

## 06—Tre på rad

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-lerretet til tegning.

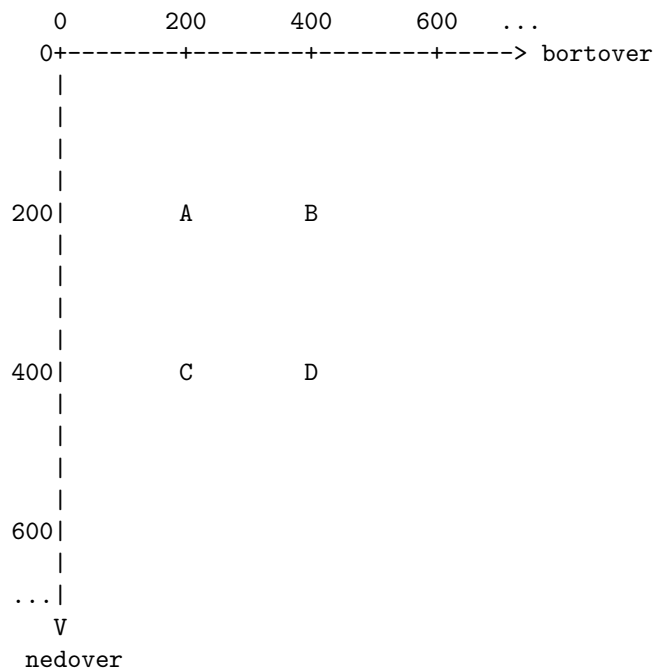
1. Åpne IDLE, lag en ny fil og lagre den som ‘xox.py’
2. Skriv følgende kode

```
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()
```

3. Lagre og kjør programmet ditt. Du vil se et rutenett tegnet på skjermen!

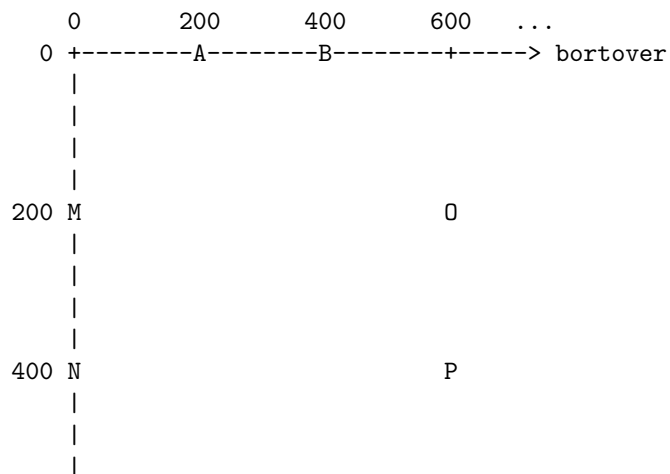
### Lerretet

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

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)`.



```

        |
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

1. I den samme filen vil vi nå legge til en prosedyre som kan tegne 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()

```

2. Kjør koden din, og klikk et sted i rutenettet. Hva skjer?

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

3. 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.

4. 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: Tegne et kryss

1. I den samme filen legger vi nå inn kode som først tegner en sirkel, deretter et kryss, deretter en sirkel, ...

```
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)

shape = "O"

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

    if shape == "O":
        c.create_oval(
            across * 200, down * 200,
            (across+1) * 200, (down+1) * 200
        )
        shape = "X"
    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
        )
        shape = "O"

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

mainloop()
```

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

Vi har brukt en ny kommando i python, `global` lar oss endre variabelen `shape` inne i prosedyren `click`.

Dersom du endrer variabler som er definert utenfor funksjoner og prosedyrer må du bruke `global` inne i funksjonen eller prosedyren.

3. Hva skjer om du trykker på samme rute to ganger på rad?

Dette skjer fordi koden vår enda ikke følger med på hva som faktisk har blitt tegnet eller hvor spillerne har flyttet. Vi må skrive litt mer kode for å holde styr på dette.

## Steg 4: Hvor er det allerede klikket?

For å hindre at spillerene klikker i samme rute to ganger vil vi følge med på hvor de klikker. For å gjøre dette bruker vi en liste kalt `grid`.

1. I den samme filen,

```
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)

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

def click(event):
    global shape, grid
    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(
        cross * 200, down * 200,
        (across+1) * 200, (down+1) * 200
    )
    grid[square] = "O"
    shape = "X"
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"
    shape = "O"

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

mainloop()

```

2. Kjør programmet ditt. Hva skjer nå om du prøver å klikke i samme rute to ganger på rad?

## Steg 5: Å finne en vinner

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

1. 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)

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

def click(event):
    global shape, grid
    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"
        shape = "X"
    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"
        shape = "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 == "000":
            return True

    for down in range(3):
        line = grid[down] + grid[down+3] + grid[down+6]
        if line == "XXX" or line == "000":
            return True

    line = grid[0] + grid[4] + grid[8]
    if line == "XXX" or line == "000":
        return True

    line = grid[2] + grid[4] + grid[6]
    if line == "XXX" or line == "000":
        return True

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

mainloop()

```

2. 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:

- (a) Sjekk hver rad om det er tre X'er eller O'er,
- (b) Sjekk hver kolonne om det er tre X'er eller O'er,
- (c) Sjekk diagonalen fra øvre venstre til nedre høyre hjørne,
- (d) 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.