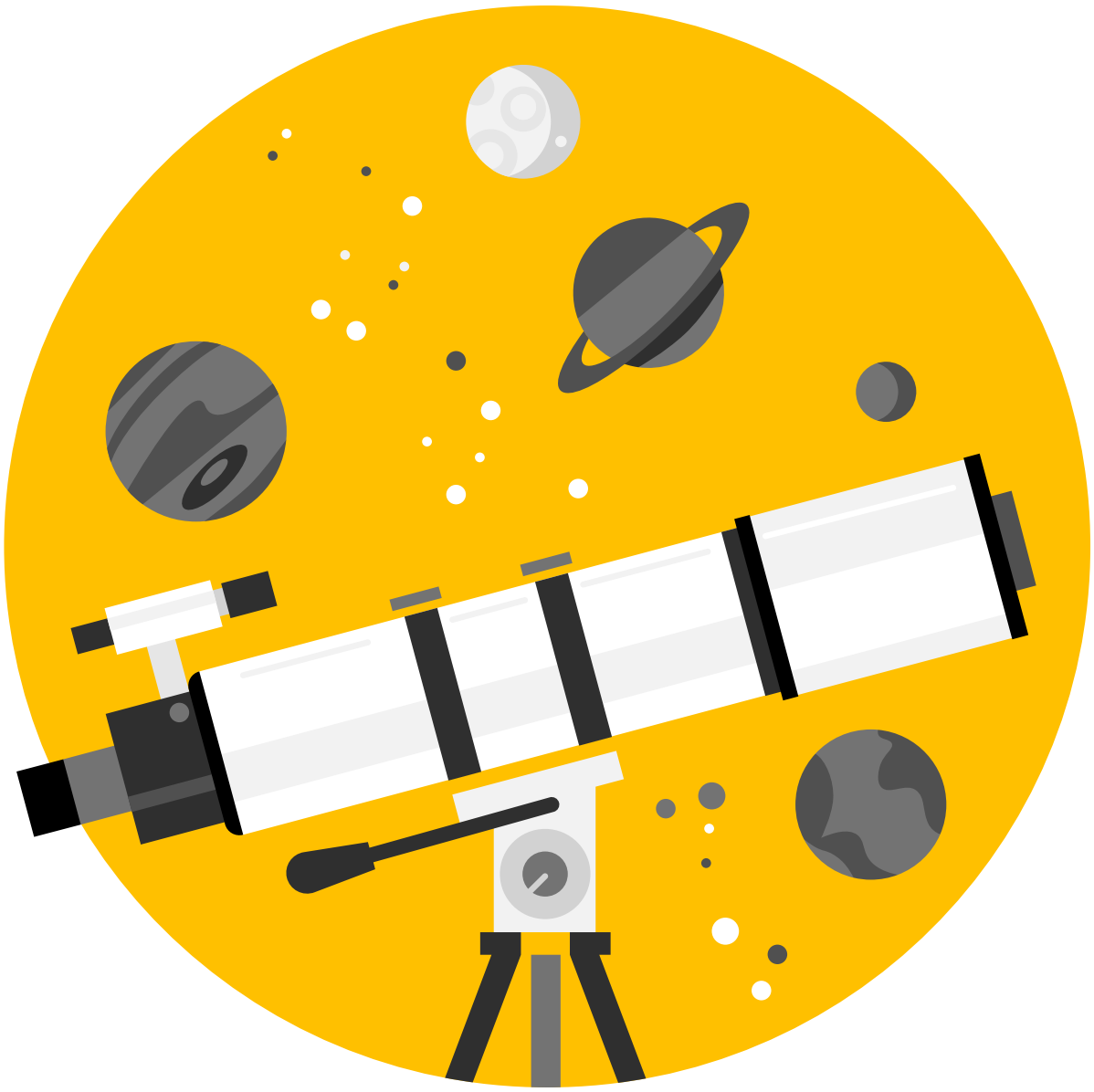
Lag solsystemet med Python

Programmering sammen – steg for steg

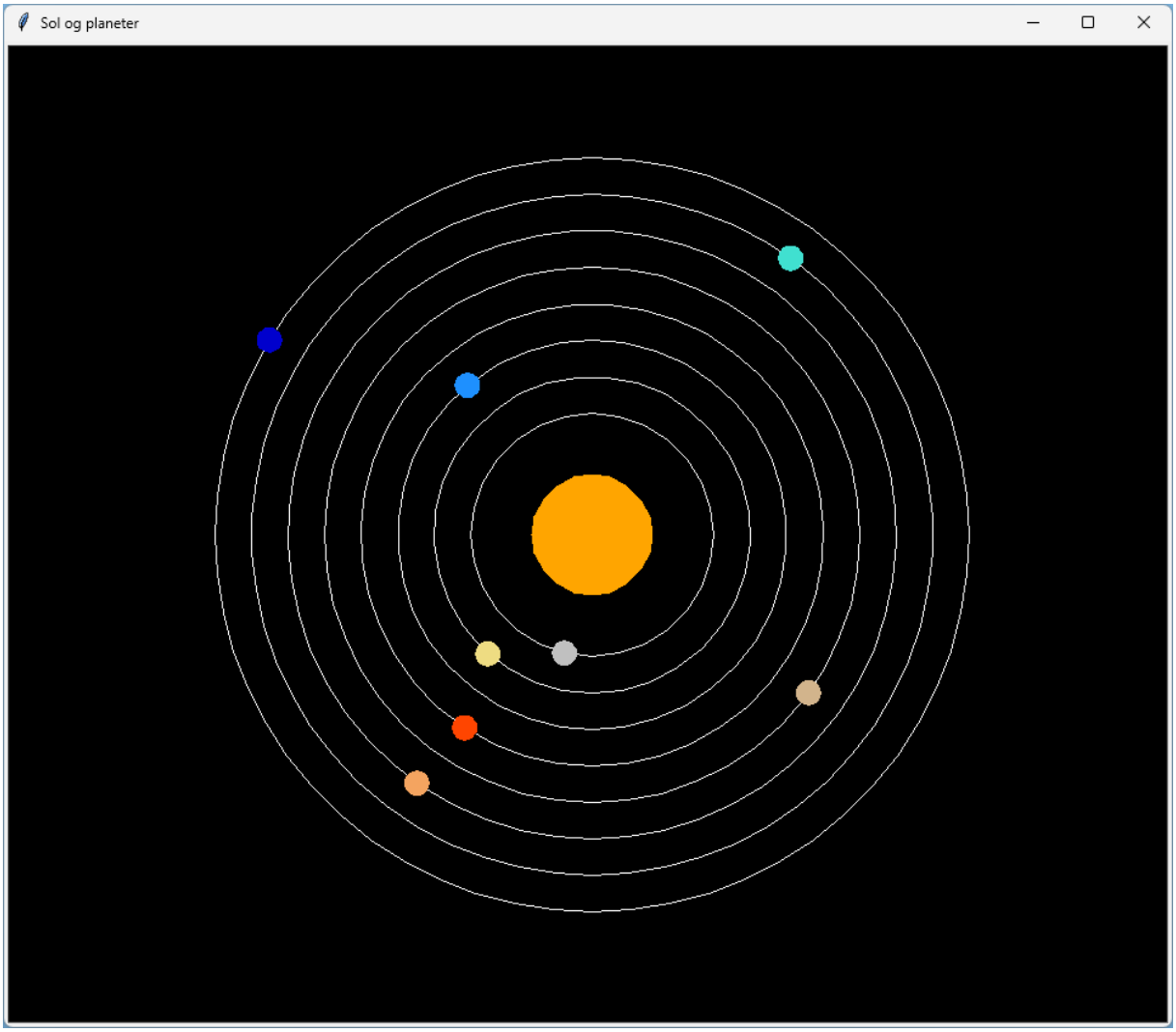


– en utforskende læringsreise i verdensrommet

LÆRERVEILEDNING

Kort om aktiviteten

Denne aktiviteten handler om å tegne solsystemet vårt gjennom en metodikk som kalles livekoding. Aktiviteten er bygd opp slik at du som lærer bygger opp Python-kode fra bunnen av, steg for steg og ender opp med en enkel grafikk av solsystemet.



Figur 1:Solsystemet med solen og planetenes baner. Kilde: Andøya Space

Innhold

Teoridel

Turtle funksjoner

En rask forklaring på noen av de innebygde funksjonene fra turtle, et grafisk bibliotek for Python som lar oss tegne og illustrere med kode.

Funksjon / Objekt	**Forenklet forklaring**
`Screen()`	Lager et «lerret» – et område der vi kan tegne.
`Turtle()`	Lager en «penn» (eller figur) som kan tegne på skjermen. Gjerne ulike penner i bruk.
`screensize(width, height, color)`	Setter størrelsen og bakgrunnsfargen på tegneområdet.
`turtle.done()`	Holder vinduet åpent til du lukker det selv.
`speed(n)`	Bestemmer hvor raskt «pennen» tegner (0 = raskest).
`hideturtle()`	Skjuler selve figuren, men den kan fortsatt tegne.
`penup()`	Løfter pennen slik at du ikke tegner samtidig som du flytter den til et annet sted.
`pendown()`	Setter pennen ned for å begynne å tegne.
`begin_fill()`	Starter fylling av en form med en farge.
`end_fill()`	Avslutter fylling av formen.
`goto(x, y)`	Flytter pennen til en bestemt posisjon i tegneområdet.

Posisjoner i tegneområdet

Posisjonen til pennen i et Turtle-tegneområde fungerer som et koordinatsystem, omtrent som i et matematisk diagram.

(0, 0) er midten av skjermen – akkurat som origo i et koordinatsystem.

- x-aksen går horisontalt:
    - Positive verdier (+) til høyre og negative verdier (–) til venstre.

- y-aksen går vertikalt:
    - Positive verdier (+) oppover og negative verdier (–) nedover.

Python er pirkete

Python reagerer på bruk av mellomrom og tabulator som innrykkstegn. Det er viktig å være konsekvent, for ellers vil Python klage. Det samme gjelder bruk av hermetegn. Du kan velge å bruke doble eller enkle hermetegn, men ikke blande de.

Hexadesimale fargekoder

En hexadesimal fargekode er en måte å skrive farger på med tall og bokstaver, for eksempel #FF4500. Den starter med # og har 6 tegn: De to første (FF) er rød, de to neste (45) er grønn og de siste to (00) er blå.

Verdiene går fra 00 (ingen farge) til FF (maks farge). Eksempel: #FF0000 er rød, #00FF00 er grønn, #0000FF er blå. Vis elevene at "blue" og "#0000FF" gir samme farge, men hex-koder gir mulighet til å lage kule farger som ikke har navn. I Kilder-kapittelen er det lenker til flere steder å finne slike fargekoder.

Vi går i gang med kodingen

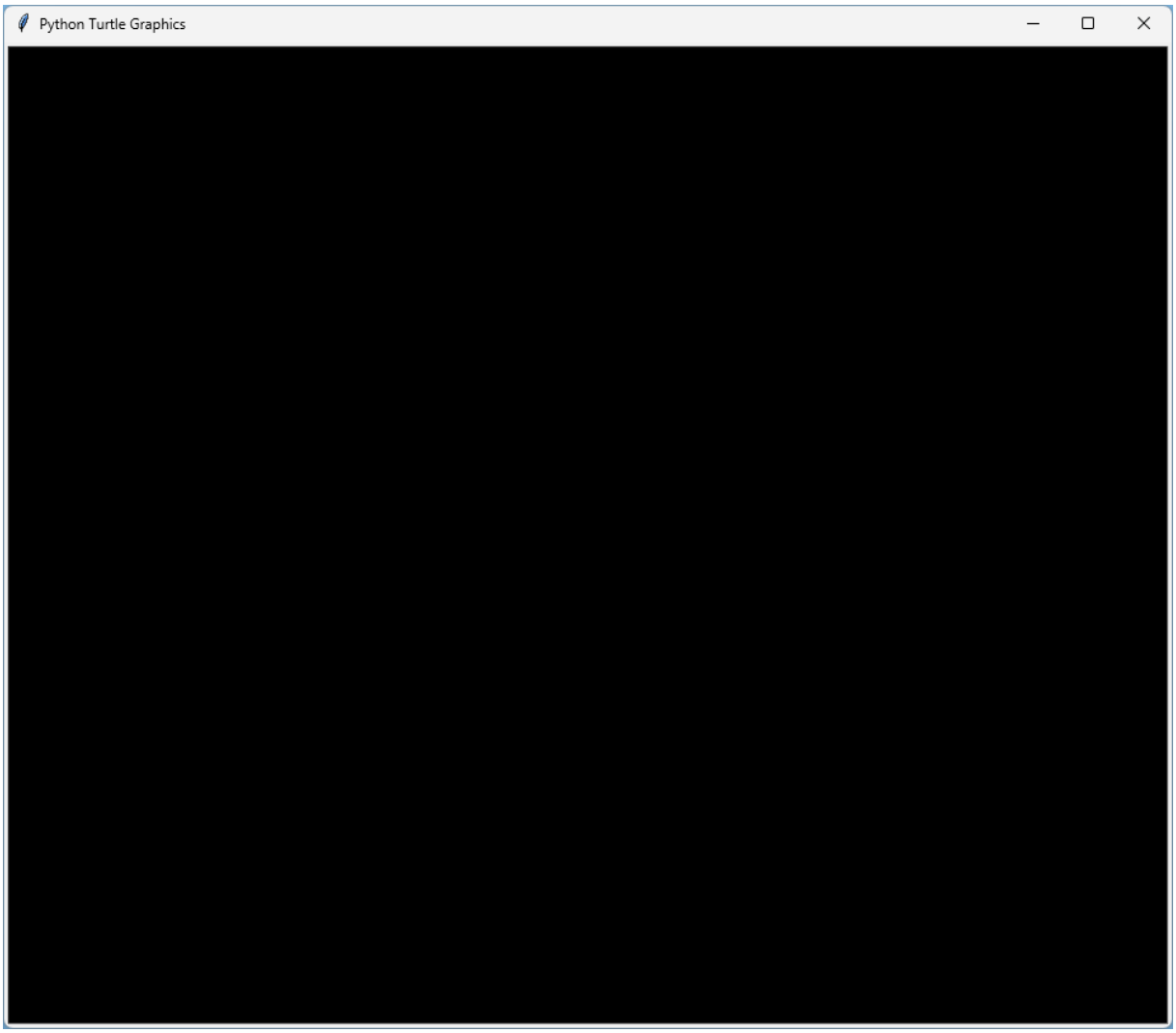
Steg 1 – Lag et område å tegne i

Sett opp vinduet (skjerm) i passende størrelse. Her har jeg valgt et kvadratisk vindu på 800 x 800 piksler. Vi starter med å importere turtle biblioteket og setter opp lerretet. Siste del av programmet skal være turtle.done() som passer på at programmet holder seg kjørende og åpent til vi lukker det.

```
import turtle

# Sett opp vinduet (skjerm) i passende størrelse
skjerm = turtle.Screen()
skjerm.screensize(800, 800, "black")

# Avslutter riktig
turtle.done()
```



Figur 2: Tom visning før tegning av solsystemet. Kilde: Andøya Space

Steg 2 – Tegn solen

Nå skal vi tegne inn solen midt i det vinduet vi har opprettet. Denne koden setter opp et nytt tegneobjekt *sol* og hvilken farge det skal tegnes med. Posisjonen blir satt og en sirkel fylles med gulfargen i heksadesimal kode. Plasser denne koden rett før turtle.done().

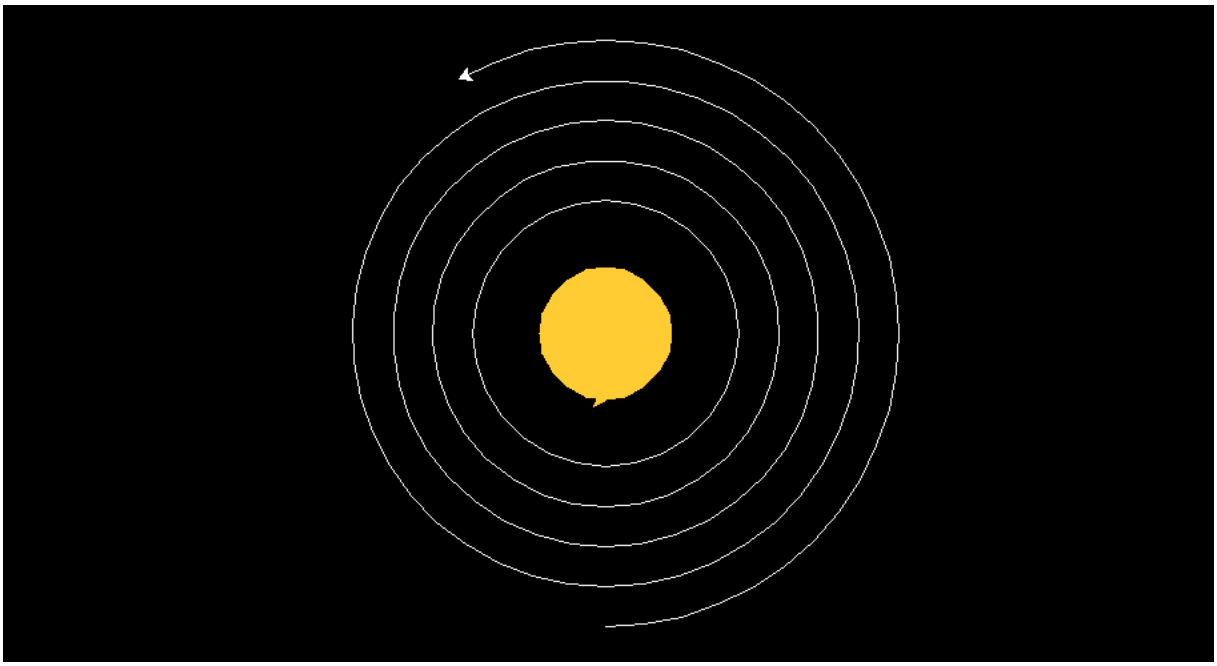
```
# Tegn solen
sol = turtle.Turtle()
sol.color("#FFCC33")
sol.penup()
sol.goto(0, -50)
sol.begin_fill()
sol.circle(50) # Solens radius
sol.end_fill()
```

Figur 3: Solen som sentrum i solsystemet. Kilde: Andøya Space

Steg 3 – Tegne planetbanene

Vi lager enda et nytt Turtle objekt, denne gangen *bane*. Vi starter med å sette fargen og radiusen til den første banen. Solen hadde en radius på 50 piksler, så vi velger en litt større på 100. Så går vi litt algoritmisk til verks og setter opp en løkke som tegner en sirkel, først på den valgte radiusen, og deretter øker med 30 piksler, til den har tegnet åtte planetbaner.

```
# Tegn planetbanene
bane = turtle.Turtle()
bane.color("white")
planetbane = 100
for i in range(8):
    bane.penup()
    bane.goto(0, - planetbane)
    bane.pendown()
    bane.circle(planetbane)
    planetbane += 30 # Øker til større bane
```

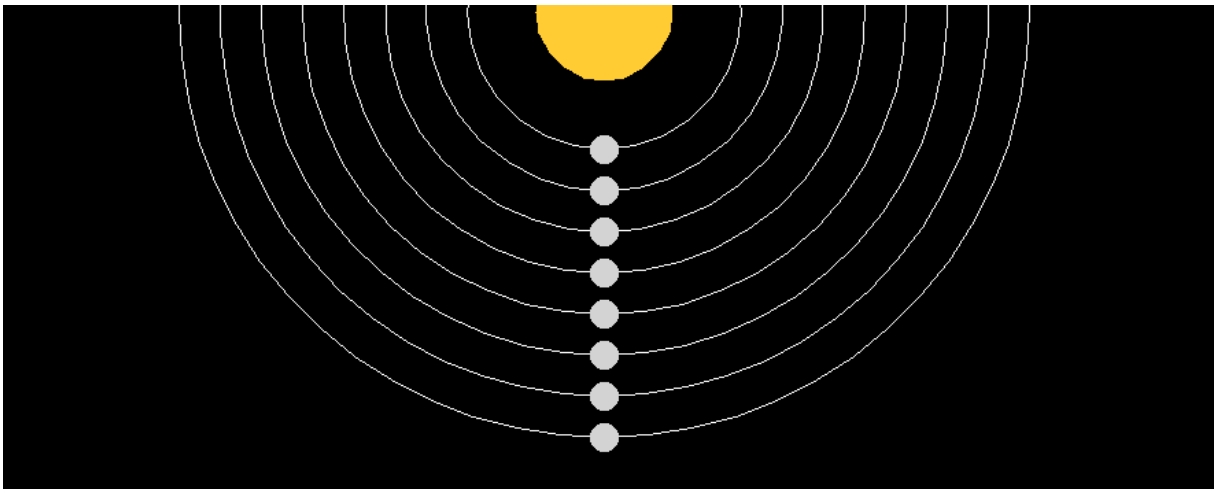


Figur 4: Solen med tegning av planetbaner. Kilde: Andøya Space

Steg 4 – Plassere ut planetene

Slik som planetbanene økte i radius, skal vi nå sette pennen i økende avstand fra solen. Denne gangen tegner vi ikke en sirkel, men bruker en av de innebygde formene til pennen og *stempler* en sirkel på hver bane.

```
# Tegn planetene
planetbane = 100
for i in range(8):
    planet = turtle.Turtle()
    planet.shape("circle")
    planet.color("lightgrey")
    planet.penup()
    planet.goto(0, -planetbane)
    planetbane += 30
```



Figur 5: Solen og åtte planeter i grått. Kilde: Andøya Space

Nå kunne vi vært fornøyde med koden slik den er samlet:

```
import turtle

# Sett opp vindu en passende størrelse
skjerm = turtle.Screen()
skjerm.screensize(800, 800, "black")

# Tegne solen
sol = turtle.Turtle()
sol.color("#FFCC33")
sol.penup()
sol.goto(0, -50)
sol.begin_fill()
sol.circle(50) # Solens radius
sol.end_fill()


# Tegn planetbanene
bane = turtle.Turtle()
bane.color("white")
planetbane = 100
for i in range(8):
    bane.penup()
    bane.goto(0, -planetbane)
    bane.pendown()
    bane.circle(planetbane) # Tegner en sirkel for hver bane
    planetbane += 30 # Øker til større bane

# Tegn planetene
planetbane = 100
for i in range(8):
    planet = turtle.Turtle()
```

```
    planet.shape("circle")
    planet.color("lightgrey")
    planet.penup()
    planet.goto(0, -planetbane)
    planetbane += 30

# Avslutter riktig
turtle.done()
```

Steg 5 – Forbedringer i koden

Tegne raskere

Du har kanskje lagt merke til at det tar fryktelig lang tid å tegne alt. Det kan vi endre på ved å sette hastigheten til tegneobjektene sol og bane med `speed()` funksjonen. Setter vi parameteren til 0 tegner den veldig raskt. Legg inn linjene etter tegneobjektet settes opp.

```
# Tegne solen
sol = turtle.Turtle()
sol.speed(0)

...

# Tegn planetbanene
bane = turtle.Turtle()
bane.speed(0)
```

Stor forskjell eller hva?

Fjerne pennemarkøren

Når en turtle-penn har gjort seg ferdig, blir den værende igjen en markør, som er satt til å være en pil. Det er ingen grunn til at den skal vises, så den kan vi skjule med `hideturtle()` funksjonen. Legg inn etter `speed()`

```
# Tegne solen
sol.speed(0)
sol.hideturtle()

...

# Tegn planetbanene
bane.speed(0)
```

```
bane.hideturtle()
```

Navn på vinduet

Du har kanskje lagt merke til at vinduet det tegnes i har et litt kjedelig navn, *Python Turtle Graphics*, et navn vi helt fint kan endre ved å legge inn i vinduoppsettet, etter at vinduobjektet er satt opp:

```
skjerm = turtle.Screen()
skjerm.title("Sol og planeter")
```

Figur 6: Programvindu for illustrasjon av solsystemet. Kilde: Andøya Space

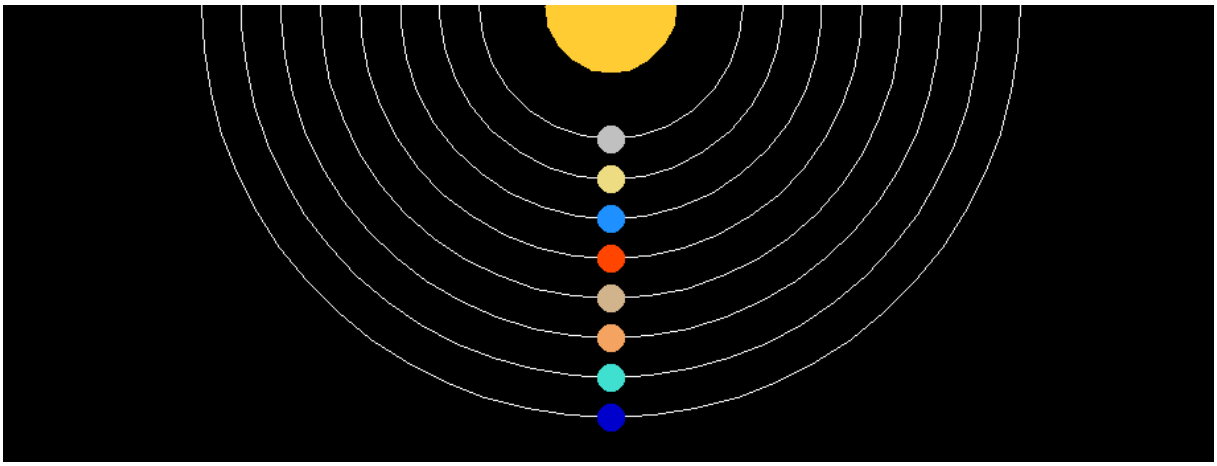
Fargerike planeter

Heldigvis er det ikke bare kjedelige grå planeter i solsystemet vårt, men hver planet har en litt gjenkjennbar farge som vi kan ta i bruk i programmet vårt. Når planetene tegnes inn fra nærmest solen og utover, kan vi gå samtidig gå gjennom en liste med farger som vi tar i bruk. Først lager vi en liste med farger og deretter endrer vi på to linjer i for-løkken.

Vi ønsker at løkken bare skal tegne inn de åtte planetene og tilsvarende åtte farger. Vi kan be for-løkken gå gjennom listen med farger som har åtte elementer og setter `planet.color(planetfarger[i])` til den gjeldende fargen `i`. Se endringene under.

```
# Tegn planetene
planetfarger = [
    "#C0C0C0", # Merkur – metallisk grå
    "#EEDC82", # Venus – sandgul/beige
    "#1E90FF", # Jorden – havblå
    "#FF4500", # Mars – rust-oransje
    "#D2B48C", # Jupiter – lys brun
    "#F4A460", # Saturn – gyllen brun
    "#40E0D0", # Uranus – turkis
    "#0000CD"  # Neptun – dyp blå
]

for i in range(len(planetfarger)):
    ...
    planet.color(planetfarger[i])
```



Figur 7: Solen og planeter med farger som skiller dem fra hverandre. Kilde: Andøya Space

Tilfeldig plassering av planetene

Vi kan åpne matematikkboken vår (`import math`) og regne ut x og y med cos og sin, gitt en tilfeldig vinkel (`import random`). Be programmet ta med kunnskap fra to nye biblioteker.

```
import turtle
import math
import random
```

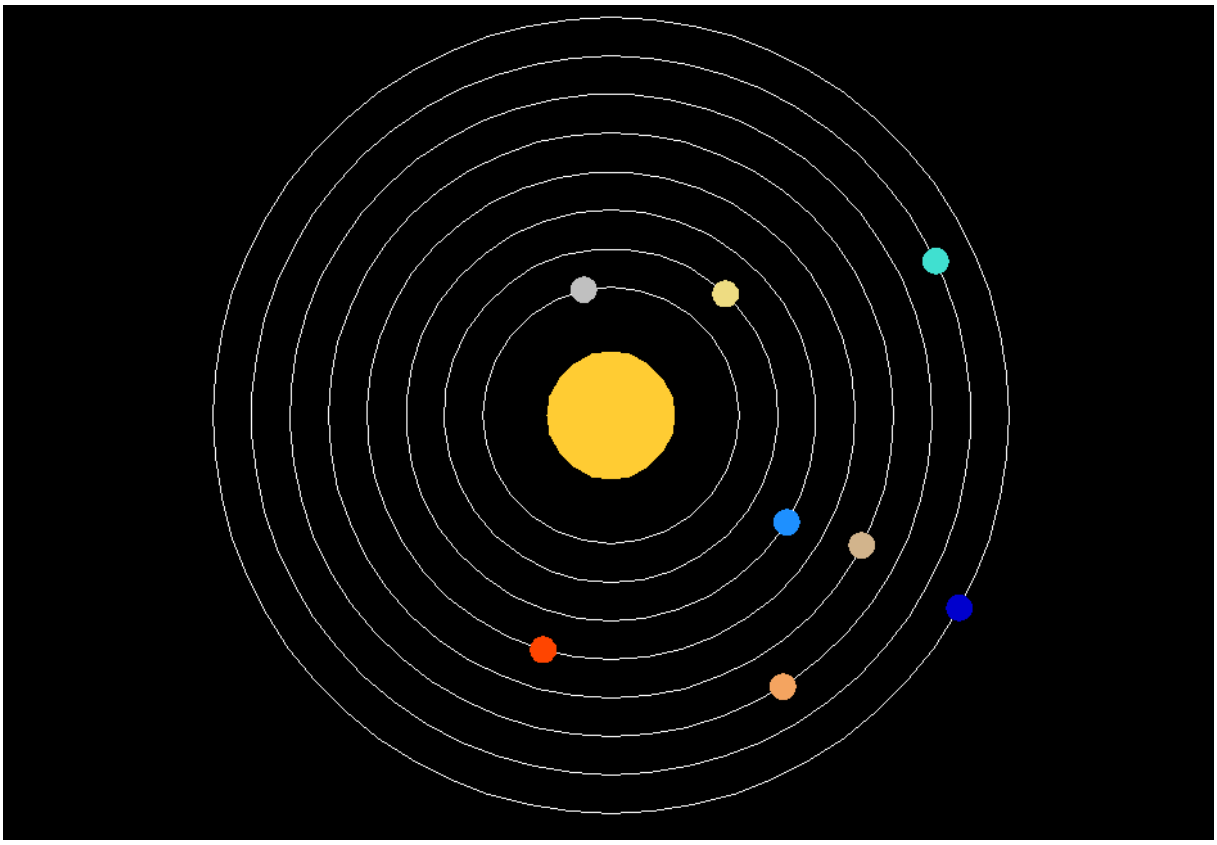
Endre plasseringen på planetbanene fra

```
planet.goto(0, -planetbane)
planetbane += 30
```

Til

```
vinkel = random.uniform(0, 360)
x = planetbane * math.cos(math.radians(vinkel))
y = planetbane * math.sin(math.radians(vinkel))
planet.goto(x, y)
planetbane += 30
```

Vi velger en tilfeldig vinkel, gjør den om til radianer, og bruker sinus og cosinus for å finne hvor planeten skal ligge på sirkelen: **x-posisjon** (sideveis) og en **y-posisjon** (opp/ned). Slik vil programmet ha regnet ut en x, y posisjon i vinduet gitt en avstand fra sentrum (planetbane). Dette handler faktisk om å kunne regne ut hvor en planet vil være til ethvert tidspunkt i banen rundt solen.



Figur 8: Ferdig illustrasjon av solsystemet med fargelagte planeter. Kilde: Andøya Space

Ulike solfarger

Hvis du vil, kan du i tillegg til å velge tilfeldig plassering av planetene, velge en tilfeldig farge på solen. Når vi plasserte planetene, valgte vi en tilfeldig verdi mellom 0 og 360 med `random.uniform()` funksjonen. Nå bruker du `random.choice()` funksjonen til å velge blant elementer i en gitt liste.

```
# Tegne solen
solfarger = ["#FFD700", "#FFCC33", "#FFA500"]
...
sol.color(random.choice(solfarger))
```

Lagre bildet som en fil

Du kan ta et skjermbilde av den ferdige grafikken, men programmet kan også lagre det ferdige produktet som en grafisk fil, såkalt EPS/Ghostscript. Noen operativsystemer kan ikke uten videre åpne de uten en ekstra programvare, men det er flere fine gratisprogrammer for det, som GIMP og Inkscape, se Kilder bakerst.

For å be programmet lagre til en fil, legg inn en linje mot slutten, rett før `turtle.done()`

```
# Lagre som EPS/PostScript
skjerm.getcanvas().postscript(file="solsystemet.eps")
```

Lærerveiledning

Selv om denne aktiviteten er ment som en livekodeaktivitet der en innbygger i Python-land (det er deg som lærer eller pedagog) veileder en gruppe elever i oppbyggingen av en kode stegvis fra blanke ark, så kan den også fungere som en stegvis veiledning for en elev å følge alene eller parvis.

Resten av lærerveiledningen tar for seg denne livekodemetoden.

Programmering og algoritmisk tenkning

Programmering og algoritmisk tenkning er synliggjort i fagene på ulikt vis i Læreplanverket for Kunnskapsløftet 2020. Elevene får en introduksjon til sentrale konsepter og grunnstrukturer i programmering i matematikkfaget, men de skal lære og anvende kompetansen også i andre fag. Det er eksplisitte kompetansemål i naturfag, kunst og håndverk og musikk. Programmering som verktøy kan også styrke andre fag, som for eksempel samfunnsfag og norsk.

Livekoding som didaktisk metode

Som en motsetning til en tradisjonell gjennomgang av et tema, der elevene passivt følger med på det du gjør, kan dere sammen skrive et dataprogram. Kode på direkten.

Forskning (se Kilder) sier at livekoding kan være en god metode for å vise elevene hvordan de ulike begrepene og programmering er i praksis. Det blir mer interaktivt, og du kan gå tilbake og endre på koden underveis. Du kan la elevene få delta i oppbyggingen av koden. De kan komme med forslag eller forsøke å forutsi hva som vil skje dersom koden kjøres på ulike tidspunkt. Skulle du som lærer gjøre en feil underveis, er det helt greit. Det vil vise elevene at det er greit om de også gjør feil. Elevene kan også inkluderes i feilsøkingen. Python vil dessuten fortelle hva som er feil slik at det kan rettes opp. Det tar tid å bli god til å programmere, men alle kan lære det.

Læremål

- Kjenne rekkefølgen til planetene i solsystemet.
- Forstå hva solsystemet består av (sol, planeter, baner).
- Beskrive hvordan planetene beveger seg i baner rundt solen.
- Lære grunnleggende programmering i Python (variabler, løkker, funksjoner).
- Bruke turtle-biblioteket til å tegne grafikk.
- Utforske hvordan koordinatsystemet fungerer i et tegneområde (origo i midten).

- Bruke matematiske funksjoner som sinus og cosinus for å plassere objekter på en sirkel.
- Forstå forskjellen på grader og radianer.
- Trene på feilsøking og problemløsing i kode.
- Eksperimentere med farger (navn og hex-koder) i programmering.
- Oppleve sammenhengen mellom naturfag og teknologi (modellere solsystemet med kode).
- Utvikle kreativitet ved å endre farger, avstander og legge til egne elementer.

Fremgangsmåte i livekoding

Start med å sette elevene inn i tematikken ved å gi kort introduksjon om solsystemet, planetene og solen. Finn støtteaktiviteter på https://www.esero.no/tema/solsystemet-vart.

Gå rolig fram steg for steg, forklare det du skriver inn underveis. Ikke alt trenger å forklares, men få frem viktigheten av å skille mellom store og små bokstaver, tegn, mellomrom og skrive helt riktig. Det kan med fordel kjøres underveis, se hva IDE sier om feilmeldinger, snakke om det og rette opp, for så kjøre igjen og se at det hjelper. Også en stor fordel å ha forsøkt å skrive koden for hånd inn steg for steg i forkant, slik at du er kjent med det. Snakke om hvor de trykker de forskjellige spesielle tegnene som trengs i koden.

Bruk av KI for økt forståelse

Hvis du opplever at elevene ikke forstår helt funksjonene de skriver inn, og at du selv ikke strekker til å forklare det, kan du be dem bruke KI / LLM til å forklare og forenkle hva bestemte linjer gjør. Dette kan gi en økt forståelse for koden de skriver. KI kan også være nyttig i feilsøking, for å forklare en feilmelding fra Python skulle de oppstå.

En plan for kodingen

Det kan være veldig lurt å ha en plan for hva du skal kode underveis, og i de påfølgende stegene går vi frem steg for steg. En ramme for koden, hvor ting kommer inn underveis i stegene, blir som følger:

```
importering

# 1. Sette opp vinduet
# 2. Tegne solen
# 3. Tegne planetbanene
# 4. Tegne planetene

# Avslutter riktig
turtle.done()
```

Vurdering av modeller

Alle modeller kommer med sine styrker og svakheter, det skal vanskelig gjøres å finne en modell som ikke har det. Det fine er at det er stor lærdom i å vurdere modellene, og er jo også en del av naturfaget.

For mange modeller for solsystemet (og verdensrommet generelt) er gjerne størrelser og avstander en utfordring, fordi avstandene er så enormt store. Skal avstanden mellom planetene og størrelsen på planetene være relative, må du enten ha enormt god plass eller planetene blir veldig små.

Som et eksempel: Dersom du forminsker solsystemet 1 milliard (1 000 000 000) ganger, så blir avstanden mellom sola og den mest fjerntliggende planeten, Neptun, omtrent 5 km, sola får en diameter på omtrent 1,5 meter og den minste planeten Merkur får en diameter på omtrent 0,5 cm. Og da snakker vi en modell som er 5 km lang! Hvordan blir størrelsene på planetene dersom du forminsker slik at solsystemet får plass i skolegården (500 meter), eller i klasserommet?

Når dere har laget modellen deres av solsystemet, med eller uten forbedringene i steg 5, er det fint å bruke det ferdige bildet og reflektere rundt hva som gjør denne modellen god og hva som er svakhetene ved denne modellen. Det kan stilles som et helt åpent spørsmål, eller hjelpe elevene med noen veiledende spørsmål.

Hvis du sammenligner modellen med virkeligheten:

1. Hva stemmer i modellen? (farger, rekkefølge, størrelse på planetene, avstandene osv.)
2. Hvordan kunne du gjort modellen bedre?
3. Hva er utfordringene med å forbedre modellen?

Fullstendig kode med alle forbedringer

```
import turtle
import math
import random

# Sett opp vindu en passende størrelse
skjerm = turtle.Screen()
skjerm.title("Sol og planeter")
skjerm.screensize(800, 800, "black")

# Tegne solen
solfarger = ["#FFD700", "#FFCC33", "#FFA500"]
sol = turtle.Turtle()
sol.speed(0)
sol.hideturtle()
sol.color(random.choice(solfarger))
sol.penup()
sol.goto(0, -50)
sol.begin_fill()
sol.circle(50)  # Solens radius
sol.end_fill()

# Tegn planetbanene
bane = turtle.Turtle()
bane.speed(0)
bane.hideturtle()
bane.color("white")
planetbane = 100
for i in range(8):
    bane.penup()
    bane.goto(0, -planetbane)
    bane.pendown()
    bane.circle(planetbane)
    planetbane += 30

# Tegn planetene
planetfarger = [
    "#C0C0C0", # Merkur – metallisk grå
    "#EEDC82", # Venus – sandgul/beige
    "#1E90FF", # Jorden – havblå
    "#FF4500", # Mars – rust-oransje
    "#D2B48C", # Jupiter – lys brun
    "#F4A460", # Saturn – gyllen brun
    "#40E0D0", # Uranus – turkis
    "#0000CD"  # Neptun – dyp blå
]
```

```
planetbane = 100
for i in range(8):
    planet = turtle.Turtle()
    planet.shape("circle")
    planet.color(planetfarger[i])
    planet.penup()
    vinkel = random.uniform(0, 360)
    x = planetbane * math.cos(math.radians(vinkel))
    y = planetbane * math.sin(math.radians(vinkel))
    planet.goto(x, y)
    planetbane += 30

# Lagre som EPS/PostScript
skjerm.getcanvas().postscript(file="solsystemet.eps")

# Avslutter riktig
turtle.done()
```

Kilder

Steder å finne fargekoder til planeter og solen:

- https://htmlcolorcodes.com
- https://www.w3schools.com/colors/colors_picker.asp
- https://color.adobe.com/nb/explore

Noen hjelpesider om Python Turtle:

- https://docs.python.org/3/library/turtle.html
- https://realpython.com/beginners-guide-python-turtle/

Gratisprogrammer som kan åpne EPS og lagre som andre formater:

- Inkscape: https://inkscape.org
- GIMP: https://gimp.org

Kilder fra udir:

- https://www.udir.no/kvalitet-og-kompetanse/digitalisering-skole/algoritmisk-tenkning/
- https://www.udir.no/kvalitet-og-kompetanse/digitalisering-skole/kunstig-intelligens-ki-i-skolen/kunstig-intelligens-ki-i-skolen/
- https://www.udir.no/kvalitet-og-kompetanse/digitalisering-skole/kompetansepakker-for-digital-kompetanse-i-skolen/

Om livekoding:

- «Ten quick tips for teaching programming» 05.04.2018,
- https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006023
- «The effectiveness of live-coding to teach introductory programming» 06.03.2013,
- https://dl.acm.org/doi/10.1145/2445196.2445388

Kodeverktøyet Thonny: https://thonny.org

Solsystemet vårt: https://www.esero.no/tema/solsystemet-vart/

Lisensiering:

Dette verket er lisensiert under en Creative Commons Navngivelse-IkkeKommersiell-IngenBearbeidelse 4.0 Internasjonal lisens (CC BY-NC-ND 4.0).

Du kan dele dette materialet så lenge du krediterer oss, ikke bruker det kommersielt, og ikke endrer det.