△ Python: Tilfeldig

PÅ BOKMÅL | LAST NED PDF

Introduksjon

Er du lei av at micro:bit-en din gjer det same kvar gong? Kanskje vil du at utfallet skal bli litt meir spanande? I denne oppgåva skal me lære korleis me kan få eininga til å oppføre seg tilsynelatande tilfeldig.

MicroPython har ein `random`-modul som gjer det enkelt å inkludere tilfeldigheiter og litt kaos inn i koden din. Til dømes er dette kode for å scrolle eit tilfeldig namn over displayet:

```
from microbit import *
import random

namn = ["Mari", "Yolanda", "Jakob", "Sofie", "Kushal", "Mei Xiu", "William" ]

display.scroll(random.choice(names))
```

Lista (`namn`) inneheldt sju namn som er definert som tekstar. Den siste linja består av nøsta funksjonar (tenk på det som ein "lauk", slik me har nemnt tidlegare). Metoden `random.choice` tek inn lista `namn` som argument, og returnerer eit tilfeldig element. Dette elementet blir argumentet til `display.scroll`.

Prøv sjølv

- [ ] Endre på lista og inkluder dine eigne namn.
- [ ] Kva bruksområde kan du kome på for ei slik liste? Prøv å finne minst to tilfelle der dette kan vere nyttig.
- [ ] Inkluder same namn fleire gonger i lista. Kva skjer?

Tilfeldige tal

Tilfeldige tal er veldig nyttige og er vanlege i spel. Veit du om andre plassar me brukar terningar?

MicroPython kjem ferdig utrusta med ei rekkje nyttige metodar for å lage tilfeldige tal. Her ser du korleis du kan lage ein enkel terning:

```
from microbit import *
import random

display.show(str(random.randint(1, 6)))
```

Kvar gong me startar micro:bit-en på nytt, så viser displayet eit tal mellom 1 og 6. Du har begynt å bli kjent med omgrepet *nøsting*, så det er viktig å hugse på at `random.randint` returnerer eit heiltal mellom to argument, inkludert endepunkta. Argumentet heiter `ranint`, ei forkorting for *random integer*, og *integer* tyder heiltal. Merk at sidan `display.show` forventar tekst så må me bruke `str`-funksjonen til å gjere om talverdien til ein tekst (til dømes å endre `6` til `"6"`).

Viss du veit at du alltid vil ha eit tal mellom `0` og `N`, så kan me bruke `random.randrange`-metoden. Den tek inn eit tal `N`, og returnerer eit tilfeldig tal opp til, men ikkje inkludert verdien til argumentet `N`. Dette er altså litt forskjellig frå `random.randint`.

Nokre gonger treng du eit desimaltal. Programmerarar kallar gjerne desse for *flyttal*, og det er mogleg å generere slike tal ved å bruke `random.random`-metoden. Dette returnerer ein verdi mellom `0.0` og `1.0` inkludert endepunkta. Viss du treng større tilfeldige desimaltal kan du leggje saman resultata frå `random.randrange` og `random.random` slik som dette

```
from microbit import *
import random

answer = random.randrange(100) + random.random()
display.scroll(str(answer))
```

Prøv sjølv

- [ ] Skriv ned det største talet programmet over kan lage.
- [ ] Skriv ned det minste talet programmet over kan lage.
- [ ] Skriv om koden for terningen over slik at den brukar `randrange` i staden for `randint`. Merk at du må gjere nokre endringar slik at du får verdiar frå 1 til 6, og ikkje 0 til 5.

HINT

Kaostilstandar

Når datamaskina di lagar tilfeldige tal er dei ikkje heilt tilfeldige. Dei gir berre tilsynelatande tilfeldige resultat gitt ein *starttilstand* (gjerne kalla *seed* på engelsk). Tilstanden er ofte laga frå tilnærma tilfeldige tal, slik som temperaturen i datamaskina di, muserørslene dei siste minutta eller klokkeslettet. Alle desse metodane kan kombinerast.

Nokre gonger vil du ha tilfeldig oppførsel som gjentek seg, ei kjelde av tilfeldigheit som er reproduserbar. Det er som å seie at du treng dei same fem tilfeldige verdiane kvar gong du kastar ein terning.

Det er heldigvis enkelt når me programmerer, sidan me kan setje starttilstanden direkte i MicroPython. Gitt ein fast starttilstand, så vil den generere dei same tilfeldige tala kvar gong. Starttilstanden set me med `random.seed` og eit positivt heiltal. Denne versjonen av terningprogrammet vil alltid produsere dei same resultata.

```
from microbit import *
import random

random.seed(1337)
while True:
```

```
if button_a.was_pressed():
    display.show(str(random.randint(1, 6)))
```

Litt juks

- [ ] Skriv om koden over slik at den alltid gir `6` dersom ein heldt inne både A- og B-knappen.
- [ ] Skriv om koden over slik at kvar gong du gjer ei bestemt rørsle viser den `6`, elles skal den virke som før.

  Python kjenner rørslene `up`, `down`, `left`, `right`, `face up`, `face down`, `freefall`, `3g`, `6g`, `8g` og `shake` viss du brukar funksjonen `accelerometer.is_gesture("rørsle")`.

HINT