

Oppgaver

I denne seksjonen finner du oppgaver som hører til dag 2 av Kodeskolens kræsjkurs i programmering.

Tema for andre dag er funksjoner, rekursjon og plotting. Dersom du står fast er det bare å spørre. Oppgaver markert som bonusoppgaver er litt mer utfordrende og du velger selv om du har lyst til å prøve deg på dem. God koding!

Plot

Oppgave 1 *Plotte populasjonstall*

Befolkningstall i Norge for årene 2010-2020 er gitt i tabellen under (tall fra Statistisk Sentralbyrå):

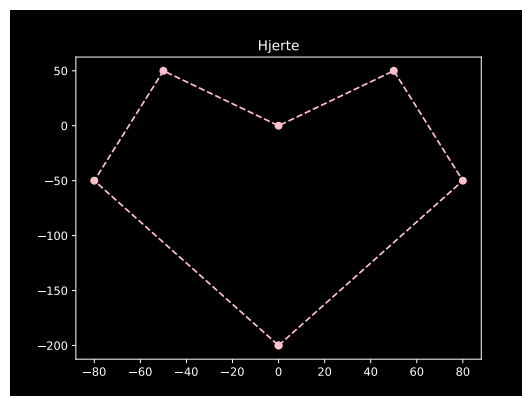
År	2015	2016	2017	2018	2019	2020
Befolkning.	5 165 802	5 213 985	5 258 317	5 295 619	5 328 212	5 367 580

- Lag en liste befolkningstall som inneholder befolkningstall og en liste tidspunkt som inneholder året befolkningstallet ble målt
- Bruk `plot` funksjonen i `matplotlib.pyplot` til å lage et plot med år på x-aksen og befolkningstall på y-aksen.
- Bruk `xlabel` og `ylabel` funksjonene i `matplotlib.pyplot` til å legge merkelapper på x- og y-aksen.
- Importer `savefig` funksjonen fra `matplotlib.pyplot` og bruk den til å lagre plottet som en png fil

Oppgave 2 *Hjerteplott*

x	0	80	50	0	-50	-80	0
y	-200	-50	50	0	50	-50	-200

- a) Lag to lister `x_verdier` og `y_verdier` som inneholder tallene i tabellen over
- b) Bruk `plt.style.use` for å skifte stilen til matplotlib til `'dark_background'`
- c) Bruk `plot` funksjonen i `matplotlib.pyplot` for å plote x-verdiene og y-verdiene mot hverandre og bruk `title` for å gjenskape figuren under (fargen er `'pink'`).



- d) Bruk `savefig` funksjonen til å lagre plottet som en pdf fil

Funksjoner

Oppgave 3 *Matematiske funksjoner*

- a) Lag en funksjon `kvadrat(x)` som tar inn et tall x og returnerer kvadratet x^2 .
- b) Gjenta oppgaven over, men med funksjonen `kubikk(x)` som returnerer x^3 .
- c) Lag en funksjon `f(x)`, som returner $x^2 + 3x - 1$

Oppgave 4 *Tegn mangekanter*

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

```
1 from turtle import *
2
3 for _ in range(3):
4     forward(100)
5     right(120)
```

- a) Legg koden inn i en funksjon. La sidelengden (antall steg) gis som et parameter. Prøv deg frem med å tegne trekanter i ulik størrelse. Kall funksjonen 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 som tegner en generell mangekant, `tegn_mangekant`, som tar et argument n og et argument `sidelengde`. Gjør den det samme som `tegn_trekant`, `tegn_firkant` og `tegn_femkant` for $n = 3, 4, 5$?
- e) Modifiser funksjonen så den tar inn omkrets istedet for sidelengde og så regner ut sidelengden ved å dele omkrets på antall sider

Oppgave 5 *Blomstereng*

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

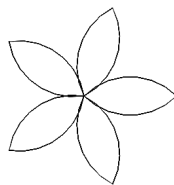
- a) Koden under tegner et blomsterblad i turtle:

```
1 radius = 100
2 right(45)
```

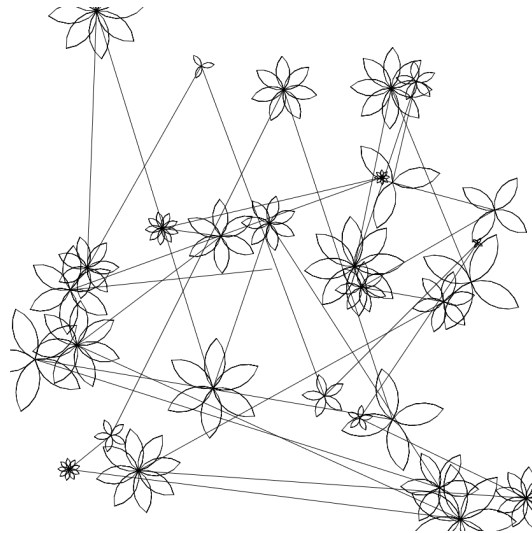
```
3 circle(radius, 90)
4 left(90)
5 circle(radius, 90)
6 left(135)
```

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 sirkel-segmentene) 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_bloerst` 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:



Rekursjon og fraktaler

Oppgave 6 *Fakultet med rekursjon*

Rekursive funksjoner er funksjoner som kaller på seg selv. I matematikken finnes det mange eksempler på rekursive sammenhenger. I denne oppgaven skal vi se på hvordan vi kan beskrive fakultet ved hjelp av rekursjon. Fakultet er definert som

$$\text{tall!} = \text{fakultet}(\text{tall}) = \text{tall} \cdot (\text{tall} - 1) \cdot (\text{tall} - 2) \cdot \dots \cdot 1$$

Dette kan vi beskrive rekursivt:

$$\text{tall!} = \text{fakultet}(\text{tall}) = \text{tall} \cdot \text{fakultet}(\text{tall} - 1)$$

- a) Lag en funksjon `fakultet(tall)`. Dersom `tall` er `0` skal funksjonen bare returnere `1`. Dersom `tall` er større enn `0` skal funksjonen “kalle på seg selv” og returnere `tall*fakultet(tall-1)`
- b) Bruk funksjonen til å regne ut fakultet av 10, 1, og 13

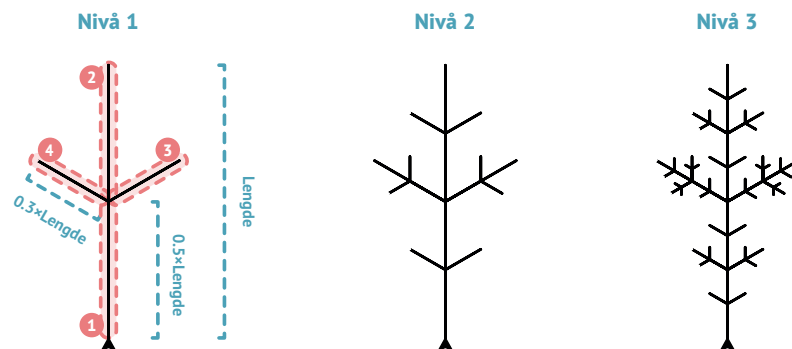
Oppgave 7 *Fraktale snøfnugg*

I denne oppgaven skal vi bruke rekursjon og fraktaler til å lage snøfnugg. Hvis

du studerer bilde av et snøfnugg ser du at de består av seks “krystall-armer”. Hver av armene har forgreininger og hver av forgreiningene ser også ut som en arm med flere forgreininger osv. Rekursjon egner seg altså godt for å tegne snøfnugg!



La oss begynne med å se på en og en arm av gangen og så kan vi sette det sammen til et snøfnugg til slutt. For armene ønsker vi et slags fraktal-tre hvor hver arm består av flere armer. Jo flere nivåer vi tegner, jo mer komplisert snøfnugg får vi (se figuren under)



- a) La oss begynne på en `tegn_arm(skilpadde, lengde, nivå)` funksjon. Den skal ta inn en skilpadde som brukes til å tegne, lengden på armen og hvilket nivå som tegnes.

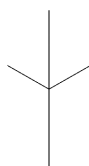
- b) Det første vi kan gjøre inne i `tegn_arm` funksjonen er å sjekke om vi er på det laveste nivået med `if nivå < 1`. Da skal armen kun være en rett strek. Bruk `forward(lengde)` etterfulgt av `backward(lengde)` for å først tegne en rett strek og så flytte skilpadden tilbake til start. Deretter returnerer vi med `return`.

Hvis vi ikke er på laveste nivå skal armen tegnes i fire deler (se figuren over):

- c) Del en tegner vi ved å kalle på `tegn_arm` funksjonen med samme skilpadde, `0.5*lengde` og `nivå-1`.
- d) For å tegne del to må vi først flytte opp til enden av del 1 med `forward(lengde)`. Så tegner vi del to ved å kalle på `tegn_arm` med samme skilpadde `0.5*lengde` og `nivå-1`
- e) For å tegne del tre roterer vi først `60` grader mot høyre og så tegner vi en arm med samme skilpadde, `0.3*lengde` og `nivå-1`
- f) For å tegne del fire må vi rotere `-60` grader mot høyre (altså `60` grader mot venstre) for å peke oppover igjen og så `-60` grader mot høyre for å peke mot venstre og så tegner vi en arm likt som i del tre
- g) Til slutt i funksjonen flytter vi skilpadden tilbake til start med å først rotere `60` grader mot høyre skilpadden den peker oppover igjen og så gå bakover `lengde*0.5` steg.
- h) Nå har vi en `tegn_arm` funksjon. La oss teste funksjonen ved å kalle på den slik:

```
1 import turtle
2 penn = turtle.Turtle()
3 penn.setheading(90) # Så armen peker oppover
4
5 tegn_arm(penn, 100, 1)
6 turtle.done()
7 turtle.bye() # Nødvendig hvis vi bruker Spyder
```

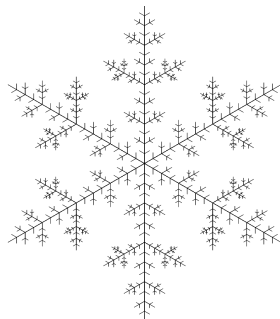
Da skal du få følgende tegning:



i) Endre startsnivå fra 1 til 4, da skal du få følgende tegning:



j) Bruk en **for**-løkke til å tegne 6 armer med 60 grader mellom for å tegne et snøfnugg som ser slik ut:



Prøv deg frem med forskjellig antall rekursjonsnivåer og lengde på armene!