Oppgaver – dag 4

I dette heftet finner du oppgaver som hører til dag 4 av Kodeskolens programmeringskurs for lærere.

Tilfeldighet og sannsynlighet

Oppgave 1 *Kaste to terninger*

I denne oppgaven skal du bruke simuleringer for å estimere sannsynligheten av å kaste 2 terninger og få nøyaktig syn øyne til sammen.

a) Lag en funkjson `kast_2d6` som bruker inpythonrandint (fra `random` pakken) til å simulere at man kaster 2 terninger returnerer summen av øynene

b) Bruk en for-løkke til å simulere kast av terning 100 ganger og tell opp antall ganger du fikk akkurat 7 med hjelp av en if test og en tellevariabel

c) Regn ut estimert sannsynlighet ved å dele antall suksesser på antall forsøk. Hva tenker du om nøyaktigheten til den estimerte sannsynligheten?

d) Endre antall runder i løkka til 10. Hva tror du om nøyaktighten nå?

e) Endre antall runder i løkka til 1000. Hvordan påvirker dette estimatet ditt?

Oppgave 2 *Kaste mange terninger*

I denne oppgaven skal du bruke simuleringer for å estimere sannsynligheten av å kaste 8 terninger og få færre enn 20 øyne tilsammen

a) Lag en funksjon `kast_1d6` som bruker inpythonrandint til å simulere at man kaster 1 terning og returnere antall øyne

b) Lag en funksjon `kast_Nd6(N)` som bruker funksjonen over sammen med en for løkke til å simulere kast av N terninger og returnerer summen av øynene

c) Bruk en for-løkke til å simulere at du kaster terningene 1000 ganger og teller opp antall suksess (færre en 20 øyne) ved hjelp av en if test og en tellevariabel.

d) Regn ut estimert sannsynlighet. Virker dette som en sannsynlig hendelse.

Geometri

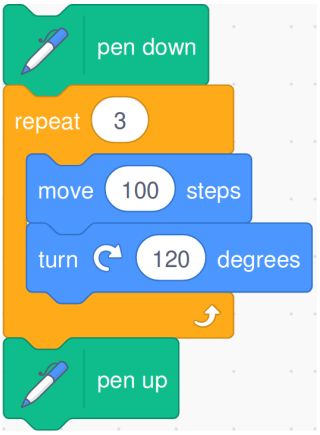
Oppgave 3 *Tegn mangekanter*

For å tegne en trekant i Python ved hjelp av skilpaddegrafikk kan vi skrive inn følgende kode:

```
from turtle import *

for _ in range(3):
    forward(100)
    right(120)
```

og for å tegne en trekant i Scratch kan vi bruke blokker, slik:



I de følgende oppgavene kan du selv velge mellom Python og Scratch.

a) Legg koden inn i en funksjon (Python) / blokk (Scratch). La sidelengden (antall steg) gis som et parameter. Prøv deg frem med å tegne trekanter i ulik størrelse. Kall funksjonen / blokken for `tegn_trekant`.

b) Lag en lignende funksjon / blokk som tegner en firkant, `tegn_firkant`, og en som tegner en femkant, `tegn_femkant`.

c) På papir: Hvis du skal tegne en generell mangekant, hvor stor blir hver vinkel? Finn en formel som sier hvor mange grader man må snu i hvert hjørne for å tegne en mangekant med n kanter.

d) Lag en funksjon / blokk som tegner en generell mangekant, `tegn_mangekant`. Gjør den det samme som `tegn_trekant`, `tegn_firkant` og `tegn_femkant` for $n = 3, 4, 5$?

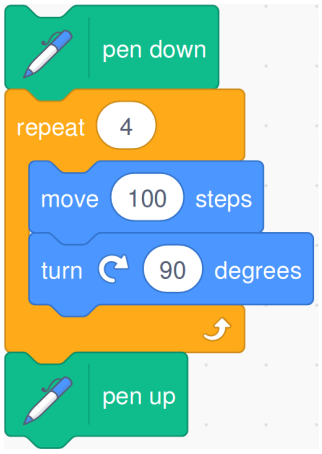
Oppgave 4 *Utforsk ulike typer firkanter*

For å tegne en firkant, eller mer presist et kvadrat, i Python ved hjelp av skilpaddegrafikk kan vi skrive inn følgende kode:

```
from turtle import *

for _ in range(4):
    forward(100)
    right(90)
```

og for å tegne en firkant i Scratch kan vi bruke blokker, slik:



I de følgende oppgavene kan du selv velge mellom Python og Scratch.

a) Legg koden over inn i en funksjon / blokk `tegn_kvadrat`, med `sidelengde` som argument. Prøv å kalle funksjonen med ulike tall som sidelengde.

b) Skriv en funksjon / blokk `tegn_rektangel` som tar inn to argumenter, `sidelengde_1` og `sidelengde_2`, som representerer lengden på hver av de parallelle sidene.

Ting å tenke på: Hva må du gi som argumenter for å få samme figur som blir tegnet av `tegn_kvadrat`?

c) For å tegne et parallellogram så trenger vi tre størrelser: de to sidelengdene og vinkelmodifikasjonen. Når vi tegner parallellogrammet, gjør vi det samme som når vi tegner et rektangel, bortsett fra at du ikke har 90° mellom linjene. Istedenfor har vi 90° + vinkelmodifikasjon mellom den første og andre linjen, 90° − vinkelmodifikasjon mellom den andre og tredje linjen, 90° + vinkelmodifikasjon mellom den tredje og fjerde linjen og 90° − vinkelmodifikasjon mellom den fjerde og første linjen. Skriv en funksjon / blokk `tegn_parallellogram` som tar inn tre argumenter, `sidelengde_1`, `sidelengde_2` og `vinkelmodifikasjon` og som tegner et parallellogram basert på disse størrelsene (husk at du ikke må å bruke en løkke her hvis du synes det er vanskelig).

Ting å tenke på: Hva må du gi som argumenter for å få samme figur som blir tegnet av `tegn_kvadrat` og `tegn_rektangel`?

d) Når vi programmerer vil vi gjerne unngå å *gjenta* kode, men samtidig er det nyttig å ha spesialiserte funksjoner. Kan du skrive om funksjonene /blokkene over slik at både `tegn_kvadrat`, `tegn_rektangel` kaller på `tegn_parallellogram`, istedenfor å utføre tegningen selv?

Ting å tenke på: Skal hver funksjon / blokk kalle på `tegn_figur`, eller skal de kalle på hverandre?

Oppgave 5 *Koordinatsystemer*

Oppgaven utforsker enkle koordinatsystemer i Python.

a) Bruk kommandoene `xlim` og `ylim` i `matplotlib.pyplot`-pakken til å lage et koordinatsystem fra −10 til 10. (Hint: `xlim` og `ylim` kan kalles på følgende vis `xlim((fra, til))`, der `fra` og `til` er verdiene aksen går fra og til. Ikke glem `show()` tilslutt.

b) Sett navn på aksene ved kommandoen `xlabel("navn")`.

c) Plasser et punkt i origo.

d) Lag et program som velger en tilfeldig x- og y-kooordinat som er et heltall fra −10 og 10, plotter det på skjermen, og deretter ber brukeren

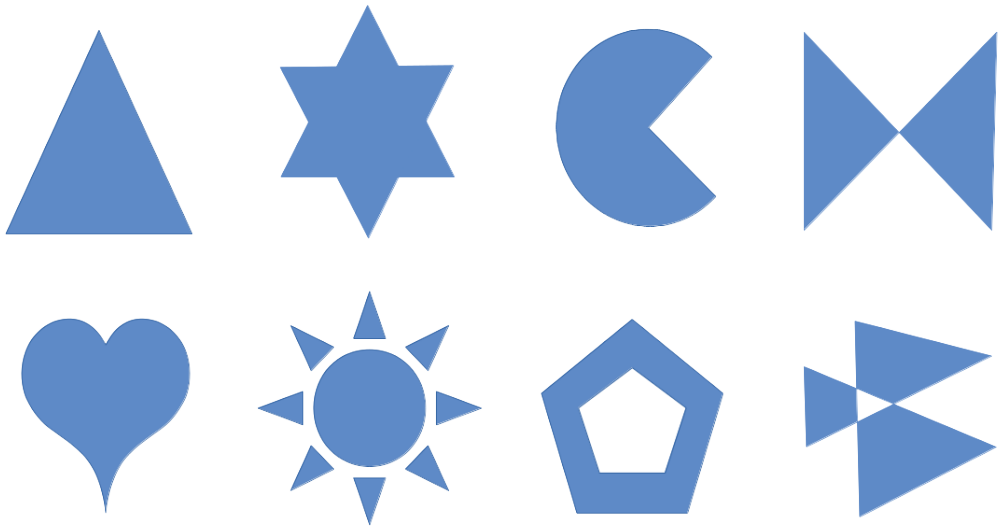
gjette hvor punktet er. (Hint: her kan det være en fordel å ha brukt kommandoen `grid()` for å få et rutenett.)

e) Modifiser nå programmet ditt fra i sted slik at programmet i stede for spør om hvilken kvadrant punktet ligger i.

f) En viktig ide i programmering er å kunne håndtere alle tilfeller som kan skje. Hva skjer med programmet ditt i forrige deloppgave dersom punktet havner i $(0, 0)$? Modifiser programmet ditt til å ta hensyn til origo.

Sammensatte oppgaver

Oppgave 6 *Areal av en polygon*



Ved hjelp av *Monte Carlo*-metoden kan man beregne π ved hjelp av sannsynlighet, der man velger et tilfeldig tall og finner ut om det er innenfor eller utenfor sirkelen. Ved samme metode kan man beregne arealet av sirkelen – det var det vi brukte for å utlede π. Og dette kan brukes for å finne arealet av andre figurer også – du må bare vite sikkert hva som er *inni* og *utenfor* figuren.

Denne oppgaven er en såkalt «frakoblet aktivitet», dvs. det er meningen å løse den uten å bruke programmering. Det er selvfølgelig også mulig å programmere dette, men det krever en relativ høy forståelse både av programmering og geometri (og det kan uansett ta litt tid).

a) En polygon, eller en mangekant, er figurer slik som trekanter, firkanter, femkanter, ... og så videre. Men hva ligger i «og så videre»? Hva er en polygon?

Matematikerne har faktisk aldri blitt helt enige med seg selv hvordan dette skal defineres – kan det f.eks. defineres med buer? Kan det ha «hull»? Kan linjene som definerer polygonet krysse hverandre? Se på figuren over. Hvilke av disse ville du sagt er en polygon? Kan du finne frem til en definisjon? Hvorfor er det viktig å definere dette nøyaktig når man skal programmere?

b) Tegn opp noen polygoner, etter din definisjon. Velg et punkt P utenfor polygonen, og et annet punkt som enten kan være inni eller utenfor figuren. Tegn en linje mellom disse to og tell hvor mange ganger du krysser en linje som definerer omrisset av polynomen. Gjenta med flere punkter. Kan du ved hjelp av dette tallet avgjøre om punktet ligger innefor eller utenfor?

c) Skisser en algoritme som ved hjelp av å velge tilfeldige punkter og tegne linjer til et punkt utenfor beregner arealet til en polygon.