

# Titel: Komet dræberen

## Spiloplevelse:

Det forgår i det ydre rum. Du flyver rundt i et rumskib. Der er kometer som flyver på kryds og tværs. Du skal skyde så mange som muligt ned inden en af kometerne rammer dig og spillet er ovre.

## Elementer i spillet:

- Stjerne baggrund
- Rumskib
- En eller flere kometer
- Skud (flere skud samtidig)
- Point system

## Del mål:

1. Sprites til spillet
2. Spil vindue med stjernefyldt baggrund
3. Rumskib der kan roter og flyve fremad
4. Rumskibet kan skyde
5. Kometer der kommer fra alle side med forskelligt tids interval
6. Komet ødelægges når den rammes med et skud
7. Point system

## Del mål 1 - Sprites til spillet

Der skal bruges sprites til følgende

- rumskib
- skud
- lille meteor
- stor meteor

Gå til <https://kenney.nl/assets/simple-space> og download pakken

Find i pakken følgende filer og omdøb dem til navn i ():

ship_F.png	(ship.png)
star_small.png	(bullet.png)
meteor_detailedLarge.png	(meteor_large.png)
meteor_detailedSmall.png	(meteor_small.png)

Opret en mappe til dit spil – kald mappen komet\_dræber)

opret i denne mappe en ny mappe – kald mappen images

placer de fire png filer i mappen

## Del mål 2 - Spil vindue med stjernefyldt baggrund

Vi definerer spil vinduets størrelse som også bliver rammen for området hvor i stjerner placeres.

Opret en fil kaldet main.py i folderen komet\_dræber

Tilføj denne kode til main.py

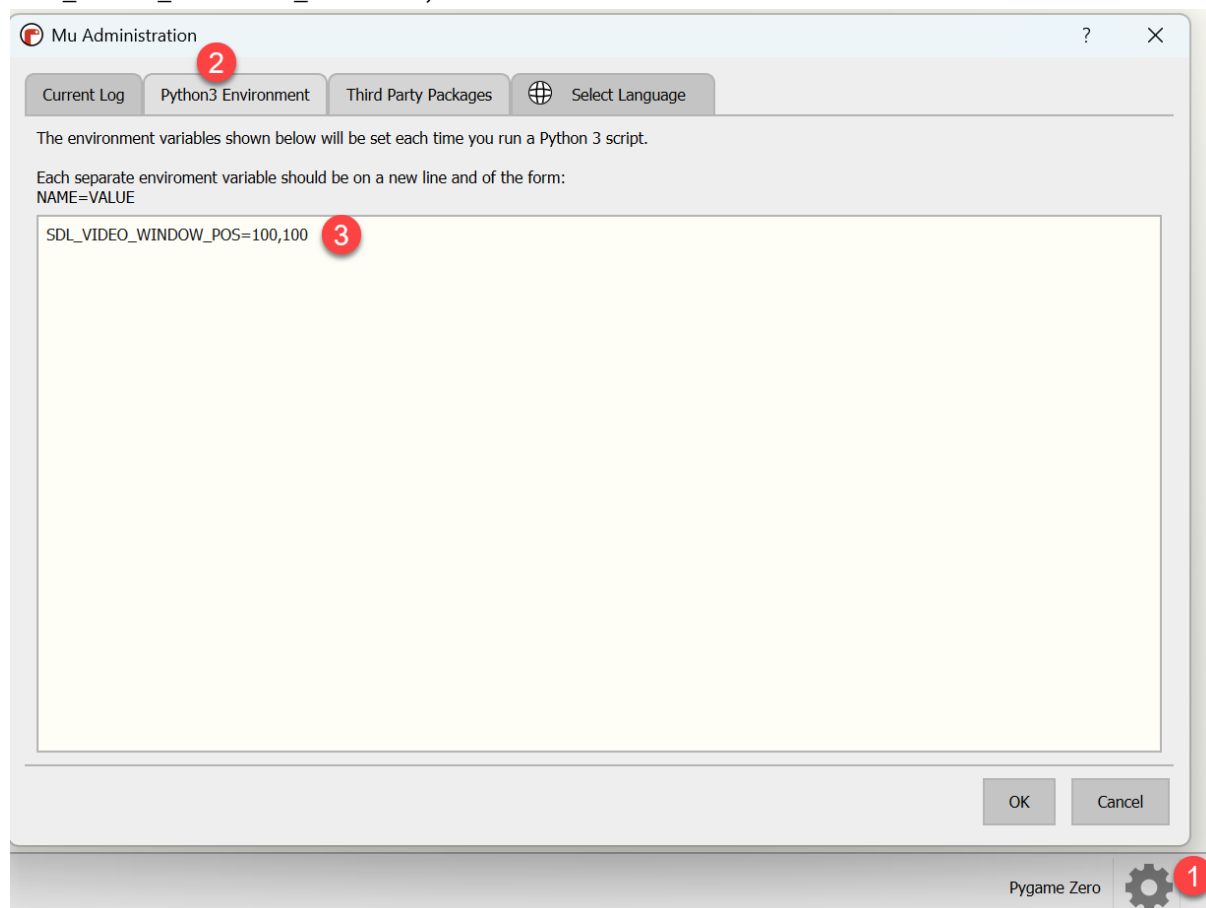
```
1 #Komet dræber spil - skyd kometerne ned inden de rammer dig!
2
3 WIDTH = 600
4 HEIGHT = 800
5
6 def draw():
7     pass
```

Tilpas størrelsen efter din skærms opløsning

Afprøv koden (husk at sætte mode til pygame zero

Det kan være nødvendigt at ændre dit spil vindues start position med denne kode

SDL\_VIDEO\_WINDOW\_POS=100,100



Hvis det virker som det skal, så lad os tilføje nogle stjerner til baggrunden

Vi vil have stjernerne til at placer sig tilfældigt så vi skal bruge random biblioteket

```
1 #Komet dræber spil - skyd kometerne ned inden de rammer dig!
2 import random
3
4 WIDTH = 600
5 HEIGHT = 800
```

Stjerner skal placeres på tilfældige x og y koordinater og skal have forskellige størrelser.  
vi starter med at generer disse parameter for 100 stjerner, som vi placer i et array kaldet stars

```
7 # Stjerner
8 stars = []
9 for _ in range(100):
10     stars.append({
11         "x": random.randint(0, WIDTH),
12         "y": random.randint(0, HEIGHT),
13         "size": random.choice([1, 2]),
14     })
15
```

Der efter vil vi have dem tegnet på skærmen så vi tilføjer følgende til draw()

```
17 def draw():
18     # Tegn stjerner
19     for star in stars:
20         screen.draw.filled_circle((star["x"], star["y"]), star["size"], "white")
21
```

husk at slette linjen med "pass"

prøv nu spillet af.

ændre lidt på antal og størrelsesintervallet og se resultatet

## Del mål 3 - Rumskib der kan roter og flyve fremad

Vi starter med at definere vores spillers rumskib objekt. Fremadrettet vil vi referere til det som ship

Skibet er en actor og grafiken er vores ship.png fil

Skibet skal til at starte med være placeret midt på skærmen

Tilføj denne kode til starten af programmet lige efter HEIGHT linjen

```
7 ship = Actor('ship')
8 ship.pos = (WIDTH // 2, HEIGHT // 2)
```

Hvis du undere dig over // i stedet for en enkelt /  
så er det en indbygget funktion der sikrer at resultatet er et helt tal eks

$5 / 2 = 2.5$  mens  $5 // 2 = 2$

Opdater draw() med følgende kode

```
26 # Tegn Spiller
27 ship.draw()
```

Test at rumskibet nu vises på skærmen.

Prøv at eksperimentere med forskellen ved at have ship.draw() koden før / efter koden der tegner stjernerne på baggrunden. Kan du se forskellen (hint: sætter stjerne antal op og skift farven)

Nu skal vi have rumskibet til at rotere når man bruger venstre og højre piletaster

Til det skal vi bruge en angle parameter på vores ship objekt.

Tilføj denne kode

```
7 # Spiller
8 ship = Actor('ship')
9 ship.pos = (WIDTH // 2, HEIGHT // 2)
10 ship.angle = 0
```

Og tilføj en ny update() funktion med følgende kode

```
31 def update():
32     # Roter spiller
33     if keyboard.left:
34         ship.angle += 4
35     if keyboard.right:
36         ship.angle -= 4
```

Prøv det af og se om det virker efter hensigten?

Det gør det ikke



Skibet roter, men det gentegner sig bare oven i sig selv.  
Det er fordi vi hver gang draw() kaldes, skal huske at rense spil vinduet.

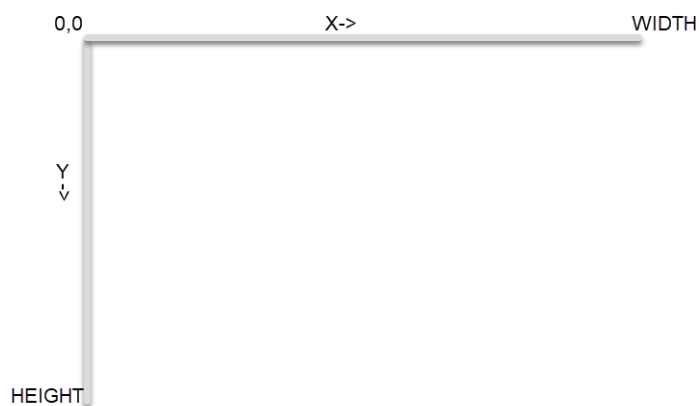
Tilføj denne kode som det første når draw() kaldes

```
23 def draw():
24     # Rens vinduet
25     screen.clear()
26
```

Prøv om det virker bedre nu

Nu skal vi have rumskibet til at kunne flyve fremad

Vi kender koordinat systemet i spillet



Det kan virke lige til at bare ændre ship.y koordinat når der trykkes på piltast op

Prøv at indsætte denne linje i update()

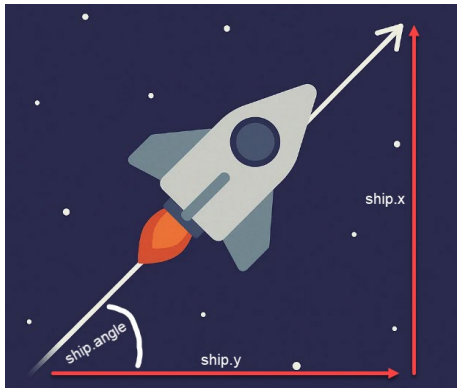
```
42     # Bevæg spiller fremad
43     if keyboard.up:
44         ship.y -= 4
```

Se hvad der sker.

Det virker fint i fremadretning, men hvad nu når skibet er roteret?

Fordi skibet kan rotere bliver vi nød til at nedbryde fremad bevægelsen til hvor meget bevægelsen påvirker skibets x og y værdi

det kan vi gøre på baggrund af retning (ship.angle) og lidt matematik



Ændret fremad koden du lige sat ind til det her

```
42     # Bevæg spiller fremad
43     if keyboard.up:
44         rad = math.radians(ship.angle)
45         ship.x += -math.sin(rad) * 1
46         ship.y -= math.cos(rad) * 1
```

Og da vi bruger math biblioteket skal vi have det sat ind i toppen af filen

```
2     import random
3     import math
```

Prøv nu spillet ad. Er det bedre?

Du kan nok regne ud hvordan du ændre hastigheden på rumskibet!

Når, men i sådan et rumskib spil skal rumskibet jo ikke bare stoppe når man slipper pil op tasten. Så vi skal have skibet til at få fart på og lige så stille bremse op.

vi skal ændre lidt på vores kode fra før så vi gemmer vores x y ændring i nogle nye parametre vx og vy

```
43     # Bevæg spiller fremad
44     if keyboard.up:
45         rad = math.radians(ship.angle)
46         ship.vx += -math.sin(rad) * 1
47         ship.vy -= math.cos(rad) * 1
```

De parametre skal vi tilføje vores ship objekt

```

8 # Spiller
9 ship = Actor('ship')
10 ship.pos = (WIDTH // 2, HEIGHT // 2)
11 ship.angle = 0
12 ship.vx = 0
13 ship.vy = 0

```

I update() indsættes denne nye kode del.

```

43 # Bevæg spiller fremad
44 if keyboard.up:
45     rad = math.radians(ship.angle)
46     ship.vx += -math.sin(rad) * 1
47     ship.vy -= math.cos(rad) * 1
48
49 # Friktion og position
50 ship.x += ship.vx
51 ship.y += ship.vy
52 ship.vx *= 0.98
53 ship.vy *= 0.98

```

bemærk der kun er et enkelt indryk.

det vil sige at fremad handling kun registreres når der trykkes på pil op.

bagefter bliver ændring lagt ind på skibets position.

derefter sker der en reduktion af vx og vy hver gang update() kaldes. (hvilket pygame zero gør automatisk)

Det giver bremse effekten.

Prøv nu spillet

Flyver rumskibet lidt hurtigt? Prøv dig lidt frem med forskellig hastighedsværdier.

Det er lidt irriterende skibet flyver uden for skærm området.

Lad os hurtigt fikse det så rumskibet kommer ud over vindues grænsen flyttes til modsatte kant.

Tilføj denne kode til sidst i update()

```

55 # Skærmgrænser (wrap around)
56 if ship.x < 0:
57     ship.x = WIDTH
58 if ship.x > WIDTH:
59     ship.x = 0
60 if ship.y < 0:
61     ship.y = HEIGHT
62 if ship.y > HEIGHT:
63     ship.y = 0

```

Prøv det af



## Del mål 4 - Rumskibet kan skyde

Nu hvor skibet kan bevæge sig, skal vi også have det til at skyde.  
og skudet skal bevæge sig i den retning skibet har, når skuddet bliver afgivet.

Det første vi gør er at oprette et array til have gemme vores skud i.  
opret arrayet i toppen af program lige over afsnittet "Stjerner"

```
14
15 #Skud
16 bullets = []
17
18 # Stjerner
```

Nu skal vi have skuddet til at blive "skabt" når man klikker på SPACE  
opret denne nye kode i bunden af programmet

```
67 def on_key_down(key):
68     if key == keys.SPACE:
69         bullet = Actor('bullet')
70         bullet.pos = ship.pos
71         bullets.append(bullet)
```

Det vi laver her er at vores funktion venter på en handling (Event)  
handlingen er at en knap bliver trykket ned.  
når en knap bliver trykket ned, så ser vi om det er SPACE  
hvis det er, så oprettes et nyt objekt som vi kalder bullet  
bullet gives samme position som ship.  
derefter tilføjer vi objektet bullet til vores array, som så bliver en liste med skud.

Hvis du starter spillet op nu og trykker på SPACE vil du se at der ikke sker noget.  
Vi bliver nød til først at tegne skuddet på skærmen.

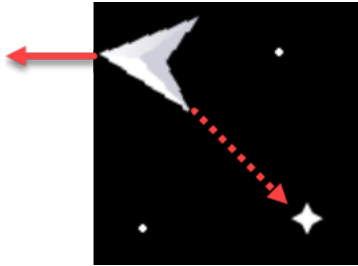
```
27 def draw():
28     # Rens vinduet
29     screen.clear()
30
31     # Tegn stjerner
32     for star in stars:
33         screen.draw.filled_circle((star["x"], star["y"]), star["size"], "white")
34
35     # Tegn Spiller
36     ship.draw()
37
38     # Tegn skud
39     for b in bullets:
40         b.draw()
```

tilføj den markerede kode i draw()  
med koden løber programmet arrayet med skud igennem og tegner alle de skud som er lagt ind i bullets. Prøv nu spillet.

Skuddet bliver tegnet midt i rumskibet, men skuddet bevæger sig ikke.  
tilføj denne kode nederst i update())

```
71     # Skud
72     for b in bullets:
73         b.x += 2
74         b.y += 2
```

Prøv nu spillet



Yes, vi har et skud som bevæger sig. Men det bevæger sig ikke i samme retning som skibet.

Vi må lave lidt kode ændring.

tilføj denne kode til vores event fra tidligere. Bemærk at den sidste linje skal blive ved med at være den sidste linje.

```
79     def on_key_down(key):
80         if key == keys.SPACE:
81             bullet = Actor('bullet')
82             bullet.pos = ship.pos
83             bullet.angle = ship.angle
84             angle_rad = math.radians(bullet.angle)
85             bullet.vx = -math.sin(angle_rad) * 8
86             bullet.vy = -math.cos(angle_rad) * 8
87             bullets.append(bullet)
```

Først sikre vi os at bullet har samme retning som ship.

så laves der omregning til radianer for at vi kan beregne vx, vy, ændrings faktorer.

lige som med ship skal vi nu have de nye ændring i skuddets placering på skærmen opdateret.

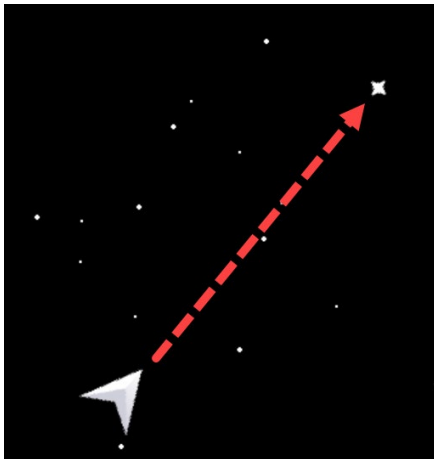
I update() laves denne ændring

<pre>71     # Skud 72     for b in bullets: 73         b.x += 2 74         b.y += 2</pre>	→	<pre>71     # Skud 72     for b in bullets: 73         b.x += b.vx 74         b.y += b.vy</pre>
---	---	---

prøv nu spillet

Virker det?

Ja det ser godt ud.



Men der er lige en ting som skal rettes op. Det kan ikke ses på skærmen men det sker i baggrunden.

Tilføj denne linje kode

```
71     # Skud
72     for b in bullets:
73         b.x += b.vx
74         b.y += b.vy
75     print(len(bullets))
```

kør spillet og se i terminalen. Hver gang vi afgiver et skud bliver skuddet tilføjet arrayet bullets. men skuddet bliver aldrig slettet der fra. Dvs. spillet bliver ved med at bruge ressourcer på skuddet, selvom vi ikke længere har det på skærmen.

lad os ændre koden så skuddet fjernes når det kommer uden for skærmen.

```
71     # Skud
72     for b in bullets:
73         b.x += b.vx
74         b.y += b.vy
75     print(len(bullets))
76     for b in bullets:
77         if not (0 <= b.x <= WIDTH and 0 <= b.y <= HEIGHT):
78             bullets.remove(b)
```

tilføj den markeret kode og prøv spillet af.

nu skal terminalen vise et stigen tal, når et skud afgives, men automatisk tælle ned når skud kommer uden for skærm området.

fjern nu denne linje kode

```
75     print(len(bullets))
```

Nu er del 4 færdig.

## Del mål 5 - Kometer der kommer fra alle side med forskelligt tids interval

Nu kan vores rumskib flyve rund og skyde. Nu skal det have noget at skyde på!

Vi skal have nogle kometer med i spillet.

det er lidt en gentagelse af sidste del mål med skuddene.

1. vi skal oprette et array til at holde styr på kometerne.
2. vi skal oprette nogle kometer (spawn).
3. Vi skal tegne kometerne
4. vi skal bevæge kometerne og slette dem igen når de er uden for skærm området.

Først tilføjer vi kode til at lave et komet array der hedder meteors

```
15 # Skud
16 bullets = []
17
18 # Kometer
19 meteors= []
20
21 # Stjerner
```

Sidst i update() tilføjer vi denne kode som er en trigger der kalder vores meteor spawner

```
82 # Spawn komet
83 if random.random() < 0.03:
84     spawn_meteor()
85
```

Nu skal vi have skrevet vores spawn funktion.

sidst i program filen tilføjes denne kode som er delt i fire dele.

Første del bruger bruger en random funktion til at bestemme om det er en lille eller stor komet.

Anden del bruger først en random funktion til at bestemme hvileken skærm kant kometen skal start fra og derefter en random funktion til at sætte hvor på kanten den starter.

Tredje del sætter en tilfældig retnings ændrings værdi på kometen i henholdsvis x og y retning.

Fjerde del tilføjer meteoren til vores array.

```

97 def spawn_meteor():
98     size = random.choice(["meteor_small", "meteor_large"])
99     meteor = Actor(size)
100
101     start_side = random.choice(['top', 'bottom', 'left', 'right'])
102     if start_side == 'top':
103         meteor.x = random.randint(0, WIDTH)
104         meteor.y = 0
105     elif start_side == 'bottom':
106         meteor.x = random.randint(0, WIDTH)
107         meteor.y = HEIGHT
108     elif start_side == 'left':
109         meteor.x = 0
110         meteor.y = random.randint(0, HEIGHT)
111     elif start_side == 'right':
112         meteor.x = WIDTH
113         meteor.y = random.randint(0, HEIGHT)
114
115     meteor.vx = random.uniform(-1.5, 1.5)
116     meteor.vy = random.uniform(1, 3)
117
118     meteors.append(meteor)

```

Nu er vi klar til at tegne kometerne på skærmen.

Side i draw() tilføjes denne kode

```

41     # Tegn skud
42     for b in bullets:
43         b.draw()
44
45     #Tegn kometer
46     for m in meteors:
47         m.draw()

```

Hvis du starter spillet nu, skal der gerne blive tegnet nogle kometer rundt langs kanten.

Så er vi klar til at få dem til at bevæge sig og slette dem når de kommer udenfor skærmen.

I update() tilføjes denne kode

```

86     # Spawn kometer
87     if random.random() < 0.03:
88         spawn_meteor()
89
90     # Kometerne
91     for m in meteors:
92         m.x += m.vx
93         m.y += m.vy
94         if not (0 <= m.x <= WIDTH and 0 <= m.y <= HEIGHT):
95             meteors.remove(m)

```

Så er del 5 færdig.

Men du kan prøve at lave lidt optimering.

1. juster spawn hastighed
2. ændre så kometer ikke fjernes når de er "halvvejs" udenfor skærm kanten.

## Del mål 6 - Komet ødelægges når den rammes med et skud

Så har vi fået kometer med i vores spil. Nu skal vi kunne skyde dem ned inden de rammer skibet!

Tilføj denne kode i bunden af update()

```
97     # Kollision mellem skud og komet
98     for b in bullets[:]:
99         for m in meteors[:]:
100             if b.colliderect(m):
101                 bullets.remove(b)
102                 meteors.remove(m)
```

Koden løber alle bullets igennem og sammen ligner hver skud med hver komet. Hvis kometen er inden for skydets kollision boks fjernes begge fra deres respektive arrays. Det vil sige at ved næste gentegning kommer de ikke med på skærmen.

Der er dog en ting vi kan optimere her.

Når vi først har fundet en kollision. Så behøver vi ikke tjekke flere kollisioner med andre kometer. Med en break kan vi stoppe den igangværende for loop

```
97     # Kollision mellem skud og komet
98     for b in bullets[:]:
99         for m in meteors[:]:
100             if b.colliderect(m):
101                 bullets.remove(b)
102                 meteors.remove(m)
103                 break
```

Prøv spillet og se hvordan det virker.

Hvad synes du om det?

Det er som om skuddet ikke kommer tæt nok på kometen inden de begge forsvinder fra skærmen.

Lad os bygge lidt mere kode på der tjekker afstanden mellem skud og komet når der er detekteret en kollision. Og kun hvis afstanden er mindre en kometens halve brede slettes skud og komet fra skærmen.

```
97     # Kollision mellem skud og komet
98     for b in bullets[:]:
99         for m in meteors[:]:
100             if b.colliderect(m):
101                 dx = m.x - b.x
102                 dy = m.y - b.y
103                 distance = math.hypot(dx, dy)
104                 if distance <= (m.width/2):
105                     bullets.remove(b)
106                     meteors.remove(m)
107                     break
```

Det er noget bedre.

men kig lige en gang på koden. I den bruges skuddets og kometens x,y værdier.

de beregnet ovenfor kollisions koden. Men det er først bagefter update() at draw() som tegner på skærmen kaldes.

det betyder at rækkefølgende er således

1. beregn ny pos af skud
2. beregn ny pos af komet
3. tag de nye pos og tjek for kollision
4. slet skyd og komet hvis der er kollision
5. tegn på skærm hvis ikke der er kollision

det er lidt underligt. Dette vil give mere mening

1. tag de gamle pos og tjek for kollision
2. slet skyd og komet hvis der er kollision
3. beregn ny pos af skud
4. beregn ny pos af komet
5. tegn på skærm

der er heldigvis nemt at ændre.

Flyt vores kode op foran afsnittet med "Skud" i update()

```
78 # Kollision mellem skud og komet
79 for b in bullets[:]:
80     for m in meteors[:]:
81         if b.collidect(m):
82             dx = m.x - b.x
83             dy = m.y - b.y
84             distance = math.hypot(dx, dy)
85             if distance <= (m.width/2):
86                 bullets.remove(b)
87                 meteors.remove(m)
88                 break
89
90 # Skud
```



Nu er det meget nemt at kopier koden og lave få tilpasning så en komet der rammer skibet også får en komet til at forsvinde.

lige efter den indsatte kode fra før, tilføj denne kode

```
90     # Kollision med komet og spiller
91     for m in meteors:
92         if m.collidect(ship):
93             dx = ship.x - m.x
94             dy = ship.y - m.y
95             distance = math.hypot(dx, dy)
96             if distance <= (m.width/2):
97                 meteors.remove(m)
98                 ship.pos = WIDTH/2, HEIGHT/2
99                 break
```

Bemærk at i stedet for at fjerne skibet flyttes skibet tilbage til midten af skærmen.

Hermed er del mål 6 færdig.

## Del mål 7 - Point system

Så er vi kommet til den sidste del af spillet point system.  
der er flere trin i det.

1. Spil start 0 point
2. point score skal vises i skærmen stop
3. når man skyder en komet ned skal point score øges med 1

Lad os starte med at definere en variable til at holde vores score

Tilføj i starten af program filen denne kode efter "Stjerner" afsnittet

```
30 # Point
31 score = 0
```

Tilføj sidst i draw()

```
52 # Point
53 screen.draw.text(f"Point: {score}", (10, 10), fontsize=40, color="white")
```

Find nu koden for kollision mellem skud og komet og tilføj disse to kode linjer

```
84 # Kollision mellem skud og komet
85 for b in bullets[:]:
86     for m in meteors[:]:
87         if b.collidect(m):
88             dx = m.x - b.x
89             dy = m.y - b.y
90             distance = math.hypot(dx, dy)
91             if distance <= (m.width/2):
92                 global score
93                 score += 1
94                 bullets.remove(b)
95                 meteors.remove(m)
96                 break
```

Prøv spillet. Virker det?

Så er vi færdig med del mål 7

## Bonus Del mål – Liv og game over, genstart

Nu har vi et spil og det er super godt.

men for at lave det lidt mere interessant vil vi implementere live og game over mode.

Vi vil gøre det i denne rækkefølge

1. Variable for liv
2. Reduktion af liv når rumskibet rammes af en komet
3. Game over når man ikke har flere liv
4. Genstart af spillet

Lave disse ændring

Variable for liv




```
30 # Point
31 score = 0
```

```
30 # Game
31 score = 0
32 lives = 3
```

Visning af liv


```
52 # Point
53 screen.draw.text(f"Point: {score}", (10, 10), fontsize=40, color="white")
```



```
54 # Game
55 screen.draw.text(f"Point: {score}", (10, 10), fontsize=40, color="white")
56 screen.draw.text(f"Lives: {lives}", (10,50), fontsize=40, color="white")
```

Prøv det af og om nødvendigt flyt lidt rundt på teksten

Træk en fra lives når skibet rammes af komet



```
90 # Kollision med komet og spiller
91 for m in meteors:
92     if m.colliderect(ship):
93         dx = ship.x - m.x
94         dy = ship.y - m.y
95         distance = math.hypot(dx, dy)
96         if distance <= (m.width/2):
97             meteors.remove(m)
98             ship.pos = WIDTH/2, HEIGHT/2
99             break
```


```
103 # Kollision med komet og spiller
104 for m in meteors:
105     if m.colliderect(ship):
106         dx = ship.x - m.x
107         dy = ship.y - m.y
108         distance = math.hypot(dx, dy)
109         if distance <= (m.width/2):
110             meteors.remove(m)
111             global lives
112             lives -= 1
113             ship.pos = WIDTH/2, HEIGHT/2
114             break
```

Prøv det af

Nu skal vi have lavet så spillet er game over hvis man har mindre end 0 liv

Opret variable for game over


```
30 # Game
31 score = 0
32 lives = 3
```



```
30 # Game
31 score = 0
32 lives = 1
33 game_over = False
```

sæt variablen true hvis man ikke har flere liv

```
103 # Kollision med komet og spiller
104 for m in meteors:
105     if m.collidrect(ship):
106         dx = ship.x - m.x
107         dy = ship.y - m.y
108         distance = math.hypot(dx, dy)
109         if distance <= (m.width/2):
110             meteors.remove(m)
111             global lives
112             lives -= 1
113             ship.pos = WIDTH/2, HEIGHT/2
114             break
```



```
110 # Kollision med komet og spiller
111 for m in meteors:
112     if m.collidrect(ship):
113         dx = ship.x - m.x
114         dy = ship.y - m.y
115         distance = math.hypot(dx, dy)
116         if distance <= (m.width/2):
117             meteors.remove(m)
118             global lives
119             lives -= 1
120             if lives < 0:
121                 game_over = True
122             else:
123                 ship.pos = WIDTH/2, HEIGHT/2
124             break
```

Sæt denne kode ind i toppen af update()


den stopper alt opdatering i spillet og fastfryser alt på skærmen

```
63 def update():
64     global game_over
65     if game_over:
66         return
```

Prøv spillet. Det kan være en fordel at sætte lives = 1 når man tester og man kan prøve sig frem om man vil lave  
if lives < 0 eller if lives < 1


Det vil selvfølgelig undre spilleren. Så lav denne ændring for at vise forklarende tekst på skærmen

```
54 # Game
55 screen.draw.text(f"Point: {score}", (10, 10), fontsize=40, color="white")
56 screen.draw.text(f"Lives: {lives}", (10, 50), fontsize=40, color="white")
```



```
54 # Game
55 if game_over:
56     screen.draw.text("GAME OVER", center=(WIDTH/2, HEIGHT/2), fontsize=60, color="red", owidth=1.0)
57     screen.draw.text("Tryk R for at genstarte", center=(WIDTH/2, HEIGHT/2+50), fontsize=40, color="white")
58 else:
59     screen.draw.text(f"Point: {score}", (10, 10), fontsize=40, color="white")
60     screen.draw.text(f"Lives: {lives}", (10, 50), fontsize=40, color="white")
```

Nu siger teksten at man kan genstarte spillet når man trykker på R  
så det må vi lave en håndtering af



```
146 def on_key_down(key):
147     # Afgiv skud
148     if key == keys.SPACE:
149         bullet = Actor('bullet')
150         bullet.pos = ship.pos
151         bullet.angle = ship.angle
152         angle_rad = math.radians(bullet.angle)
153         bullet.vx = -math.sin(angle_rad) * 8
154         bullet.vy = -math.cos(angle_rad) * 8
155         bullets.append(bullet)

146 def on_key_down(key):
147     # Afgiv skud
148     if key == keys.SPACE:
149         bullet = Actor('bullet')
150         bullet.pos = ship.pos
151         bullet.angle = ship.angle
152         angle_rad = math.radians(bullet.angle)
153         bullet.vx = -math.sin(angle_rad) * 8
154         bullet.vy = -math.cos(angle_rad) * 8
155         bullets.append(bullet)
156     elif key == keys.R and game_over:
157         reset_game()
```

Nu kaldes reset\_game() når spillet er game over og man trykker på R

Tilføj derfor reset\_game() i bunden af program filen

```
182 def reset_game():
183     global game_over, score, lives, meteors, bullets
184     game_over = False
185     score = 0
186     lives = 3
187     meteors = []
188     bullets = []
189     ship.pos = (WIDTH/2, HEIGHT/2)
190     ship.angle = 0
191     ship.vx = 0
192     ship.vy = 0
```

I denne funktion sætter vi spillet tilbage til start indstillinger.

## Flere del mål – Arbejdet selv videre

Hvad kan du selv finde på at tilføje?

Eks. eksplosion ved kollision, power-ups, ekstra liv, øget sværhedsgrad