Simulering med Python.

Du kan vælge at få Python med Anaconda eller på den nørdede måde.

Med Anaconda:

Download Anaconda på anaconda.com
Anaconda | The World's Most Popular Data Science Platform
Når du har installeret anaconda (det kan tage lidt tid) opdaterer du til version 2.2.0 eller nyere. Dernæst vælger du Spyder, her er python programmet.

Den nørdede måde er beskrevet til sidst, det er påkrævet, hvis du vil have de nyeste Python pakker.

Vi starter med et simpelt program, hvor vi tegner punkter på linjen y=4x+3, ved at simulere trinnene stepvis.

Ikke fordi I skal benytte det til rette linjer, men fordi, her kan i let se princippet.

Du åbner Spyder. Hvis du vælger den nørdede måde Åbner nu IDLE.

Vælg File, New file

Læs programmet. Hvor meget forstår du? (matplotlib er krævet , hvis vi skal have tegnet en graf.)

Skriv programmet ind.

Først vil vi kun beregne y, og få udskrevet en ”tabel”.

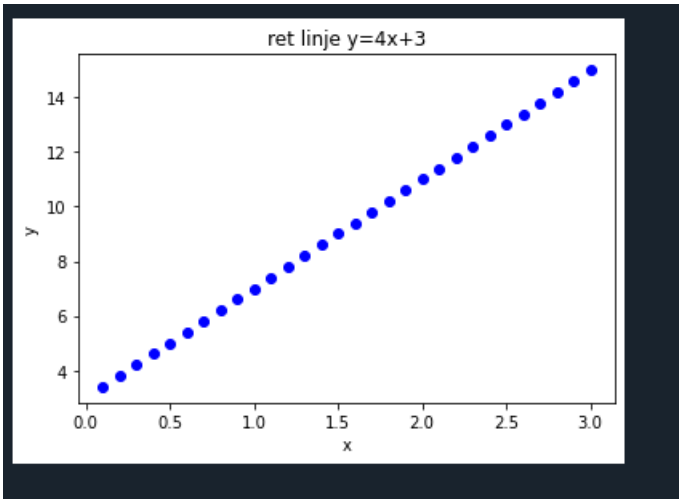
```
8
9   a=4
10
11  y=3
12
13  x=0
14
15  dx=0.1
16
17  for i in range (30):
18
19      x=x+dx  # kunne også skrives x+=dx
20      dy=a*dx
21      y=y+dy  # kunne også skrives y+=dy
22
23      print(x,y)
24
25
26
```

```
In [4]: runfile('C:/Users/edith/Documents/ret linj
0.1 3.4
0.2 3.8
0.30000000000000004 4.2
0.4 4.6000000000000005
0.5 5.000000000000001
0.6 5.400000000000001
0.7 5.800000000000002
0.7999999999999999 6.200000000000002
0.8999999999999999 6.600000000000002
0.9999999999999999 7.000000000000003
1.0999999999999999 7.400000000000003
1.2 7.800000000000003
1.3 8.200000000000003
1.4000000000000001 8.600000000000003
1.5000000000000002 9.000000000000004
1.6000000000000003 9.400000000000004
1.7000000000000004 9.800000000000004
1.8000000000000005 10.200000000000005
1.9000000000000006 10.600000000000005
2.0000000000000004 11.000000000000005
2.1000000000000005 11.400000000000006
2.2000000000000006 11.800000000000006
2.3000000000000007 12.200000000000006
2.400000000000001 12.600000000000007
2.500000000000001 13.000000000000007
2.600000000000001 13.400000000000007
2.700000000000001 13.800000000000008
2.800000000000001 14.200000000000008
2.9000000000000012 14.600000000000009
3.0000000000000013 15.000000000000009

In [5]:
```

Dernæst vil vi have tegnet en graf

```
from matplotlib import pyplot as plt

a=4

y=3

x=0

dx=0.1

for i in range (30):

    x=x+dx  # kunne også skrives x+=dx
    dy=a*dx
    y=y+dy  # kunne også skrives y+=dy

    plt.title("ret linje y=4x+3")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.plot(x,y,'bo')
plt.show()
```

Kør programmet og I vil få denne graf:



Opgave 1 : prøv at tegne punkter på linjen y=3x-7

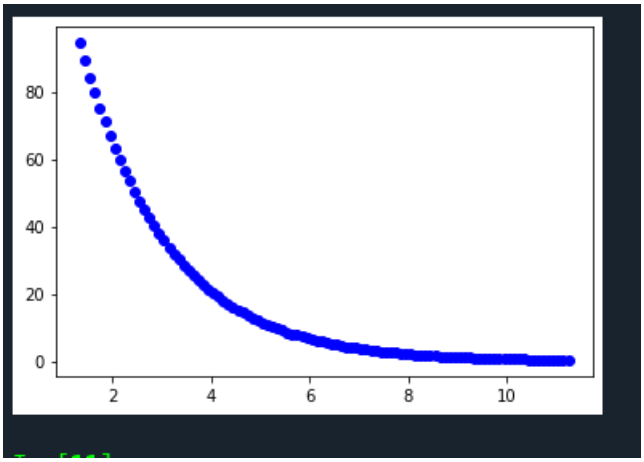
Radioaktivt henfald

Læs programmet. Hvor meget forstår du?

Python kender ikke ln , vi skal have importeret math, hvis vi skal regne med logaritmer,

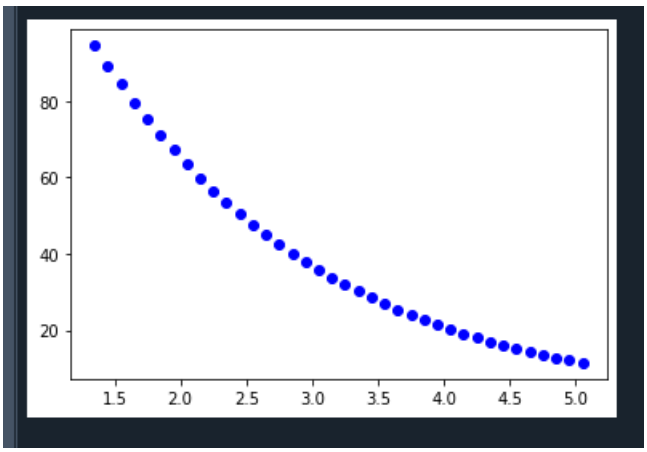
ln(2) skrives som math.log(2,e) .

```
from matplotlib import pyplot as plt
import math

e=math.e

n=100

t=0
dt=0.1
t=1.25  #halveringstid for Te-131 er 1.25 døgn
k=math.log(2,e)/t


for i in range (100):
  a=k*n
  dn=a*dt
  n=n-dn
  t=t+dt
  plt.plot(t,n,'bo')


plt.show()
```



Dette program gør det samme, blot kun i 5 døgn:

```
from matplotlib import pyplot as plt
import math

e=math.e

n=100

t=0
dt=0.1
t=1.25  #halveringstid for Te-131 er 1.25 døgn
k=math.log(2,e)/t


while t<5:
  a=k*n
  dn=a*dt
  n=n-dn
  t=t+dt
  plt.plot(t,n,'bo')


plt.show()
```



ADVARSEL, Hvis I benytter while, skal I være sikre på at betingelsen bliver opfyldt, ellers får I en uendelig løkke, der belaster computeren. Det sker ikke med en ”For i en range (100)”, da den kun gennemløber løkken 100 gange.

Skulle uheldet være ude, kan I stoppe programmet, ved at klikke på den røde firkant i nederste vindue for oven til højre.  .

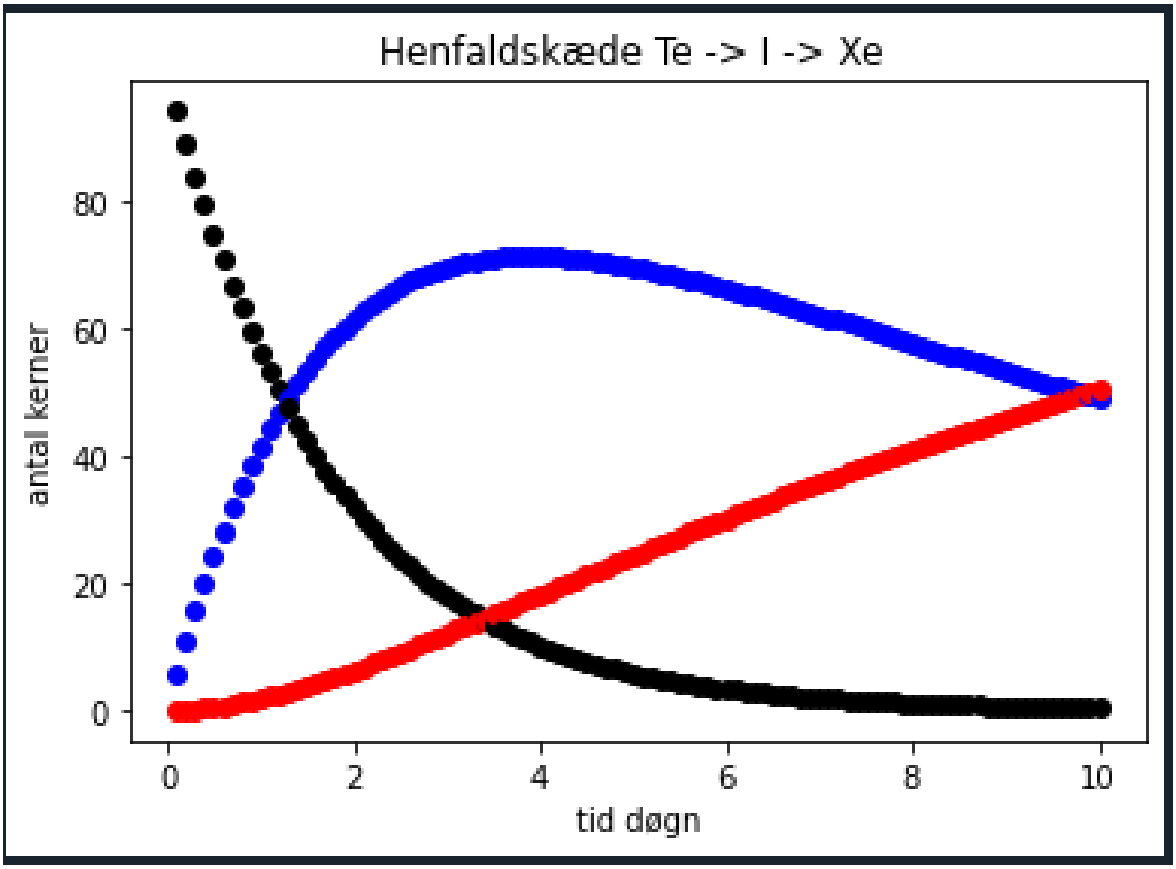
Opgave 2: sæt navne på akserne, kør sidste program i 10 døgn

Henfaldskæder

Vi vil se på henfaldskæden Tellur-131 -> Iod -131-> Xenon-131

```python
from matplotlib import pyplot as plt
import math
import sys
e=math.e

n1=100 #antal Te-131
n2=0   #antal I-131
n3=0    #antal Xe-131
t=0
dt=0.1
t1=1.25  #halveringstid for Te-131 er 1.25 døgn
k1=math.log(2,e)/t1  #k1=ln(2)/t1
t2=8.0     #halveringstid for I-131 er 8 døgn
k2=math.log(2,e)/t2

for i in range (100):
  a1=k1*n1
  a2=k2*n2
  dn1=a1*dt
  dn2=a2*dt
  n1+=-dn1
  n2+=dn1-dn2
  n3+=dn2
  t+=dt
  plt.plot(t,n2, 'bo')
  plt.plot(t,n1,  marker ='o', color='k')
  plt.plot(t,n3,  marker = 'o', color='r')
  plt.title("Henfaldskæde Te -> I -> Xe ")
  plt.xlabel("tid døgn")
  plt.ylabel("antal kerner")
  if t>10:
     sys.exit()
plt.show()
```

Her er et resultat af en kørsel:



Opgave : Skriv programmet og kør programmet.

Prøv med andre henfaldskæder.

nuclide	half-life (a=year)	product of decay
^{232}Th	1.405×10^{10} a	^{228}Ra
^{228}Ra	5.75 a	^{228}Ac
^{228}Ac	6.25 h	^{228}Th
^{228}Th	1.9116 a	^{224}Ra
^{224}Ra	3.6319 d	^{220}Rn
^{220}Rn	55.6 s	^{216}Po
^{216}Po	0.145 s	^{212}Pb
^{212}Pb	10.64 h	^{212}Bi
^{212}Bi	60.55 min	^{212}Po ^{208}Tl
^{212}Po	299 ns	^{208}Pb
^{208}Tl	3.053 min	^{208}Pb
^{208}Pb	.	.

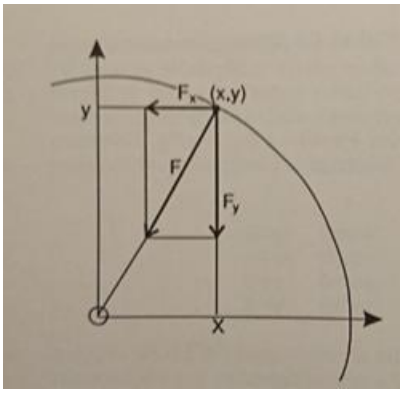
Satellitbevægelser

Månen om Jorden

Den kraft, der påvirker planeter eller satellitter, gravitationskraften, har hele tiden retning mod et bestemt punkt Jordens eller Solens centrum.
Vi vil her se på månens bevægelse om Jorden.
Jorden befinder sig i punktet (0,0) og satellitten, her månen, befinder sig i punktet (x,y) .
Kraften F peger mod (0,0) , og den har størrelsen:

$F = G \cdot \frac{M \cdot m}{r^2}$ og $r = \sqrt{x^2 + y^2}$, M er her centrallegemets masse, her Jorden men kan også være Solen.

F kan opfattes som en sum af to kræfter: F_x i x-aksens retning og F_y i y-aksens retning (se figur)



radius i cirklen er r. Figurbillede fra Det Kosmiske urværk.

Af trekanten ses:

$\frac{F_x}{F} = \frac{x}{r} <=> F_x = F \cdot \frac{x}{r}$ og $\frac{F_y}{F} = \frac{y}{r} <=> F_y = F \cdot \frac{y}{r}$

Vi vil nu finde startbetingelserne:
Månebanens radius er $3{,}84 \cdot 10^8 m$, og den omkreds bliver $2 \cdot \pi \cdot 3{,}84 \cdot 10^8 m = 2{,}41 \cdot 10^9 m$

Månens omløbstid om Jorden er 27,3 dage= $2{,}36 \cdot 10^6 s$
Månen har altså hastigheden:

$vx = \frac{2{,}41 \cdot 10^9 m}{2{,}36 \cdot 10^6 s} = 1022 \frac{m}{s}$

Geostationær bane

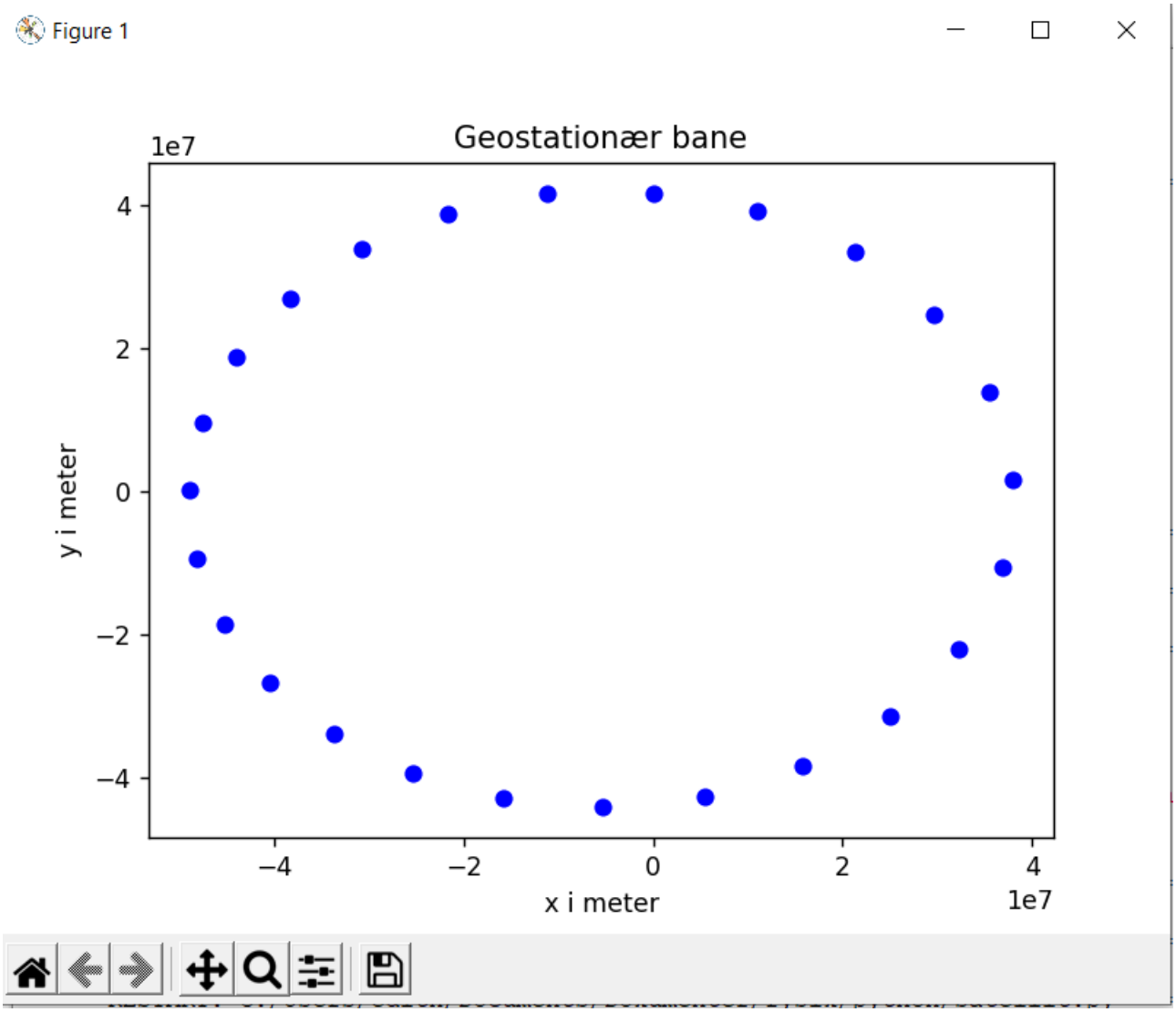
Vi vil ny simulere en satellitbevægelse i den geostationære bane:

Her har vi vx:

$$\frac{2 \cdot \pi \cdot 42200000 \cdot \_m}{1 \cdot \_day} \blacktriangleright 3068.87 \cdot \frac{\_m}{\_s}$$

```
from math import sqrt
from matplotlib import pyplot as plt
import sys

mj=5.974e24
G=6.6743e-11
m=1
t=0
x=0
y=42200000  #geostationær bane
vx=3068.87  #m/s  2*pi*ystart/(1døgn)
vy=0
dt=3600   # 1 time

for i in range (25):
    r=sqrt(x*x+y*y)
    F=G*mj*m/(r*r)
    Fx=-F*x/r
    Fy=-F*y/r
    ax=Fx/m
    ay=Fy/m
    dvx=ax*dt
    dvy=ay*dt
    vx+=dvx
    vy+=dvy
    dx=vx*dt
    dy=vy*dt
    t+=dt
    x+=dx
    y+=dy

    plt.plot(x,y,'bo') # blå o
    plt.title("Geostationær bane")
    plt.xlabel("x i meter")
    plt.ylabel("y i meter")

    if t>3000000:
        sys.exit()

plt.show()
```

Her er et resultat af en kørsel:



Opgave: Prøv selv at simulere Månen om Jorden.

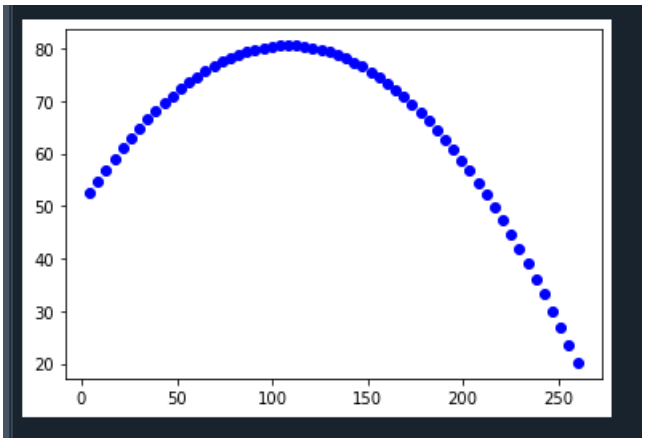
Opgave:
Prøv at ændre programmet så I simulerer nogle af de første satellitter:

	afsendelsesdag	Afstand i km	Omløbstid	vægt
Sputnik 1 (Russisk)	4.10.57	220-959	96,2min	83.6kg
Explorer (Amr)	31.1.58	320-2600	114,5min	13,5 kg
geostationær		42200	1 døgn	

Prøv med en af satellitterne at simulere bevægelsen.
husk at regne i SI enheder.
Prøv at ændre vx og se hvad der sker hvis den er for lille eller for stor.

Skråt kast

```python

from matplotlib import pyplot as plt
import math

pi=math.pi

x=0
y=50
t=0
dt=0.1
ax=0
ay=-9.82
v0=50  #m/s
vinkel=30*pi/180  #30 grader
vx=v0*math.cos(vinkel)
vy=v0*math.sin(vinkel)
for i in range (60):
    dvx=ax*dt
    dvy=ay*dt
    vx=vx+dvx
    vy=vy+dvy
    dx=vx*dt
    dy=vy*dt
    x=x+dx
    y=y+dy
    t=t+dt
    plt.plot(x,y,'bo')

plt.show()

```



Vi prøver nu med luftmodstand på en golfkugle.

Luftmodstanden virker direkte modsat kuglens hastighed (v_x,v_y). Vi indfører kuglens fart

$v = \sqrt{{v_x}^2 + v_y^2}$. Vi kan så skrive luftmodstandskraften på formen

$F_x = -k \cdot v \cdot v_x$ og $F_y = -k \cdot v \cdot v_y$, hvor $k = \frac{1}{2} \cdot C_w \cdot \rho \cdot A$

A er arealet, Cw er formfaktoren, ρ er luftens densitet.

-Vi kan kontrollere formlen:

$|F| = \sqrt{F_x^2 + F_y^2} = \sqrt{(-k \cdot v \cdot v_x)^2 + (-k \cdot v \cdot v_y)^2}$

$$= \sqrt{(k \cdot v)^2 \cdot (v_x^2 + v_y^2)} \quad = \quad k \cdot v \cdot v = \frac{1}{2} \cdot C_w \cdot \rho \cdot A \cdot v^2$$

Vi har da for c=k/m:

$$v = \sqrt{vx^2 + vy^2}$$

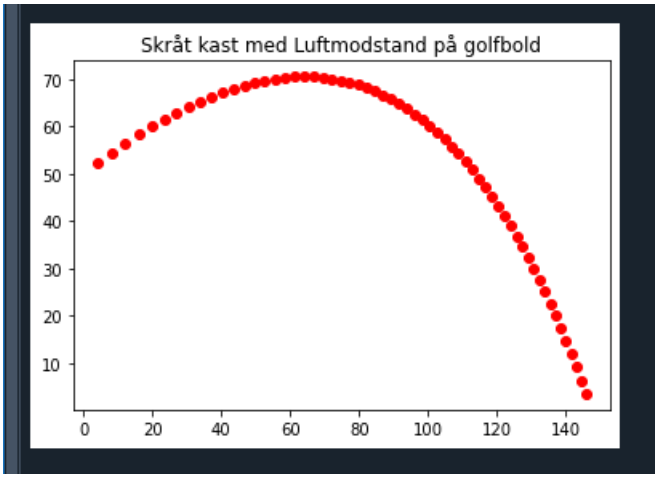
$$ax = -c \cdot v \cdot vx$$

$$ay = -g - c \cdot v \cdot vy$$

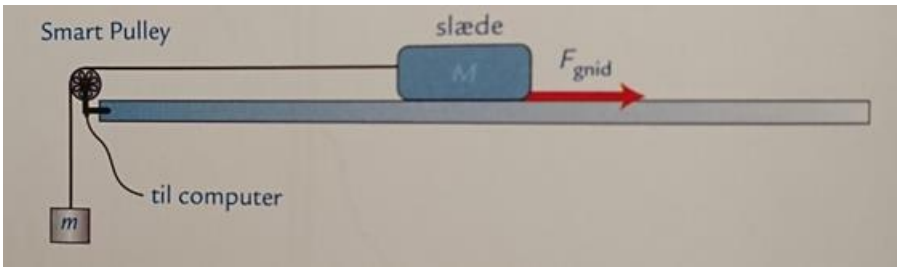
Vi får følgende program:

```
from matplotlib import pyplot as plt
import math

pi=math.pi
x=0
y=50
t=0
cw=0.3
radius=0.0215 #kuglens radius
rho= 1.3
A= pi*radius**2
m=0.045 #masse i kg
c=.5*A*rho*cw/m

dt=0.1
v0=50  #m/s
vinkel=30*pi/180  #30 grader i radianer
vx=v0*math.cos(vinkel)
vy=v0*math.sin(vinkel)

for i in range (60):
    v=math.sqrt(vx**2+vy**2)
    ax=-c*v*vx
    ay=-9.82-c*v*vy

    dvx=ax*dt
    dvy=ay*dt
    vx=vx+dvx
    vy=vy+dvy
    dx=vx*dt
    dy=vy*dt
    x=x+dx
    y=y+dy
    t=t+dt
    plt.plot(x,y,'ro')
    plt.title("Skråt kast med Luftmodstand på golfbold")

plt.show()
```

Med denne graf:



Klods på bord med gnidning



Figurbillede fra Orbit BA.

Simulering af forsøget

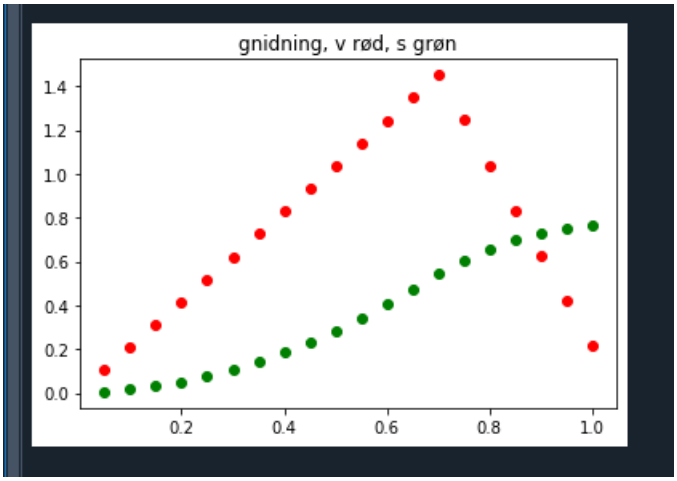
Mb er trækloddet

Ma er slæden

Prøv om du kan forstå programmet, her ser vi en if else forgrening, kør programmet.

```
from matplotlib import pyplot as plt
import sys

Ma=5
Mb=4
my=0.42
g=9.82
t=0
dt=0.05
v=0
s=0
for i in range (20):
    if s<0.5:
        Fres= Mb*g-my*Ma*g
        M=Ma+Mb

    else:
        Fres=-my*Ma*g
        M=Ma

    t=t+dt
    a=Fres/M
    v=v+a*dt
    s=s+v*dt

    plt.plot(t,s,'go')
    plt.plot(t,v,'ro')
    plt.title("gnidning, v rød, s grøn")
```

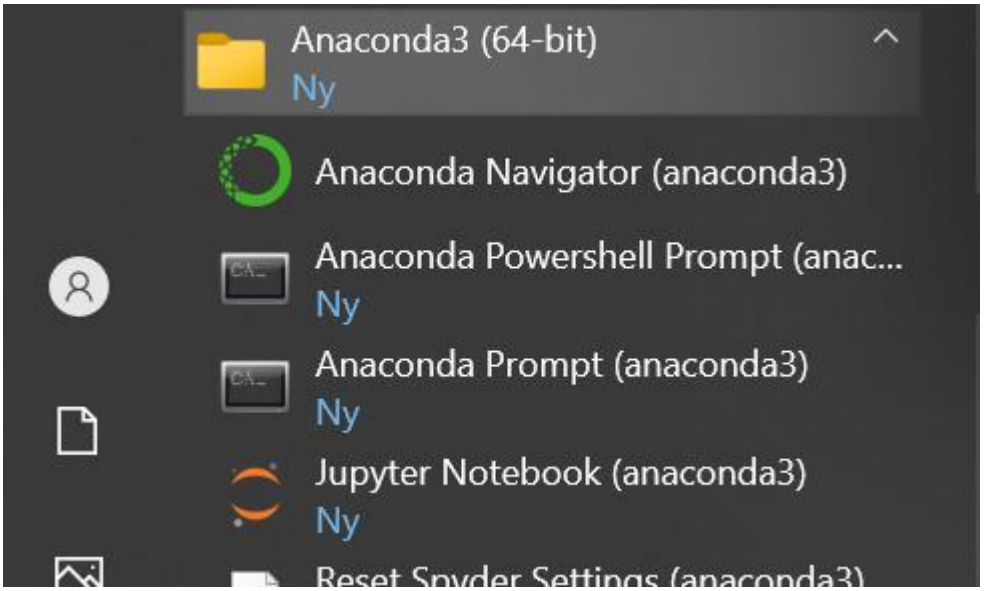
```
if v<0:
    sys.exit()

plt.show
```



Senere kan du få brug for biblioteker der ikke er i Anacondas Spider.

Så kan du evt installere dem med Anaconda promten

Vælg Anaconda Promt i Windows-menuen



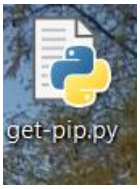
Skriv conda install og navnet på det du vil installere i promten.

Hvis du vil have helt nye pakker, så skal du installere Python på en anden måde:

På den nørdede måde:

Download Python programmet og læg mærke til, hvor det bliver lagt. Husk at sætte fluebenet i Add to PATH, når I skal køre Exefilen. (Se video). (72) How to Download and Install Python latest version in Windows 10 - YouTube www.youtube.com/watch?v=VWgs_iTojoA

Du kan nu bruge Python til simple programmer , men vi skal importere nogle biblioteker.
Importer pip biblioteker: se følgende
[Download and install pip Latest Version - GeeksforGeeks]
[How to Install PIP for Python on Windows - Liquid Web]

Download nu 

Højreklik læg på skrivebord
Læg nu filen i den mappe hvor I har Python programmet.
Skriv CMD i søgefeltet nederst til venstre.
Åben kommando-promten
Tast python og tast enter.
Hvis du ikke kan starte python fra kommandopromten, så se videoen om installation af python igen.
Hvis du har python installeret skal du nu
Taste exit() for at stoppe pythonprogrammet.
Skriv dernæst I kommandopromten:
python get-pip.py
For at tegne grafer skal vi benytte matplotlib.
Importer matplotlib skriv I kommandopromten

```
pip install matplotlib
```

Senere kan du her få brug for at importere astropy herunder numpy, den er i Anaconda, men hvis du benytter den **nørdede installation skal du**
Skriv I CMD promt:

To install astropy from source into a existing Python installation without using Anaconda, use the following:

```
pip install astropy
```

More detailed installation instructions (e.g., for building from source code locally) are in the documentation.

Tilsvarende med andre som ikke er i Anaconda.

Inspiration hentet fra

FPro3_en kort introduktion.pdf (fys.dk)

www.fys.dk/fpro3/Tekster/FPro3_en%20kort%20introduktion.pdf

Her kan du også læse om spin og Magnuskraften.

EH 21-09-2022