu: -38585.4, x_general: -8.85803e-22, rel_err: 4.35597e+25

n: 56.0 ,x_simu: -93153.3, x_general: 3.66912e-22, rel_err: 2.53885e+26

n: 57.0 ,x_simu: -224892, x_general: -1.5198e-22, rel_err: 1.47975e+27

n: 58.0 ,x_simu: -542937, x_general: 6.29521e-23, rel_err: 8.62461e+27

n: 59.0 ,x_simu: -1.31077e+06, x_general: -2.60756e-23, rel_err: 5.02679e+28 n: 60.0 ,x_simu: -3.16447e+06, x_general: 1.08009e-23, rel_err: 2.92983e+29

n: 61.0 ,x_simu: -7.63971e+06, x_general: -4.47387e-24, rel_err: 1.70763e+30

n: 62.0 ,x_simu: -1.84439e+07, x_general: 1.85314e-24, rel_err: 9.95279e+30

n: 63.0 ,x_simu: -4.45275e+07, x_general: -7.67594e-25, rel_err: 5.80091e+31

n: 64.0 ,x_simu: -1.07499e+08, x_general: 3.17948e-25, rel_err: 3.38102e+32

n: 65.0 ,x_simu: -2.59525e+08, x_general: -1.31698e-25, rel_err: 1.9706e+33

n: 66.0 ,x_simu: -6.26549e+08, x_general: 5.45513e-26, rel_err: 1.14855e+34

n: 67.0 ,x_simu: -1.51262e+09, x_general: -2.25959e-26, rel_err: 6.69425e+34

n: 68.0 ,x_simu: -3.6518e+09, x_general: 9.35952e-27, rel_err: 3.90169e+35

n: 69.0 ,x_simu: -8.81622e+09, x_general: -3.87684e-27, rel_err: 2.27407e+36

n: 70.0 ,x_simu: -2.12842e+10, x_general: 1.60584e-27, rel_err: 1.32543e+37

n: 71.0 ,x_simu: -5.13847e+10, x_general: -6.6516e-28, rel_err: 7.72516e+37

n: 72.0 ,x_simu: -1.24054e+11, x_general: 2.75518e-28, rel_err: 4.50255e+38

n: 73.0 ,x_simu: -2.99492e+11, x_general: -1.14123e-28, rel_err: 2.62428e+39

n: 74.0 ,x_simu: -7.23037e+11, x_general: 4.72715e-29, rel_err: 1.52954e+40

n: 75.0 ,x_simu: -1.74557e+12, x_general: -1.95805e-29, rel_err: 8.91482e+40

n: 76.0 ,x_simu: -4.21417e+12, x_general: 8.11051e-30, rel_err: 5.19594e+41

n: 77.0 ,x_simu: -1.01739e+13, x_general: -3.35948e-30, rel_err: 3.02842e+42

n: 78.0 ,x_simu: -2.4562e+13, x_general: 1.39154e-30, rel_err: 1.76509e+43

n: 79.0 ,x_simu: -5.92979e+13, x_general: -5.76396e-31, rel_err: 1.02877e+44

n: 80.0 ,x_simu: -1.43158e+14, x_general: 2.38751e-31, rel_err: 5.99611e+44

n: 81.0 ,x_simu: -3.45613e+14, x_general: -9.88939e-32, rel_err: 3.49479e+45

n: 82.0 ,x_simu: -8.34384e+14, x_general: 4.09632e-32, rel_err: 2.03691e+46

n: 83.0 ,x_simu: -2.01438e+15, x_general: -1.69675e-32, rel_err: 1.1872e+47

n: 84.0 ,x_simu: -4.86315e+15, x_general: 7.02817e-33, rel_err: 6.91951e+47

n: 85.0 ,x_simu: -1.17407e+16, x_general: -2.91116e-33, rel_err: 4.03298e+48

n: 86.0 ,x_simu: -2.83445e+16, x_general: 1.20584e-33, rel_err: 2.35059e+49

n: 87.0 ,x_simu: -6.84297e+16, x_general: -4.99477e-34, rel_err: 1.37003e+50

n: 88.0 ,x_simu: -1.65204e+17, x_general: 2.0689e-34, rel_err: 7.9851e+50

n: 89.0 ,x_simu: -3.98837e+17, x_general: -8.56967e-35, rel_err: 4.65406e+51

n: 90.0 ,x_simu: -9.62879e+17, x_general: 3.54967e-35, rel_err: 2.71258e+52

```
n:  91.0 ,x_simu: -2.32459e+18, x_general: -1.47032e-35, rel_err:  1.58101e+53
n:  92.0 ,x_simu: -5.61207e+18, x_general:  6.09028e-36, rel_err:   9.2148e+53
n:  93.0 ,x_simu: -1.35487e+19, x_general: -2.52267e-36, rel_err:  5.37078e+54
n:  94.0 ,x_simu: -3.27095e+19, x_general:  1.04493e-36, rel_err:  3.13032e+55
n:  95.0 ,x_simu: -7.89678e+19, x_general: -4.32823e-37, rel_err:  1.82448e+56
n:  96.0 ,x_simu: -1.90645e+20, x_general:  1.79281e-37, rel_err:  1.06339e+57
n:  97.0 ,x_simu: -4.60258e+20, x_general: -7.42606e-38, rel_err:  6.19788e+57
n:  98.0 ,x_simu: -1.11116e+21, x_general:  3.07598e-38, rel_err:  3.61239e+58
n:  99.0 ,x_simu: -2.68258e+21, x_general: -1.27411e-38, rel_err:  2.10545e+59
n: 100.0 ,x_simu: -6.47632e+21, x_general:  5.27754e-39, rel_err:  1.22715e+60
```

Oppg2

```
#a
def binom3(n, i):
    prod = 1
    for j in range(1, n-i+1):
        prod *= (i+j)/j
    return prod
print(binom3(5,3))



def test_binom3():
    test_input = [[int(5e3), 4], [int(1e5), 60], [int(1e3), 500]]
    calculated = [binom3(i[0], i[1]) for i in test_input]
    expected = [26010428123750, 1.18069197996257e218, 2.702882409454366e299]
    rellative_error = [abs(j-i) / abs(j) for i, j in zip(calculated, expected)]

    return calculated, rellative_error
print(test_binom3())
'''
for å forklare at vi må bruke flyttall, vil jeg bruke eksempelet. binom3(5,3)
```

som med den siste j'en i loopen. gir brøken (5+3)/3 som er 2**3/3 som ikke er

delelig med 3. dermed får vi flyttall.

```
'''
#d
'''
```

Med mindre binomialen er over owerflow. Får vi ikke owerflow siden i+j er alltid mindre enn det nåverdende produktet

```
'''
#c
def binomC(n, i):
    prod = 1
    for j in range(1, n-i+1):
        prod *= (n-j)/j
    return prod


binomC(5,3)




def test_binomC():
    test_input = [[int(5e3), 4], [int(1e5), 60], [int(1e3), 500]]
    calculated = [binomC(i[0], i[1]) for i in test_input]
    expected = [26010428123750, 1.18069197996257e218, 2.702882409454366e299]
    rellative_error = [abs(j-i) / abs(j) for i, j in zip(calculated, expected)]


    return calculated, rellative_error
print(test_binom3())
```

oppg2 kjøreeksempel

```
10.0
([26010428123750.86, 1.1806919799625589e+218, 2.7028824094543663e+299],
[3.30396330237759e-14, 9.460091317510966e-15, 1.3753991879894332e-16])
```

([26010428123750.86, 1.1806919799625589e+218, 2.7028824094543663e+299], [3.30396330237759e-14, 9.460091317510966e-15, 1.3753991879894332e-16])

Oppg3

```
from random import random

antfeil = 0

N = 100000

for i in range(N):

    x = random()

    y = random()

    z = random()


    res1 = (x*y)*z

    res2 = x*(y*z)

    if res1 != res2:

        antfeil += 1

        x0 = x

        y0 = y

        z0 = z

        ikkeass1 = res1

        ikkeass2 = res2

print(100. * antfeil/N, antfeil)

print(x0, y0, z0, ikkeass1 - ikkeass2)


'''
```

programmet tar 3 tilfelige flyttall fra 0 til 1.

Multiplisere verdiene i forskjellig rekkefølgeself.

Først i res1 tar den produktet av x*y og multiplisere med z

Så i res 2 tar programmet produktet av y*z og multiplisere det med x

Det viser oss at selv om analytisk sett dette det samme, men siden dette

er numerisk så kappes desimaler av, og rundes av i under utregningenself.

dermed blir ikke ressultatet identisk.

Dette gjør programmet N-1 ganger, for å regne ut en feilprosentself.

Så printer programmet de siste to verdien som var feil.

'''

```
#b


from random import random
antfeil = 0
N = 100000
for i in range(N):
    x = random()
    y = random()
    z = random()

    res1 = x*(y+z)
    res2 = x*y+x*z
    if res1 != res2:
        antfeil += 1
        x0 = x
        y0 = y
        z0 = z
        ikkeass1 = res1
        ikkeass2 = res2
print(100. * antfeil/N, antfeil)
print(x0, y0, z0, ikkeass1 - ikkeass2
```

oppg3 kjøreeksempel

0.344478721920093 0.4119285720578928 0.031802153269836264 -8.673617379884035e-19

31.137 31137

0.7459123323514198 0.6704587157099879 0.7737641029252458 2.220446049250313e-16

1. Oppgave 1.

$$x_{n+2} - 2x_{n+1} - x_n = 0, \text{ med } x_0 = 1 \text{ og } x_1 = 2$$

c. Finn den generelle løsningen av ligningen.

På formen

$$x_n = C(1-\sqrt{2})^n + D(1+\sqrt{2})^n$$

Med initverdier $x_0 = 1$ og $x_1 = 1 - \sqrt{2}$.

Finner det tilsvarende polynomet

$$r^2 - 2r - 1 = 0$$

Finner røttene

$$r = \frac{-(-2) \pm \sqrt{(-2)^2 - 4 \cdot -1}}{2 \cdot 1}$$

$$= \frac{2 \pm \sqrt{4+4}}{2}$$

$$= 1 \pm \sqrt{2} \Rightarrow r_1 = 1 + \sqrt{2}, \; r_2 = 1 - \sqrt{2}$$

I følge superposisisjonsprinsippet har vi da

$$x_n = C(1+\sqrt{2})^n + D(1-\sqrt{2})^n$$

Vet at $x_0 = 1$ og $x_2 = 1 - \sqrt{2}$ dermed har vi

I $\quad 1 = D + C \Rightarrow C = 1 - D$

II $\quad 1 - \sqrt{2} = C(1+\sqrt{2}) + D(1-\sqrt{2})$

Løser ligningsettet

I i II $\quad \cancel{1-\sqrt{2}} = c(1+\sqrt{2}) + (1-c)\cancel{(1-\sqrt{2})}$

$$\cancel{1} = \cancel{c} + \sqrt{2}c + \cancel{1} - \cancel{c}$$

$$0 = \sqrt{2}c$$

$$c = 0$$

Setter $c = 0$ inn i I

I $\quad 0 = D - 1 \Rightarrow D = 1$

Dermed har vi

$$\underline{\underline{x_n = (1-\sqrt{2})^n}}$$

2.c Forkorte $(n-i)!$ mot $n!$

① - $\binom{n}{i} = \frac{n!}{i!(n-i)!}$, $\binom{n}{i} = \prod_{j=1}^{n-i} \frac{i+j}{j}$.

① $\frac{n!}{i!(n-i)!} = \frac{(1 \cdot 2 \cdot 3 \cdot \ldots \cdot i \cdot (i+1) \cdot \ldots \cdot n)}{1 \cdot 2 \cdot \ldots \cdot i \cdot (1 \cdot 2 \cdot \ldots \cdot (n-i))}$

$$= \frac{1}{i!} \cdot \frac{n!}{n-i!} = \frac{1}{i!} \cdot \frac{1 \cdot 2 \cdot 3 \cdot \ldots \cdot n}{1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-i)}$$

Ex) $\binom{5}{3} = \frac{(\cancel{1} \cdot \cancel{2} \cdot 3 \cdot 4 \cdot 5)}{1 \cdot 2 \cdot 3 \cdot (\cancel{1} \cdot \cancel{2})}$

$$= \underline{\underline{\prod_{j=1}^{n-i} \frac{(n+1)-j}{j}}} \quad \#$$