Python

Tre på rad

Skrevet av: Omsett frå Code Club UK (//codeclub.org.uk)

Oversatt av: Stein Olav Romslo

Kurs: Python
Tema: Tekstbasert, Spill
Fag: Programmering
Klassetrinn: 8.-10. klasse

Introduksjon

På tide med eit nytt spel! I dag skal me lage tre på rad, der spelarane etter tur merker ruter med `X` eller `0` til ein av spelarane får tre på rad.

Steg 1: Teikne rutenettet

Me vil teikne fire linjer, i eit #-mønster, som dette:

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

Me kunne brukt skjelpadde-kommandoar for å teikne rutenettet, men i dag skal me i staden lære å bruke tk-biblioteket til teikning.

Sjekkliste

- [ ] Åpne IDLE, lag ei ny fil og lagre den som `xox.py`.

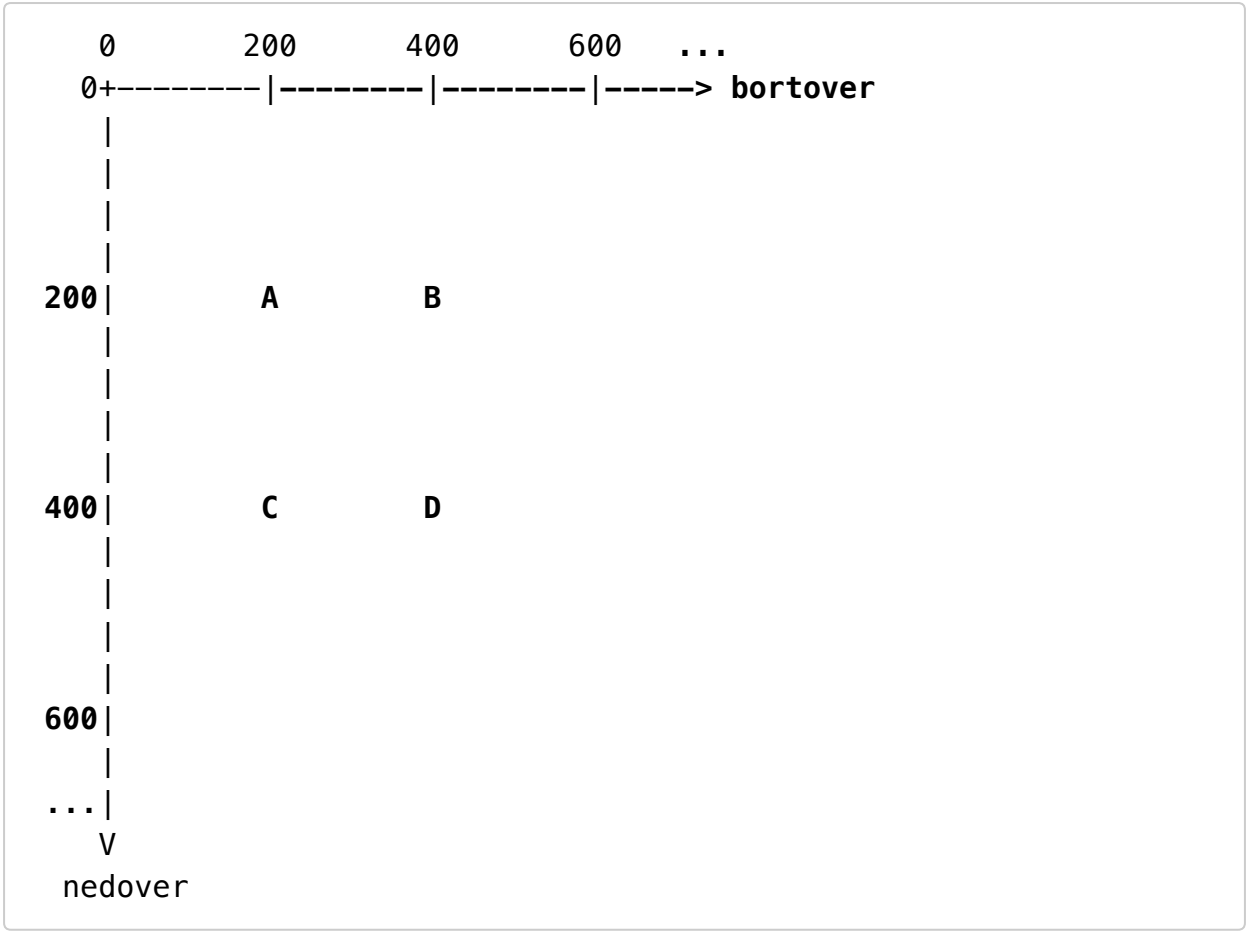
- [ ] Skriv følgjande kode

```
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 køyr programmet ditt. Du vil sjå eit rutenett som er teikna på skjermen! Steng vindauget rutenettet vart teikna i for å avslutte programmet ditt.

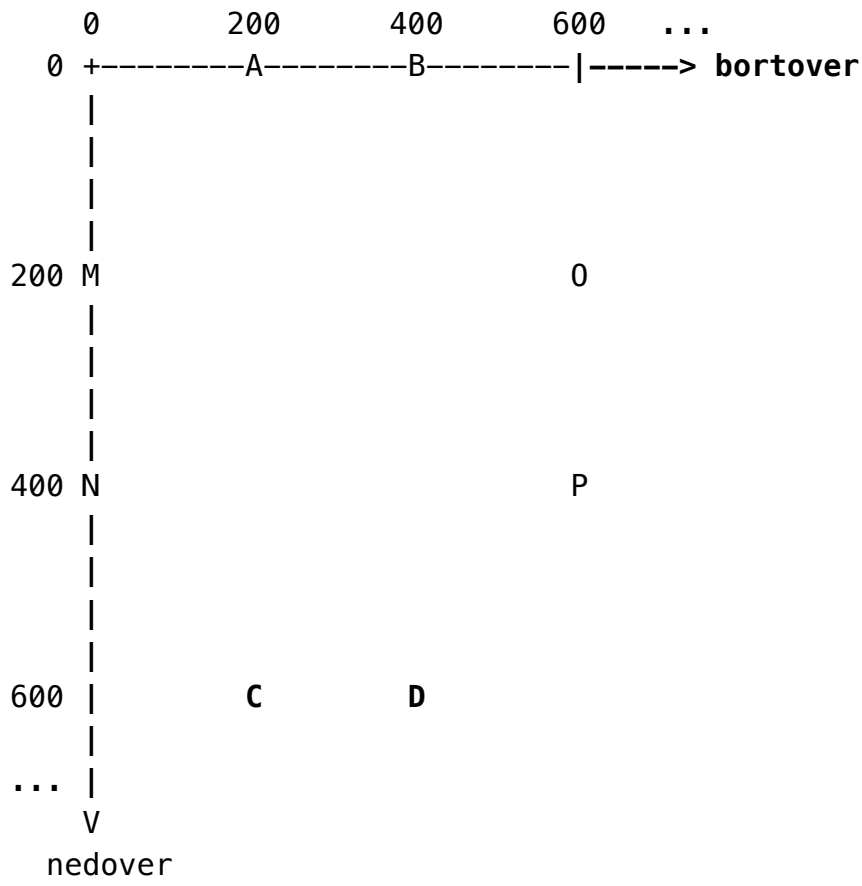
Lerretet

På same måte som me brukte `turtle` -biblioteket då me teikna med skjelpadder brukar me `tkinter` -biblioteket her. Me lagar eit `600x600` pikslar stort lerret teikna i eit vindauge med kommandoen `c = Canvas(main, width=600, height=600)` . For datamaskina ser det slik ut:



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.

Kvar av kodelinjene `c.create_line(bortover1, nedover1, bortover2, nedover2)` teiknar ei linje på skjermen, der dei fire tala beskriv kor linja starta og sluttar. Til dømes, om me vil teikne ei linje frå `A` til `D` kan me bruke `c.create_line(200, 200, 400, 400)`.



Med punkta som i den siste figuren vil me teikne linjer frå `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 me koder kallar me ofte bortover for `x`, medan nedover ofte kallast `y`. Dette rutenettet likvar ganske mykje på koordinatane du kanskje har lært om i mattetimen. Skilnaden er at her startar me i øvre, ikkje nedre, venstre hjørne, slik at `y` blir større når me går nedover.

Steg 2: Teikne ein sirkel

Sjekkliste

- [ ] I den same fila vil me no leggje til ei prosedyre som kan teikne ein sirkel når du klikkar med musa!

```
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()
```

- [ ] Køyr koden din, og klikk ein sted i rutenettet. Kva skjer?

  Du skal sjå ein sirkel i den midterste ruta på skjermen.

- [ ] La oss endre på koden slik at me teiknar sirkelen i den same ruta som du klikkar i.

  For å gjere dette må me finne posisjonen til musepeikaren og reikne ut kva rute i rutenettet dette tilsvarar. Dette gjer me ved å endre på `click` -prosedyra.

```
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()
```

Linja `int(c.canvasx(event.x) / 200)` finn fyrst posisjonen til musepeikaren, `event.x`, gjer om denne til ein lerret-posisjon, `c.canvas(event.x)`, og deler denne på 200 og runder nedover slik at me får eit tall som er anten 0, 1 eller 2. Dette tallet forteller oss i kva kolonne musepeikaren er. Linja `int(c.canvasy(event.y) / 200)` finn på same måte ut kva rad musepeikaren befinn seg i.

- [ ] Køyr koden. Legg merke til at kvar gong du klikkar i ei rute blir det teikna ein sirkel i den ruta.

Koden `c.create_oval(across * 200, down * 200, (across+1) * 200, (down+1) * 200)` gjer om `bortover 1, nedover 2` til posisjonar på lerretet som `bortover 200, nedover 400`.

Steg 3: Halde oversikta

Tilsvarande slik me gjorde då me laga Hangman, vil me no innføre ei liste som kan halde oversikt over kor me allereie har klikka. Dette vil vere viktig når me seinare sjekkar om ein har tre på rad.

Sjekkliste

- [ ] Me lagar fyrst ei liste `grid` med ni element, eitt for kvar rute. Legg til følgjande kode rett før definisjonen av prosedyra `click`:

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

Me kunne ha starta lista med ni tomme strengar, `grid = ["", "", "", "", "", "", "", "", ""]`, men ved å skrive listen som me gjer er det enklare å hugse korleis rutene på brettet er nummerert.

- [ ] No vil me registrere at me teiknar sirklar i denne lista. Bytt ut `click`-prosedyra med følgjande:

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

    if grid[square] == "0":
        print("Du har allereie klikka i rute " + str(square))
    else:
        print("Du klikka i rute " + str(square))

    c.create_oval(
            across * 200, down * 200,
            (across+1) * 200, (down+1) * 200
    )
    grid[square] = "0"
```

For å teste at lista virkar brukar me ein enkel `print`-kommando som fortel oss kva rute me klikkar i, og om me klikkar i same rute to gonger. Med `str` gjer me om eit tal til tekst (ein streng) slik at den kan skrivast ut saman med den forklarande teksten.

- [ ] Køyr koden. Klikk i forskjellige ruter slik at du forstår korleis me har nummerert rutene på brettet.

Steg 4: Teikne eit kryss

No vil me leggje til ein spelar til, som teiknar kryss i staden for sirklar.

Sjekkliste

- [ ] Me lagar ei prosedyre som bestemmer kven sin tur det er. Med `choose_shape` undersøker me `grid` -lista og let det vere `X` sin tur viss det allereie er fleire `0` enn `X` i lista.

- [ ] Me utvidar `click` -prosedyra slik at den kan teikne både sirklar og kryss. No ser koden 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 "0"

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

mainloop()
```

- [ ] Køyr programmet ditt. Prøv å trykkje på ei rute. Det skal blir teikna ein O. Klikk på ei anna rute. No blir det teikna ein X.

Steg 5: Å finne ein vinnar

No er me nesten ferdige med spelet, me manglar berre å sjekke om nokon får tre på rad!

Sjekkliste

- [ ] I den same fila vil me no skrive ei ny prosedyre `winner`. Me kallar denne etter kvart klikk for å sjekke om ein av spelarane har vunne.

  Den ferdige koden ser ut som følgjer:

```
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 spele spelet slik at du får tre på rad. Kan du klikke i fleire ruter?

Prosedyra `winner` undersøker dei fire ulike måtane ein kan få tre på rad på:

- Sjekk kvar rad om det er tre X-ar eller O-ar,
- sjekk kvar kolonne om det er tre X-ar eller O-ar,
- sjekk diagonalen frå øvre venstre til nedre høgre hjørne,
- sjekk diagonalen frå øvre høgre til nedre venstre hjørne.

Ferdig

Du er ferdig med ein enkel versjon av tre på rad! Prøv å endre koden, til dømes slik at den teiknar andre symbol.