

## Lærerveiledning - Kalkulator

Kurs: Python

## Om oppgaven

I denne oppgaven skal elevene lage en kalkulator helt på egenhånd. Det er meningen at kalkulatoren skal kunne legge sammen, trekke fra, gange og dele to tall på hverandre.



#### Oppgaven passer til:

Fag: Matematikk, Programmering

Anbefalte trinn:7.-10. trinn

**Tema**: Aritmetikk, if-setninger, brukerinteraksjon, funksjoner

Tidsbruk: Dobbeltime

1/		°
Kom	petans	emai

Kompetansemai		
Matematikk, 4. trinn: bruke matematiske symboler og uttrykksmåter for å uttrykke matematiske sammenhenger i oppgaveløsning		
Programmering, 10. trinn: bruke flere programmeringsspråk der minst ett er tekstbasert		
Programmering, 10. trinn: bruke grunnleggende prinsipper i programmering, slik som løkker, tester, variabler, funksjoner og enkel brukerinteraksjon		
Programmering, 10. trinn: omgjøre problemer til konkrete delproblemer, vurdere hvilke delproblemer som lar seg løse digitalt, og utforme løsninger for disse		

# Forslag til læringsmål Elevene klarer å dele opp programmet sitt i logiske funksjoner Elevene klarer å skrive selvstendig kode, basert på små hint Elevene er i stand til å bruke matematiske symboler til å kode en enkel kalkulator Forslag til vurderingskriterier Eleven viser middels måloppnåelse ved å fullføre oppgaven. Eleven viser høy måloppnåelse ved å videreutvikle egen kode basert på oppgaven, for eksempel ved å gjøre en eller flere av variasjonene nedenfor. Forutsetninger og utstyr Forutsetninger: Kjennskap til while-løkker, if-setninger, funksjoner. **Utstyr**: Datamaskin med Python installert.

### Fremgangsmåte

Her kommer tips, erfaring og utfordringer til de ulike stegene i den faktiske oppgaven. Klikk her for å se oppgaveteksten. (../kalkulator/kalkulator.html)

## Sammenlikning av tekst

En naturlig måte å løse oppgaven for elevene på er å sammenlikne en tek ststreng med en operator. Et hint en kan gi elevene er å spørre hva føl gende kode gir ut:

```
a = '*'
print(a == '*')
```

Der det logiske uttrykket som står i parentesene kan kombineres med en if-setning for å finne ut hvilken operator brukeren prøver å bruke. En som er mer erfaren med Python vil kanskje bemerke seg at == ikke er veldig "Pythonisk". Filosofien til python er at en skal i størst mulig grad strebe etter å uttrykke seg i klartekst med ord. Det er derfor vi for eksempel skriver for element in list. Med dette i tankene kan koden ovenfor skrives som

```
a = '*'
print(a is '*')
```

Men merk at det er noen fallgruver med denne metoden. Hva skjer om du kjører følgende kode?

```
a = 19998989890
b = 19998989889 +1
print(a is b)
```

Grunnen til at koden ovenfor gir ut False handler om at is sammenligner to *objekter* i minnet, mens == sammenligner *verdien* til objektene. Kort oppsumert bruker vi gjerne is for å sammenlikne teststrenger eller objekter mens == forbeholdes størrelser.

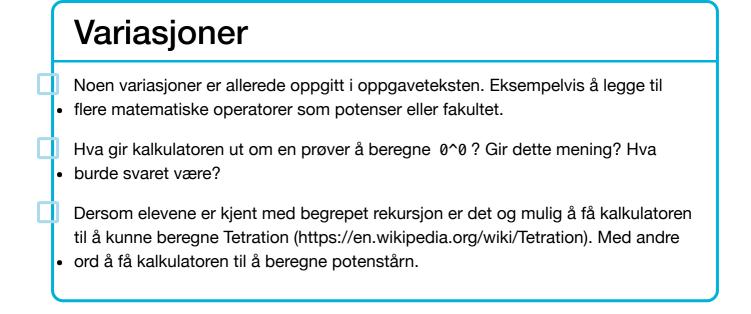
## Robusthet med tekst som input

Noen elever ergrer seg kanskje over at om en skriver inn \* så godtar ikke programmet dette som gyldig innputt. Dette er fordi Python sammenlikner om strengene er *nøyaktig* lik, så mellomrom godtas ikke. En måte å løse dette problemet på er å bruker strip() kommandoen som fjerner tomromm foran og bak tekstrengen. En annen mulighet er å bruke replace()

Forskjellen kan du sjekke ved å kjøre kodesnutten under

```
text = ' The quick brown fox jumps over the lazy dog '
print(text)
print(text.strip())
print(text.replace(' ','')
```

Spiller det noen rolle i denne oppgaven om en bruker replace() eller strip()?



#### Eksterne ressurser

Foreløpig ingen eksterne ressurser