



Introduksjon

La oss lage et spill: Hangman! Datamaskinen vil velge et ord og du kan gjette det bokstav for bokstav. Dersom du gjetter feil for mange ganger taper du.

Steg 1: Velg et ord

Først må vi få datamaskinen til å velge et tilfeldig ord, så la oss begynne.

✓ Sjekkliste

☐ Åpne IDLE, og åpne et nytt vindu

☐ Skriv inn følgende kode:

```
from random import choice

word = choice(["kode", "kurs"])

print(word)
```

☐ Lagre programmet ditt og kjør det. Hvilket ord skrives ut?

☐ Kjør programmet en gang til. Skriver det ut et annet ord?

Hver gang du kjører dette programmet vil det velge et tilfeldig ord fra listen `["kode", "kurs"]` ved hjelp av `choice`-funksjonen.

Steg 2: Gjett en bokstav

Nå har vi valgt et ord, la oss finne ut hvordan vi gjetter en bokstav.

✓ Sjekkliste

☐ I den samme filen, endre koden så den ser ut som følger

```
from random import choice

word = choice(["kode", "kurs"])

out = ""

for letter in word:
    out = out + "_"

print("Gjett en bokstav i ordet:", out)
```

☐ Lagre og kjør programmet.

☐ Du burde se `Gjett en bokstav i ordet: ____`, i output-vinduet (det andre vinduet, ikke vinduet du har skrevet programmet ditt i).

Vi bruker en **for**-løkke for å bygge en tekst hvor hver bokstav i ordet er byttet med en understrek **_**. Ordet **kode** vil da for eksempel skrives som **_ _ _** til skjermen.

☐ La oss gjette på en bokstav! Endre koden så den ser ut som dette

```
from random import choice

word = choice(["kode", "kurs"])

out = ""

for letter in word:
    out = out + "_"

print("Gjett en bokstav i ordet, avslutt med enter:", out)

guess = input()

if guess in word:
    print("Yay")
else:
    print("Nope")
```

Vi bruker en ny prosedyre **input()** for å finne ut hvilken bokstav spilleren skriver. Vi bruker **if** for å sjekke om bokstaven er i ordet.

Da har vi gjort det viktigste, la oss fortsette videre.

Python 2 tips:

Bruk **raw_input** i stedet for **input** dersom du bruker en gammel versjon av python.

Steg 3: Husk bokstavene som er gjettet

Nå skal vi bruke to nye komponenter i python, lister og **while**-løkker.

✓ Sjekkliste

☐ I den samme filen, endre koden så den ser slik ut:

```

from random import choice

word = choice(["kode", "kurs"])

guessed = []

while True:
    out = ""
    for letter in word:
        if letter in guessed:
            out = out + letter
        else:
            out = out + " _ "

    if out == word:
        print("Du gjettet", word)
        break

    print("Gjett en bokstav i ordet:", out)
    guess = input()

    if guess in guessed:
        print("Bokstaven er allerede gjettet på:", guess)
    elif guess in word:
        print("Yay")
        guessed.append(guess)
    else:
        print("Nope")

print()

```

- ☐ Kjør koden og prøv å gjette bokstavene.

Vi har laget en **while True**-løkke, tilsvarende **for alltid** i scratch. Denne vil i utgangspunktet fortsette å spørre spilleren om å gjette bokstaver for alltid. For å komme ut av løkken bruker vi kommandoen **break** når ordet har blitt gjettet.

Vi bruker også en liste, **guessed**, hvor vi legger til bokstavene som er riktige for å huske dem senere.

Steg 4: Tell feilene

For at Hangman skal holde oversikt over alle bokstavene som er gjettet på må vi også huske på når spilleren gjetter feil.

✓ Sjekkliste

- ☐ Endre filen du jobber med slik at den blir seende ut som dette:

```

from random import choice

word = choice(["kode", "kurs"])

guessed = []
wrong = []

while True:
    out = ""
    for letter in word:
        if letter in guessed:
            out = out + letter
        else:
            out = out + " _"

    if out == word:
        print("Du gjettet", word)
        break

    print("Gjett en bokstav i ordet:", out)

    guess = input()

    if guess in guessed or guess in wrong:
        print("Bokstaven er allerede gjettet på:", guess)
    elif guess in word:
        print("Yay")
        guessed.append(guess)
    else:
        print("Nope")
        wrong.append(guess)

    print()

```

Vi bruker en ny liste `wrong` som tar vare på alle bokstavene vi har gjettet som er feil.

Steg 5: Bare noen få forsøk

Bare en ting gjenstår før spillet er ferdig, vi vil begrense hvor mange forsøk man har til å gjette.

✓ Sjekkliste

☐ Endre filen for å legge til en ny variabel, `tries` :

```

from random import choice

word = choice(["kode", "kurs"])

guessed = []
wrong = []

tries = 7

while tries > 0:
    out = ""
    for letter in word:
        if letter in guessed:
            out = out + letter
        else:
            out = out + "_"

    if out == word:
        break

    print("Gjett en bokstav i ordet:", out)
    print(tries, "forsøk igjen")

    guess = input()

    if guess in guessed or guess in wrong:
        print("Bokstaven er allerede gjettet på:", guess)
    elif guess in word:
        print("Yay")
        guessed.append(guess)
    else:
        print("Nope")
        tries = tries - 1
        wrong.append(guess)

    print()

if tries:
    print("Du gjettet", word)
else:
    print("Du klarte ikke å gjette", word)

```

- ☐ Kjør programmet, og se hva som skjer når du gjetter feil bokstaver.

Legg merke til at vi endret **while**-løkken ved å legge inn en forutsetning, **while tries > 0**. Dette betyr at løkken bare kjøres så lenge variabelen **tries** er større enn 0. Kikker du litt rundt i koden ser du at **tries** starter med verdien 7, også blir den 1 mindre for hver feil bokstav som gjettes. Altså vil spilleren kunne gjette opp til 7 bokstaver feil før spillet er slutt.

Steg 6: Legg til nye ord

✓ Sjekkliste

- ☐ Finn linjen i programkoden som sier:

```
word = choice(["kode", "kurs"])
```

- ☐ Vi kan endre denne linjen for å legge til flere ord i spillet. Prøv for eksempel

```
word = choice(["kode", "kurs", "robot", "klubb"])
```

Husk at ordene må stå i anførselstegn og at det må være komma mellom ordene for å lage en liste. Legg til flere ord som du finner på selv.

