

# Tre på rad

## Introduksjon

På tide med et nytt spill! I dag skal vi lage tre på rad, hvor spillerne et 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 d 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()

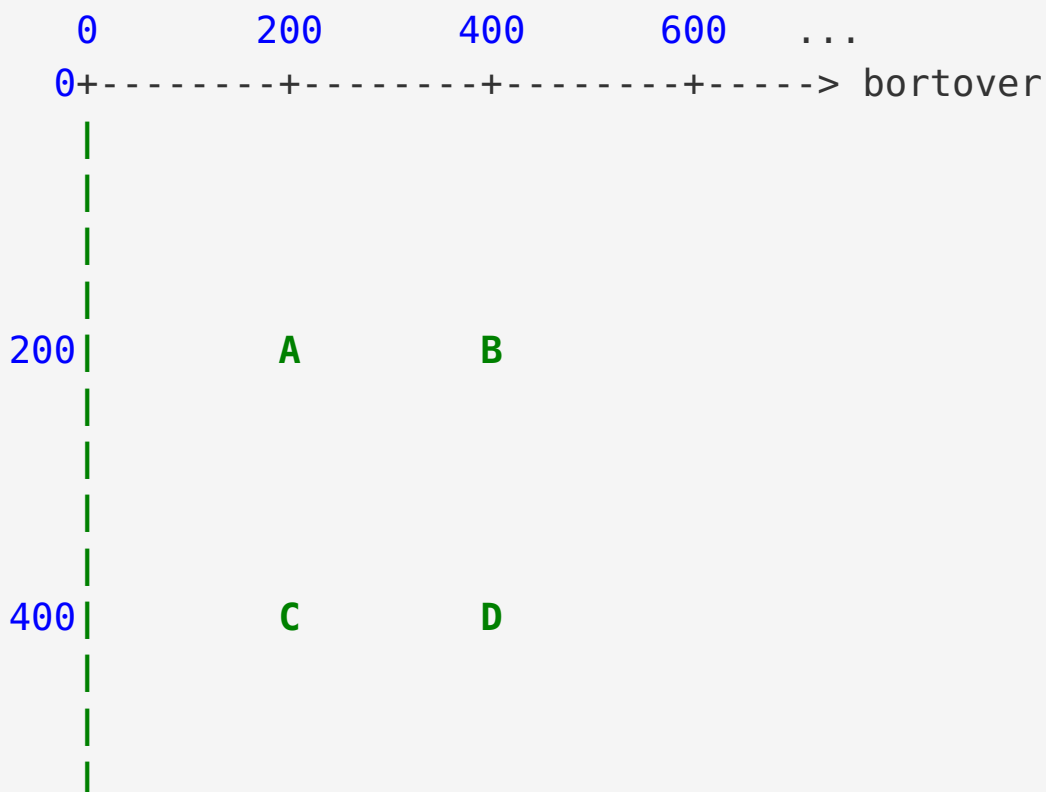
```

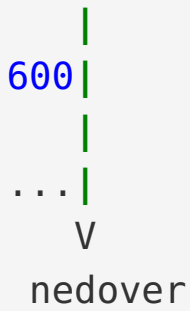


Lagre og kjør programmet ditt. Du vil se et rutenett tegnet på skrivebordet. Avslutt programmet ditt.

## Lerretet

På samme måte som vi brukte `turtle`-biblioteket når vi tegnet med turtle, kan vi lagre et 600 ganger 600-piksler lerret som tegnes i et vindu med `height=600`). For datamaskinen ser dette slik ut:





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

Hver av kodelinjene `c.create_line(bortover1, nedover1, bortov`  
de fire tallene beskriver hvor linjer starter og slutter. For eksempel  
`c.create_line(200, 200, 400, 400)`.



Med punktene som i den siste figuren vil vi tegne linjer fra A til C, E

```
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 koordinatene du kanskje har lært om i mattetimen. Forskjellen er 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

```
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.  
For å gjøre dette må vi finne posisjonen til muspekeren og regne ut hvor vi skal tegne sirkelen. Vi vil gjøre dette 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 posisjon, `c.canvas(event.x)` og deler denne på 200 og runder. Dette tallet forteller oss i hvilken kolonne muspekeren er. Linjen må ut hvilken rad muspekeren befinner seg i.

- ☐ Kjør koden. Legg merke til at hver gang du klikker i en rute tegner Koden `c.create_oval(across * 200, down * 200, (across+1) * 200, (down+1) * 200)` Nedover 2' til posisjoner på lerretet som Bortover 200, Nedover 200.

## Steg 3: Holde oversikten

Tilsvarende slik vi gjorde i forrige leksjon om Hangman, vil vi nå innføre en liste som holder oversikten over hvilke ruter allerede har blitt klikket. Dette vil være viktig når vi senere vil sjekke om en rute er klikket.

### ✓ Sjekkliste

- ☐ Vi lager først en liste `grid` med ni elementer, en for hver rute. I prosedyren `click`:

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

Vi kunne ha startet listen med ni tomme strenger, `grid = ["", "", "", "", "", "", "", "", ""]`, men 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` med:

```
def click(event):
    across = int(c.canvasx(event.x) / 200)
```

```

down = int(c.canvassy(event.y) / 200)
square = across + (down * 3)

if grid[square] == "0":
    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] = "0"

```

For å teste at listen virker bruker vi en enkel `print`-kommando klikker i samme rute to ganger. `str` gjør om et tall til tekst (en forklarende teksten).

- ☐ Kjør koden. Klikk i forskjellige ruter slik at du skjønner hvordan v

## 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. `choc` `X` sin tur dersom det allerede er flere `0` enn `X` i listen.
- ☐ Vi utvider også `click`-prosedyren slik at den kan tegne både si

```

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] == "0":
        return

    if shape == "0":
        c.create_oval(
            across * 200, down * 200,
            (across+1) * 200, (down+1) * 200
        )
        grid[square] = "0"
    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 e

## Steg 5: Å finne en vinner

Nå er vi nesten ferdige med spillet, vi mangler bare å sjekke om noen

### Sjekkliste

- ☐ I den samme filen vil vi nå skrive en ny prosedyre `winner`. Vi ka

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("0") > grid.count("X"):
        return "X"
    else:
        return "0"

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()

```

- ☐ Prøv å spill spillet slik at du får tre på rad. Kan du klikke i noen f
- Prosedyren `winner` undersøker de fire forskjellige måtene man
- ☐ 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,

**Lisens:** [Code Club World Limited Terms of Service](#) **Forfatter:** O  
**Øversetter:** Geir Arne Hjelle