Excited

Centre for Excellent IT Education

Hvordan komme i gang med tekstbasert programmering?

Guttorm Sindre

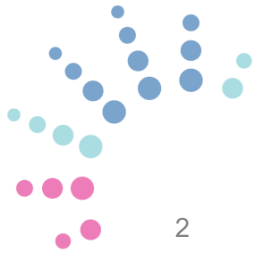
NORD University

Centre for Excellence in Education

NTNU

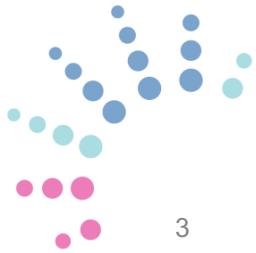
Disposisjon

- Utfordringer med tekstbasert programmering
- Ideer til løsning
- Små kodeeksempler med grafikk (turtle-modulen)
- Feilmeldinger, hvordan bli venn med dem?
- Eksempel på et litt større prosjekt
  - Program som foreslår matchende farger

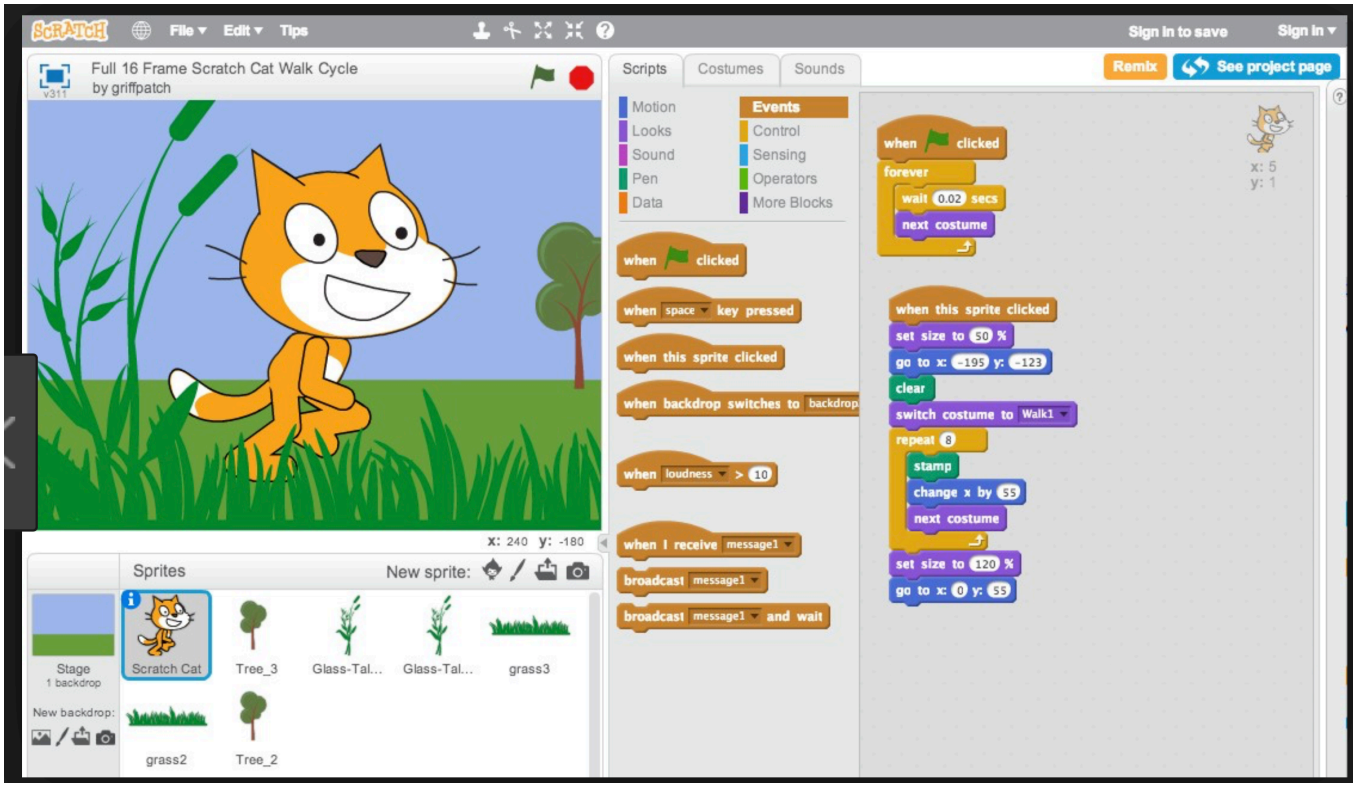


Disposisjon

- Utfordringer med tekstbasert programmering
- Ideer til løsning
- Små kodeeksempler med grafikk (turtle-modulen)
- Feilmeldinger, hvordan bli venn med dem?
- Eksempel på et litt større prosjekt
  - Program som foreslår matchende farger



Blokkbasert programmering

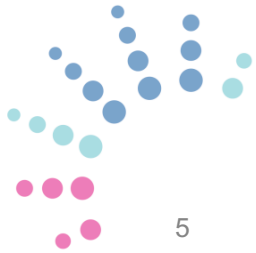


Tekstbasert programmering

temp.py - /Users/guttorm/Documents/temp.py (3.6.2)

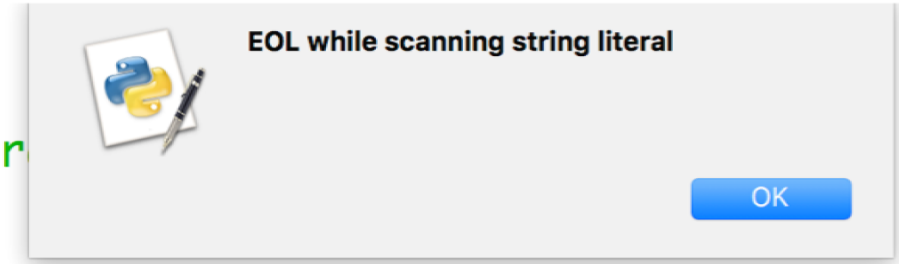
```
C = input("Hvor mange grader Celsius er det i dag? ")
C = float(C)
F = C * 1.8 + 32
print("Det blir", F, "grader Fahrenheit")
```

```
========== RESTART: /Users/guttorm/Documents/temp.py
Hvor mange grader Celsius er det i dag? 19
Det blir 66.2 grader Fahrenheit
>>>
```



Tekstbasert programmering

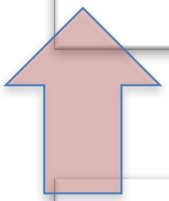
```
C = input("Hvor mange gr
C = float(C)
F = C * 1.8 + 32
print("Det blir", F, "grader Fahrenheit)
```



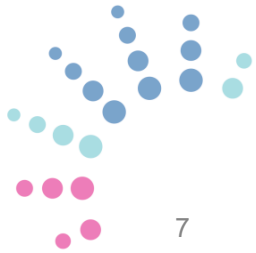
Tekstbasert programmering

temp.py - /Users/guttorm/Documents/temp.py (3.6.2)

```
C = input("Hvor mange grader Celsius er det i dag? ")
C = float(C)
F = C * 1.8 + 32
Print("Det blir", F, "grader Fahrenheit")
```



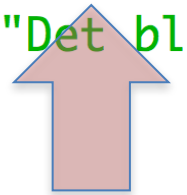
```
========== RESTART: /Users/guttorm/Documents/temp.py ==
Hvor mange grader Celsius er det i dag? 13
Traceback (most recent call last):
  File "/Users/guttorm/Documents/temp.py", line 4, in <module>
    Print("Det blir", F, "grader Fahrenheit")
NameError: name 'Print' is not defined
>>>
```



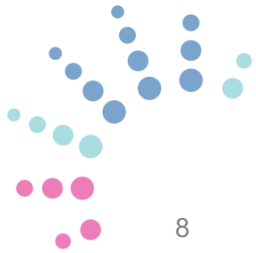
Tekstbasert programmering

temp.py - /Users/guttorm/Documents/temp.py (3.6.2)

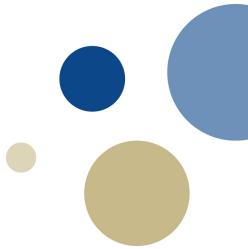
```
C = input("Hvor mange grader Celsius er det i dag? ")
C = float(C)
F = C * 1,8 + 32
print("Det blir", F, "grader Fahrenheit")
```



```
================== RESTART: /Users/guttorm/Documents/temp.py
Hvor mange grader Celsius er det i dag? 11
Det blir (11.0, 40) grader Fahrenheit
>>>
```



Forskjeller



Blokkbasert

- Kule ting kjapt
- Beskyttes mot feil
- Fargede blokker gjør det lett å se strukturen i koden

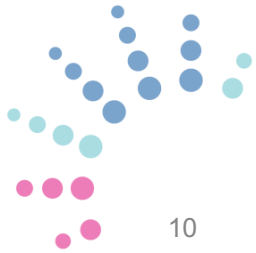
Tekstbasert

- Mye slit for lite
- Lett å gjøre feil
- Vanskeligere å holde oversikt

- Generelt anvendelig
- Jobbrelevans

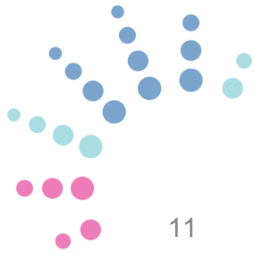
Typisk progresjon

- I programmering for nybegynnere, f.eks. Python
  - Enkel utskrift på skjerm (f.eks. ”Hello World!”)
  - Enkel input fra bruker
  - Verdier, datatyper, variable, tilordning
  - Aritmetiske uttrykk, presedens, parentesbruk
  - Valg (if-setninger), logiske betingelser
  - **Løkker (while, for)**
  - Funksjoner
  - Tekstfiler
  - Komplekse datatyper (lister, mengder, ...)
  - Unntaksbehandling
  - Rekursjon
- Mye kode for å gjøre lite... helt til løkker



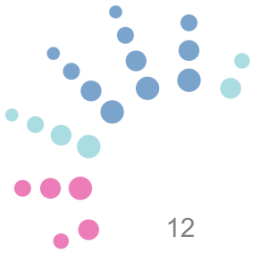
Disposisjon

- Utfordringer med tekstbasert programmering
- Ideer til løsning
- Små kodeeksempler med grafikk (Turtle)
- Feilmeldinger, hvordan bli venn med dem?
- Eksempel på et litt større prosjekt
  - Program som foreslår matchende farger



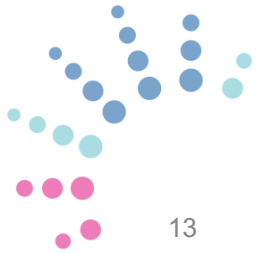
Ideer til løsning

- Gi rask mestring
  - ...ved å ha ferdig kode som elevene kan kjøre og endre
  - Koden må ha *endringspotensial;*
    - Små endringer kan gjøre koden bedre
- Finn eksempler som gjør mye med få linjer kode
  - ...ved å gå kjapt til *løkker*
  - Kode som gir lett synlige resultat, f.eks. grafikk
  - Biblioteker med ferdige funksjoner der det passer
    - Brukerinput og randomfunksjoner kan få program til å oppføre seg ulikt fra gang til gang
- Håndtere feilmeldinger på en positiv måte?
  - Hver feil en mulighet til å lære noe



Disposisjon

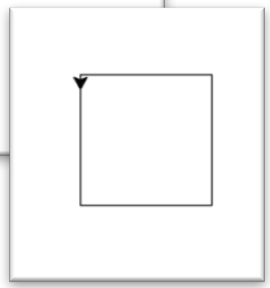
- Utfordringer med tekstbasert programmering
- Ideer til løsning
- Små kodeeksempler med grafikk (turtle-modulen)
- Feilmeldinger, hvordan bli venn med dem?
- Eksempel på et litt større prosjekt
  - Program som foreslår matchende farger



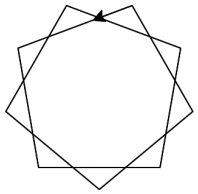
Eksempel 1: Evig firkant

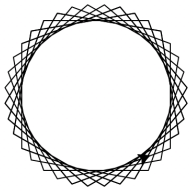
uendelig.py - /Users/guttorm/Documents/uendelig.py (3.6.2)

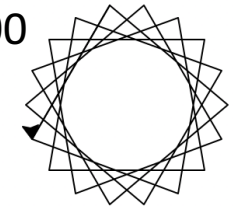
```
from turtle import *     #importer tegnefunksjoner
while True:              #repeter uendelig
    forward(100)
    left(90)
```

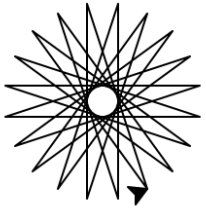


- Ikke spesielt interessant i seg selv
- Men potensial for endringer
  - Mange andre tall enn 90 : mer interessant figur

80 

70 

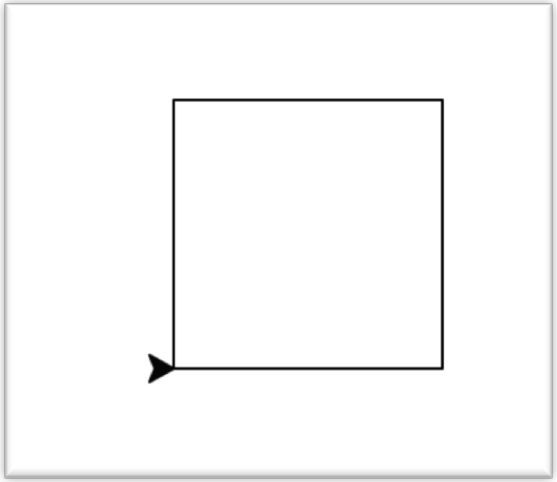
100 

162 

Eksempel 2: Firkant m. for-løkke

firkant1.py - /Users/guttormsindre/Documents/firkant1.py (3.6.1)

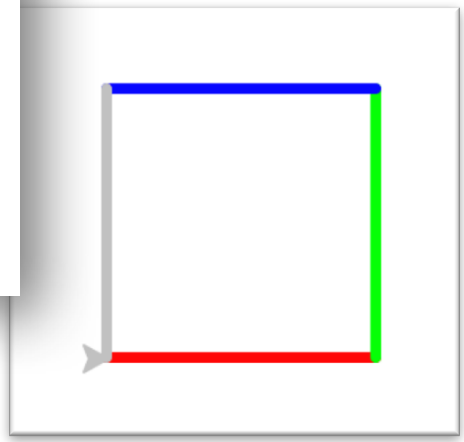
```
from turtle import *     #importer tegnefunksjoner

for i in range(4):       #repeter 4 ganger
    forward(100)
    left(90)
```



Eksempel 2a: synliggjøre iterasjon

firkant1.py - /Users/guttormsindre/Documents/firkant1.py (3.6.1)

```
from turtle import *    #importer tegnefunksjoner

pensize(4)

for i in ["red", "green", "blue", "grey"]:
    color(i)
    forward(100)
    left(90)
```

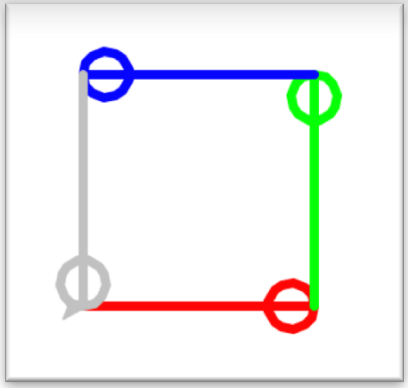


- Gjør det visuelt tydelig at løkka itererer gjennom de fire elementene i lista

Eksempel 2b,c: betydning av innrykk

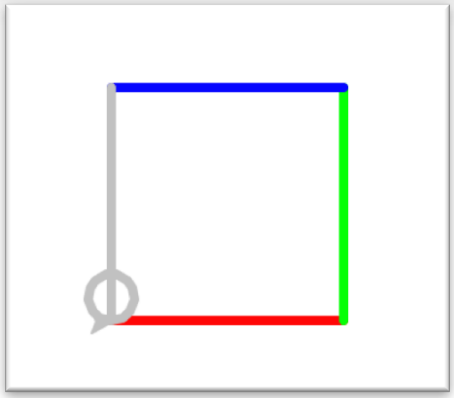
firkant1.py - /Users/guttormsindre/Documents

```
from turtle import *    #importer tegnefunks
pensize(4)

for i in ["red", "green", "blue", "grey"]:
    color(i)
    forward(100)
    left(90)
    circle(10)
```



firkant1.py - /Users/guttormsindre/Documer

```
from turtle import *    #importer tegnefunks
pensize(4)

for i in ["red", "green", "blue", "grey"]:
    color(i)
    forward(100)
    left(90)
circle(10)
```

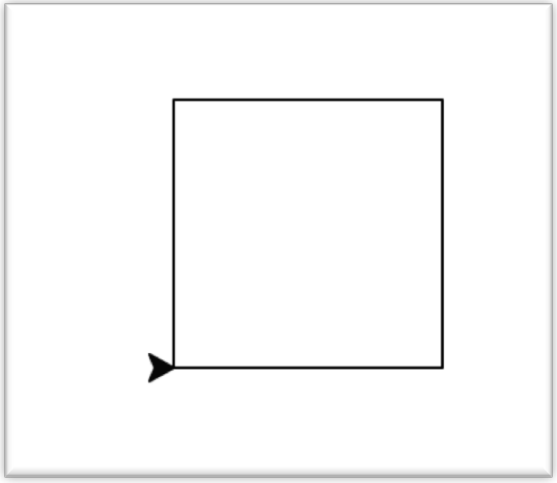


- ...og hvordan innrykk avgjør omfanget av løkka

Eksempel 2d: Firkant, 50 runder

*firkant2.py - /Users/guttormsindre/Documents/firkant2.p

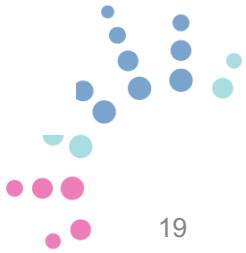
```
from turtle import *     #importer tegnefunksjoner
for i in range(200):     #repeter 200 ganger
    forward(100)
    left(90)
```



Kan gjøres mer spennende ved å bruke **i** også inni løkka. Hvordan?

Farger i Python Turtle

```
from turtle import *    #importer tegnefunksjoner

pensize(5)

color("green") # ved fargenavn på engelsk
circle(90)

color(1,0.5,0.5) # verdier 0-1 for R(ødt), G(rønt) og B(lått)
circle(70)

colormode(255)
color(30, 100, 255) # verdier 0-255 for R, G, B
circle(50)

color("#00509e") # som hex-verdier
circle(30)
```



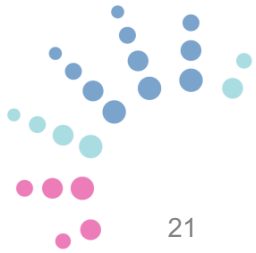
Ett eksempel med farger

farger2.py - /Users/guttormsindre/Desktop/filer-2/farger2.py (3.6.5)

```
from turtle import *    #importer tegnefunksjoner

bgcolor("black")
pensize(3)
fargeliste = ["pink", "yellow", "dark red"]

for i in range(200):
    color(fargeliste[i%3])
    forward(i)
    left(59)
```

Disposisjon

- Utfordringer med tekstbasert programmering
- Ideer til løsning
- Små kodeeksempler med grafikk (turtle-modulen)
- Feilmeldinger, hvordan bli venn med dem?
- Eksempel på et litt større prosjekt
  - Program som foreslår matchende farger



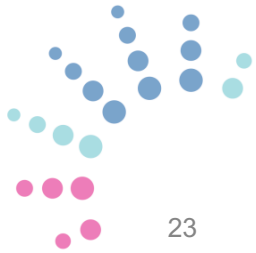
Bli venn med feilmeldinger

- Ser ofte store og ekle ut,
  - mye rød skrift
  - kryptisk innhold
- Hver feil en sjanse til å lære noe nytt
- Kanskje ha en “Error Dex”?
  - Seen: … Corrected: ...

- Noen tips:
  - Gjør en del feil selv
    - få erfaring med å rette dem
  - Feilmeldinger bør ofte leses nedenfra
    - mest relevant info sist

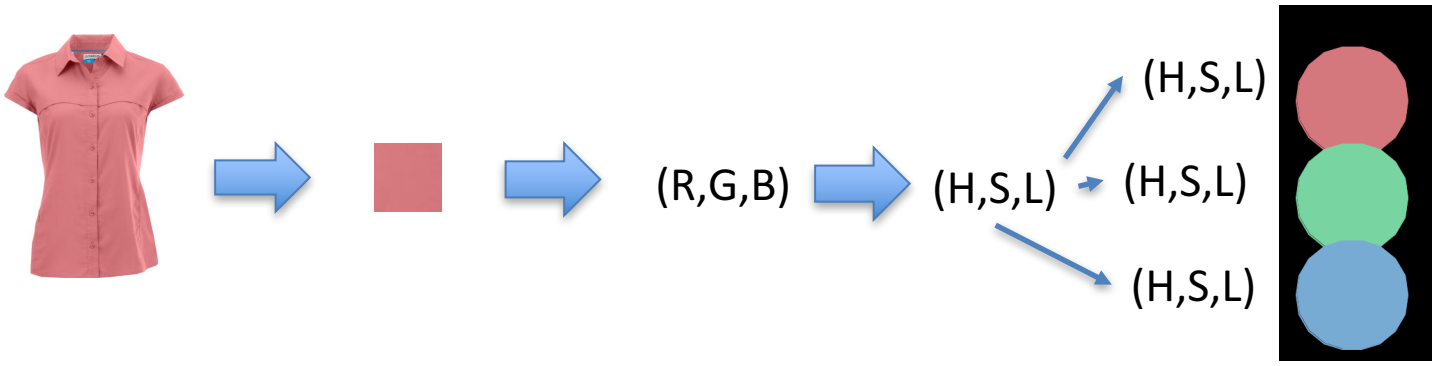
Disposisjon

- Utfordringer med tekstbasert programmering
- Ideer til løsning
- Små kodeeksempler med grafikk (turtle-modulen)
- Feilmeldinger, hvordan bli venn med dem?
- Eksempel på et litt større prosjekt
  - Program som foreslår matchende farger

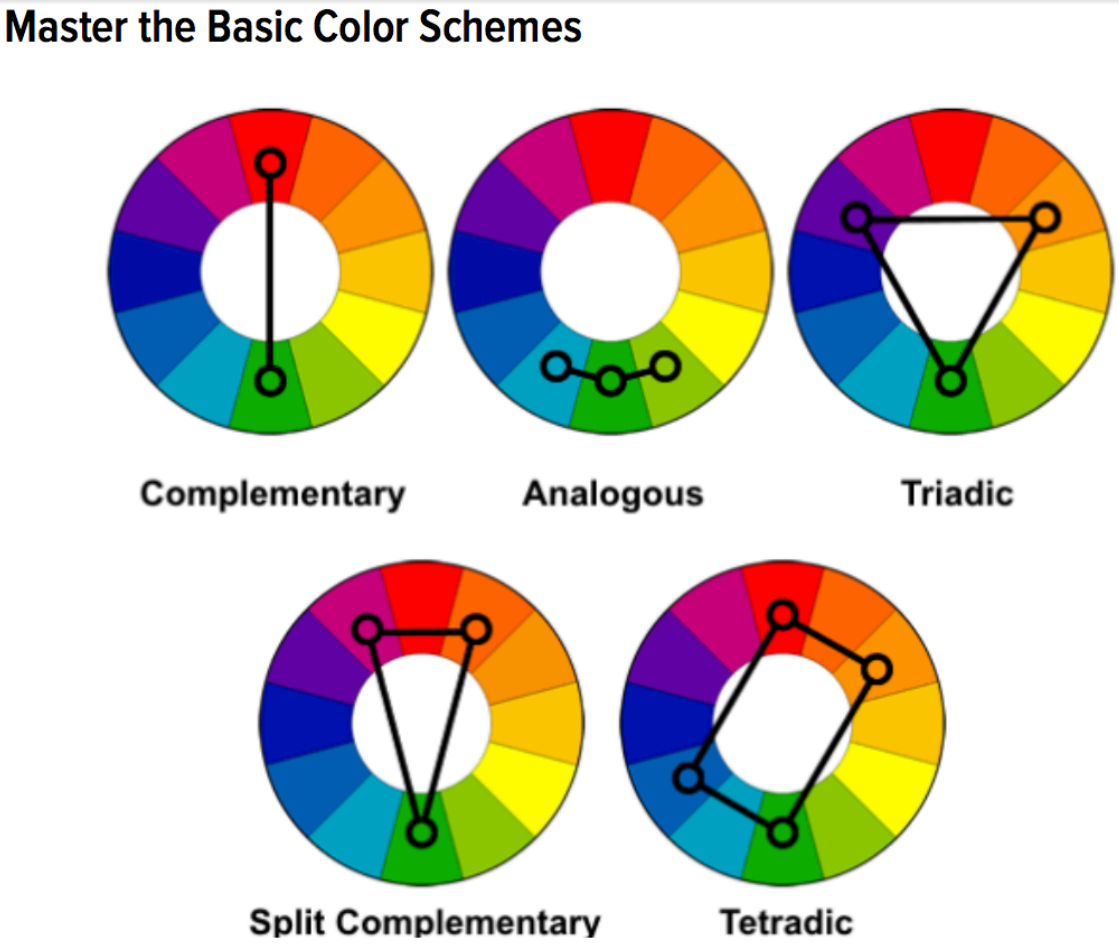


Større eksempel

- Ønsker å lage et program som kan foreslå matchende farger
- F.eks.
  - Elev tar bilde av et klesplagg med mobiltelefon og laster bildet inn på datamaskinen
  - Et Python-program finner ut hvilken farge klesplagget har
  - …regner deretter ut andre farger som kan matche med dette
  - ...og skriver mulige fargekombinasjoner ut på skjermen



Forstå problemet



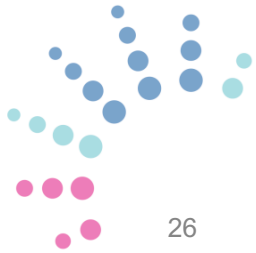
Hvis farger er oppgitt i HSL, (hue, saturation, lightness) er det lett å regne ut farger som matcher. F.eks.

Opprinnelig: (H, S, L)
Komplementær (H+0.5, S, L)
Triadisk: H+0.33, H+0.67
Analog: H+1/12, H-1/12

Men farger i Python Turtle er typisk oppgitt som (R, G, B)

Trenger dermed

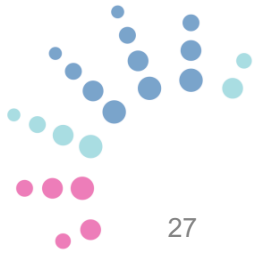
- Kode for å finne den dominerende fargen i et bilde
  - Websøk antyder modulen PIL (Pillow) som bra for dette, med metoden **Image.getcolors()**
- Kode for å oversette mellom HSL og RGB
  - Websøk antyder modulen colorsys med metodene hsl_to_rgb() og rgb_to_hsl()
- Kode for å regne ut matchende farger
  - Dette blir ganske enkel regning i HSL
- Kode for å vise resultatet pent på skjermen
  - Lys farge: sort bakgrunn, hvit skrift
  - Mørk farge: hvit bakgrunn, sort skrift



Konverteringen

color_match.py - /Users/guttormsindre/Desktop/filer-2/color_match.py (3.6.5)

```
from colorsys import hls_to_rgb, rgb_to_hls

def matching_hue(rgb, x):
    # INN:
    #  rgb: et tuppel med tall 0-1 for rødt, grønt og blått
    #  x: et tall mellom 0-1 som angir forflytning rundt på fargehjulet
    #     f.eks. 0.5 motsatt side av fargehjulet, som rødt -> grønt
    #     0.33 en tredjedel rundt fargehjulet, som rødt -> blått
    # UT: den nye fargen

    # oversetter fargen fra RGB til HLS
    hls = rgb_to_hls(rgb[0], rgb[1], rgb[2])
    # regner ut ny "hue", modulo 1.0 for å unngå tall > 1
    new_hue = (hls[0] + x) % 1.0
    # oversetter ny farge tilbake HLS -> RGB
    return hls_to_rgb(new_hue, hls[1], hls[2])
```

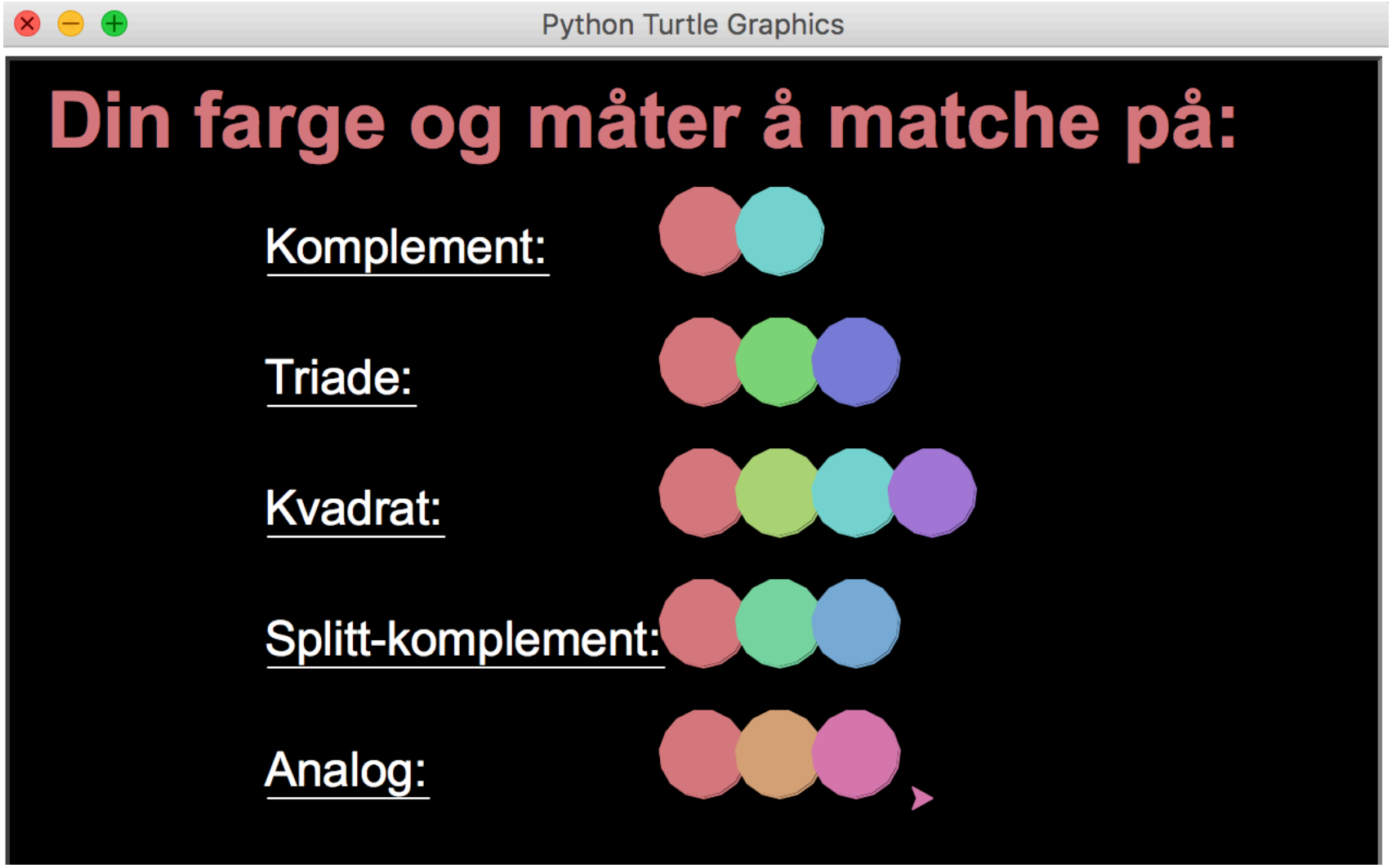


Finne farge fra bilde

color_image.py - /Users/guttormsindre/Desktop/filer-2/color_image.py (3.6.5)

```
from PIL import Image

def color_of_image(fil):
    bilde = Image.open(fil) #åpner bildefila
    liste = bilde.getcolors()
    # getcolors returnerer en liste av tupler (n, (r,g,b))
    # hvor n er antall forekomster av fargen (r,g,b) i bildet
    # og r, g, b er gitt som heltall 0-255
    maxi = 0
    # går derfor i løkke for å finne hvilken farge som har høyest n
    for i in range(1, len(liste)):
        if liste[i][0] > liste[maxi][0]:
            maxi = i
    farge = liste[maxi][1]
    # deler på 255 for å få fargen som tall 0-1
    return (farge[0]/255, farge[1]/255, farge[2]/255)
```

Eksempel på kjøring



Konklusjoner

- Gi elevene programmer der de kan oppnå noe ved små endringer
- Gå fort til løkker for å få programmer som gjør mye med lite kode
- Grafikk og farger kan være gøy, men er bare ett eksempel – andre ideer kan være like bra
- For større prosjekter, prøv å finne noe som viser en nyttig bruk av teknologien