Tre på rad

Journeyman Python PDF

Introduksjon

På tide med et nytt spill! I dag skal vi lage tre på rad, hvor spillerne etter tur merker ruter med X eller O inntil en av spillerne får tre på rad.

Steg 1: Tegne rutenettet

Vi vil tegne fire linjer, i et #-mønster, som dette:

```
_|_|_
_|_|_
 | |
```

Vi kunne brukt skilpadde-kommandoer for å tegne rutenettet, men i dag skal vi i stedet lære å bruke tk-biblioteket til tegning.

Sjekkliste

- [ ] Åpne IDLE, lag en ny fil og lagre den som 'xox.py'
- [ ] Skriv følgende kode

```
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)

mainloop()
```

- [ ] Lagre og kjør programmet ditt. Du vil se et rutenett tegnet på skjermen! Steng vinduet rutenettet ble tegnet i for å avslutte programmet ditt.

Lerretet

På samme måte som vi brukte `turtle`-biblioteket når vi tegnet med skilpadder bruker vi her `tkinter`-biblioteket. Vi lager et 600 ganger 600-piksler lerret som tegnes i et vindu med kommandoen `c = Canvas(main, width=600, height=600)`. For datamaskinen ser dette slik ut:

```
  0     200     400     600  ...
 0+--------+--------+--------+-----> bortover
  |
  |
  |
  |
200|       A       B
  |
  |
  |
  |
400|       C       D
  |
  |
  |
  |
600|
  |
...|
  V
 nedover
```

Her er punkt **A** ved 200 bortover, 200 nedover. Punkt B er ved 400 bortover, 200 nedover. Punkt **C** er ved 200 bortover, 400 nedover. Til slutt er punkt **D** ved 400 bortover, 400 nedover.

Hver av kodelinjene `c.create_line(bortover1, nedover1, bortover2, nedover2)` tegner en linje på skjermen, hvor de fire tallene beskriver hvor linjer starter og slutter. For eksempel, om vi vil tegne en linje fra **A** til **D** kan vi bruke `c.create_line(200, 200, 400, 400)`.

```
   0     200     400     600  ...
 0 +--------A--------B--------+-----> bortover
   |
   |
   |
   |
200 M                  O
   |
   |
   |
   |
400 N                  P
   |
   |
   |
   |
600 |       C       D
   |
... |
   V
  nedover
```

Med punktene som i den siste figuren vil vi tegne linjer fra A til C, B til D, M til O og N til P.

```
c.create_line(200, 0, 200, 600) # A til C
c.create_line(400, 0, 400, 600) # B til D

c.create_line(0, 200, 600, 200) # M til O
c.create_line(0, 400, 600, 400) # N til P
```

Når vi koder kaller vi ofte bortover for `x`, mens nedover ofte kalles **y**. Dette rutenettet ligner ganske mye på koordinatene du kanskje har lært om i mattetimen. Forskjellen er at her begynner vi i øvre, i stedet for nedre, venstre hjørne, slik at **y** blir større når vi går nedover.

Steg 2: Tegne en sirkel

Sjekkliste

- [ ] I den samme filen vil vi nå legge til en prosedyre som kan tegne en sirkel når du klikker med musen!

```
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)

def click(event):
    c.create_oval(200, 200, 400, 400)

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

mainloop()
```

- [ ] Kjør koden din, og klikk et sted i rutenettet. Hva skjer?

  Du skal se en sirkel i den midterste ruta på skjermen.

- [ ] La oss endre på koden slik at vi tegner sirkelen i den samme ruta som du klikker i.

  For å gjøre dette må vi finne posisjonen til muspekeren og regne ut hvilken rute i rutenettet dette tilsvarer. Dette gjør vi ved å endre på `click`-prosedyren.

```
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)

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

    c.create_oval(
        across * 200, down * 200,
        (across+1) * 200, (down+1) * 200
    )

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

mainloop()
```

  Linjen `int(c.canvasx(event.x) / 200)` finner først posisjonen til muspekeren `event.x`, gjør om denne til en lerret-posisjon, `c.canvas(event.x)` og deler denne på 200 og runder nedover slik at vi får et tall som er enten 0, 1 eller 2. Dette tallet forteller oss i hvilken kolonne muspekeren er. Linjen `int(c.canvasy(event.y) / 200)` finner på samme måte ut hvilken rad muspekeren befinner seg i.

- [ ] Kjør koden. Legg merke til at hver gang du klikker i en rute tegnes en sirkel i den ruten.

  Koden `c.create_oval(across * 200, down * 200, (across+1) * 200, (down+1) * 200)` gjør om 'Bortover 1, Nedover 2' til posisjoner på lerretet som Bortover 200, Nedover 400.

Steg 3: Holde oversikten

Tilsvarende slik vi gjorde i forrige leksjon om Hangman, vil vi nå innføre en liste som kan holde oversikten over hvor vi allerede har klikket. Dette vil være viktig når vi senere vil sjekke om man har tre på rad.

Sjekkliste

- [ ] Vi lager først en liste `grid` med ni elementer, en for hver rute. Legg til følgende kode rett før definisjonen av prosedyren `click`:

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

Vi kunne ha startet listen med ni tomme strenger, `grid = ["", "", "", "", "", "", "", "", ""]`, men ved å skrive listen som vi gjør er det enklere å huske hvordan rutene på brettet er nummerert.

- [ ] Nå vil vi registrere at vi tegner sirkler i denne listen. Bytt ut `click`-prosedyren med følgende:

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

    if grid[square] == "O":
        print("Du har allerede klikket i rute " + str(square))
    else:
        print("Du klikket i rute " + str(square))

    c.create_oval(
            across * 200, down * 200,
            (across+1) * 200, (down+1) * 200
    )
    grid[square] = "O"
```

For å teste at listen virker bruker vi en enkel `print`-kommando som forteller oss hvilken rute vi klikker i, og om vi klikker i samme rute to ganger. `str` gjør om et tall til tekst (en streng) slik at den kan skrives ut sammen med den forklarende teksten.

- [ ] Kjør koden. Klikk i forskjellige ruter slik at du skjønner hvordan vi har nummerert rutene på brettet.

Steg 4: Tegne et kryss

Vi vil nå legge til en spiller til, som tegner kryss i stedet for sirkel.

Sjekkliste

- [ ] Vi lager en prosedyre som bestemmer hvem sin tur det er. `choose_shape` undersøker `grid`-listen vår og lar det være `X` sin tur dersom det allerede er flere `O` enn `X` i listen.

- [ ] Vi utvider også `click`-prosedyren slik at den kan tegne både sirkler og kryss. Koden ser nå slik ut:

```
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 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"

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

mainloop()
```

- [ ] Kjør programmet ditt. Prøv å trykk på en rute. Det skal tegnes en O. Klikk på en annen rute. Nå tegnes en X.

Steg 5: Å finne en vinner

Nå er vi nesten ferdige med spillet, vi mangler bare å sjekke om noen får tre på rad!

Sjekkliste

- [ ] I den samme filen vil vi nå skrive en ny prosedyre `winner`. Vi kaller denne etter hvert klikk for å sjekke om en av spillerene

har vunnet.

Den ferdige koden ser ut som følger:

```
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()
```

- [ ] Prøv å spill spillet slik at du får tre på rad. Kan du klikke i noen flere ruter?

  Prosedyren `winner` undersøker de fire forskjellige måtene man kan få tre på rad på:

  - [ ] Sjekk hver rad om det er tre X'er eller O'er,
  - [ ] Sjekk hver kolonne om det er tre X'er eller O'er,
  - [ ] Sjekk diagonalen fra øvre venstre til nedre høyre hjørne,
  - [ ] Sjekk diagonalen fra øvre høyre til nedre venstre hjørne.

Steg 6:

Du er ferdig med en enkel versjon av tre på rad! Prøv å endre koden, for eksempel slik at den tegner andre symboler.

Lisens: Code Club World Limited Terms of Service **Forfatter:** Oversatt fra Code Club UK **Oversetter:** Geir Arne Hjelle