Munthe

Programmering og diskusjon

Jeg har lenge vært interessert i programmering, og siden 2017 har jeg forsket på om og hvordan programmering kan bidra til økt matematisk forståelse. Programmering er, i mine øyne, et verktøy til å få elevene til å utforske, diskutere og forstå matematikk. Jeg er positiv til at programmering har kommet inn i matematikkfaget, men er kritisk til hvordan det har blitt gjennomført, spesielt med tanke på at matematikkfaget har ansvar for å lære elevene programmering.

Jeg vil her vise eksempler på programmering fra timene mine. Eksemplene er koblet til videregående opplæring, men erfaringene er overførbare til flere trinn. Dette er ikke ment som undervisningsopplegg, for avhengig av nivå kan elevene programmere større eller mindre deler av disse programmene selv. Jeg gir her hele koden slik at lærere selv kan bruke metoder som beskrevet gjennom PRIMM (Flø, 2021) eller UMC-modellen.

Morten Munthe
NMBU
morten.munthe@nmbu.no

Morten Munthe er årets vinner av Holmboeprisen.

Eksempel 1 – Sentralmål (9. trinn og 2P)

Hvordan kan vi bruke programmering slik at det bidrar til forståelse for matematikken? Dette var et spørsmål jeg tenkte mye på da jeg begynte å tenke på oppgaver til P-matte. Utfordringen er at programmeringen ofte blir tvunget inn i oppgaver som bedre kan løses ved hjelp av Excel, GeoGebra, kalkulator eller papir og blyant. Da forsvinner litt av gleden og nytten ved å bruke verktøyet. Når vi først skal bruke programmering i 2P, får vi se om vi kan lage noe som kan bidra positivt.

I 2P skal elevene lære sentralmålet median for et gitt sett med data. La oss ta en liste med data over høyden til en gruppe på 12 personer:

[173, 165, 183, 170, 171, 167,
166, 173, 168, 182, 183, 157]

For å finne median for hånd kunne vi sortert listen og funnet det midterst leddet eller gjennomsnittet av de to midterste leddene. Dette går fint for korte lister, men for lengre lister blir det fort tungvint. Når vi jobber med programmering, ønsker vi at elevene skal se metoden som ligger bak, slik at vi unngår en «black box» som for eksempel å skrive listen inn i Excel og bruke «=median».

Vi kan selvsagt løse denne oppgaven med programmering, og det bør være et mål at vi kan bygge opp programmet slik at det blir mer

Munthe, M.. (2023). Programmering og diskusjon. *Tangenten – tidsskrift for matematikkundervisning, 33*(4), 9–12.

www.caspar.no/tangenten

```
1  handel = [173,165,183,170,171,167,166,173,168,182,183,157]
2
3  antall = len(handel)                          # Antall tall i settet
4  handel.sort()                                 # Her sorterer vi datasettet i stigende rekkefølge
5
6  if antall % 2 == 0:                           # Hvis antall tall er et partall ...
7      midt = round(antall / 2)                  # ... finner den høyre midtverdien
8      median = (handel[midt] + handel[midt-1]) / 2   # ... regner gjennomsnittet av de to verdiene i midten
9  else:                                         # Ellers hvis tallet er et oddetall ...
10     midt = round((antall+1) / 2)              # ... finner midtverdien
11     median = handel[midt]                     # ... finner median
12
13 print("Median er", median)
```

```
Median er 170.5
```

Figur 1

```
1  handel = [173,165,183,170,171,167,166,173,168,182,183,157, 169]
2
3  antall = len(handel)                          # Antall tall i settet
4  handel.sort()                                 # Her sorterer vi datasettet i stigende rekkefølge
5
6  while antall > 2:                             # Så lenge listen har fler enn 2 tall ...
7      del handel[0]                             # ... sletter vi første tall i rekken
8      del handel[-1]                            # ... sletter siste tall i rekken
9      antall = antall - 2                       # ... oppdaterer antall tall i listen
10     print(handel)                             # ... printer ut rekken
11
12 if len(handel) == 1:                          # Hvis det er ett igjen ...
13     median = sum(handel)                      # ... så er median det tallet
14 else:                                         # Hvis det er to tall igjen ...
15     median = sum(handel)/2                    # ... regner vi gjennomsnittet av de to tallene
16
17 print("Median er", median)
```

```
[165, 166, 167, 168, 169, 170, 171, 173, 173, 182, 183]
[166, 167, 168, 169, 170, 171, 173, 173, 182]
[167, 168, 169, 170, 171, 173, 173]
[168, 169, 170, 171, 173]
[169, 170, 171]
[170]
Median er 170
```

Figur 2

«gjennomsiktig» enn Excel. Hva mener jeg med «gjennomsiktig»? La meg først vise et program som fungerer, men som ikke er «gjennomsiktig» (Figur 1).

Deler av koden er fin, men spesielt fra linje 6 og utover blir det mange kommandoer som krever en del av leseren. Modulus og litt komplisert liste-leking er elementer som er utfordrende å forstå. Hvordan kan vi gjøre dette mer «gjennomsiktig»?

La meg nå vise en kode som jeg mener er mye mer pedagogisk, og som legger mer til rette for læring (Figur 2).

Det andre programmet er lengre enn det første, og det er fortsatt en del ting som skjer i det skjulte, for eksempel sorterer vi listen i linje 4 og bruker en litt rar kommando til å slette siste ledd i listen i linje 8. Likevel vil jeg argumentere for at denne koden er bedre. Spesielt utskriften av svaret gjør det lettere for elevene å se hva programmet gjør. Programmet sorterer listen, og sletter så en etter en den største og minste verdien, før man står igjen med én eller to verdier som man bruker til å regne ut median. Her er det mer åpent hva som skjer, og kanskje elevene får en bedre forståelse av hva median er, og hvordan vi kan regne den ut. Det er også en fordel at man lett kan endre på antall elementer i listen og så se hvordan utregningen endres.

Munthe, M.. (2023). Programmering og diskusjon. *Tangenten – tidsskrift for matematikkundervisning, 33*(4), 9–12.

```python
handel = [173,165,183,170,171,167,166,173,168,182,183,157, 169]

antall = len(handel)                          # Antall tall i settet
handel.sort()                                 # Her sorterer vi datasettet i stigende rekkefølge

while antall > 2:                             # Så lenge listen har fler enn 2 tall ...
    del handel[0]                             # ... sletter vi første tall i rekken
    del handel[-1]                            # ... sletter siste tall i rekken
    antall = len(handel)                      # ... oppdaterer antall tall i listen
    print(handel)                             # ... printer ut rekken

if len(handel) == 1:                          # Hvis det er ett igjen ...
    median = sum(handel)                      # ... så er median det tallet
else:                                         # Hvis det er to tall igjen ...
    median = sum(handel)/2                    # ... regner vi gjennomsnittet av de to tallene

print("Median er", median)
```

```
[165, 166, 167, 168, 169, 170, 171, 173, 173, 182, 183]
[166, 167, 168, 169, 170, 171, 173, 173, 182]
[167, 168, 169, 170, 171, 173, 173]
[168, 169, 170, 171, 173]
[169, 170, 171]
[170]
Median er 170
```

Figur 3

```python
from random import randint                    # Importerer kommandoen randint

tall = randint(1, 100)                        # Programmet velger et tilfeldig tall i [1, 100]
ant = 1                                       # Setter antall gjett til 1
gjett = int(input("Gjett på et tall: "))      # Ber spilleren gjette på et tall

while gjett != tall:                          # Så lenge gjettet ikke er likt tallet ...
    if gjett > tall:                          # Hvis gjettet er større enn tallet ...
        print("Du gjettet for høyt, prøv igjen.")  # ... så skriver den ut det
    else:                                     # Ellers (hvis tallet er mindre enn tallet) ...
        print("Du gjettet for lavt, prøv igjen")   # ... så skriver den ut det
    gjett = int(input("Gjett på et tall: "))  # Spilleren gjetter på et nytt tall
    ant = ant + 1                             # Antall gjett økes med en

print("Du fikk riktig, yay. Du brukte", ant, "gjett.")
```

Figur 4

La oss si at vi legger til en person som har høyden 169 cm. Da kan vi se hvordan programmet håndterer et odde antall listeelementer (Figur 3).

Vi kan videre bruke programmet til å utforske spørsmål som «Hva trenger man å legge til for at median skal bli 172? Hvilke høyder? Hvor mange høyder?». Her finnes det sikkert mange flere spørsmål, og nettopp slike spørsmål kan gjøre at programmering legger til rette for diskusjon. Om dette gir bedre forståelse enn å regne for hånd, er vanskelig å si, men hvis vi først skal bruke programmering i 2P, er dette ikke det verste.

Eksempel 2 – Oppvarming til halveringsmetoden (R1)

Her skal vi se på hvordan vi kan lage et litt morsomt program som leder elevene inn i tankesettet til halveringsmetoden. Programmet egner seg godt til R1, men kan fint brukes helt ned til ungdomstrinnet.

Programmet lager først et ukjent tilfeldig tall mellom 1 og 100. Deretter ber programmet spillerne om å gjette på hvilket tall de tror det er. Programmet sjekker så om gjetningen er korrekt, før det gir tilbakemelding til spilleren om hvorvidt svaret er riktig, for høyt eller for lavt. La oss se på et eksempel (Figur 4):

Munthe, M.. (2023). Programmering og diskusjon. *Tangenten – tidsskrift for matematikkundervisning, 33*(4), 9–12.

- Programmet lager et ukjent tall (la oss si at det ble 73)
- Spilleren gjetter på tallet 50
- Programmet sjekker svaret og skriver til spilleren at «Du gjettet for lavt, prøv igjen»
- Spilleren gjetter igjen, nå på tallet 80
- Programmet sjekker svaret og skriver til spilleren at «Du gjettet for høyt, prøv igjen» <dette forsetter inntil …>
- Spilleren gjetter igjen, nå på tallet 73
- Programmet sjekker svaret og skriver til spilleren at «Du fikk riktig …»

Programmet virker i utgangspunktet kanskje ikke spesielt relevant for matematikk, men programmet har en god del matematikk man kan diskutere med elevene.

- Hva er den beste taktikken for gjetting?
- Hvor mange gjetninger trenger man maksimalt hvis man bruker en best mulig taktikk?
- Hvor mange gjetninger trenger man for et program som ber spilleren om å gjette på et tall mellom 1 og n (der n er et valgfritt heltall)?

Dette passer veldig fint inn i den algoritmiske tenkeren for både ungdomstrinnet og videregående skole. Målet er at dette skal lede opp til halveringsmetoden i R1, la oss se på et eksempel.

Eksempel

Anta at vi har en funksjon f som har ett nullpunkt i intervallet [0, 10].

- Hvordan kan vi dele dette opp på best mulig måte? (Dele opp i [0, 5] og [5, 10] som er akkurat det samme vi gjorde i det forrige programmet.)
- Hvordan vet vi i hvilket intervall nullpunktet er? (Vi trenger å vite funksjonsverdien for å avgjøre i hvilket av intervallene vi skal fortsette å lete. Dette er likt forrige program, der vi fikk beskjed om «for høyt» eller «for lavt».)
- Hvordan deler vi opp det nye intervallet? (Vi tar alltid midtpunktet i det foregående intervallet.)

Dette fortsetter vi med til vi har oppnådd ønsket nøyaktighet.

Programmet er metodisk veldig likt halveringsmetoden. Når elevene senere skal lære (og programmere) halveringsmetoden, vil prinsippene i metoden være kjent. Dette gjør at elevene har en lettere inngang til metoden.

Diskusjon, utforsking og forståelse er sentralt i eksemplene over, men programmering er fortsatt utfordrende å gjennomføre på en god måte i klasserommet. Vi ønsker både å fremme faget og å motivere elevene. I mine øyne er vi fortsatt i oppstartsfasen, og det vil komme gode eksempler på bruk av programmering etter hvert. I denne fasen er det mer enn noen gang viktig at lærere som har gode eksempler, deler disse med andre. Selv holder jeg stadig på å utvikle ulike opplegg og kodesnutter som jeg tror og håper kan være nyttige for andre lærere, og legger dem ut på EduData.no. Håpet mitt er at vi etter hvert kan få laget en ressurs som kan bidra til at alle som vil bruke programmering i sin undervisning, uavhengig av fag, kan finne eksempler de kan dra nytte av. Inntil det håper jeg at eksemplene her kan bidra til kreativitet og inspirasjon for mange nye ideer for både bruk av programmering og diskusjon i klasserommet.

Referanse

Flø, E. E. (2021). Programmering i LK20. *Tangenten – tidsskrift for matematikkundervisning, 32*(1), 3–9.

Munthe, M.. (2023). Programmering og diskusjon. *Tangenten – tidsskrift for matematikkundervisning, 33*(4), 9–12.

www.caspar.no/tangenten