Tre på rad mot datamaskinen

Erfaren Python

Introduksjon

I dag skal vi prøve å skrive kode slik at datamaskinen kan spille tre på rad mot oss. Datamaskinen vil ikke spille så bra i begynnelsen, men etterhvert som den lærer noen triks vil den kanskje klare å vinne mot deg!

Steg 1: Vi fortsetter fra forrige gang

I leksjon 6 skrev vi et tre-på-rad spill for to spillere. Vi brukte *Tk lerretet* fra `tkinter`-biblioteket for å tegne på skjermen. La oss se på hva vi allerede har før vi begynner å skrive ny kode.

Sjekkliste

- [ ] Åpne IDLE. Åpne filen fra forrige leksjon og lagre den med et nytt navn. Eller om du ikke kan finne den filen kan du kopiere inn følgende:

```
from tkinter import *

main = Tk()

c = Canvas(main, width=600, height=600)
c.pack()

c.create_line(200, 0, 200, 600)
c.create_line(400, 0, 400, 600)

c.create_line(0, 200, 600, 200)
c.create_line(0, 400, 600, 400)

grid = [
  "0", "1", "2",
  "3", "4", "5",
  "6", "7", "8",
]

def click(event):
  shape = choose_shape()
  across = int(c.canvasx(event.x) / 200)
  down = int(c.canvasy(event.y) / 200)
  square = across + (down * 3)

  if grid[square] == "X" or grid[square] == "O":
    return

  if winner():
    return

  if shape == "O":
    c.create_oval(across * 200, down * 200,
      (across+1) * 200, (down+1) * 200)
    grid[square] = "O"
  else:
    c.create_line(across * 200, down * 200,
      (across+1) * 200, (down+1) * 200)
    c.create_line(across * 200, (down+1) * 200,
      (across+1) * 200, down * 200)
    grid[square] = "X"

def choose_shape():
  if grid.count("O") > grid.count("X"):
```

```
        return "X"
    else:
        return "O"

def winner():
    for across in range(3):
        row = across * 3
        line = grid[row] + grid[row+1] + grid[row+2]
        if line == "XXX" or line == "OOO":
            return True

    for down in range(3):
        line = grid[down] + grid[down+3] + grid[down+6]
        if line == "XXX" or line == "OOO":
            return True

    line = grid[0] + grid[4] + grid[8]
    if line == "XXX" or line == "OOO":
            return True

    line = grid[2] + grid[4] + grid[6]
    if line == "XXX" or line == "OOO":
            return True

c.bind("<Button-1>", click)

mainloop()
```

- [ ] Lagre, og kjør programmet, slik at du er sikker på at det virker!

  Du skal kunne klikke i rutene for å plassere sirkler og kryss inntil noen får tre på rad.

- [ ] Før vi begynner med dagens kode vil vi gjøre en liten opprydning i koden for at vi enklere skal kunne lese hva som skjer i prosedyren `click`. Vi flytter koden som tegner sirkler og kryss til en egen prosedyre. Bytt prosedyren `click` ut med disse to prosedyrene:

```
def click(event):
    shape = choose_shape()
    across = int(c.canvasx(event.x) / 200)
    down = int(c.canvasy(event.y) / 200)
    square = across + (down * 3)

    if grid[square] == "X" or grid[square] == "O":
        return

    if winner():
        return

    grid[square] = shape
    draw_shape(shape, across, down)

def draw_shape(shape, across, down):
    if shape == "O":
        c.create_oval(across * 200, down * 200,
            (across+1) * 200, (down+1) * 200)
    else:
        c.create_line(across * 200, down * 200,
            (across+1) * 200, (down+1) * 200)
        c.create_line(across * 200, (down+1) * 200,
            (across+1) * 200, down * 200)
```

  Kjør koden og test at den fortsatt fungerer på samme måte som tidligere. Dette er et eksempel på noe som kalles refaktorering. Vi har endret på selve koden, men ikke endret hvordan programmet fungerer.

Steg 2: Spill tilfeldig

Før vi kan lære datamaskinen hvordan den gjør gode trekk vil vi lære den hvordan den gjør trekk i det hele tatt. Vi begynner med å la datamaskinen finne en tilfeldig ledig rute, og deretter spille der.

Husk at vi har en variabel som heter `grid` som kan fortelle oss hvordan brettet ser ut. Det er en liste som starter som `["0", "1", "2", ... ]`, hvor vi putter inn `"X"` og `"O"` etterhvert som vi spiller. Vi begynner med å finne ledige ruter i denne listen for deretter å spille en slik rute.

Sjekkliste

- [ ] Vi vil først lage en ny prosedyre, `free_squares`, som kan finne ledige ruter. Legg til denne koden nedenfor prosedyren `winner`, men over linjen `c.bind(...)`:

```
def free_squares():
    output = []
    for position, square in enumerate(grid):
        if square != "X" and square != "O":
            output.append(position)
    return output
```

Denne prosedyren lager en tom liste. Deretter går den gjennom hele rutenettet og sjekker hver rute om den er tom.

Kommandoen `enumerate` kan fortelle oss posisjonen til hvert element i `grid`-listen. For eksempel vil `enumerate` gjøre om en liste `['A','B','C']` til parene `(0, 'A'), (1,'B'), (2, 'C')` slik at vi ikke trenger å telle elementene selv.

- [ ] På toppen av filen vil vi importere `random`-biblioteket, som vi vil bruke for å tilfeldig velge et trekk

```
from tkinter import *
import random
```

Du husker kanskje at vi brukte `random.choice` i en tidligere leksjon om Hangman.

- [ ] Nå skriver vi en prosedyre `play_move()` som kan spille i en tilfeldig tom rute. Legg til denne prosedyren etter `free_squares` men før linjnen `c.bind(...)`

```
def play_move():
    moves = free_squares()
    square = random.choice(moves)

    across = square % 3
    down = square // 3

    grid[square] = "X"
    draw_shape("X", across, down)
```

Først bruker vi `free_squares` til å lage en liste over de tomme rutene. Deretter velger vi en tilfeldig av disse rutene. Vi vil nå oversette dette rutenummeret til rad- og kolonne-nummer. Dette gjør vi ved å bruke `%` og `//` operatorene. La oss se litt nærmere på hvordan dette virker:

```
  0 1 2
  -----
0 | 0 1 2
1 | 3 4 5
2 | 6 7 8
```

For eksempel er rute nummer 5 i rad 1 og kolonne 2. Hvis vi deler 5 på 3 får vi 1 med 2 i rest.

`5 // 3` er 1, `6 // 3` er 2, og så videre. Operatoren `//` forteller oss hvor mange ganger et tall deler et annet, men ser bort i fra resten. Siden vi har 3 kolonner forteller `5 // 3` oss i hvilken rad rute 5 er.

`5 % 3` er 2, `6 % 3` er 0. Operatoren `%` forteller oss hva resten er når vi deler et tall med et annet. Dette gir oss kolonnenummeret.

Legg merke til at de to linjene

```
across = square % 3
down = square // 3
```

gjør den motsatte utregningen av

```
square = across + (down * 3)
```

som vi allerede har brukt i `click`.

- [ ] Til slutt endrer vi `click`-prosedyren slik at den kaller `play_move`. På denne måten vil først spilleren gjøre sitt trekk, og deretter gjør datamaskinen sitt trekk.

```
def click(event):
    across = int(c.canvasx(event.x) / 200)
    down = int(c.canvasy(event.y) / 200)
    square = across + (down * 3)

    if grid[square] == "X" or grid[square] == "O":
        return

    if winner():
        return

    grid[square] = "O"
    draw_shape("O", across, down)

    if winner():
        return

    play_move()
```

Vi sjekker først om spilleren har vunnet, og hvis ikke lar vi datamaskinen gjøre sitt trekk.

- [ ] Lagre programmet og kjør det. Datamaskinen vil nå trekke etter deg. Den vil ikke spille spesielt bra siden den bare gjør tilfeldige trekk.

Steg 3: Velg et trekk som vinner

Datamaskinen spiller nå tre på rad, men den er ikke spesielt flink. La oss hjelpe den litt. I stedet for å bare velge trekk helt tilfeldig, la datamaskinen velge trekk som gjør at den vinner om de finnes. Ideen er at vi kan sjekke alle de mulige trekkene til datamaskinen, og om ett av disse vil vinne spillet lar vi datamaskinen spille det.

Sjekkliste

- [ ] Endre prosedyren `winner` slik at den tar et argument `grid`:

```
def winner(grid):
    for across in range(3):
        row = across * 3
        line = grid[row] + grid[row+1] + grid[row+2]
        if line == "XXX" or line == "OOO":
            return True

    for down in range(3):
        line = grid[down] + grid[down+3] + grid[down+6]
        if line == "XXX" or line == "OOO":
            return True

    line = grid[0] + grid[4] + grid[8]
    if line == "XXX" or line == "OOO":
            return True

    line = grid[2] + grid[4] + grid[6]
    if line == "XXX" or line == "OOO":
            return True
```

Du trenger bare å endre den første linjen i prosedyren. Dette betyr at `winner` vil bruke en liste vi sender til den, i stedet for `grid` som husker hvordan dette spillet ser ut. Dermed kan `winner` også undersøke trekk som ikke er blitt spilt enda.

- [ ] Nå må vi forandre `click` så den sender inn riktig liste.

```
def click(event):
    across = int(c.canvasx(event.x) / 200)
    down = int(c.canvasy(event.y) / 200)
    square = across + (down * 3)

    if grid[square] == "X" or grid[square] == "O":
        return

    if winner(grid):
        return

    grid[square] = "O"
    draw_shape("O", across, down)

    if winner(grid):
        return

    play_move()
```

Alle steder vi har `winner()` i koden bytter vi det ut med `winner(grid)`.

- [ ] Kjør koden, den skal fortsatt virke akkurat som før for vi har enda ikke endret hvordan datamaskinen spiller.

- [ ] La oss hjelpe datamaskinen ved å legge til noen linjer i `play_move` som kan lete etter vinnende trekk!

```
def play_move():
    moves = free_squares()
    square = random.choice(moves)

    # Bruk et vinnende trekk om det eksisterer
    for possible in moves:
        new_grid = list(grid)
        new_grid[possible] = "X"
        if winner(new_grid):
            square = possible
            break

    across = square % 3
    down = square // 3

    grid[square] = "X"
    draw_shape("X", across, down)
```

For hver ledige rute lager vi en kopi av `grid`-listen med kommandoen `list(grid)`. Deretter plasserer vi en X i denne ledige ruten og bruker `winner` for å undersøke om dette vil være et vinnende trekk!

- [ ] Kjør programmet ditt og test det flere ganger. Datamaskinen skal ha blitt litt flinkere til å spille nå.

Steg 4: Velg et trekk som blokkerer

Den andre strategien vi vil lære datamaskinen er å blokkere trekk som gjør at vi vil vinne. Dette gjør vi på nesten samme måte, men nå ser vi hva som skjer om vi plasserer ut O i de ledige rutene.

Sjekkliste

- [ ] Legg til litt mer kode i `play_move` som blokkerer trekk som gjør at spilleren kan vinne.

```
def play_move():
    moves = free_squares()
    square = random.choice(moves)

    # Bruk et blokkerende trekk om det eksisterer
    for possible in moves:
        new_grid = list(grid)
        new_grid[possible] = "O"
        if winner(new_grid):
            square = possible
            break

    # Bruk et vinnende trekk om det eksisterer
    for possible in moves:
        new_grid = list(grid)
        new_grid[possible] = "X"
        if winner(new_grid):
            square = possible
            break

    across = square % 3
    down = square // 3

    grid[square] = "X"
    draw_shape("X", across, down)
```

Legg merke til at datamaskinen først plukker en tilfeldig ledig rute. Deretter sjekker den om den kan blokkere, og hvis den kan det så ombestemmer den seg. Til slutt sjekker den om den kan vinne, og dersom den kan det så ombestemmer den seg en gang til!

- [ ] Kjør koden og se om du klarer å vinne mot datamaskinen! Det har nå blitt mye vanskeligere.

Hele programmet

Det ferdige programmet ditt vil nå se omtrent ut som dette!

```
from tkinter import *
import random

main = Tk()

c = Canvas(main, width=600, height=600)
c.pack()

c.create_line(200, 0, 200, 600)
c.create_line(400, 0, 400, 600)

c.create_line(0, 200, 600, 200)
c.create_line(0, 400, 600, 400)

grid = [
    "0", "1", "2",
    "3", "4", "5",
    "6", "7", "8",
]

def click(event):
    across = int(c.canvasx(event.x) / 200)
    down = int(c.canvasy(event.y) / 200)
    square = across + (down * 3)

    if grid[square] == "X" or grid[square] == "O":
        return

    if winner(grid):
        return
```

```
    grid[square] = "O"
    draw_shape("O", across, down)

    if winner(grid):
        return

    play_move()

def draw_shape(shape, across, down):
    if shape == "O":
        c.create_oval(across * 200, down * 200,
            (across+1) * 200, (down+1) * 200)
    else:
        c.create_line(across * 200, down * 200,
            (across+1) * 200, (down+1) * 200)
        c.create_line(across * 200, (down+1) * 200,
            (across+1) * 200, down * 200)

def winner(grid):
    for across in range(3):
        row = across * 3
        line = grid[row] + grid[row+1] + grid[row+2]
        if line == "XXX" or line == "OOO":
            return True

    for down in range(3):
        line = grid[down] + grid[down+3] + grid[down+6]
        if line == "XXX" or line == "OOO":
            return True

    line = grid[0] + grid[4] + grid[8]
    if line == "XXX" or line == "OOO":
        return True

    line = grid[2] + grid[4] + grid[6]
    if line == "XXX" or line == "OOO":
        return True

def free_squares():
    output = []
    for position, square in enumerate(grid):
        if square != "X" and square != "O":
            output.append(position)
    return output

def play_move():
    moves = free_squares()
    square = random.choice(moves)

    # Bruk et blokkerende trekk om det eksisterer
    for possible in moves:
        new_grid = list(grid)
        new_grid[possible] = "O"
        if winner(new_grid):
            square = possible
            break

    # Bruk et vinnende trekk om det eksisterer
    for possible in moves:
        new_grid = list(grid)
        new_grid[possible] = "X"
        if winner(new_grid):
            square = possible
            break

    down = square // 3
    across = square % 3

    grid[square] = "X"
    draw_shape("X", across, down)
```

```
c.bind("<Button-1>", click)

mainloop()
```

Utfordring

Det er fortsatt mulig å vinne mot datamaskinen. Kan du gjøre endringer som gjør at den spiller enda bedre? Kanskje du kan lære datamaskinen å spille perfekt?