Python Game

Opdrachtbeschrijving

[gees\_at\_stanislascollege\_punt\_nl](mailto:gees@stanislascollege.nl)  
29-12-2023, laatste update 15-7-2024

Afbeelding met elektronica, Invoerapparaat, Elektronisch apparaat, randapparaat

Automatisch gegenereerde beschrijving

Plaatje van <https://openai.com/chatgpt/>

Inhoudsopgave

[Inleiding 3](#_Toc171954009)

[Leeswijzer 3](#_Toc171954010)

[Geschiedenis 3](#_Toc171954011)

[Voorkennis en naslagmateriaal 4](#_Toc171954012)

[Tools 5](#_Toc171954013)

[Werkwijze 5](#_Toc171954014)

[Inleveren 6](#_Toc171954015)

[Beoordeling 6](#_Toc171954016)

[Basisstappen 0 t/m 3: de bal 8](#_Toc171954017)

[Stap 0. Run de startcode 8](#_Toc171954018)

[Stap 1. Voeg commentaar toe 9](#_Toc171954019)

[Stap 2. Beweeg de bal schuin 11](#_Toc171954020)

[Stap 3. Stuiter de bal tegen de onder- en bovenkant van het scherm 11](#_Toc171954021)

[Basisstappen 4 t/m 9: de plank 13](#_Toc171954022)

[Stap 4. Teken de plank 13](#_Toc171954023)

[Stap 5. Beweeg de plank 14](#_Toc171954024)

[Stap 6. Stop de plank aan de randen van het scherm 14](#_Toc171954025)

[Stap 7. Stuiter de bal tegen de plank 15](#_Toc171954026)

[Stap 8. Stop het spel als je af bent 16](#_Toc171954027)

[Stap 9. Toon een bericht als je af bent 16](#_Toc171954028)

[Basisstappen 10 t/m 14: het eerste blok 18](#_Toc171954029)

[Stap 10. Teken een blok in het veld 18](#_Toc171954030)

[Stap 11. Detecteer als de bal het blok raakt 18](#_Toc171954031)

[Stap 12. Stuiter de bal omhoog als hij het blok raakt 19](#_Toc171954032)

[Stap 13. Stuiter de bal omlaag als hij het blok raakt 20](#_Toc171954033)

[Stap 14. Stuiter de bal links of rechts als hij het blok raakt 20](#_Toc171954034)

[Basisstappen 15 t/m 19: meer blokken 22](#_Toc171954035)

[Stap 15. Maak een tweede blok 22](#_Toc171954036)

[Stap 16. Zet blokken in genummerde lijsten 23](#_Toc171954037)

[Stap 17. Gebruik for-loop bij blokken tekenen 23](#_Toc171954038)

[Stap 18. Gebruik for-loop bij botsing tegen blokken 24](#_Toc171954039)

[Stap 19. Maak een veld met 24 blokken 24](#_Toc171954040)

[Basisstappen 20 t/m 21: winnen 25](#_Toc171954041)

[Stap 20. Haal een blok weg als de bal een blok raakt 25](#_Toc171954042)

[Stap 21. Toon bericht als je wint 26](#_Toc171954043)

[Uitbreidingen 27](#_Toc171954044)

[Uitbreiding: Bal die steeds sneller gaat 27](#_Toc171954045)

[Uitbreiding: Blokken die van kleur veranderen 27](#_Toc171954046)

[Uitbreiding: Beter kaats-algoritme voor plankje 27](#_Toc171954047)

[Uitbreiding: Uitlegscherm en gameoverscherm 28](#_Toc171954048)

[Uitbreiding: Animaties als een blok verdwijnt 28](#_Toc171954049)

[Uitbreiding: Powerups die omlaag vallen 28](#_Toc171954050)

[Uitbreiding: Meerdere levels 28](#_Toc171954051)

[Uitbreiding: Meerdere ballen tegelijk 28](#_Toc171954052)

[Uitbreiding: Schieten 28](#_Toc171954053)

[Uitbreiding: Overleg met je docent voor andere uitbreidingen 28](#_Toc171954054)

# Inleiding

## Leeswijzer

Dit is een opdracht waar je één of twee maanden aan gaat werken. De opdracht bestaat uit basisstappen en uitbreidingen. Met de basisstappen maak je een eenvoudig spel waarin je één level kunt spelen. Daarna ga je verder met de uitbreidingen. Je kunt zelf kiezen welke uitbreidingen je gaat maken.

A video game screen with a colorful heart

Description automatically generatedVoorbeeld van een spelletje.

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!

## Geschiedenis

A group of arcade games

Description automatically generated

Arcadekasten.   
Plaatje van <https://commons.wikimedia.org/wiki/File:Four_Arcade_Games.jpg>

In de jaren 80 van de vorige eeuw waren Arcadekasten erg populair. Arcadekasten vond je in uitgaansgelegenheden, soms stonde ze in grote aantallen bij elkaar in een arcadehal. Elke kast had zijn eigen spelletje. Als je geld in de kast stopte, dan kon je een spelletje spelen. Toen er steeds betere spelcomputers bij mensen thuis kwamen, verloor de arcadekast zijn populariteit.

A screenshot of a video game

Description automatically generated

Arkanoid.   
Schermafdruk van <https://www.youtube.com/watch?v=QCfnri9hefQ>

Een voorbeeld van een spel dat je kon spelen is Arkanoid. Arkanoid is een game waarbij je blokjes moet wegstoten met een bal. De bal kaatst tegen een plank. De speler beweegt de plank. Als de bal langs de plank schiet, dan ben je af.

## Voorkennis en naslagmateriaal

Voor deze opdracht heb je basiskennis nodig van programmeren in Python. Je hebt basiskennis als je geoefend hebt met variabelen, selectie (if-jes), iteratie (for-loop), 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) die je voor deze opdracht gaat gebruiken.

Je kunt deze opdracht maken in GitHub Codespaces. Dit is een professionele online omgeving die door veel programmeurs gebruikt wordt. GitHub 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 Codespaces vind je op <https://stanislas.informatica.nu/help/codespaces/>

Versiebeheer in GitHub is gebaseerd op Git. 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)

Mogelijk geeft je docent aan dat je een andere programmeeromgeving moet gebruiken voor deze opdracht. Er zijn veel andere programmeeromgevingen die geschikt zijn om deze opdracht op te maken. Bijvoorbeeld op een computer waar Python en het Pygame package lokaal zijn geïnstalleerd. Op Replit kan deze opdracht niet gemaakt worden. De gratis versie van Replit biedt niet voldoende CPU-capaciteit om een aktiespel dat in Python en Pygame gemaakt is soepel te spelen.

## Werkwijze

* Je maakt deze opdracht in een team van twee personen. Je docent geeft aan hoe de teams gemaakt worden.
* Je gebruikt de startcode die je voor deze opdracht krijgt van je 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.
* Jullie verdelen 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 je 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 je 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, als richtlijn kun je aanhouden dat je maximaal vijf regels code overneemt van anderen of van een tutorial.

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. Je docent kan tijdens het nakijken aanpassingen doen aan het beoordelingsmodel of de berekening van het cijfer.

|  |  |  |  |
| --- | --- | --- | --- |
| **Onderdeel** | **Punten havo** | **Punten vwo** | **Toelichting** |
|  |  |  |  |
| **Code** |  |  |  |
| Stijl | 5 | 5 | - Netjes uitgelijnd,  - helder commentaar,  - logische naamgeving van variabelen,  - logische volgorde van opdrachten,  - consistente code. |
| Basiscode | 20 | 15 | - If-opdrachten goed toegepast, vergelijkingsoperatoren </>/== gebruikt, logische operatoren and/or gebruikt,  - lijsten toegepast,  - for (of while)-opdrachten toegepast,  - code kan omgaan met verschillende lengtes van lijsten (len goed gebruikt), - code wijzigt aantal elementen in lijst tijdens het runnen. |
| Uitbreidingen (maantjes) | 15 | 20 | - Uitbreidingen gemaakt met technieken die in de opdracht behandeld zijn, zoals meer loops, meer lijsten, meer functies (1 maantje) - uitbreidingen gemaakt met technieken die niet in de opdracht behandeld zijn en je je zelfstandig hebt eigen gemaakt, zoals dicts, geneste loop, lijst in lijst (2 maantjes). |
|  |  |  |  |
| **Functionaliteit** |  |  |  |
| Speelbaarheid | 5 | 5 | - geen glitches - soepele bediening |
| Basisfunctionaliteit | 20 | 15 | - Bal stuitert tegen randen,  - plank beweegt en stopt aan de rand van het scherm en laat bal stuiteren,  - 1 blok waartegen bal stuitert, - meer dan 10 blokken waartegen bal stuitert, - blokken verdwijnen, je kunt af gaan en winnen. |
| Uitbreidingen (zonnetjes) | 15 | 20 | Uitbreidingen in het spel die zorgen voor diversiteit en hoe beter je wordt hoe leuker of moeilijker het wordt,  - voorbeelden van kleine uitbreidingen (1 zonnetje) zijn steeds snellere bal, verschillende kleuren blokken, - voorbeelden van grotere uitbreidingen (2 zonnetjes) zijn meer levels, ingewikkelde powerups zoals blokken wegschieten, veel ballen. |
|  |  |  |  |
| **Proces** |  |  |  |
| Planning | 5 | 5 | Uit de commits en/of tijdens de les blijkt dat het stappenplan gevolgd is en dat er regelmatig is gewerkt. |
| Samenwerking | 5 | 5 | Uit de commits en/of tijdens de les blijkt dat het werk gelijk verdeeld is onder teamleden. Als één teamlid beduidend minder doet, dan is dat een gezamenlijke verantwoordelijkheid van het team om dat op te lossen of te melden bij de docent. Maak vanaf het begin afspraken en spreek elkaar daarop aan. Als je er niet uitkomt, dan meld je dat zo snel mogelijk bij je docent. |

Cijfer = 1 + 9 \* behaalde punten / maximale punten

# Basisstappen 0 t/m 3: de bal

We beginnen met een bal die stuitert.

A black screen with a black background

Description automatically generated

Schermafdruk van spelletje met basisstap 0 t/m 3.

De basisstappen zijn genummerd van stap 0 tot en met stap 21. In elke basisstap staat onder ‘opdracht’ wat je code moet doen. Lees dit aandachtig door, voordat je begint te coderen. De toelichting geeft aan hoe je de opdracht kunt maken. De tips kun je bekijken als je er niet uitkomt.

## Stap 0. Run de startcode

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. De tips kun je bekijken als je er niet uitkomt.

### Opdracht

Verander in de code de achtergrondkleur van je spel. Doe het volgende:

* Run de startcode. De achtergrond is zwart, bovenin beweegt een blauwe bal.
* Verander in de code de achtergrondkleur van je spel.
* Start je code opnieuw.
* Sla de nieuwe versie van je code op in GitHub.

### Toelichting

Voor deze opdracht krijg je startcode. De startcode beweegt een bal bovenin het scherm van links naar rechts naar links enzovoort. De startcode moet je uitvoeren (Engels: run) en wijzigen (Engels: edit) in een ontwikkelomgeving. Je docent vertelt hoe je aan de startcode komt en welke ontwikkelomgeving je mag gebruiken.

### Tips

* Wijzig de screen.fill('black') in het bestand main.py om de achtergrondkleur te veranderen.
* Bij de startcode zit een README.md bestand. Hierin staat hoe je code kunt starten, stoppen, wijzigingen en opslaan.
* Meer informatie over de ontwikkelomgeving GitHub Codespaces vind je op <https://stanislas.informatica.nu/help/codespaces/>

## Stap 1. Voeg commentaar toe

### Opdracht

Breidt het commentaar in je startcode uit. In de toelichting vind je voorbeelden van stukjes code met extra commentaar. Het gaat er bij deze opdracht om dat je de startcode leert begrijpen.

### Toelichting

Stukje code uit main.py die constanten en variabelen maakt:

# definitions

FPS = 30 # Frames Per Second

SCREEN\_WIDTH = 1280 # screensize in x-direction in pixels

SCREEN\_HEIGHT = 720 # screensize in y-direction in pixels

BALL\_WIDTH = 16 # ballsize in x-direction in pixels

BALL\_HEIGHT = 16 # ballsize in y-direction in pixels

ball\_x = 0 # x-position of ball in pixels

ball\_speed\_x = 6 # speed of ball in x-direction in pixels per frame

Stukje code uit main.py die game te 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()

Stukje code uit main.py die plaatjes inleest:

# read images

# read spritesheet containing all images

# convert\_alpha increases speed of blit and keeps transparency of .png

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

# create empty image of 64 x 64 pixels, SRCALPHA supports transparency

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

# copy part (x-left=1403, y-top=652, width=64, height=64) from spritesheet to ball\_img at (0,0)

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

# resize ball\_img from 64 x 64 pixels to BALL\_WIDTH x BALL\_HEIGHT

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

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

running = True

while running:

# read events

# move everything

# handle collisions

# draw everything

# wait until next frame

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

# read events

for event in pygame.event.get(): # read all events

if event.type == pygame.QUIT: # GUI is closed

running = False # end programm

keys = pygame.key.get\_pressed() # read which keys are down

Stukje code uit gameloop die alles beweegt:

# move everything

# move ball

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

# bounce ball against edges of screen

if ball\_x < 0 : # left edge

ball\_speed\_x = abs(ball\_speed\_x) # positive x-speed = move right

if ball\_x + BALL\_WIDTH > SCREEN\_WIDTH: # right edge

ball\_speed\_x = abs(ball\_speed\_x) \* -1 # negative x-speed = move left

Plek in gameloop om later code toe te voegen die botsingen afhandelt:

#

# handle collisions

#

Stukje code uit gameloop die alles tekent:

# draw everything

# 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:

# wait until next frame

fps\_clock.tick(FPS) # Sleep the remaining time of this frame

### Tips:

* Alles tussen # en het einde van de regel, wordt door de computer overgeslagen.
* 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 diagonaal (schuin) over het scherm beweegt. Voeg de variabelen ball\_y en ball\_speed\_y toe.

### 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 functie die de bal op het scherm zet. Dit is de functie 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.

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

### Opdracht

De bal verdwijnt nu uit 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.

### Toelichting

Code die zorgt dat bal stuitert tegen zijkant van scherm:

if ball\_x < 0 : # left edge

ball\_speed\_x = abs(ball\_speed\_x) # positive x-speed = move right

if ball\_x + BALL\_WIDTH > SCREEN\_WIDTH: # right edge

ball\_speed\_x = abs(ball\_speed\_x) \* -1 # negative x-speed = move left

* De functie abs(getal) geeft als resultaat een getal zonder min. Dus abs(-2) is 2 en abs(2) is ook 2. Dus abs(ball\_speed\_x) geeft de positieve waarde van de x-snelheid. abs(ball\_speed\_x) \* -1 geeft de negatieve waarde van de x-snelheid.

### 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.
* Gebruik BALL\_HEIGHT om rekening te houden met de hoogte van de bal.

# Basisstappen 4 t/m 9: de plank

De bal beweegt, maar je kunt het spel nog niet spelen. Daarom gaan we een plank toevoegen.

A screenshot of a computer

Description automatically generated

Schermafdruk van spelletje met basisstap 4 t/m 9.

## Stap 4. Teken de plank

### Opdracht

Teken de plank (Engels: paddle) 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

Kopieer de code van de bal en pas de kopie aan, zodat hij werkt voor de plank.

Regels uit de startcode die het plaatje van de bal inlezen:

ball\_img = pygame.Surface((64, 64), pygame.SRCALPHA) # create new image

ball\_img.blit(spritesheet, (0, 0), (1403, 652, 64, 64)) # copy part of sheet to image

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

* De eerste regel maakt een variabele ball\_img met een leeg plaatje erin. (64, 64) zijn de breedte en hoogte van het plaatje. pygame.SRCALPHA zorgt ervoor dat de achtergrond van je plaatje transparant kan zijn.
* De tweede regel kopieert een stuk uit het spritesheet naar het lege plaatje. (0, 0) betekent dat het stukje spritesheet linksboven in je lege plaatje komt. (1403, 652, 64, 64) zijn de x, y, width en height van het stuk dat je uit het spritesheet kopieert.
* De derde regel verkleint het plaatje tot de afmeting die je in de game nodig hebt. (BALL\_WIDTH, BALL\_HEIGHT) is de breedte en hoogte die je nodig hebt in je game.

Regels uit de startcode die het plaatje van de bal op het scherm zetten:

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

* ball\_img is de naam van de variabele waar het plaatje in zit dat je op het scherm wil zetten. ball\_x en ball\_y zijn de linkerbovenhoek van de plek waar het plaatje op het scherm wordt gezet.

### Tips:

* De breedte van de plank in het spritesheet is 243 pixels, de hoogte is 64 pixels. Dat is de afmeting die het nieuwe plaatje moet hebben.
* De plank staat in het spritesheet op x is 1158 en y is 396. Dat is de linkerbovenhoek van het stuk dat je uit het spritesheet moet kopieren.
* Definieer de constanten PADDLE\_WIDTH en PADDLE\_HEIGHT en geef ze de waarde 144 en 32. Dit is de afmeting waarnaar je het plaatje van de paddle moet verkleinen.
* Maak de variabelen paddle\_x en paddle\_y om bij te houden waar de paddle staat. Geef ze een startwaarde, bijvoorbeeld SCREEN\_WIDTH / 2 en SCREEN\_HEIGHT – 100. Gebruik paddle\_x en paddle\_y om de plank op de goede plaats op het scherm te zetten.

## Stap 5. Beweeg de plank

### Opdracht

Beweeg de plank als de D-toets (rechts) of A-toets (links) is ingedrukt. Beweeg de plank met één pixel per frame. Gebruik keys uit de startcode om te zien welke toets momenteel worden ingedrukt.

### Toelichting

Kijken of de D-toets is ingedrukt doe je als volgt:

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

### Tips:

* Als de D-toets is ingedrukt, beweeg je de plank naar rechts door bij paddle\_x tien op te tellen. De plank wordt daardoor verderop in de code op een nieuwe plek getekend.
* Als de A-toets is ingedrukt, beweegt 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 de plank te laten stoppen aan de rechterkant van het scherm.

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

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

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

Pseudo-code om de plank te laten stoppen aan de linkerkant 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. Het algoritme uit deze stap kijkt of een stukje van de bal op dezelfde plek is als een stukje van de plank. Als dat zo is dan wordt de y-richting van de bal negatief gemaakt, zodat de bal omhoog beweegt.

### 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 = de positieve waarde van bal\_snelheid\_y keer -1

### Tips:

* Gebruik de pseudo-code. Gebruik nu geen ingebouwde functie van pygame om een botsing te detecteren.
* 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.
* De y-snelheid van de bal moet negatief (kleiner dan nul) zijn om te zorgen dat de bal omhoog beweegt. Want bovenaan het scherm is y gelijk aan nul en hoe verder naar onderen je komt hoe groter de y wordt.
* Je kunt de y-snelheid negatief maken door een eventuele min eraf te halen en dan keer -1 te doen. De functie abs(getal) geeft als resultaat een getal zonder min. Dus abs(-2) is 2 en abs(2) is ook 2. Dus abs(ball\_speed\_y) \* -1 geeft de negatieve waarde van de y-snelheid.
* Wat doet het algoritme als de bal tegen de zijkant van de plank kaatst? Wat zou je in het echt verwachten?
* Hou het algoritme voor nu simpel, je kunt het bij de uitbreidingen 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 onder afhandelen van botsingen en voor je begint met tekenen. Dat is op de plek van de <<hier toevoegen>> in onderstaande stukjes.

# move everything

...

# handle collisions

<<hier toevoegen>>

# draw everything

...

### Tips:

* Bovenaan het scherm is y gelijk aan 0. Hoe verder je naar beneden op het scherm komt, hoe groter het getal y wordt. De y-coördinaat van de bovenkant van de bal bewaar je in de variabele ball\_y. De hoogte van de bal bewaar je in BALL\_HEIGHT. Hoe bereken je de y-coördinaat van de onderkant van de bal?
* Wat gebeurt er met de code die 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

Deze drie stukjes code zetten een bericht op het scherm. Kopieer en plak ze 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().

# Basisstappen 10 t/m 14: het eerste blok

Het spel bevat alleen nog maar een plank en een bal, maar geen blokken… Het wordt tijd om het eerste blok toe te voegen!

A black screen with a white letter

Description automatically generated

Schermafdruk van spelletje met basisstap 10 t/m 14.

## 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))

### Tips:

* Bekijk “Stap 7. Stuiter de bal tegen de plank” als je niet meer weet hoe je kunt zien dat twee voorwerpen elkaar raken.
* 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.
* 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 stappen 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 ) :

maak snelheid\_y omhoog

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 stappen 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.
* maak snelheid\_y omhoog betekent dat snelheid\_y een negatief getal (een getal kleiner dan 0) moet zijn. Gebruik daarvoor de functie abs(). Kijk in stap 7 en stap 3 als je niet meer weet hoe je abs() kunt gebruiken
* Kun je uitleggen waarom de code bij de tweede “als” kijkt of er een stukje bal buiten het blok zit, in plaats van te 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 ) :*

*maak snelheid\_y omhoog*

anders als ( de bal naar boven beweegt en

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

maak snelheid\_y omlaag

De schuingedrukte regels heb je al gemaakt in de vorige stap. De rest moet je nog toevoegen. Bij de vorige stap checkt je code of er een stuk van de bal *binnen* het blok zat. De code die je in deze stap toevoegt checkt welk stuk van de bal *buiten* het blok zit.

### Tips:

* Met elif 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.
* maak snelheid\_y omlaag betekent dat snelheid\_y een positief getal (een getal groter dan 0) moet zijn. Gebruik daarvoor de functie abs(). Kijk in stap 7 en stap 3 als je niet meer weet hoe je abs() kunt gebruiken

## 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 ) :*

*maak snelheid\_y omhoog*

*anders als ( de bal naar boven beweegt en*

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

*maak snelheid\_y omlaag*

anders als ( de bal naar rechts beweegt en

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

maak snelheid\_x naar links

anders als ( de bal naar links beweegt en

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

maak snelheid\_x naar rechts

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.
* maak snelheid\_x links betekent dat snelheid\_x een negatief getal (een getal kleiner dan 0) moet zijn. maak snelheid\_x rechts betekent dat snelheid\_x een postief getal (een getal groter dan 0) moet zijn. Gebruik daarvoor de functie abs(). Kijk in de vorige stap als je niet meer weet hoe je deze functie gebruikt.
* 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?

# Basisstappen 15 t/m 19: meer blokken

Eén blok is leuk, maar meer blokken is nog leuker! We gaan stap voor stap blokken toevoegen.

A screenshot of a video game

Description automatically generated

Schermafdruk van spelletje met basisstap 15 t/m 19.

## Stap 15. Maak een tweede blok

### Opdracht

Maak een tweede blok. Gebruik daarvoor de waarden van de variabelen brick\_x2 en brick\_y2. In dit spel zien alle blokken er hetzelfde uit, daardoor kun je 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 botsing 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. Zet blokken in genummerde lijsten

### Opdracht

Gebruik een lijst met de naam bricks\_x voor de variabelen brick\_x1 en brick\_x2. Gebruik een genummerde lijst met de naam bricks\_y voor de variabelen brick\_y1 en brick\_y2. Deze stap is een voorbereiding op het gebruik van loops. Die ga je gebruiken in de volgende stap. Met loops verdwijnt de dubbele code die je in de vorige stap voor het tweede blok gemaakt hebt.

### Toelichting

Code om een lijst met twee elementen te definiëren:

bricks\_x = [96, 192]

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

Code om een element uit een lijst te gebruiken:

bricks\_x[1]

Het eerste element van de lijst bricks\_x heeft als index het getal nul, het tweede element heeft als index één, het derde element heeft index twee enzovoort. Deze code geeft de waarde van het tweede element uit de lijst. Die waarde is 192.

Maak een lijst bricks\_y op dezelfde manier als je bricks\_x hebt gemaakt.

### Tips:

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

## Stap 17. Gebruik for-loop bij blokken tekenen

### Opdracht

Maak met het commando for een loop die alle blokken tekent.

### Toelichting

Eenvoudige code om met een loop alle elementen in de lijst af te drukken:

for brick\_x in bricks\_x :

print(str(brick\_x))

Omdat wij ook bricks\_y gebruiken in de loop, heb je het nummer van het element in de lijst nodig binnen de for-loop.

Code die je in je game kunt gebruiken om met een loop alle elementen in de lijst af te drukken:

for i in range(0, len(bricks\_x)) :

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

Deze code doorloopt de lijst van voor naar achter en slaat het nummer van het element uit de lijst op in de variabele i.

### Tips:

* Voeg bij code voor het tekenen van de blokken een for-loop toe die alle blokken tekent. Daarvoor heb je één teken-commando binnen de for-loop nodig.
* Let op dat je de teken-commando inspringt.
* Vergeet de oude regels die de blokken tekende niet te verwijderen.

## Stap 18. Gebruik for-loop bij botsing tegen blokken

### Opdracht

Maak met het commando for een loop die botsingen van de bal met alle blokken afhandelt.

### Tips:

* Kijk bij de vorige stap hoe je een for-loop maakt.
* Let op dat je goed inspringt.

## Stap 19. Maak een veld met 24 blokken

### Opdracht

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

### Toelichting

In de vorige stap wordt het aantal elementen in de lijst berekend met len(bricks\_x). Daardoor 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 lijsten.

### Tips:

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

# Basisstappen 20 t/m 21: winnen

Om het spel uit te kunnen spelen, moeten de blokken verdwijnen. Laten we dat gaan toevoegen.

A screenshot of a video game

Description automatically generated

Schermafdruk van spelletje met basisstap 20 t/m 21.

## 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. Stop de for-loop met break, zodra er een blok verwijderd is.

### Toelichting

Code om het 1e element te verwijderen uit een lijst:

bricks\_x.pop(0)

Code om het 2e element te verwijderen uit een lijst:

bricks\_x.pop(1)

### 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.
* Als je nadat je een blok verwijderd hebt, het commando break gebruikt, dan wordt er maar één blok per frame verwijdert. De bal kan twee blokken tegelijk raken, maar elk frame gaat zo snel, dat je daar niks van ziet. Als je geen break gebruikt, dan krijg je een “index out of range” foutmelding, omdat je verderop in de for-loop probeert een element uit de lijst te gebruiken dat er niet meer is.

## 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 dat het level is uitgespeeld.

### Toelichting

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

A video game screen with colorful blocks

Description automatically generated

Voorbeeld van een spelletje met 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, maar je kunt ook uitbreidingen toevoegen die je zelf hebt bedacht. Het aantal 🌜(maan) geeft aan hoe moeilijk de code voor de uitbreiding is. Het aantal 🌞 (zon) geeft aan hoeveel functionaliteit het toevoegt. Overleg met je docent welke uitbreidingen je kunt gaan maken.

## Uitbreiding: Bal die steeds sneller gaat

Een eenvoudige 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. Op deze manier kan het met twee regels code en maakt je spelletje een heel klein beetje spannender. Dit is een makkelijke uitbreiding, nog niet genoeg om een maantje of zonnetje te verdienen.

## Uitbreiding: Blokken die van kleur veranderen 🌜🌞

Een eenvoudige 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 eenvoudig 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 game-over-scherm als je gewonnen of verloren hebt. Dit kan bijvoorbeeld door 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: Powerups 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 levels 🌜🌜🌞🌞

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

## Uitbreiding: Meerdere ballen tegelijk 🌜🌜🌞

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

Gebruik array’s om meer dan twee ballen tegelijk te kunnen hebben. Gebruikt een geneste loop om de botsingen van alle ballen tegen alle blokken te maken

## Uitbreiding: Schieten 🌜🌜🌞

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

Gebruikt array’s om meerdere kogels tegelijk te kunnen schieten.

## 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.