Gratis download
voor alle Elektor-leden: PDF-special van 60 pagina's
met deel 1 t/m 10 van deze serie. Zie:
www.elektormagazine.nl/news/GUI-PDF-Special

Deel 05

GUI's maken met Python: Boter-kaas-en-eieren

Gebruik je GUI om een eenvoudig spel te besturen.

MAKER
Laura Sach

Laura leidt het A Level-team bij de Raspberry Pi Foundation, waar ze leermiddelen maakt voor leerlingen om te leren over computerwetenschappen.

@CodeBoom

MAKER
Martin O'Hanlon

Martin werkt in het leerteam van de Raspberry Pi Foundation, waar hij online cursussen, projecten en leermiddelen creëert.

@martinohanlon

Nu je geleerd hebt hoe je een basis-GUI maakt, is het tijd om wat meer programmeerlogica achter de schermen toe te voegen om met je GUI een spelletje boter-kaas-en-eieren ('Tic tac toe' in het Engels) te besturen.

Maak een nieuw bestand met de volgende code:

```
# Imports ---------------
from guizero import App

# Functions -------------

# Variables -------------

# App -------------------
app = App("Tic tac toe")

app.display()
```

Opzetten van het bord

We beginnen met het maken van de widgets die het speelbord zullen vormen. Een traditioneel boter-kaas-en-eieren bord ziet eruit zoals in **Figuur 1**.

Je gebruikt knoppen om elke positie op het bord weer te geven, zodat de speler op een van de knoppen kan klikken om aan te geven waar hij heen wil. Om de knoppen in een raster te kunnen plaatsen, maken we een nieuw type guizero-widget, een Box.

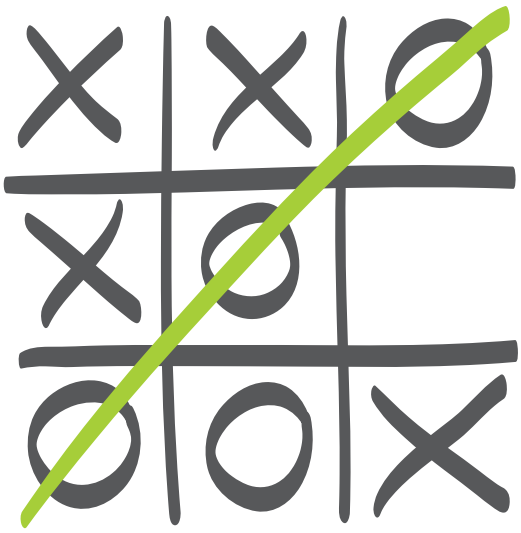
Een Box is een container-widget. Dit betekent dat hij gebruikt wordt om er andere widgets in te stoppen en te groeperen. Voeg hem toe aan de sectie imports bovenaan je code:

```
from guizero import App, Box
```

Stel de box in op een rasterlay-out en voeg hem toe aan je app - voor de `app.display()` regel, zoals met alle widgets.

```
board = Box(app, layout="grid")
```

Figuur 1



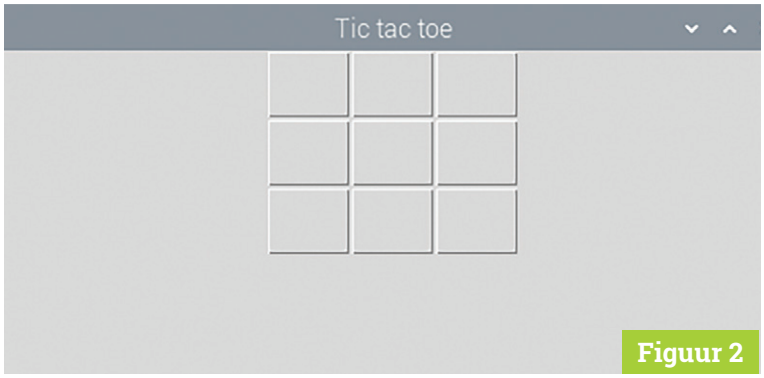
▲ **Figuur 1** Een typisch spelletje boter-kaas-en-eieren.

Als je je programma op dit moment uitvoert, zul je niets op het scherm zien omdat de Box zelf onzichtbaar is.

We maken nu de knoppen die er in moeten. Je hebt in totaal negen knoppen nodig. Dus in plaats van ze allemaal apart te maken, kun je een geneste lus gebruiken om ze automatisch te genereren en coördinaten te geven. Voeg eerst PushButton toe aan je lijst van te importeren widgets en voeg dan deze code toe onmiddellijk na de code voor het bord dat je net hebt gemaakt.

```
for x in range(3):
    for y in range(3):
        button = PushButton(
            board, text="", grid=[x, y],
width=3
        )
```

Merk op dat er twee lusvariabelen zijn: `x` van 0 tot 2 en `y` van 0 tot 2. Terwijl we itereren



▲ **Figuur 2** Een raster van negen knoppen om boter-kaas-en-eieren te spelen.

tictactoe1.py

DOWNLOAD DE VOLLEDIGE CODE:
magpi.cc/guizerocode

Taal: **Python 3**

```
001.  # Imports ---------------
002.  from guizero import App, Box, PushButton
003.
004.  # Functions -------------
005.
006.  # Variables -------------
007.
008.  # App -------------------
009.  app = App("Tic tac toe")
010.
011.  board = Box(app, layout="grid")
012.  for x in range(3):
013.          for y in range(3):
014.          button = PushButton(
      board, text="", grid=[x, y], width=3)
015.
016.  app.display()
```

tictactoe2.py

Taal: **Python 3**

```
001.  # Imports ---------------
002.  from guizero import App, Box, PushButton
003.
004.  # Functions -------------
005.  def clear_board():
006.      new_board = [[None, None, None], [None, None, None], [
      None, None, None]]
007.      for x in range(3):
008.          for y in range(3):
009.              button = PushButton(
010.                  board, text="", grid=[x, y], width=3)
011.              new_board[x][y] = button
012.      return new_board
013.
014.  # Variables -------------
015.
016.  # App -------------------
017.  app = App("Tic tac toe")
018.
019.  board = Box(app, layout="grid")
020.  board_squares = clear_board()
021.
022   app.display()
```

en knoppen genereren, zal elke knop worden toegevoegd aan het bord, dat wil zeggen de Box-container die je eerder hebt gemaakt. De knop krijgt de rastercoördinaten `x,y,` wat betekent dat elke knop netjes op een raster wordt geplaatst, elk op een andere positie!

Je code zou er nu uit moeten zien als **tictactoe1.py**. Het resultaat als je het uitvoert, is te zien in **Figuur 2**.

Onderliggende datastructuur

Het zal je opvallen dat wanneer je de knoppen aanmaakt via een lus, je automatisch negen knoppen aanmaakt en dat ze allemaal `button` heten. Hoe kun je in het programma naar elk van deze knoppen verwijzen?

Het antwoord is dat je een onderliggende datastructuur nodig hebt met een verwijzing naar elke knop. Hiervoor zullen we een tweedimensionale lijst gebruiken.

Laten we een functie maken die we kunnen aanroepen om het bord leeg te maken. Het is een goed idee om dit in een functie te doen, zodat je de code kunt hergebruiken nadat het spel gespeeld is om het bord te resetten en de speler een nieuw spel te laten beginnen.

Voeg in de sectie functions een nieuwe functie toe met de naam `clear_boar`.

```
def clear_board():
```

Je eerste taak in deze functie is om de datastructuur voor het bord te initialiseren. Laten we aannemen dat je op dit punt nog geen knoppen hebt gemaakt. Dus je kunt elke positie op het bord initialiseren als `None` - het element in de lijst bestaat nu, maar heeft nog geen waarde. Voeg de volgende regel, ingesprongen, toe aan je functie.

```
new_board = [[None, None, None], [None,
None, None], [None, None, None]]
```

Verplaats vervolgens de geneste lus-code van je app-sectie naar de `clear_board`-functie. Zorg ervoor dat de inspringing correct is.

Voeg binnen de binnenste (y)-lus een regel code toe om een verwijzing naar elke knop op te slaan op zijn x,y-coördinaatpositie binnen de tweedimensionale lijst, zodat je er later naar kunt verwijzen.

```
new_board[x][y] = button
```

Tot slot, nadat de lussen zijn afgelopen, zorg je met return dat de functie het new_board teruggeeft dat je zojuist gemaakt hebt. Je functie zou er als volgt uit moeten zien:

Het spel resetten

In het begin heb je een functie geschreven genaamd **`clear_board`**. Dit leek op dat moment misschien onnodig, maar in feite dacht het vooruit tot wanneer het spel afgelopen was. Aangezien boter-kaas-en-eieren een vrij kort spel is, is het waarschijnlijk dat iemand meer dan één spel achter elkaar wil spelen. Kun je een reset-knop toevoegen aan je spel, die alleen verschijnt als iemand het spel heeft gewonnen, of als het gelijkspel is geworden? De knop moet de functie **`clear_board`** aanroepen en de variabele **`turn`** resetten, evenals de melding wiens beurt het is.

Hint: je zult de documentatie van guizero moeten raadplegen om uit te vinden hoe je widgets kunt verbergen en tonen, zodat je knop niet de hele tijd zichtbaar is tijdens het spel.

Hint: maak een nieuwe functie die zorgt voor alles wat je moet doen om het spel te resetten, en roep die functie aan wanneer de reset-knop wordt ingedrukt. Vergeet niet dat je in je functie enkele variabelen als global moet specificeren.

```
def clear_board():
    new_board = [[None, None, None],
                 [None, None, None],
                 [None, None, None]]
    for x in range(3):
        for y in range(3):
            button = PushButton(
                board, text="", grid=[x,
y], width=3
            )
            new_board[x][y] = button
    return new_board
```

In de app-sectie initialiseer je nu een lijst genaamd `board_squares` en stel je deze in om de nieuwe functie aan te roepen die je zojuist hebt gemaakt.

```
board_squares = clear_board()
```

Deze variabele krijgt de waarde van `new_board` dat je in de functie hebt aangemaakt, en dat moet een leeg bord zijn met negen knoppen. Zorg ervoor dat je deze variabele aanmaakt na de code voor het maken van de Box, anders zul je proberen knoppen toe te voegen aan een container die nog niet bestaat.

Je code zal er nu uit zien als **tictactoe2.py**. Sla het programma op en voer het uit en je zou een identiek resultaat moeten zien als op het einde van de vorige stap, maar nu heb je een verborgen datastructuur in een tweedimensionale lijst om naar de knoppen te verwijzen en ze te manipuleren.

Als je wilt zien hoe je 2D-lijst er uit ziet, zou je een print commando kunnen toevoegen om de lijst `board_squares` af te drukken: `print(board_squares)`. Je zou dan negen keer `[PushButton] object with text ""` moeten zien in de shell.

tictactoe3.py

Taal: **Python 3**

```
001.  # Imports ---------------
002.  from guizero import App, Box, PushButton, Text
003.
004.  # Functions -------------
005.  def clear_board():
006.      new_board = [[None, None, None], [None, None, None], [None,
      None, None]]
007.      for x in range(3):
008.          for y in range(3):
009.              button = PushButton(board, text="", grid=[x, y],
      width=3, command=choose_square, args=[x,y])
010.              new_board[x][y] = button
011.      return new_board
012.
013.  def choose_square(x, y):
014.      board_squares[x][y].text = turn
015.      board_squares[x][y].disable()
016.
017.  # Variables -------------
018.  turn = "X"
019.
020.  # App -------------------
021.  app = App("Tic tac toe")
022.
023.  board = Box(app, layout="grid")
024.  board_squares = clear_board()
025.  message = Text(app, text="It is your turn, " + turn)
026.
027.  app.display()
```

Laat de knoppen werken

Op dit moment doen de knoppen niets als je er op drukt. Laten we een functie maken om aan de knop te koppelen, zodat wanneer hij wordt ingedrukt, de knop een X of een O laat zien, afhankelijk van welke speler hem heeft gekozen.

Maak eerst een variabele in de sectie variables om vast te leggen wiens beurt het is. Je kunt met elke speler beginnen, maar wij kiezen voor X.

```
turn = "X"
```

Dit betekent nu dat je op de GUI moet laten zien wiens beurt het is (**Figuur 3**), zodat de spelers niet in de war raken. Voeg Text toe aan je lijst van te importeren widgets:

```
from guizero import App, Box, PushButton,
Text
```

Voeg dan een nieuwe Text widget toe in de app sectie om de beurt weer te geven.

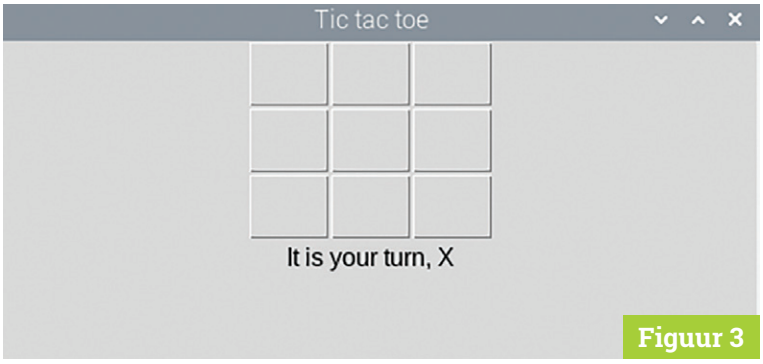
```
message = Text(app, text="It is your turn,
" + turn)
```

tictactoe4.py

Taal: **Python 3**

```
001.  # Imports ---------------
002.  from guizero import App, Box, PushButton, Text
003.
004.  # Functions -------------
005.  def clear_board():
006.      new_board = [[None, None, None], [None, None, None], [None,
      None, None]]
007.      for x in range(3):
008.          for y in range(3):
009.              button = PushButton(board, text="", grid=[x, y],
      width=3, command=choose_square, args=[x,y])
010.              new_board[x][y] = button
011.      return new_board
012.
013.  def choose_square(x, y):
014.      board_squares[x][y].text = turn
015.      board_squares[x][y].disable()
016.      toggle_player()
017.
018.  def toggle_player():
019.      global turn
020.      if turn == "X":
021.          turn = "O"
022.      else:
023.          turn = "X"
024.      message.value = "It is your turn, " + turn
025.
026.  # Variables -------------
027.  turn = "X"
028.
029.  # App -------------------
030.  app = App("Tic tac toe")
031.
032.  board = Box(app, layout="grid")
033.  board_squares = clear_board()
034.  message = Text(app, text="It is your turn, " + turn)
035.
036.  app.display()
```

Ga naar de sectie functions en maak een nieuwe functie genaamd `choose_square`.



Figuur 3

▼ **Figuur 3** Laat je spelers weten wiens beurt het is.

```
def choose_square(x, y):
```

Je ziet dat deze functie twee argumenten accepteert - x en y. Dit is om te weten op welk vakje op het bord is geklikt.

Voeg de volgende code (ingesprongen) toe in de functie om de tekst in de knop waarop geklikt is in te stellen op het symbool van de huidige speler, en vervolgens de knop uit te schakelen zodat er niet meer op geklikt kan worden.

```
board_squares[x][y].text = turn
board_squares[x][y].disable()
```

Tot slot, verbind deze functie met de knop. Zoek deze regel code in je functie `clear_board`:

```
button = PushButton(board, text="",
grid=[x, y], width=3)
```

Wijzig hem zodat hij eruit ziet als de regel hieronder:

```
button = PushButton(board, text="",
grid=[x, y], width=3, command=choose_square,
args=[x,y])
```

Je hebt hier twee dingen toegevoegd. Ten eerste, je hangt er een commando aan, net als eerst. Wanneer de knop wordt ingedrukt, zal de functie met deze naam worden aangeroepen. Ten tweede geef je ook argumenten aan deze functie, namelijk de coördinaten x en y van de knop die werd ingedrukt, zodat je die knop weer kunt vinden in de lijst.

Je programma zou er nu uit moeten zien als **tictactoe3.py**. Sla het op en start het. Je kunt nu op een knop klikken en die zal veranderen in een X. Helaas is het in deze versie van het spel permanent de beurt van X!

Afwisselen tussen spelers

Als een speler eenmaal aan de beurt is, moet de variabele `turn` veranderen in die van de andere speler. Hier is een functie die van X naar O wisselt en vice versa.

```
def toggle_player():
    global turn
    if turn == "X":
        turn = "O"
    else:
        turn = "X"
```

Voeg de code toe in de sectie functions. Let op de

tictactoe5.py

Taal: **Python 3**

```
001.  # Imports ---------------
002.  from guizero import App, Box, PushButton, Text
003.
004.  # Functions -------------
005.  def clear_board():
006.      new_board = [[None, None, None], [None, None,
      None], [None, None, None]]
007.      for x in range(3):
008.          for y in range(3):
009.              button = PushButton(board, text="",
      grid=[x, y], width=3, command=choose_square,
      args=[x,y])
010.              new_board[x][y] = button
011.      return new_board
012.
013.  def choose_square(x, y):
014.      board_squares[x][y].text = turn
015.      board_squares[x][y].disable()
016.      toggle_player()
017.      check_win()
018.
019.  def toggle_player():
020.      global turn
021.      if turn == "X":
022.          turn = "O"
023.      else:
024.          turn = "X"
025.      message.value = "It is your turn, " + turn
026.
027.  def check_win():
028.      winner = None
029.
030.      # Vertical lines
031.      if (
032.          board_squares[0][0].text ==
      board_squares[0][1].text == board_squares[0][2].text
033.      ) and board_squares[0][2].text in ["X", "O"]:
034.          winner = board_squares[0][0]
035.      elif (
036.          board_squares[1][0].text ==
      board_squares[1][1].text == board_squares[1][2].text
037.      ) and board_squares[1][2].text in ["X", "O"]:
038.          winner = board_squares[1][0]
039.      elif (
040.          board_squares[2][0].text ==
      board_squares[2][1].text == board_squares[2][2].text
041.      ) and board_squares[2][2].text in ["X", "O"]:
042.          winner = board_squares[2][0]
043.
044.      # Horizontal lines
045.      elif (
046.          board_squares[0][0].text ==
       board_squares[1][0].text == board_squares[2][0].text
047.      ) and board_squares[2][0].text in ["X", "O"]:
048.          winner = board_squares[0][0]
049.      elif (
050.          board_squares[0][1].text ==
      board_squares[1][1].text == board_squares[2][1].text
051.      ) and board_squares[2][1].text in ["X", "O"]:
052.          winner = board_squares[0][1]
053.      elif (
054.          board_squares[0][2].text ==
      board_squares[1][2].text == board_squares[2][2].text
055.      ) and board_squares[2][2].text in ["X", "O"]:
056.          winner = board_squares[0][2]
057.
058.      # Diagonals
059.      elif (
060.          board_squares[0][0].text ==
      board_squares[1][1].text == board_squares[2][2].text
061.      ) and board_squares[2][2].text in ["X", "O"]:
062.          winner = board_squares[0][0]
063.      elif (
064.          board_squares[2][0].text ==
      board_squares[1][1].text == board_squares[0][2].text
065.      ) and board_squares[0][2].text in ["X", "O"]:
066.          winner = board_squares[0][2]
067.
068.      if winner is not None:
069.          message.value = winner.text + " wins!"
070.
071.  # Variables -------------
072.  turn = "X"
073.
074.  # App -------------------
075.  app = App("Tic tac toe")
076.
077.  board = Box(app, layout="grid")
078.  board_squares = clear_board()
079.  message = Text(app, text="It is your turn, " + turn)
080.
081.  app.display()
```

eerste regel in de functie: `global turn`. Je moet dit specificeren zodat je de globale versie van de variabele `turn` mag wijzigen, d.w.z. die ene die je al hebt aangemaakt. Als je dit niet specificeert, maakt Python een lokale variabele genaamd `turn` en wijzigt die in plaats daarvan, maar die wijziging wordt niet opgeslagen als de functie wordt afgesloten.

Je moet er ook voor zorgen dat de Text widget de beurt van de huidige speler nauwkeurig weergeeft. Na het if/else statement in de functie `toggle_player` pas je de melding als volgt aan:

```
message.value = "It is your turn, " + turn
```

Ga terug naar de functie `choose_square` en roep de functie `toggle_player` aan - met `toggle_player()` - zodra je de tekst hebt ingesteld en de knop hebt uitgeschakeld. Je code zou nu moeten lijken op **tictactoe4.py**. Bewaar en test het programma opnieuw en je zou moeten merken dat je op vierkantjes kunt klikken en dat ze om beurten met X of O aangeduid worden.

Hebben we een winnaar?

Tot slot moet je een functie schrijven die controleert of er een rij, kolom of diagonaal van drie X'en of O's is, en zo ja, de winnaar van het spel meldt.

Hoewel het erg onelegant lijkt, de gemakkelijkste manier om te controleren of iemand gewonnen heeft, is het hard coderen van de controles voor elke verticale, horizontale en diagonale lijn apart.

De volgende code is voor één verticale lijn, één horizontale lijn, en één diagonale - kun jij de rest toevoegen?

Globale variabelen

Het is misschien een slecht idee om globale variabelen te gebruiken, want als je veel functies in een groot programma hebt, kan het erg verwarrend worden welke code de waarde van een variabele wijzigt en wanneer. In een klein programma als dit, is het niet al te moeilijk om het bij te houden.

Onthoud dat het mogelijk is om de waarde van een globale variabele te lezen en te gebruiken vanuit een functie zonder deze als global te declareren, maar om de waarde te wijzigen moet je dit expliciet declareren. De functies in dit programma (en de meeste GUI-programma's in deze tutorial serie) wijzigen eigenlijk de waarden van je widgets als globale variabelen. Bijvoorbeeld, wanneer iemand het spel wint, stel je de waarde van de melding in om weer te geven wie er gewonnen heeft:

```
message.value = winner.text + " wins!"
```

In dit voorbeeld is **message** een globale variabele. Dus hoe kunnen we zijn waarde wijzigen zonder hem als global te declareren? Het antwoord is dat we gebruik maken van een eigenschap van de widget message, de eigenschap value. In wezen is wat deze code zegt: "Hé Python, ken je die widget daar die message heet? Nou, zou je zijn value-eigenschap kunnen wijzigen alsjeblieft?" Python staat wijziging toe via objecteigenschappen in de global scope, maar hij staat niet toe dat je rechtstreeks de waarde van een variabele wijzigt zonder deze als global te declareren.

Python 3 for Science and Engineering Applications

Als je de basis van Python onder de knie hebt en de taal verder wilt uitdiepen, dan is dit boek iets voor jou. Aan de hand van concrete voorbeelden die in verschillende toepassingen worden gebruikt, illustreert het boek vele aspecten van het programmeren (bv. algoritmen, recursie, datastructuren) en helpt het bij probleemoplossende strategieën.
www.elektor.nl/python-3-for-science-and-engineering-applications

```
def check_win():
    winner = None

    # Vertical lines
    if (
        board_squares[0][0].text == board_
squares[0][1].text == board_squares[0][2].text
    ) and board_squares[0][2].text in ["X", "O"]:
        winner = board_squares[0][0]

    # Horizontal lines
    elif (
        board_squares[0][0].text == board_
squares[1][0].text == board_squares[2][0].text
    ) and board_squares[2][0].text in ["X", "O"]:
        winner = board_squares[0][0]

    # Diagonals
    elif (
        board_squares[0][0].text == board_
squares[1][1].text == board_squares[2][2].text
    ) and board_squares[2][2].text in ["X", "O"]:
        winner = board_squares[0][0]
```

Merk op dat de functie begint met het creëren van een booleaanse variabele genaamd winner. Als tegen de tijd dat het lange if/elif-statement is uitgevoerd, de waarde van deze variabele True is, weet je dat iemand het spel heeft gewonnen.
Na het toevoegen van de resterende checks op de winnende lijn, voeg je aan het eind van de functie wat code toe om de melding op het scherm te veranderen als er een winnaar is geweest:

```
if winner is not None:
    message.value = winner.text + " wins!"
```

Je moet er nu voor zorgen dat deze functie wordt aangeroepen telkens als er een X of O wordt geplaatst, wat overeenkomt met elke keer dat er op een knop wordt gedrukt. Voeg aan het einde van de functie `choose_square` een aanroep van `check_win` toe, voor het geval dat het gekozen vierkant het winnende vierkant was.

Je programma zou er nu uit moeten zien als tictactoe5.py. Start het en test het spel. Als je de checks in de functie `check_win` correct hebt geschreven, zou je moeten merken dat het spel correct detecteert wanneer een speler gewonnen heeft.

Gelijkspel

Op dit moment zal het spel je toelaten om verder te spelen zelfs nadat het gewonnen is, totdat alle vierkanten geselecteerd zijn. Het zal je ook niet vertellen of het spel gelijkspel was. Je zou op dit punt kunnen stoppen, maar als je echt de kers op de taart wilt zetten, zou het toevoegen van nog een paar kleine details je spel netter kunnen maken.

Laten we eerst wat code toevoegen om te detecteren of het spel gelijkspel is. Het spel is gelijkspel als alle vakjes een X of een O bevatten, en niemand heeft gewonnen. Maak in de sectie functions een nieuwe functie genaamd `moves_taken`:

```
def moves_taken():
```

Je gaat deze functie gebruiken om het aantal zetten te tellen dat gedaan is, dus laten we een variabele starten om de telling bij te houden, aanvankelijk beginnend bij 0.

```
def moves_taken():
  moves = 0
```

Weet je nog, toen we `board_squares` maakten, we een geneste lus gebruikten om alle vierkanten op het raster te maken? We hebben hier nog een geneste lus nodig om elk vierkant te controleren en te bepalen of het ingevuld is met een X of O, of dat het leeg is. Voeg deze code voor een geneste lus toe aan de functie `moves_taken`:

```
for row in board_squares:
    for col in row:
```

Binnen de lus moeten we controleren of dat specifieke vierkant is ingevuld met een X of een O. Als dat zo is, voeg dan 1 toe aan de variabele `moves` om te registreren dat het vierkant is geteld.

```
if col.text == "X" or col.text == "O":
```

Noot van de redactie: dit artikel verscheen oorspronkelijk in MagPi tijdschrift.
Elektor publiceert de Nederlandse, Franse en Duitse edities van MagPi.

06-tictactoe.py

Taal: **Python 3**

```
# Imports ---------------
from guizero import App, Box, PushButton, Text

# Functions -------------
def clear_board():
    new_board = [[None, None, None], [None, None, None], [None, None, None]]
    for x in range(3):
        for y in range(3):
            button = PushButton(board, text="", grid=[x, y], width=3, command=choose_square, args=[x,y])
            new_board[x][y] = button
    return new_board

def choose_square(x, y):
    board_squares[x][y].text = turn
    board_squares[x][y].disable()
    toggle_player()
    check_win()

def toggle_player():
    global turn
    if turn == "X":
        turn = "O"
    else:
        turn = "X"
    message.value = "It is your turn, " + turn

def check_win():
    winner = None

    # Vertical lines
    if (
        board_squares[0][0].text == board_squares[0][1].text == board_squares[0][2].text
    ) and board_squares[0][2].text in ["X", "O"]:
        winner = board_squares[0][0]
    elif (
        board_squares[1][0].text == board_squares[1][1].text == board_squares[1][2].text
    ) and board_squares[1][2].text in ["X", "O"]:
        winner = board_squares[1][0]
    elif (
        board_squares[2][0].text == board_squares[2][1].text == board_squares[2][2].text
    ) and board_squares[2][2].text in ["X", "O"]:
        winner = board_squares[2][0]

    # Horizontal lines
    elif (
        board_squares[0][0].text == board_squares[1][0].text == board_squares[2][0].text
    ) and board_squares[2][0].text in ["X", "O"]:
        winner = board_squares[0][0]
    elif (
        board_squares[0][1].text == board_squares[1][1].text == board_squares[2][1].text
    ) and board_squares[2][1].text in ["X", "O"]:
        winner = board_squares[0][1]
    elif (
        board_squares[0][2].text == board_squares[1][2].text == board_squares[2][2].text
    ) and board_squares[2][2].text in ["X", "O"]:
        winner = board_squares[0][2]

    # Diagonals
    elif (
        board_squares[0][0].text == board_squares[1][1].text == board_squares[2][2].text
    ) and board_squares[2][2].text in ["X", "O"]:
        winner = board_squares[0][0]
    elif (
        board_squares[2][0].text == board_squares[1][1].text == board_squares[0][2].text
    ) and board_squares[0][2].text in ["X", "O"]:
        winner = board_squares[0][2]

    if winner is not None:
        message.value = winner.text + " wins!"
    elif moves_taken() == 9:
        message.value = "It's a draw"

def moves_taken():
    moves = 0
    for row in board_squares:
        for col in row:
            if col.text == "X" or col.text == "O":
                moves = moves + 1
    return moves


# Variables -------------
turn = "X"

# App -------------------
app = App("Tic tac toe")

board = Box(app, layout="grid")
board_squares = clear_board()
message = Text(app, text="It is your turn, " + turn)

app.display()
```

```
    moves = moves + 1
```

Tot slot, zodra de lussen beëindigd zijn, voeg je een `return`-statement toe om het aantal gemaakte zetten terug te geven.

```
return moves
```

Roep nu deze functie aan binnen de functie `check_win`, om te controleren op een gelijkspel. Voeg deze code toe na de code die controleert op een winnaar:

```
if winner is not None:
    message.value = winner.text + " wins!"

# Add this code
elif moves_taken() == 9:
    message.value = "It's a draw"
```

Je code zou moeten lijken op **06-tictactoe.py**. Als het spel wordt uitgevoerd, zal het nu controleren of er negen zetten zijn gedaan; als dat zo is, zal het de melding veranderen om te vertellen dat de partij gelijkspel was.