



Gemaakt door: Aagje Reynders



GAMES ONTWIKKELEN

We gaan leren werken met pygame, dit is gelijkaardig aan tkinter, alleen uitgebreider en meer gefocust op het maken van spelletjes.

In deze cursus gaan we proberen stap voor stap een spelletje te maken. Ik ga er vanuit dat de basis van python gekend is en leg ze hieronder niet meer in detail uit. Je kunt deze nalezen in de kennismakingscursus van Python.

Het eerste spelletje dat we gaan maken is Snake.



Er wordt veel gebruik gemaakt van HOOFDLETTERS en inspringingen, als er iets niet werkt of je krijgt fouten, kijk dan zeker na of je de variables of termen correct hebt geschreven en of je inspringingen correct zijn. Als je iets wilt uitvoeren in een while-loop let dan op dat er de juiste inspringingen zijn toegepast.

PYGAME INSTALLEREN EN IMPORTEREN

Pygame zit niet standaard inbegrepen bij Python. Hierdoor gaan we iets moeten downloaden, maar we gaan eerst onderscheid maken tussen Mac en Windows gebruikers.

Windows:

Controleer welke versie je hebt van python, dit zie je aan de IDLE die je opent. Kijk ook of je 32bit hebt of 64bit.

Heb je 2.7? Ga dan naar : <http://www.pygame.org/download.shtml>
en download dit bestand: `pygame-1.9.1.win32-py2.7.msi`
Indien je het gedownload hebt, zou het moeten werken, ga naar pagina 4.



Heb je 3.5?

Ga dan naar : <http://www.lfd.uci.edu/~gohlke/pythonlibs/#pygame>
download dit bestand: `pygame-1.9.2b1-cp35-cp35m-win32.whl` of `pygame-1.9.2b1-cp35-cp35m-win_amd64.whl` (het eerste voor 32 bit en het tweede voor 64bit)

Heb je dit gedownload? **Roep een coach er even bij dan gaan we de hieronderstaande stappen samen doen:**

Het gedownloade bestand moeten we verplaatsen naar de python map, een voorbeeld van zo'n structuur is:

`C:\Users\{{User}}\AppData\Local\Programs\Python\Python35`

Als we dat gedaan hebben openen we cmd, slepen we de locatie daar naartoe en schrijven we: `pip install <naam van bestand>`

Gelukt? Ga dan naar pagina 4.

MAC:

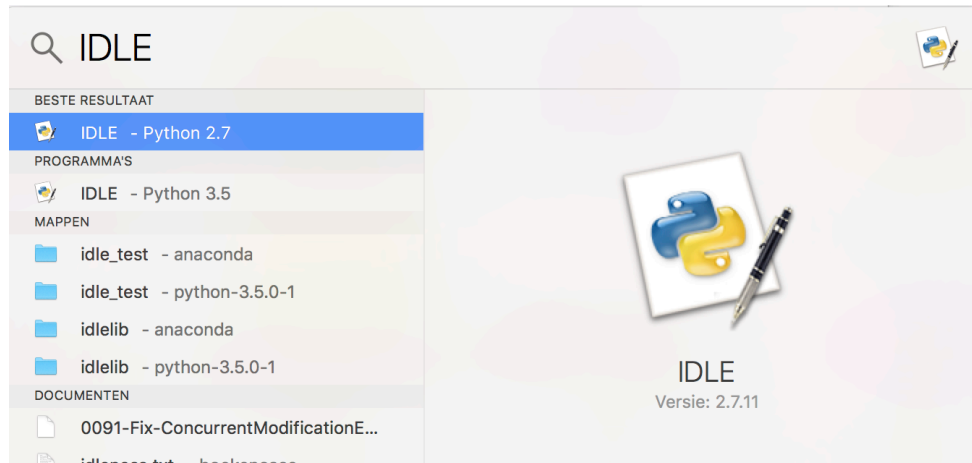
Voor MAC moeten we een 2.7 of minder versie van python hebben. Deze kun je installeren via python website.

Daarna installeer je pygame.

Ga nu naar finder en klik op macintosh HD. Open nu de onderstaande mappen.

Macintosh HD > Bibliotheek > Frameworks > Python.framework > Versions > 2.7 > bin >

Als je hier bent, zie je allemaal bestanden staan. We moeten hier twee namen aanpassen. zoek python 2.7 en python2.7-32. De python2.7 veranderen we van naam en dit wordt python2.7-64. De python2.7-32 veranderen we nu naar python2.7. Nu gaan we de IDLE openen van python 2.7. Dit gaat via de spotlight door IDLE in te geven. Het kan zijn dat je twee IDLE's hebt (van twee verschillende python versies. Voor pygame kiezen we voor 2.7)



We moeten nu nog één ding downloaden en dat vind je op deze pagina :

<https://www.libsdl.org/download-1.2.php>

We hebben dit nodig zodat pygame goed kan werken. Eens we het gedownload hebben plaatsen we dit in de map Frameworks.

Macintosh HD > Bibliotheek > Frameworks > |

Hij gaat misschien zeggen dat er al zo'n bestand bestaat, maar we drukken op samenvoegen met het bestaande bestand.

**Als je het geïnstalleerd hebt:**

We gaan even controleren of alles wel werkt. Open de IDLE van python en maak een nieuw bestand aan. Daarin type je

```
import pygame
```

Probeer dit te runnen, als hij geen errors geeft en niets uitvoert, wilt het zeggen dat het prima werkt. Als python een error geeft dat hij pygame niet kan vinden, vraag dan even hulp aan een coach.

Opgelet: Benoem je bestand nooit 'pygame.py', want python gaat hierdoor in de war zijn. Benoem het bestand bijvoorbeeld 'snake.py'.



STRUCTUUR

Om het in de komende pagina's gemakkelijker te maken wil ik eerst een bepaalde structuur opstellen. Hierdoor zetten we bepaalde titels in commentaar (# ervoor zetten) zodat het gemakkelijker is om bepaalde dingen op de juiste plaats te zetten. Als er staat "zet dit bij variables" dan weet je dat het onder de #variables moet staan.

```
#import
#variables
#kleuren
#functies
#whileloop
```

Wanneer je deze titels hebt, voegen we de code die hieronder staat toe aan onze basisstructuur. De foto die je hieronder ziet moet je altijd in het begin hebben als je werkt met pygame.

```
*Untitled*
#import
import pygame
pygame.init() #pygame wordt ingeladen

#variables
gameDisplay = pygame.display.set_mode((800,600)) # Een venster wordt geopend

#kleuren

#functies

#whileloop

pygame.display.update() #python laadt venster constant opnieuw in
                        #om veranderingen aan te passen

pygame.quit() #programma sluiten |
quit() #effectief sluiten
```

Als je dit uitvoert opent er een scherm, maar wordt dit scherm ook meteen gesloten.



SNAKE

Wat is snake? Snake is een spelletje waar een slang moet proberen **appels te eten**. Deze appels komen **willekeurig op het scherm**, met de **pijltjestoetsen** probeer je met de slang **naar de appel te kruipen** zonder tegen **de randen of jezelf te botsen**. Per keer dat je een **appel eet wordt je lichaam langer**. Alles in het vet zijn verschillende belangrijke acties die in de code gaan komen.

We gaan stap voor stap dit spel bereiken. Dit zal ongeveer 2 coderdojo lessen duren, dus neem geregeld pauzes en stel vragen als je iets niet goed snapt!

Werken met events

Events zijn bepaalde acties die de gebruiker uitvoert. Zoals een knop indrukken, pijltjes indrukken, met de muis over het venster gaan,...

Om even kort te laten zien hoeveel verschillende events er zijn gaan we even wat uitproberen. Plaats de variable gameExit onder de #variables en de while loop onder pygame.display.update()

```
gameExit = False  
  
while not gameExit:  
    for event in pygame.event.get():  
        print(event)
```

Bij het maken van games maken we gebruik van een while-loop. Zolang we het spel spelen zal de while-loop lopen. Wanneer gameExit gelijk is aan True zal het spel zich sluiten. Dit zullen we in één van de volgende hoofdstukken schrijven.

Op het volgende lijntje hebben we een for-loop. Hierbij gaan we alle events die er zijn doorlopen en deze weergeven wanneer ze aangesproken worden. Als je dit programmaatje uitprobeert zal je iets gelijkaardig zien als hieronder, probeer met de muis te bewegen over het veld of wat letters in te drukken.



```
*Python 2.7.11 Shell*
<Event(4-MouseMotion {'buttons': (0, 0, 0), 'pos': (71, 0), 'rel': (-51, 0)})>
<Event(4-MouseMotion {'buttons': (0, 0, 0), 'pos': (70, 0), 'rel': (-1, 0)})>
<Event(4-MouseMotion {'buttons': (0, 0, 0), 'pos': (61, 0), 'rel': (-9, 0)})>
<Event(4-MouseMotion {'buttons': (0, 0, 0), 'pos': (47, 0), 'rel': (-14, 0)})>
<Event(4-MouseMotion {'buttons': (0, 0, 0), 'pos': (44, 0), 'rel': (-3, 0)})>
<Event(4-MouseMotion {'buttons': (0, 0, 0), 'pos': (38, 0), 'rel': (-6, 0)})>
<Event(4-MouseMotion {'buttons': (0, 0, 0), 'pos': (36, 0), 'rel': (-2, 0)})>
<Event(4-MouseMotion {'buttons': (0, 0, 0), 'pos': (29, 0), 'rel': (-7, 0)})>
<Event(4-MouseMotion {'buttons': (0, 0, 0), 'pos': (23, 0), 'rel': (-6, 0)})>
<Event(4-MouseMotion {'buttons': (0, 0, 0), 'pos': (19, 0), 'rel': (-4, 0)})>
<Event(4-MouseMotion {'buttons': (0, 0, 0), 'pos': (15, 0), 'rel': (-4, 0)})>
<Event(4-MouseMotion {'buttons': (0, 0, 0), 'pos': (13, 0), 'rel': (-2, 0)})>
<Event(4-MouseMotion {'buttons': (0, 0, 0), 'pos': (14, 0), 'rel': (1, 0)})>
<Event(12-Quit {})>
<Event(4-MouseMotion {'buttons': (0, 0, 0), 'pos': (12, 0), 'rel': (-2, 0)})>
<Event(4-MouseMotion {'buttons': (0, 0, 0), 'pos': (0, 11), 'rel': (-12, 11)})>
<Event(4-MouseMotion {'buttons': (0, 0, 0), 'pos': (0, 22), 'rel': (0, 11)})>
<Event(1-ActiveEvent {'state': 3, 'gain': 0})>
<Event(1-ActiveEvent {'state': 2, 'gain': 1})>
```

Als je drukt op het kruisje van het venster van het spel, dan gebeurt er niets, behalve dat (zoals je hierboven ziet) het event "quit" wordt aangegeven op het scherm. Het stoppen van het programmaatje is dus een event en om het programmaatje effectief te laten stoppen moeten we dit nog schrijven. (Anders blijft hij in de while-loop)

Om het programmaatje dus nu te kunnen stoppen drukken we op het kruisje van de shell (zie foto hierboven)

Spel afsluiten

De laatste twee regels van ons programmaatje zorgen ervoor dat het programmaatje stopt. Omdat we in een while-loop zitten, blijft die dus oneindig lopen en komen we nooit aan die twee laatste regels. We gaan hier verandering in brengen!

```
while not gameExit:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            gameExit = True
```

We voegen twee nieuwe regels toe en laten de print() weg. Deze twee regels zorgen ervoor dat het moment dat we op het kruisje drukken (en het event QUIT activeren) gameExit gelijkstellen aan True. Wat wilt zeggen dat de while-loop stopt en we verder kunnen gaan met de twee laatste regels (en hierdoor sluit het programmaatje).



Tekenen op het scherm (werken met objecten, kleuren, ...)

- Werken met kleuren:

Bij tkinter hadden we al gezien dat aan de hand van termen zoals “red, yellow, blue, green,...” we objecten een bepaalde kleur konden geven. Bij pygame werkt dit anders. We krijgen een bepaalde kleur aan de hand van drie verschillende getallen. Deze drie getallen verduidelijken aan de computer hoeveel rood, blauw en groen in de kleur zitten. De maximale waarde is 255. Hieronder zie je wat voorbeelden:

Zwart = (0, 0, 0)	→ Geen kleur aanwezig, vandaar 0
Wit = (255, 255, 255)	→ Als alle waarden maximaal zijn krijg je wit
Rood = (255, 0, 0) ->	→ Alleen de waarde rood is maximaal
Blauw = (0, 255, 0)	→ Alleen de waarde blauw is maximaal
Groen = (0, 0, 255)	→ Alleen de waarde groen is maximaal

Laten we de achtergrondkleur veranderen.

Maak een variable achtergrondkleur = (getal1, getal2, getal3). Plaats deze onder #kleuren. Vervang getal1, getal2 en getal3 door drie willekeurige getallen tussen 0 en 255 die je zelf uitkiest. Welke kleur verkrijg jij? Voorbeeld:

`achtergrondkleur = (25, 78, 114)`

```
achtergrondkleur = (25,78,114) #Kies zelf de getallen! |
gameExit = False

while not gameExit:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            gameExit = True
    gameDisplay.fill(achtergrondkleur) #Achtergrond wordt ingekleurd met bepaalde kleur
    pygame.display.update() #python laadt venster constant
                                #opnieuw in om veranderingen aan te passen
```

- **OPDRACHT:** Om het gemakkelijker te maken in de komende hoofdstukken schrijf boven achtergrondkleur ook andere kleuren (zwart, wit, rood, groen, blauw)

- Objecten tekenen

Voor het spel snake gaan we een rechthoek tekenen die onze slang moet voorstellen. Dit doen we zo:

`pygame.draw.rect(gameDisplay, zwart, [400, 300, 10, 10])`

→ Dit plaatsen we achter de fill() maar voor de update() in de code. Waarom? Stel dat we het voor de fill() zouden zetten doet het programma dit:

- tekent rechthoekje
- kleurt het venster met de fill-kleur (over het rechthoekje)
- met update wordt het eindresultaat van de code weergegeven en gaat hij terug naar het begin van de while.



Dit doet hij zo snel dat we het niet zien gebeuren, maar het rechthoekje zal niet zichtbaar zijn.



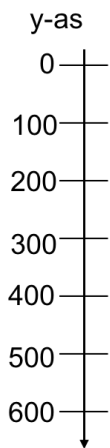
Laten we het lijntje hierboven eens beter bekijken. 'draw' wilt zeggen tekenen en rect betekent een rechthoek. We zeggen dus letterlijk dat we via pygame een rechthoek willen tekenen. Daarna hebben we wat waarden tussen de haakjes. Via welk venster werken we? Welke kleur gaat de rechthoek krijgen en waar en hoe groot gaat de rechthoek zijn. Aangezien we in het begin gezegd hebben dat het venster 800 op 600 groot is, maken we nu gebruik van de waarden 400 en 300, zodat de rechthoek in het midden begint. Je kunt ook andere getallen ingeven, zolang ze binnen de 800 en 600 zitten! De twee tien op het einde moeten de hoogte en breedte voorstellen van het rechthoekje, waardoor het een vierkant wordt.

De rechthoek laten bewegen

Iets laten bewegen wilt zeggen dat door een bepaalde actie hij naar links, rechts, boven of onder opschuift. Hierbij worden de coördinaten van de rechthoek aangepast (400 op 300).

Naar links gaan dan doen we huidige locatie X min 10, naar rechts gaan doen we huidige locatie Y plus 10.





We maken twee variable `coordX` en `coordY`, deze twee houden bij wat de positie is van de rechthoek zowel op de x als y-as.

```
coordX = 400
```

```
coordY = 300
```

Bij `pygame.draw.rect()` vervangen we de 400 en 300 in `coordX` en `coordY`.

Nu willen we dat de slang geregeld opschuift van plaats. Hierdoor moeten we bepaalde dingen toevoegen. Bij `pygame` variable voeren we het hieronderstaande toe:

```
clock = pygame.time.Clock()
```

in onze while loop plaatsen we `coordX -= 10` en onder `update()` plaatsen we `clock.tick(15)`. Wat doet dit? Dit zorgt ervoor dat 15 keer in één seconde het spel wordt geupdated. De slang zal dan ook 15 keer in één seconde opschuiven.

Als we het spel nu uitproberen zien we het rechthoekje naar links opschuiven.

Maar nu willen we nog de richting aanpassen. Onder de if-statement waar we controlleren wanneer we het event "afsluiten" uitvoeren, gaan we nu controlleren of we een knop indrukken en hierdoor gaan we van richting veranderen. Maak een variable genaamd `richting` en deze is standaard gelijk aan "left". De code hieronder toevoeg je aan de while-loop, binnen de for-lus `for event in pygame.event.get():` die er al in staat.

```
if event.type == pygame.KEYDOWN:
```

```
    if event.key == pygame.K_LEFT and richting != "right":  
        richting = "left"
```

```
    elif event.key == pygame.K_RIGHT and richting != "left":  
        richting = "right"
```

```
    elif event.key == pygame.K_UP and richting != "down":  
        richting = "up"
```

```
elif event.key == pygame.K_DOWN and richting != "up":  
    richting = "down"
```

De code hieronder moet buiten de forlus van de events:

```
if richting == "left":  
    coordX -= 10
```

```
elif richting == "right":  
    coordX += 10
```

```
elif richting == "up":  
    coordY -= 10
```

```
elif richting == "down":  
    coordY += 10
```



Wat doen we hierboven? De richting wordt aangepast naargelang welke knop we indrukken, maar we kunnen nooit in de tegenoverstelde richting gaan! Hij kan dus nooit achteruit gaan en kan alleen van route afwijken. Daarom dat in de if-statement staat dat hij enkel de richting mag aanpassen als de oorspronkelijke richting niet het omgekeerde is (dus als hij naar rechts gaat en jij drukt op links, zal hij niet van richting veranderen). Nu we de richting weten, moeten we ervoor zorgen dat de juiste coördinaten worden aangepast!

Randen herkennen

één van de moeilijkheden binnen snake is dat we de randen niet mogen aanraken. We willen dus controleren wanneer de slang buiten de grenzen zit. Dit wilt zeggen minder dan nul of groter dan 800 of 600.

Voorlopig gaan we instellen dat het moment dat we tegen de randen botsen het spel zich afsluit. (Later voegen we een tekst toe)

We voegen in de while lus een nieuwe if toe (**binnen** de while maar **buiten** de for, dit wilt zeggen één inspringing naar binnen)

```
if coordX >= 790 or coordX < 0 or coordY >=590 or coordY < 0:  
    gameExit = True
```

Waarom niet 800 en 600? Als we een locatie opgeven van 400 op 300 staat de linkerbovenhoek van de rechthoek op die locatie. Dit wilt zeggen dat wanneer het rechthoekje tegen de rand staat deze op op -10 van de rand staat, aangezien het rechthoekje 10 pixels breed is. Hierdoor doen we zowel -10 bij de hoogte en de breedte.

GAME OVER

Voorlopig sluit het spel zich af van het moment dat je verloren bent, natuurlijk is dat niet zo handig als je eigenlijk opnieuw wilde proberen. Hierdoor gaan we eerst vragen of je nog verder wilt of niet. Zoniet, dan sluit het spel zich af, indien wel, dan gaan we terug opnieuw beginnen.

Voordat we dit doen gaan we eerst kijken hoe we tekst moeten toevoegen aan het spel.

We maken een functie die tekst aanmaakt met de parameters tekst en kleur. Hierdoor moeten we niet telkens dezelfde code herhalen en roepen we deze functie op. We voegen de onderstaande code toe:



```
#variable
font = pygame.font.SysFont(None, 25) → lettertype + grootte

#functies

def berichtOpSchermb(bericht, kleur):
    tekst = font.render(bericht, TRUE, kleur)
    tekstRect = tekst.get_rect()
    tekstRect.center = 400, 300
    gameDisplay.blit(tekst, tekstRect)
```

Wat doen we hierboven? We maken de tekst aan, we maken van de tekst een rechthoek zodat we hier het midden kunnen berekenen. En dan wordt de tekst geplaatst op scherm (MAAR moet nog steeds geupdate worden, zie hieronder)

Nu we tekst kunnen weergeven kunnen we ook bepaalde dingen vragen. Zoals "Wilt u opnieuw proberen druk op R of afsluiten druk op Q?"

Binnen de whileloop van gameExit toevoegen we een **nieuwe** whileloop van gameOver. Op de plaats waar gameExit = False staat, daaronder maken we ook de variable aan gameOver = False.

We maken hierdoor een verschil tussen effectief **het spel beëindigen** en **verloren** zijn, want dan kunnen we alsnog opnieuw beginnen.

Wanneer we de randen van het venster aanraken moet er gameOver = True staan in plaats van gameExit. Dit moet dus even aangepast worden in de code.

Daarna voegen we onder de update() functie de hieronderstaande code toe:

```
while gameOver:
    berichtOpSchermb("GAME OVER! Wilt u opnieuw proberen druk op R of afsluiten druk op Q?", rood)
    pygame.display.update()
    for event in pygame.event.get():
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_q:
                gameExit = True
                gameOver = False
            if event.key == pygame.K_c:
                gameOver = False
                coordX = 400
                coordY = 300
                #punten = 0
                #lengte = 0
                #...
```

Als we dit gedaan hebben zou de code er zo uit zien zoals op de volgende pagina



```
import pygame                #pygame oproepen
pygame.init()                #Pygame wordt ingeladen

gameDisplay = pygame.display.set_mode((800,600)) # Een venster wordt geopend

coordX = 400
coordY = 300
richting = "RIGHT"
wit = (255, 255, 255)
rood = (255, 0, 0)
achtergrondkleur = (25,78,114) #Kies zelf de getallen!

clock = pygame.time.Clock()
font = pygame.font.SysFont(None, 25)

def berichtOpSchermb(bericht, kleur):
    tekst = font.render(bericht, True, kleur)
    tekstRect = tekst.get_rect()
    tekstRect.center = 400, 300
    gameDisplay.blit(tekst, tekstRect)

gameExit = False
gameOver = False

while not gameExit:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            gameExit = True
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT and richting != "RIGHT":
                richting = "LEFT"
            elif event.key == pygame.K_RIGHT and richting != "LEFT":
                richting = "RIGHT"
            elif event.key == pygame.K_UP and richting != "DOWN":
                richting = "UP"
            elif event.key == pygame.K_DOWN and richting != "UP":
                richting = "DOWN"

        if richting == "LEFT":
            coordX -= 10
        elif richting == "RIGHT":
            coordX += 10
        elif richting == "UP":
            coordY -= 10
        elif richting == "DOWN":
            coordY += 10

        if coordX >= 790 or coordX < 0 or coordY >=590 or coordY < 0:
            gameOver = True

    gameDisplay.fill(achtergrondkleur) #Achtergrond wordt ingekleurd met bepaalde kleur
    pygame.draw.rect(gameDisplay, wit, [coordX, coordY, 10, 10])
    pygame.display.update() #python laadt venster constant
                                #opnieuw in om veranderingen aan te passen

    clock.tick(15)

    while gameOver:
        berichtOpSchermb("GAME OVER! Wilt u opnieuw proberen druk op R of afsluiten druk op Q?", rood)
        pygame.display.update()
        for event in pygame.event.get():
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_q:
                    gameExit = True
                    gameOver = False
                if event.key == pygame.K_r:
                    gameOver = False
                    coordX = 400
                    coordY = 300
                    #punten = 0
                    #lengte = 0
                    #...

pygame.quit() #programma sluiten
quit() #effectief sluiten
```



Een appel hier, een appel daar

Natuurlijk missen we nog een zeer belangrijk onderdeel van het spel namelijk de appels die we moeten proberen op te eten. Hierbij moeten we op een willekeurige plaats de appel plaatsen. Wanneer de appel en de slang in contact komen dan wordt de slang langer en verandert de appel van locatie.

Voor de willekeurige plaats moeten we random importeren.

```
import random
```

Bij variable maken we twee nieuwe variable aan:

```
randAppleX = round(random.randrange(0, 790)/10.0)*10.0  
randAppleY = round(random.randrange(0, 590)/10.0)*10.0
```

Wat doen we hierboven? We generen een willekeurig getal tussen 0 en de hoogte of de breedte min de breedte van het blokje, zodat de appel niet buiten de rand valt. Hij geeft een willekeurig getal dus het kan zijn dat hij op locatie 17 op 17 staat. Maar ons rechthoekje gaat altijd op locatie staan dat deelbaar is door 10. (10, 20, 30, 40, 50, ...) waardoor het rechthoekje nooit op locatie 17 op 17 gaat komen. Hierdoor voegen we "round(.../10.0)*10.0" toe. Wat doet dit? Stel nu dat het getal 17 is. Deze delen we door 10 en krijgen 1.7. We ronden dit af, waardoor we 2.0 krijgen en nu doen we opnieuw maal 10, waardoor we de locatie 20 krijgen. Hier kunnen de appel en de slang elkaar perfect tegen komen en kan de slang de appel opeten, maar eerst moeten we de appel nog tekenen. Boven de rect binnen de whileloop die we hebben voor de slang maken we een nieuwe rechthoek:

```
pygame.draw.rect(gameDisplay, rood, [randAppleX,  
randAppleY, 10, 10])
```

We gaan nu controleren of de locatie van de slang hetzelfde is van de locatie van de appel, als dit gebeurt, eet de slang de appel op en verschijnt de appel op een andere plaats. Plaats deze onder de if waar we controleren of de slang buiten het venster is.

```
if coordX == randAppleX and coordY == randAppleY:  
    randAppleX = round(random.randrange(0, 790)/10.0)*10.0  
    randAppleY = round(random.randrange(0, 590)/10.0)*10.0
```




Ons spel begint eindelijk wat vorm te krijgen! Alleen moet onze slang nog wat langer worden.

De slang langer laten worden

Om de slang langer te laten worden gaan we achter het hoofd nieuwe rechthoeken tekenen.



Locaties

1		<code>[[x , y], [x1 , y1], [x2 , y2], [x3 , y3]]</code>
2		<code>[[x1 , y1], [x2 , y2], [x3 , y3], [x4 , y4]]</code>
3		<code>[[x2 , y2], [x3 , y3], [x4 , y4], [x5 , y5]]</code>

-> eerste locatie valt weg, er komt een nieuwe locatie bij. Dit is altijd de locatie van het slangen hoofd

We gaan dus gebruik maken van een lijst waar we bijhouden wat de locaties zijn van de rechthoeken. Wanneer hij een appel eet voegen we een extra locatie toe aan de lijst. Bij de variables voegen we twee variables toe.

```
snakeList = []  
snakeLength = 1
```

We beginnen met een lege lijst en met een lengte van één rechthoek.

In de whileloop gaan we bijhouden wat de locatie is van het hoofd, deze is altijd gelijk aan coordX en coordY. Deze locaties plaatsen we dan in snakeList.

In code schrijven we dat zo (binnen de whileloop, let op voor de inspringing) :

```
snakeHead = []  
snakeHead.append(coordX)  
snakeHead.append(coordY)  
snakeList.append(snakeHead)  
  
if len(snakeList) > snakeLength:  
    del snakeList[0]  
  
for eachSegment in snakeList[:-1]:  
    if eachSegment == snakeHead:  
        gameOver = True  
snake(snakeList)
```

Wat doen we hierboven allemaal?

We maken een lijst snakeHead en hieraan voegen we coordX en coordY toe, de locatie van onze slang. Dit wordt dan toegevoegd aan de lijst van de slang. Elke keer dat de slang



opschuift, wordt de snakeHead leeggemaakt en opnieuw gevuld met de nieuwe locaties die wederom worden toegevoegd aan de lijst van de slang.

Daaronder controleren we de lengte van de slang. Als snakeLength gelijk is aan 1 zal de laatste locatie van de lijst van de slang verwijderd worden (dit is de oude locatie, de locatie waar hij de vorige keer stond en dus -10 van de locatie waar hij nu staat) waardoor er maar één segment over blijft. Wanneer we een appel eten wordt er dus snakeLength += 1 gedaan. Zoals je hieronder kunt zien:

```
if coordX == randAppleX and coordY == randAppleY:
    randAppleX = round(random.randrange(0, 790)/10.0)*10.0
    randAppleY = round(random.randrange(0, 590)/10.0)*10.0
    snakeLength += 1|
```

In het volgende lijntje code zien we een for-lus die doorloopt elke locatie van de lijst van de slang, zodat we deze kunnen vergelijken in de if-statement die eronder staat. Als er een locatie overeenkomt met de locatie van het hoofd, dan wilt dit zeggen dat het hoofd gebotst is tegen zijn eigen lichaam en dan is gameOver = True. Waarom snakeList[:-1] Dit wilt zeggen dat hij alleen het laatste deel van de lijst niet controleert, waarom? Omdat dat juist de locatie is van het slangenhoofd en dat moeten we natuurlijk niet vergelijken.

Tenslotte staat er snake(snakeList) Dit is een functie die in vervanging komt van de pygame.draw.rect(). Aangezien we nu veel meer rechthoeken moeten tekenen dan gewoon eentje. We moeten deze functie wel nog aanmaken, deze plaatsen we tussen de functies!

```
def snake(snakeList):
    for XnY in snakeList:
        pygame.draw.rect(gameDisplay, wit, [XnY[0],
XnY[1], 10, 10])
```

Simpel gezegd vragen we hier alle locaties van de segmenten op uit de lijst snakeList en maken we er verschillende rechthoeken van. XnY[0] is de X-coördinaat en XnY[1] is de Y-coördinaat per lichaamsdeel.

Het enige wat je nog moet doen is bij de gameover while-lus je snakeList en snakelength terug resetten als we terug opnieuw zouden spelen.

```
if event.key == pygame.K_r:
    gameOver = False
    coordX = 400
    coordY = 300
    snakeList = []
    snakeLength = 1
```



Als je dit allemaal gedaan hebt en het werkt, dan heb je een prima werkende snake-spel! Is dit laatste deel toch niet goed gelukt? Steek even je hand omhoog zodat de coach het met jou kan overlopen!

Wat nu? We gaan nu verder experimenteren binnen snake en pygame:

- Verander de kleuren van slang, achtergrond, tekst, ...
- Toevoeg een score die telkens +1 doet wanneer je een appel opeet en wanneer je sterft de score weergeeft.
- Maak een startscherm waar je de moeilijkheid kan aanduiden (waardoor de slang sneller of trager zou gaan).
- Maak een spel voor twee spelers, waarbij er twee slangen zijn, 1 appel en je moet proberen zoveel mogelijk punten te verzamelen, maar je mag de randen niet aanraken en de andere slang ook niet.
- ...

Ben je snake wat beu? Probeer dan een nieuw file aan te maken waarbij je experimenteert met hetgeen dat we geleerd hebben binnen deze cursus. Probeer dingen uit met de events (objecten bewegen, kleuren veranderen bij het drukken op een bepaalde knop, ...)