Python Game

Opdrachtbeschrijving

[gees\_at\_stanislascollege\_punt\_nl](mailto:gees@stanislascollege.nl)  
29-12-2023

Afbeelding met elektronica, Invoerapparaat, Elektronisch apparaat, randapparaat

Automatisch gegenereerde beschrijving

**Afbeelding met schermopname, tekst, software, Multimediasoftware

Automatisch gegenereerde beschrijving**

Inhoudsopgave

[Inleiding 3](#_Toc154763360)

[Leeswijzer 3](#_Toc154763361)

[Voorkennis en naslagmateriaal 3](#_Toc154763362)

[Tools 4](#_Toc154763363)

[Werkwijze 4](#_Toc154763364)

[Inleveren 4](#_Toc154763365)

[Beoordeling 4](#_Toc154763366)

[Basisstappen 5](#_Toc154763367)

[Stap 1. Voeg commentaar toe 5](#_Toc154763368)

[Stap 2. Beweeg de bal schuin 7](#_Toc154763369)

[Stap 3. Stuiter de bal tegen de onder- en bovenkant van het scherm 7](#_Toc154763370)

[Stap 4. Teken de plank 8](#_Toc154763371)

[Stap 5. Beweeg de plank 8](#_Toc154763372)

[Stap 6. Stop de plank aan de randen van het scherm 9](#_Toc154763373)

[Stap 7. Stuiter de bal tegen de plank 9](#_Toc154763374)

[Stap 8. Stop het spel als je af bent 10](#_Toc154763375)

[Stap 9. Toon een bericht als je af bent 10](#_Toc154763376)

[Stap 10. Teken een blok in het veld 11](#_Toc154763377)

[Stap 11. Detecteer als de bal het blok raakt 11](#_Toc154763378)

[Stap 12. Stuiter de bal omhoog als hij het blok raakt 12](#_Toc154763379)

[Stap 13. Stuiter de bal omlaag als hij het blok raakt 12](#_Toc154763380)

[Stap 14. Stuiter de bal links of rechts als hij het blok raakt 13](#_Toc154763381)

[Stap 15. Maak een tweede blok 13](#_Toc154763382)

[Stap 16. Maak een derde en vierde blok 14](#_Toc154763383)

[Stap 17. Zet blokken in genummerde lijsten 15](#_Toc154763384)

[Stap 18. Gebruik iteratie bij blokken in genummerde lijsten 15](#_Toc154763385)

[Stap 19. Maak een veld met 24 blokken 16](#_Toc154763386)

[Stap 20. Haal een blok weg als de bal een blok raakt 16](#_Toc154763387)

[Stap 21. Toon bericht als je wint 16](#_Toc154763388)

[Uitbreidingen 17](#_Toc154763389)

[Uitbreiding: Bal die steeds sneller gaat ⭐️ 17](#_Toc154763390)

[Uitbreiding: Blokken die van kleur veranderen ⭐️ 17](#_Toc154763391)

[Uitbreiding: Beter kaats-algoritme voor plankje ⭐️ 17](#_Toc154763392)

[Uitbreiding: Uitlegscherm en gameoverscherm⭐️⭐️ 17](#_Toc154763393)

[Uitbreiding: Animaties als een blok verdwijnt ⭐️⭐️ 17](#_Toc154763394)

[Uitbreiding: Meerdere levels ⭐️⭐️ 17](#_Toc154763395)

[Uitbreiding: Blokken die omlaag vallen ⭐️⭐️⭐️ 17](#_Toc154763396)

[Uitbreiding: Meerdere ballen tegelijk ⭐️⭐️⭐️ 18](#_Toc154763397)

[Uitbreiding: Blokken wegschieten ⭐️⭐️⭐️ 18](#_Toc154763398)

[Uitbreiding: Overleg met je docent voor andere uitbreidingen 18](#_Toc154763399)

# Inleiding

## Leeswijzer

Dit is een lange opdracht. Je maakt een eenvoudig spel waarin je één level kunt spelen in de basisstappen. De basisstappen zijn voor iedereen hetzelfde. Daarna kun je je spel verder uitbreiden met uitbreidingen. De uitbreidingen kunnen verschillend zijn per groepje.

Elke basisstap bestaat uit een opdracht, toelichting en tips. In de opdracht staat wat je code moet doen. Lees dit aandachtig door, voordat je begint te coderen. De toelichting geeft aan hoe je de opdracht kunt maken. Een ervaren programmeur kan vaak zonder toelichting de opdracht maken. De tips kun je bekijken als je er niet uitkomt. Bekijk de tips één voor één van boven naar beneden en probeer na elke tip of je verder met de opdracht komt. Bij sommige stappen zijn de laatste tips bedoeld om je meer aan het denken te zetten over wat je gemaakt hebt.

Bij de uitbreidingen heb je meer vrijheid. Ze zijn minder ver uitgewerkt. Daardoor zijn die moeilijker dan de basisstappen. Je leert zelfstandig kleine of grotere uitbreidingen toe te voegen aan bestaande code. Hoe beter je daarin wordt, hoe meer functies uit je eigen fantasie jij kunt maken!

## Voorkennis en naslagmateriaal

Voor deze opdracht heb je basiskennis nodig van programmeren in Python. We gaan ervan uit dat je geoefend hebt met variabelen, selectie (if-jes), iteratie (for), functies en lijsten. Je kunt deze onderwerpen nog eens nakijken in [Fundament Kernprogramma, Domein D: Programmeren -> Ontwikkelen met Python](https://fundament-online.nl/leeromgeving/hoofdstuk.php?id=10500).

Deze opdracht maakt gebruik van Pygame. Pygame is een package voor Python waarmee je gemakkelijker grafische spelletjes kunt programmeren. Je hoeft niet eerder met Pygame geoefend te hebben om deze opdracht te kunnen maken.

Als je meer over Python of Pygame wilt weten, dan kun je de volgende naslagwerken gebruiken:

* Uitgebreide informatie over Python,   
  Begin bij de Tutorial, daarna kun je de Library Reference bekijken, als je alles wilt weten dan bekijk je daarna de Language Reference  
  <https://docs.python.org/3/>
* Alle mogelijkheden van de pygame library kun je in deze reference guide opzoeken  
  <https://www.pygame.org/docs/>
* Als je de Pygame library beter wilt begrijpen, bekijk dan deze tutorial  
  <https://realpython.com/pygame-a-primer/>
* Meer over algoritmes in games leer je bij de CS50-cursus van Harvard  
  <https://cs50.harvard.edu/games/2018/>
* Meer plaatjes kun je halen van Game Art  
  <https://opengameart.org/content/breakout-brick-breaker-tile-set-free>

## Tools

Voordat je kunt programmeren moet je bekend zijn met het gereedschap (de tools) dat je voor deze opdracht gaat gebruiken.

We raden aan om deze opdracht te maken in GitHub met Codespaces. Dit is een professionele online omgeving die door veel programmeurs gebruikt wordt. GitHub met CodeSpaces werkt ook op Chromebooks. Een account op GitHub is gratis en daarmee kun je 60 uur per maand gebruik maken van Codespaces.

Meer informatie over GitHub en Codespaces vind je op <https://informatica.emmauscollege.nl/tools/github-codespaces/>.

Als je met Git wilt oefenen, dan kun je dat doen op

[Fundament - Kernprogramma A: Vaardigheden 9. Versiebeheer met Git](https://fundament-online.nl/leeromgeving/hoofdstuk.php?id=11352)

De opdracht zou moeten werken op een computer waarop Python en het Pygame package lokaal geïnstalleerd zijn, maar dit is niet getest. Op Replit kan deze opdracht niet gemaakt worden. De gratis versie van Replit biedt niet voldoende CPU-capaciteit om een spel dat in Python en Pygame gemaakt is soepel te spelen.

## Werkwijze

* Je maakt deze opdracht in een team van twee personen. De docent geeft aan hoe de teams gemaakt worden.
* Je gebruikt de startcode die je voor deze opdracht krijgt van de docent.
* Je volgt het stappenplan in de opdracht. Elke stap of uitbreiding sla je op in GitHub, je maakt per stap minimaal één commit. Uit je commit-message blijkt bij welke stap of uitbreiding de commit hoort.
* Je verdeelt het werk eerlijk over de teamleden. Elk teamlid codeert evenveel. Als je samen achter één computer zit, dan wissel je elke stap wie er typt. Elke leerling typt op zijn eigen GitHub-account.
* Problemen binnen het team meld je zo snel mogelijk bij de docent.

## Inleveren

* De deadline staat in de lesplanner.
* De versie van de main branche van je opdracht die op het moment van de deadline in GitHub staat wordt gebruikt voor de beoordeling. Zorg dat je ruim op tijd klaar bent.
* Uit de commit-historie van de main branche in GitHub blijkt wie wanneer wat gedaan heeft.
* Je kunt vragen stellen tot de laatste les voor de deadline.

## Beoordeling

Je krijgt één cijfer per team, maar de docent kan hiervan afwijken als teamleden geen gelijkwaardige bijdrage hebben geleverd.

Er gelden minimale eisen waaraan je opdracht moet voldoen:

* Je hebt je gehouden aan de voorgeschreven werkwijze.
* Je opdracht is fatsoenlijk, dus vrij van beledigende of illegale elementen.
* Je hebt de opdracht helemaal zelf gemaakt, dus zonder plagiaat.

Opdrachten die niet voldoen aan de minimale eisen krijgen het cijfer 1,0.

Een netjes uitgevoerde opdracht met alle basisstappen zal in de meeste gevallen beoordeeld worden met een voldoende. Om een hoog cijfer te halen moet je creatieve en complexe uitbreidingen toevoegen.

Een concept beoordelingsmodel staat hieronder. De docent kan tijdens het nakijken aanpassingen doen aan het beoordelingsmodel.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | **Code** | | | | **Functionaliteit** | | **Proces** | **Uitbreidingen** |
|  |  | Handmatige correctie |  | code stijl | selectie | herhaling | lijsten | speelbaarheid | basisfunctionaliteit | commits | sterren |
| **Groep** | **Namen** |  | **Cijfer** | **10** | **10** | **10** | **10** | **10** | **10** | **10** | **20** |
| **Groep 1** |  |  | 1,0 |  |  |  |  |  |  |  |  |

# Basisstappen

## Stap 1. Voeg commentaar toe

### Opdracht

Voeg commentaar toe aan de code. Bovenaan de code, onder de naam van de game, maak je een nieuwe regel met commentaar. Het commentaar is de voornaam van iedereen die aan deze code werkt. Boven elk van de vijf onderdelen (define constants, define global variables, init game, read images en game loop) zet je een beschrijving wat het doet. Een beschrijving is één of enkele regels.

### Toelichting

Code die constanten en variabelen maakt:

# define constants

FPS = 30 # Frames Per Second

SCREEN\_WIDTH = 1280

# define global variables

ball\_x = 0

ball\_speed\_x = 6

Code die de game initialiseert:

# init game

pygame.init()

font = pygame.font.SysFont('default', 64)

screen = pygame.display.set\_mode((SCREEN\_WIDTH, SCREEN\_HEIGHT), pygame.FULLSCREEN)

fps\_clock = pygame.time.Clock()

Code die plaatjes inleest:

# read spridesheet

spritesheet = pygame.image.load('Breakout\_Tile\_Free.png').convert\_alpha()

# create new image

ball\_img = pygame.Surface((64, 64))

# copy part of sheet to image

ball\_img.blit(spritesheet, (0, 0), (1403, 652, 64, 64))

# resize image

ball\_img = pygame.transform.scale(ball\_img, (BALL\_WIDTH, BALL\_HEIGHT))

Gameloop op hoofdlijnen, de code in de gameloop wordt steeds herhaalt:

running = True

while running:

# read events

# move everything

# draw everything

# wait until next frame

Stukje code uit gameloop die zorgt dat invoer vanuit besturingssysteem wordt afgehandeld:

# read all events

# to end programme when GUI is closed

# and to read keys using pygame.get.get\_pressed()

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

Stukje code uit gameloop die alles beweegt:

# move everything

# move ball

ball\_x = ball\_x + ball\_speed\_x

Stukje code uit gameloop die alles tekent:

# clear screen

screen.fill('black')

# draw ball

screen.blit(ball\_img, (ball\_x, 0))

# show screen

pygame.display.flip()

Stukje code uit gameloop die frame-snelheid regelt:

# Sleep the remaining time of this frame

fps\_clock.tick(FPS)

### Tips:

* Een beschrijving mag meerdere regels lang zijn. Elke regel begint met # .
* Hoe werkt het tekenen op het scherm in de pygame-library? In welke regels code worden de plaatjes ingelezen? In welke regels code worden plaatjes op het scherm gezet? In welke regel wordt het nieuwe scherm op de monitor getoond?
* Hoe wordt het aantal frames per seconde geregeld?
* Wat moet je veranderen aan de code om de bal op een andere plek te laten starten?

## Stap 2. Beweeg de bal schuin

### Opdracht

De bal beweegt nu horizontaal over het scherm. Voeg code toe die ervoor zorgt dat de bal schuin over het scherm beweegt. Gebruik de variabelen ball\_y en ball\_speed\_y.

### Toelichting

Regels uit de startcode die de bal horizontaal bewegen:

ball\_x = 0

ball\_speed\_x = 6

# move ball

ball\_x = ball\_x + ball\_speed\_x

# draw ball

screen.blit(ball\_img, (ball\_x, 0))

### Tips:

* Maak een variabele ball\_y en geef die een waarde, bijvoorbeeld 100.
* Gebruik de waarde van ball\_y bij de opdracht die de bal op het scherm zet. Dit is de opdracht screen.blit.
* Maak een variabele ball\_speed\_y en geef die een waarde ongelijk aan 0, bijvoorbeeld 10.
* Kopieer de regel code waarin steeds de waarde van ball\_x wordt berekend. Pas de kopie aan, zodat de waarde van ball\_y wordt berekend. Gebruik daarbij variabele ball\_speed\_y.
* Zorg dat ball

## Stap 3. Stuiter de bal tegen de onder- en bovenkant van het scherm

### Opdracht

De bal verdwijnt nu van het scherm als hij aan de bovenkant of de onderkant komt. Zorg dat de bal tegen de onderrand en bovenrand van het scherm stuitert, net zoals hij dat doet aan de zijkanten. Gebruik BALL\_HEIGHT om rekening te houden met de hoogte van de bal.

### Toelichting

Code die zorgt dat bal stuitert tegen zijkant van scherm:

# if ball is beyond left edge or ball is beyond right edge

if ball\_x < 0 or ball\_x + BALL\_WIDTH > SCREEN\_WIDTH:

# reverse horizontal direction of ball

ball\_speed\_x = ball\_speed\_x \* -1

### Tips:

* Zoek de regels code die ervoor zorgen dat de bal tegen de zijkanten van het scherm kaatst. Kopieer dat stukje code en pas het aan zodat dat de bal ook tegen de bovenkant en onderkant van het scherm kaatst.

## Stap 4. Teken de plank

### Opdracht

Teken de plank op het scherm. Gebruik de variabele paddle\_img om het plaatje van de plank in op te slaan. Gebruik de variabelen paddle\_x en paddle\_y om de linkerbovenhoek van de plank aan te geven. Maak de startpositie van de plank ongeveer midden onder op het scherm. Gebruik de constanten PADDLE\_WIDTH en PADDLE\_HEIGHT voor de breedte en hoogte van de plank. Maak je plank 144 pixels breed en 32 pixels hoog.

### Toelichting

Regels uit de startcode die het plaatje van de bal inlezen, met extra commentaar:

# create new image

# img = pygame.Surface((width, height))

ball\_img = pygame.Surface((64, 64))

# copy part of sheet to image

destination\_image.blit(source\_spridesheet,

(x, y), # of where you want your image to go, 0,0 is top left corner

(x, y, width, height)) # of part that you want from spritesheet

ball\_img.blit(spritesheet, (0, 0), (1403, 652, 64, 64))

# resize image

# new\_image = pygame.transform.scale(old\_image, (new width, new height))

ball\_img = pygame.transform.scale(ball\_img, (BALL\_WIDTH, BALL\_HEIGHT))

### Tips:

* Zoek uit op welke x en y in het spritesheet een plank zit.
* Definieer de constanten PADDLE\_WIDTH en PADDLE\_HEIGHT en geef ze de waarde 144 en 32.
* Bekijk de code in het blok onder # read images om te zien hoe je het stukje van de spritesheet waarop de plank staan kunt opslaan in de variabele paddle\_img. Je moet het plaatje kopiëren en de juiste afmeting maken.
* Maak de variabelen paddle\_x en paddle\_y en geef ze een waarde, bijvoorbeeld SCREEN\_WIDTH / 2 en SCREEN\_HEIGHT – 100.
* In het blok onder # game loop staat een regel code die het plaatje van de bal op je scherm zet. Kopieer die regel code en pas hem aan zodat de plank op het scherm verschijnt.

## Stap 5. Beweeg de plank

### Opdracht

Beweeg de plank als de D (rechts) of A (links) zijn ingedrukt. Beweeg de plank met één pixel per frame. Gebruik pygame.key.get\_pressed() om te zien welke toetsen momenteel worden ingedrukt.

### Toelichting

Kijken of de D is ingedrukt, doe je zo:

keys = pygame.key.get\_pressed()

if keys[pygame.K\_d] : # key d is down

### Tips:

* Als de toets D is ingedrukt, beweeg je de plank naar rechts door bij paddle\_x één op te tellen. De plank wordt daardoor verderop in de code op een nieuwe plek getekend.
* Als de toets A is ingedrukt, beweeg de plank naar links.
* Gebruik maar één keer de regel keys = pygame.key.get\_pressed() om de stand van de toetsen te lezen

## Stap 6. Stop de plank aan de randen van het scherm

### Opdracht

Zorg dat de plank niet verder beweegt als hij de rand van het scherm raakt. Gebruik paddle\_x en SCREEN\_WIDTH om te kijken of de plank aan de rand is. Gebruik PADDLE\_WIDTH om rekening te houden met de breedte van de plank.

### Toelichting

Pseudo-code om plank te laten stoppen aan de rechterkant van het scherm.

plank\_x = plank\_x + stapje

als plank\_x + plank\_breedte > breedte\_van\_scherm dan

plank\_x = breedte\_van\_scherm – plank\_breedte

Maak zelf de code en zet die op een logische plaats in je programma.

Pseudo-code om plank te laten stoppen aan de linker van het scherm kun je zelf maken.

### Tips:

* plank\_x in de pseudo-code is paddle\_x in je code,   
  breedte\_van\_scherm in de pseudo-code is SCREEN\_WIDTH uit je code en   
  plank\_breedte uit de pseudo-code is PADDLE\_WIDTH in je code.

## Stap 7. Stuiter de bal tegen de plank

### Opdracht

Zorg dat de bal kaatst tegen je plank. Hiervoor zijn veel algoritmen te bedenken. Ons algoritme kijkt of een stukje van de bal op dezelfde plek is als een stukje van de plank. Als dat zo is dan wordt de beweegrichting van de bal in y-richting omgedraaid. Omdraaien doe je door de waarde te vermenigvuldigen met min één.

### Toelichting

Pseudo-code:

als rechter\_kant\_bal is groter dan linker\_kant\_paddle en

linker\_kant\_bal is kleiner dan rechter\_kant\_paddle en

onderkant\_kant\_bal is groter dan boven\_kant\_paddle en

boven\_kant\_bal is kleiner dan onder\_kant\_paddle dan

bal\_snelheid\_y = bal\_snelheid\_y \* -1

### Tips:

* Gebruik de pseudo-code. Gebruik nu geen ingebouwde functie van pygame om een botsing te detecteren, want dan leer je minder goed programmeren.
* Je kunt een lange conditie achter if over meerdere regels spreiden om het leesbaarder te maken. Zet de conditie dan tussen ( en ), bijvoorbeeld  
  if (x > 100 and   
   x < 200 and   
   y > 50 and   
   y < 150) :
* rechter\_kant\_bal uit de pseudo-code is ball\_x + BALL\_WIDTH in je code.
* linker\_kant\_plank uit de pseudo-code is paddle\_x in je code.
* Wat doet ons algoritme als de bal op een hoek van de plank kaatst? Wat zou je in het echt verwachten?
* Hou het algoritme voor nu simpel, je kunt het later nog aanpassen.

## Stap 8. Stop het spel als je af bent

### Opdracht

Zorg dat je af bent als de bal de onderkant van het scherm raakt. Voeg daarvoor code toe die kijkt of de onderkant van de bal voorbij de onderkant van de plank is. Als dat waar is, dan zet je code de snelheid van de bal op 0. Op die manier stopt het spel zodra je de bal langs de plank hebt laten gaan.

### Toelichting

Een logische plek voor je nieuwe code is na het bewegen van alles en voor je begint met tekenen. Dat ziet er zo uit:

# move everything

...

# check if you lost

...

# clear screen

...

### Tips:

* Wat gebeurt er met de code de bal tegen de onderkant van het scherm laat stuiteren? Wordt die code nog uitgevoerd?

## Stap 9. Toon een bericht als je af bent

### Opdracht

Zet een bericht op het scherm als de speler af is. Dat is duidelijker voor de speler.

### Toelichting

Dit stukje code zet een bericht op het scherm. Knip en plak het op de juiste plekken.

# define global variables

game\_status\_msg = ""

# if dead

game\_status\_msg = "You lost!"

# draw game status message

game\_status\_img = font.render(game\_status\_msg, True, 'green')

screen.blit(game\_status\_img, (0, 0)) # (0, 0) is top left corner of screen

### Tips:

* Je kunt tijdens het spelen een ander bericht op het scherm zetten, bijvoorbeeld:

game\_status\_msg = "Speel met [A] en [D]"

* Je kunt het bericht netjes in het midden van de regel uitlijnen met een berekening met SCREEN\_WIDTH en game\_status\_img.get\_width.

## Stap 10. Teken een blok in het veld

### Opdracht

Teken een blok op het scherm. Gebruik de variabele brick\_img om het plaatje van het blok op te slaan. Maak en gebruik de constanten BRICK\_WIDTH en BRICK\_HEIGHT voor de breedte en hoogte van het blok. Het blok is 96 pixels breed en 32 pixels hoog. Maak en gebruik de variabelen brick\_x en brick\_y voor de linkerbovenhoek van het blok. Kies zelf op welke plaats je het blok op het scherm wilt zetten.

### Tips:

* Bekijk “Stap 4. Teken de plank” als je er niet uitkomt.

## Stap 11. Detecteer als de bal het blok raakt

### Opdracht

Druk een tekst af in de terminal als de bal het blok raakt.

Een tekst in de terminal afdrukken doe je zo:

print('brick touched at ball\_x = ' + str(ball\_x) + ' and ball\_y = ' + str(ball\_y))

Python plakt teksten aan elkaar als je ze optelt. De functie str zet een getal om in een string. Deze functie heb je nodig, omdat python getallen en strings niet automatisch bij elkaar optelt, je krijgt dan een foutmelding.

### Tips:

* Bekijk “Stap 7. Stuiter de bal tegen de plank” als je niet meer weet hoe je kunt zien of twee voorwerpen elkaar raken.
* Meer informatie over het printen van tekst vind je op <https://www.freecodecamp.org/news/python-print-variable-how-to-print-a-string-and-variable/>
* Je vraagt je misschien af hoe je het blok weghaalt. Dat leer je in “Stap 20, Haal een blok weg als de bal een blok raakt”.

## Stap 12. Stuiter de bal omhoog als hij het blok raakt

### Opdracht

Stuiter de bal omhoog als hij van boven aankomt en een blok raakt. Ga ervan uit dat de beweging van de bal in de x- en y-richting minder pixels per frame is dan de breedte en hoogte van de bal. Dat betekent dat er altijd een stukje van de bal buiten het blok is. De stuiter vanaf beneden, links en rechts ga je in de volgende opdrachten toevoegen.

### Toelichting

Pseudo-code

*als de bal het blok raakt :*

*druk een tekst af*

als ( de bal naar onder beweegt en

de bovenkant van de bal zit boven de bovenkant van het blok ) :

draai snelheid\_y om

De schuingedrukte regels heb je al gemaakt in de vorige stap. De rest moet je nog toevoegen.

Er zit twee keer “als” in de pseudo-code. De tweede “als” wordt alleen uitgevoerd als de eerste “als” waar is. Je kunt deze code combineren in één “als”, door de condities uit beide “als”-en aan elkaar te knopen met “and”. In de volgende opdrachten ga je zien dat dat in dit geval minder duidelijk is, omdat er dan dubbele code ontstaat.

### Tips:

* Met if ball\_speed\_y > 0 : kun je kijken of de bal naar onder beweegt.
* De bovenkant van de bal is ball\_y.
* De bovenkant van het blok is brick\_y.
* De snelheid draai je om door deze te vermenigvuldigen met -1. Kijk in “Stap 3. Stuiter de bal tegen de onder- en bovenkant van het scherm” als je niet meer weet hoe dat moet.
* Kun je uitleggen waarom we bij de tweede “als” kijken of er een stukje bal buiten het blok zit, in plaats van kijken of er een stukje bal binnen het blok zit?

## Stap 13. Stuiter de bal omlaag als hij het blok raakt

### Opdracht

Voeg code toe die de bal omlaag stuitert als hij van beneden aankomt en een blok raakt.

### Toelichting

Pseudo-code

*als de bal het blok raakt :*

*druk een tekst af*

*als ( de bal naar onder beweegt en*

*de bovenkant van de bal zit boven de bovenkant van het blok ) :*

*draai snelheid\_y om*

als ( de bal naar boven beweegt en

de onderkant van de bal zit onder de onderkant van het blok ) :

draai snelheid\_y om

De schuingedrukte regels heb je al gemaakt in de vorige stap. De rest moet je nog toevoegen. De tweede en de derde “als” kun je combineren in één “als”. Dat scheelt één dubbele regel code om de snelheid om te draaien, maar het wordt iets moeilijker leesbaar. Daarom kiezen ervoor om dat niet te doen.

### Tips:

* Met if ball\_speed\_y < 0 : kun je kijken of de bal naar boven beweegt.
* De onderkant van de bal is ball\_y + BALL\_HEIGHT.
* De onderkant van het blok is brick\_y + BRICK\_HEIGHT.
* Hoe zou de code er uitzien als je de tweede en derde “als” combineert?

## Stap 14. Stuiter de bal links of rechts als hij het blok raakt

### Opdracht

Voeg code toe die de bal stuitert als hij van links of rechts aankomt en een blok raakt.

### Toelichting

Pseudo-code

*als de bal het blok raakt :*

*druk een tekst af*

*als ( de bal naar onder beweegt en*

*de bovenkant van de bal zit boven de bovenkant van het blok ) :*

*draai snelheid\_y om*

*als ( de bal naar boven beweegt en*

*de onderkant van de bal zit onder de onderkant van het blok ) :*

*draai snelheid\_y om*

als ( de bal naar rechts beweegt en

de linkerkant van de bal zit links van de linkerkant van het blok ) :

draai snelheid\_x om

als ( de bal naar links beweegt en

de rechterkant van de bal zit rechts van de rechterkant van het blok ) :

draai snelheid\_y om

De schuingedrukte regels heb je al gemaakt in de vorige stap. De rest moet je nog toevoegen.

### Tips:

* Kijk naar de vorige 2 stappen, kopieer de code en pas hem aan zodat hij voor x werkt in plaats van voor y.
* Er zijn andere pseudo-codes mogelijk die een bal die ergens tegenaan botst laten stuiteren. Kun jij er eentje bedenken? Wat zijn de voor- en nadelen ten opzichte van de code uit deze stap?

## Stap 15. Maak een tweede blok

### Opdracht

Maak een tweede blok. Gebruik daarvoor de variabelen brick\_x2 en brick\_y2. In ons spel zien alle blokken er hetzelfde uit, daardoor kunnen we brick\_img, BRICK\_WIDTH en BRICK-HEIGHT gebruiken voor alle blokken.

### Toelichting

Om een tweede blok te maken, moet je drie stukken code kopiëren en aanpassen.

Code om plaats van blok te onthouden

# define global variables

brick\_x = 100

brick\_y = 200

Code om botsting van bal met blok af te handelen

if ... # ball collides with brick

# print message in terminal

# bounce

Code om blok te tekenen:

# draw bricks

screen.blit(brick\_img, (brick\_x, brick\_y))

### Tips:

* Verander eerst overal in de code brick\_x en brick\_y door brick\_x1 en brick\_y1. Als je dat goed doet, dan werkt je code nog hetzelfde. Dit helpt je om te zien waar je allemaal extra code moet toevoegen voor een tweede blok.
* Begin met het maken van variabelen brick\_x2 en brick\_y2.
* Voeg een regel toe om het nieuwe blok te tekenen
* Kopieer de code die de botsing tussen bal en blok afhandelt. Pas de kopie aan zodat hij werkt met het tweede blok.

## Stap 16. Maak een derde en vierde blok

### Opdracht

Maak een tweede blok. Gebruik daarvoor de variabelen brick\_x3, brick\_x4, brick\_y3 en brick\_y4.

### Toelichting

Je ziet dat er veel dubbele code ontstaat. Dat maakt het lastig om later nog iets te veranderen of toe te voegen. In de volgende stap leer je hoe je hetzelfde kunt doen, zonder dubbele code.

### Tips:

* Kijk naar de vorige opdracht en doe hetzelfde nog twee keer.

## Stap 17. Zet blokken in genummerde lijsten

### Opdracht

Gebruik een genummerde lijst met de naam bricks\_x voor de variabelen brick\_x, brick\_x2, brick\_x3 en brick\_x4. Gebruik een genummerde lijst met de naam bricks\_y voor de variabelen brick\_y, brick\_y2, brick\_y3 en brick\_y4. Deze stap is en voorbereiding op het gebruik van herhaal-lussen. Die gaan we gebruiken in de volgende stap. Met herhaal-lussen verdwijnt de dubbele code.

### Toelichting

Code om een genummerde lijst te definiëren:

bricks\_x = [96, 192, 288, 384]

We gebruiken meervoud voor de naam van een lijst, dat maakt de code leesbaarder.

Code om een element uit een genummerde lijst te gebruiken:

bricks\_x[1]

Deze opdracht geeft de waarde van het tweede (het eerste element heeft index nul) element uit de lijst. Die waarde is 192.

### Tips:

* Let op de index van de elementen uit de genummerde lijst. Het 1e element heeft index 0, het 2e element heeft index 1, het 3e element heeft index 2 enzovoort.

## Stap 18. Gebruik iteratie bij blokken in genummerde lijsten

### Opdracht

Maak met de opdracht for een herhaal-lus die alle blokken tekent. Maak daarna met de opdracht for een herhaal-lus die botsingen van de bal met alle blokken afhandelt.

### Toelichting

Simpele code om te itereren over alle elementen in de genummerde lijst:

for brick\_x in bricks\_x :

print(str(brick\_x))

Omdat wij ook bricks\_y gebruiken in de loop, hebben we het nummer van het element in de lijst nodig binnen de herhaal-lus. Omdat we in een latere stap een element uit de lijst willen verwijderen binnen de herhaal-lus, moeten we de lijst van achter naar voren doorlopen. Daardoor hebben wij wat ingewikkelder code nodig.

Onze code om te itereren over alle elementen in de genummerde lijst:

for i in range(len(bricks\_x)-1, -1, -1) :

print('bricks\_x[' + str(i) + '] = ' str(bricks\_x[i]))

Deze code doorloopt de lijst van achter naar voren en zorgt dat in de variabele i het nummer van het element uit de lijst waar we mee bezig zijn wordt opgeslagen.

### Tips:

* Voeg eerst een for-lus toe die alle blokken tekent. Daarvoor heb je één teken-opdracht nodig.
* Let op dat je de teken-opdracht inspringt.
* Vergeet de vier oude regels die de blokken tekende niet te verwijderen.

## Stap 19. Maak een veld met 24 blokken

### Opdracht

Voeg 20 nieuwe blokken toe, zodat je een veld krijgt met 24 blokken.

### Toelichting

Doordat we in de vorige stap het aantal elementen in de lijst berekenden met len(bricks\_x), hoef je nu maar heel weinig aan te passen om blokken toe te voegen. Je hoeft alleen elementen toe te voegen aan de definities van de genummerde lijsten.

### Tips:

* De genummerde lijsten bricks\_x en bricks\_y moeten in onze code evenveel elementen bevatten.
* De definitie van genummerde lijsten kun je over meerdere regels verdelen. Op alle plaatsen waar een spatie is toegestaan, mag je op een nieuwe regel beginnen.

## Stap 20. Haal een blok weg als de bal een blok raakt

### Opdracht

Verwijder een blok dat geraakt is uit de lijst, zodat het uit het veld verdwijnt.

### Toelichting

Code om het 3e element te verwijderen uit een lijst:

bricks\_x.pop(2)

### Tips:

* Gebruik i om het nummer van het element dat je wilt verwijderen aan te geven.
* Verwijder een element uit de lijst bricks\_x en verwijder een element uit de lijst bricks\_y.

## Stap 21. Toon bericht als je wint

### Opdracht

Als alle blokken weg zijn, dan heb je het level uitgespeeld. Stop de beweging van de bal en toon een bericht als het level uit is gespeeld.

### Toelichting

De code len(bricks\_x) geeft het aantal elementen in de genummerde lijst bricks\_x.

### Tips:

* Als je niet weet hoe je de bal stopt, kijk dan bij “Stap 8. Stop het spel als je af bent”
* Als je niet weet hoe je een bericht op het scherm zet, kijk dan bij “Stap 9. Toon een bericht als je af bent”
* Gebruik if len(bricks\_x) == 0 : om te kijken of alle blokken weg zijn.
* Kun je in plaats van if len(bricks\_x) == 0 : ook if len(bricks\_y) == 0 : gebruiken?

# Uitbreidingen

Nu je basisgame af is kan het echte werk beginnen. Voeg een aantal uitbreidingen toe aan je game. Gebruik je creativiteit en laat zien dat je complexere code aankunt. Een aantal voorbeelden van uitbreidingen zijn hieronder aangeven. Met een aantal sterren is aangegeven hoe moeilijk de uitbreiding is, hoe meer sterren hoe moeilijker. Overleg met je docent welke uitbreidingen je kunt gaan maken.

## Uitbreiding: Bal die steeds sneller gaat ⭐️

Een simpele manier om dit te doen is telkens als er een blok verwijderd wordt, de snelheid van het blok in de x en y richting te vermenigvuldigen met een kommagetal dat groter is dan 1.0.

## Uitbreiding: Blokken die van kleur veranderen ⭐️

Een simpele manier om dit te doen is plaatjes van blokken met een andere kleur inlezen en dan meerdere blokken op dezelfde plek zetten.

## Uitbreiding: Beter kaats-algoritme voor plankje ⭐️

In het basisspel kaatst de bal erg simpel tegen de plank. Als je de horizontale snelheid waarmee de bal de plank verlaat laat afhangen van de plek waar de bal de plank raakt, dan wordt het spel beter speelbaar.

## Uitbreiding: Uitlegscherm en gameoverscherm⭐️⭐️

Maak een scherm met uitleg voordat je het spel start en/of een Gameoverscherm als je gewonnen en verloren hebt. Dit kan bijvoorbeeld door je een variabele game\_status erbij te maken. Je kunt je gameloop opdelen in een stuk voor uitleg, spelen, winnen en verliezen. De waarde van de variabele game\_status bepaalt dan welk stuk van de game-loop wordt uitgevoerd.

## Uitbreiding: Animaties als een blok verdwijnt ⭐️⭐️

Laat een blok niet in één keer verdwijnen, maar toon een effect als het blok verdwijnt. Een effect kan bijvoorbeeld zijn dat het blok snel naar boven uit het scherm vliegt. Het lastige is dat je game moet doorgaan terwijl het effect getoond wordt. Je zult dus elke keer in je game-loop een stukje effect moeten tonen.

## Uitbreiding: Meerdere levels ⭐️⭐️

Maak meerdere levels in het spel. Probeer daarbij zo min mogelijk code te dubbelen.

## Uitbreiding: Blokken die omlaag vallen ⭐️⭐️⭐️

Laat bepaalde blokken omlaag vallen als de bal ze raakt en geef extra punten als je je het vallende blok opvangt met je plank.

## Uitbreiding: Meerdere ballen tegelijk ⭐️⭐️⭐️

Laat voor een bepaalde tijd meerdere ballen in het spel als er een speciaal blok geraakt wordt.

## Uitbreiding: Blokken wegschieten ⭐️⭐️⭐️

Zorg dat je plank met kogels blokken weg kan schieten als de bal een speciaal blok raakt.

## Uitbreiding: Overleg met je docent voor andere uitbreidingen

Verzin zelf een uitbreiding die je helemaal te gek vindt. Je docent kan je helpen om je idee in stappen op te splitsen, zodat het uitvoerbaar wordt.