Bonusoppgaver

I denne oppgavesamlingen finner du et knippe med bonusoppgaver for dag 2 dersom du vil ha litt mer å bryne deg på. God koding!

Funksjoner

Oppgave 1 *Beste pizzapris*

Hallgeir har lyst til å kjøpe pizza og sjekker prisen på nettet. Der står det at pizzaen koster 200 kroner for stor pizza med diameter 40 cm og 110 for liten pizza med diameter 27 cm. Hallgeir har lyst til å velge den pizzaen som er billigst per kvadratcentimeter med pizza og han vet at arealet til en sirkel er gitt ved:

$$\text{Areal} = \pi \times \text{radius}^2$$

For enkelhetsskyld så sier vi at pi er 3.14 istedenfor å importere den fra noe bibliotek.

a) Opprett en variabel, `diameter` som skal ha verdien 40

b) For å bruke formelen for arealet av en sirkel trenger vi radiusen. Vi vet at diameteren til en sirkel er det dobbelte av radiusen. Opprett ennå en variabel, `radius`, som skal være halvparten av diameteren.

c) Regn ut arealet av sirkelen, lagre resultatet i en variabel, `areal` og skriv ut hvor mange kvadratcm med pizza den store pizzaen inneholder.

d) Regn ut hvor mye pizzaen koster i kroner per kvadratcentimeter med pizza. Skriv svaret ut til skjermen.

e) Opprett et funksjon `kvadratcentimeter_pris(diameter, pris)` som tar inn diameteren til en pizza og radiusen og returnerer prisen per kvadratcentimeter.

f) Bruk funksjonen du lagde i oppgaven over til å regne ut kvadratcentimeterprisen til den lille pizzaen.

g) Hvilken pizza skal Hallgeir velge dersom han vil ha pizzaen som er billigst per kvadratcentimeter?

Skilpaddeprogrammering

Oppgave 2 *Blomstereng*

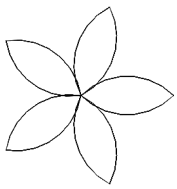
I denne oppgaven skal vi utforske hvordan vi kan tegne en blomstereng ved hjelp av funksjoner.

a) Koden under tegner et blomsterblad i `turtle`:

```
radius = 100
right(45)
circle(radius, 90)
left(90)
circle(radius, 90)
left(135)
```

Bruk koden til å lage en funksjon `tegn_blad(radius)` som tar inn en `radius` og tegner et tilhørende kronblad

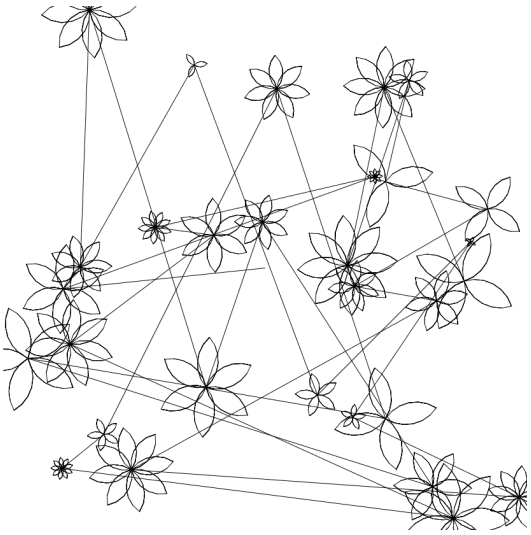
b) Bruk funksjonen du lagde i a) sammen med en `for`-løkke til å gjenskape denne blomsten:



c) Bruk koden du lagde i b) for å lage en funksjon, `tegn_blomst(radius, antall_blader)` som tar inn størrelse på bladene (radius for sirkelsegmentene) og antall blader og tegner en tilhørende blomst. Prøv ut funksjonen med forskjellige parametere.

d) Modifiser koden du skrev i forrige oppgave slik at du bruker `randint` for å tegne en blomst med tilfeldig radius mellom 3 og 30 og tilfeldig antall blader mellom 3 og 9.

e) Bruk en løkke sammen med `randint`, `goto`, `right` og `tegn_blomst` funksjonen din til å tegne *30* blomster med *tilfeldig radius*, *rotasjoner*, *antall blader* og *posisjoner*. Prøv deg frem til du finner gode intervaller å trekke de tilfeldige tallene fra. Under har du et eksempel på hvordan blomsterengen kan bli:



(OBS: Det kan ta lang tid å tegne 30 blomster, men du kan bruke `speed('fastest')` på starten av koden for at skilpadden skal bevege seg raskere)

f) La oss gjøre blomsterengen litt penere: Oppdater `tegn_blomst` til å bruke `begin_fill` før du tegner et blad og `end_fill` etter. Bruk `penup` i starten av programmet for å fjerne streken.

g) **Bonusoppgave:** farger i blomsterbeddet! Lag en liste med noen farger du liker (f.eks: `['PaleVioletRed', 'MediumVioletRed', 'Orchid', 'RosyBrown', 'DarkSlateBlue', 'Chocolate']`) Bruk `choice` til å velge en tilfeldig farge fra lista for hver blomst og `fillcolor` til å oppdatere fyllfargen til fargen du har valgt. Under er et eksempel til inspirasjon:

Oppgave 3 *Plotte med Turtle*

For å lage plot trenger vi funksjon som skalerer et tall fra en måleverdi til et koordinatsystem.

a) Lag en funksjon `omskaler(tall, a1, b1, a2, b2)` Denne funksjonen skal transformere et tall på denne måten:

$$\texttt{omskaler}(x, a_1, b_1, a_2, b_2) = a_2 + x\frac{b_2 - a_2}{b_1 - a_1} + a_2 - a_1\frac{b_2 - a_2}{b_1 - a_1} \tag{1}$$

Denne funksjonen tar inn et tall mellom `a1` og `b1` og gir ut et tall mellom `a2` og `b2`. Dersom `tall` er lik `a1` vil omskaleringen være lik `a2`. Tilsvarende, dersom `tall` er lik `b1`, vil omskaleringen være lik `b2`. Hvis `tall` ligger midt mellom `a1` og `a2`, så vil resultatet av omskaleringen ligge midt mellom `a1` og `b1`.

b) Nå skal vi teste at omskaleringsfunksjonen vår virker. Sjekk at

$$\texttt{omskaler}(0, 0, 1, 10, 20) = 10 \tag{2}$$
$$\texttt{omskaler}(1, 0, 1, 10, 20) = 20 \tag{3}$$
$$\texttt{omskaler}(0.5, 0, 1, 10, 20) = 15 \tag{4}$$
$$\texttt{omskaler}(2, 0, 10, 100, 200) = 120 \tag{5}$$

Tabellen under viser målinger av internetthastigheten i en leilighet i Grünerløkka på forskjellige tidspunkt i løpet av en dag

Klokkeslett [time]	Nedlastingshastighet [Mbps]
7	96.56
10	95.75
13	78.30
15	96.95
19	97.79

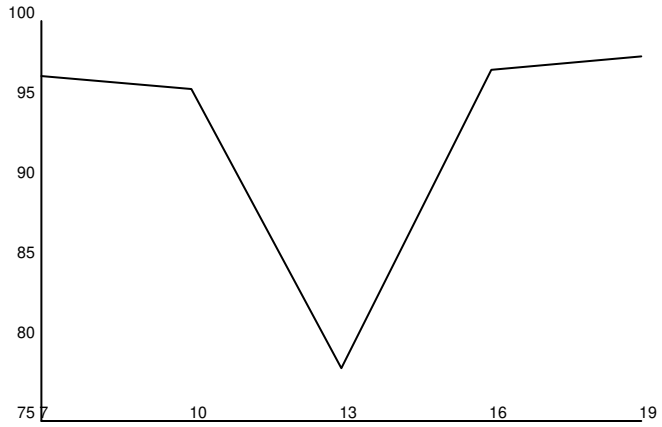
c) Lag en liste `hastigheter` som inneholder måleverdiene for internetthastighet og en liste `timer` som inneholder timene hastighetene ble målt

d) Opprett en skilpadde med `turtle.Turtle()`

e) Bruk en løkke til å gå igjennom hastighetsmålingene og timene. Inne i løkka skal du omskalere hver hastighetsmåling til en y-verdi mellom `0` og `200` og hver timeverdi til en x-verdi mellom `-300` og `300`.

f) Bruk `goto` inne i løkka til å gå til en og en x og y verdi med skilpadda di og tegne en strek (hint: Det kan være lurt å bruke `penup` og `pendown` for å flytte skilpadda til første målepunktet uten å tegne en strek).

g) **Bonusoppgave:** Prøv å gjenskape figuren under ved å tegne opp koordinataksene og bruke `write` funksjonen til å tegne opp tallverdier på aksene.


100
95
90
85
80
75
7
10
13
16
19