Begreper

Program

Et sett med operasjoner som får en datamaskin til å utføre noe.

Beskrivelse

Et program eller en app (applikasjon) er et set med operasjoner som får en datamaskin til å utføre noe. Når dette skjer sier man ofte at man *kjører* programmet. Hvis vi bruker en mobiltelefon eller en datamaskin kjører vi nesten alltid et eller annet program, f eks sms-appen til telefonen eller Microsoft Word på datamaskinen. Et program lages ved at man programmerer eller koder det og det kjøres ved at man laster programmet inn i arbeidsminne til mobiltelefonen eller datamaskinen.

Arbeidsminne

Et sted i datamaskinen hvor programmer som kjøres lagres.

Beskrivelse

Når et program kjøres lastes det inn i arbeidsminnet til en datamaskin. Er programmet komplisert, har mange operasjoner, tar det også større plass i arbeidsminnet.

Arbeidsminnet til en datamaskin kalles også for RAM (Random Access Memory).

Programmere

Å lage et program

Beskrivelse

Når du lager et program sier vi at du programmerer eller koder programmet. Da setter du sammen operasjoner som får datamaskinen til å gjøre det du vill den skal gjøre. Operasjonene kan enten skrives i en tesktfil, tekstprogrammering eller settes sammen grafiskt gjennom blokker, blokkprogrammering. Blokkprogrammering kan være enklere å bruke når du skal lære deg programmering og tekstprogrammering brukes gjerne for mer komplekse programmer. Uanset om du velger tekst eller blokk-programmering må du velge et eller annet programmeringsspråk som du koder i.

Tekstprogrammering

Programmering hvor operasjoner skrives i tekstfiler.

Beskrivelse

I tekstprogrammering består operasjoner av instruksjoner satt sammen av ord og tall. Operasjonene utføres så linje for linje i en rekkefølge som går fra toppen av filen mot bunnen og i hver linje utføres de fra høyre mot venstre. *ILLUSTRASJON* Tekstfilene kan så enten *kjøres* direkte gjennom en kommandotolk (*interpreter* eng.) og filene kalles da for et *skript*, eller kan filene gjøres om til egne program gjennom å kompilere de. Vi

kommer bare lage skript når vi bruker tekstprogrammering. Alle apper som installeres på en datamaskin er eksempler på kompilerte program.

Tekstprogrammering må følge fastlagde regler som bestemmes av hvilket programmeringsspråk du koder i. Reglene kalles for syntaksen til programmeringsspråket.

Kompilere

Lage et kjørbart program av en tekstfil.

Beskrivelse

En type av tekstprogammering hvor man lager et kjørbart program fra en tekstfil.

Blokkprogrammering

Programmering hvor operasjonene settes sammen grafisk gjennom ulike blokker.

Beskrivelse

I blokkprogrammering består operasjoner av blokker som settes sammen grafisk i et eget program. Det er oftest lettest å komme igang å programmere gjennom å bruke blokkprogrammering da betydningen til operasjonene mange ganger lett kan visualiseres gjennom utseendet til de ulike blokkene.

Syntaks

Regler for hvordan operasjoner skrives innen tekstprogrammering.

Beskrivelse

Ved tekstprogrammering skriver man alle operasjoner med bokstaver og tegn. Syntaksen til et tekst-programmeringsspråk er regler som bestemmer hvordan bokstavene kan skrives. På samme måte som syntaksen til skriftlig norsk bestemmer hvordan norsk kan skrives, f. eks. en setning starter med stor bokstav og slutter med et punkt, bestemmer syntaksen til tekstprogrammering hvordan operasjoner kan skrives i en tekstfil.

Syntaksen for å tildele to variabler hver sin verdi og å lage en kommentar er forskjellige for ulike programmeringsspråk. Her viser vi hvordan det ser ut for Python og JavaScript. Vi ser at det er flere ting som er likt men også noen ting som skiller de to språkene.

Python

```
# Alder og høyde til en person
alder = 9
høyde = 1.34
```

JavaScript

```
// Alder og høyde til en person
let alder = 9;
let høyde = 1.34;
```

Kommandotolk

Et program som kjører et skript.

Beskrivelse

Noen filer som lages ved tekstprogrammering trenger et eget program, kommandotolk, for å kunne kjøres. Tekstfilene med programmet kalles da for skript.

Et python-skript trenger en kommandotolk for å kunne kjøres. Hvis man har installert python på datamaskinen kan man kjøre skriptet gjennom å bruke kommandotolken `python` forran filnavnet til skriptet (`mitt-program.py`).

```
> python mitt-program.py
```

Programmeringsspråk

Et språk som beskriver hvordan operasjoner settes sammen i et program.

Beskrivelse

Når man programmerer må man velge et programmeringsspråk man skal bruke for å sette sammen de ulike operasjonene. Det finnes mange ulike programmeringsspråk som kan brukes til enten tekstprogrammering eller blokkprogrammering. De finnes aller flest tekst-programmeringsspråk i verden da disse er enklere å bruke når man skal lage større program.

Eksempler på tekst-programmeringsspråk som trenger en kommandotolk

- Python - brukes mye for å regne med matematikk
- JavaScript - brukes til å styre en nettside

Eksempler på tekst-programmeringsspråk som må kompileres

- C - brukes til å programmere kjernen til Android
- C++ - brukes til å programmere Windows

Eksempler på blokk-programmeringsspråk

- Scratch - brukes til å programmere enkle spill
- Blokkuino - brukes til å styre en arduino

Algoritme

En liste med operasjoner som utfør en oppgave.

Beskrivelse

En algoritme er en liste med operasjoner som tilsammens utfør en oppgave. Oppgaven kan være en matematisk oppgave som å beregne et gjennomsnitt eller å finne typetallet (største tallet i en tallmengde). Oppgaven kan også være noe værdagslig som å velge den beste oppskriften i en samling med oppskrifter.

Listen med operasjoner må komme i en gitt rekkefølge, for eksempel er følgende to operasjoner fra en oppskrift:

- blande ingrediensene sammen
- sett inn formen i ovnen Her er det opplagt at man må blande ingrediensene i en oppskrift før man setter inn formen i ovnen, men akkurat rekkefølgen i en algoritme er ofte vanskelig for en som starter å lære seg programmering å få inn.

En algoritme kan være noe konkret som noen linjer med kode som tilsammens utfør oppgaven eller en abstrakt beskrivelse av hvilke operasjoner som skal utføres og i hvilken rekkefølge de skal gjøres. TODO: Gi eksempler eller ta bort

Mange internettbedrifter bruker algoritmer når de skal presentere varer eller artikler til deg. For eksempel bruker strømmetjenester som Netflix og Spotify slike når de skal presentere filmer eller musikk de mener passer deg. Disse algoritmene baserer seg på at bedriftene husker hva du har valgt tidligere og hva andre personer som likner deg (samme alder, kjønn, bosted) har valgt. TODO: Passer dette her?

Eksempel: gjennomsnitt

- Lag en list-variabel `alle_tall` hvor du samler alle tall du vill beregne gjennomsnittet på.
- Summer alle tallene i `alle_tall` og lagre resultatet i variabelen `sum`.
- Divider verdien i `sum` med antallet tall i `alle_tall` og lagre verdien i variabelen `gjennomsnitt`

```
alle_tall = [4, 5, 3, 5, 6, 4, 3, 5]
sum = 0
for tall in alle_tall:
    sum += tall
gjennomsnitt = sum/len(alle_tall)
```

Operasjon

En enkelt ting en datamaskin skal gjøre

//: # Legge inn kommando for det faktiske kommandoet man skriver inn

Beskrivelse

En operasjon forteller datamaskinen at den skal gjøre noe. Et program er mange operasjoner som er satt sammen etter hverandre. En operasjon i hverdagslivet kan være mange ting men innen programmering er en operasjon noe forenklet begrenset til operasjoner på verdier og det å bruke kontrolstrukturer og funksjoner.

Det finnes 4 ulike hovedvarianter av operasjoner som utøfres på verdier:

1. Bearbeider en verdi (*process* eng.)
2. Lagre en verdi i en variabel
3. Hente inn en verdi (inndata), input
4. Vise en verdi (utdata), output

Bearbeide verdier

Når du bearbeider en eller flere verdier gjør datamaskinene noe med verdiene. To ulike måter å bearbeide verdier på er å enten bruke en operator for eksempel `5 + 6` eller å sende verdiene som argumenter til en funksjon for eksempel `gjennomsnitt([3, 6, 8])`.

Flere operasjoner som bearbeider verdier kan settes sammen til uttrykk for eksempel:

```
round(sqrt(3)/2, 3)
```

Lagre verdi

Verdier lagres i variabler og operasjoner lagres i funksjoner. Verdien eller operasjonene blir da lagret i arbeidsminnet og kan brukes seinere i programmet.

Når en verdi skal lagres i en variabel bruker man en tilordningsoperator (`=`).

```
alder = 9
navn = "Marie"
```

Her lagres to ulike verdier: `9, "Marie"`, til variablene: `alder, navn`.

Hente en verdi, input

En datamaskin hadde vært ganske verdiløs hvis den ikke kunne hente verdier enten fra en bruker eller fra et eller annet sted for eksempel en avstandsensor på en bil.

```
navn = input("Hva heter du?")
```

Her brukes funksjonen `input` til å spør brukeren om å skrive inn sitt navn. Programmet stopper da opp og venter på at brukeren skal skrive inn sitt navn.

Vise en verdi, output

En verdi kan vises til en bruker for eksempel ved at den vises på skjermen, eller skrives til en fil.

```
print("Jeg er 13 år!")
```

Her sendes verdien `"Jeg er 13 år!"` til funksjonen `print` som så viser den på skjermen.

Kontrollstruktur

Bestemmer hvilke deler av et program som skal utføres

Beskrivelse

Vanligvis utføres operasjonene i et program fra toppen av tekstfilen og nedover. Hver linje i tekstfilen utføres en gang. Kontrollstrukturer består av if-setninger og løkker og de kan bestemme at noen linjer skal utføres og andre ikke, eller at noen linjer skal utføres mange ganger. Slik bestemmer kontrollstrukturer flyten i et program.

Uttrykk

En eller flere operasjoner som er satt sammen og som returnerer en verdi

Beskrivelse

Et uttrykk (*expression* eng.) består av en eller flere operasjoner som er satt sammen. Alle enkelt operasjoner i et uttrykk resulterer i en verdi.

```
navn = "Petter"
setning = "Jeg heter " + navn + " og jeg er " + 16 + " år."
```

Her er to linjer som begge tilordner to variabler to verdier som begge er resultater av to uttrykk. Den første linjen består av et uttrykk som resulterer i verdien: `"Petter"`. Den andre linjen består av et uttrykk som resulterer i verdien `"Jeg heter Petter og jeg er 16 år."`.

Variabel

Er et navngitt sted i arbeidsminnet som lagrer verdier.

Beskrivelse

En variabel er et navngitt sted i arbeidsminnet som lagrer en eller flere verdier i et program. Navnet til variabelen brukes så i uttrykk for å representere verdien til variabelen.

Du bestemmer selv hva navnet til en variabel skal være, men syntaksen til programmeringsspråket legger begrensninger på hva et variabelnavn kan være. Når man lager en variabel for første gangen sier man at man definerer den og man kan ikke bruke en variabel før den er definert. Navnet til en variabel brukes når du:

1. definerer variablen
2. bruker verdien til variablen
3. endrer verdien til variablen.

Disse tre måtene kan illustreres med følgende eksempel:

```
lån = 500000.
rente_prosent = 0.05
rente = lån*rente_prosent
lån = lån + rente
```

Her defineres første tre variabler: `lån`, `rente_prosent` og `rente`. På tredje raden brukes variabelen `lån` og `rente_prosent` i et uttrykk for å regne ut hva renten skal være. På siste raden endres så verdien til variabelen `lån` at renten legges til det første lånet.

Variabel i matematikken

Ordet variabel brukes også i algebra i matematikken. Da betegner det ofte en ukjent tallverdi, f. eks. *x, y*. Innen programmering har en variabel *alltid* en verdi og representerer altså ikke noe ukjent.

Variabelnavn

Navnet til en variabel.

Beskrivelse

En variabel har altid et navn. Syntaksen til et programmeringsspråk legger noen krav til navnet.

For Python gjelder disse kravene til et variabelnavn

1. kan inneholde bokstaver, siffer og understrek.
2. kan IKKE inneholde mellomrom , bindestrek `-` eller punktum `.`: Bruk heller `_` mellom ord i variabelnavnet.
3. kan IKKE starte med en siffer.
4. må være unikt. Flere variabler kan altså ikke ha samme navn.

Tips til variabelnavn

Bruk variabelnavn som beskriver hva verdien til variabelen skal brukes til. For eksempel er navn som `a`, `b` oftest dårlige navn, mens `lån`, `navn` eller `poengsum` er bra navn.

Verdi

En konkret representasjon av data

Beskrivelse

En verdi representerer en størrelse som kan manipuleres av et program. Verdier er ofte det man til daglig kaller data. En datamaskin er altså en maskin som bruker verdier! Verdier har ulike datatyper som beskriver hva verdien kan brukes til:

Eksempler på datatyper er:

- tall: `2, -5, 0.45, 1e-3`
- tekst eller strenger: `"A", "Jeg heter Lise."`

- boolean: `True, False`
- liste: `[5, 3, 8], ["ost", 2, "melk", 5]`

Datatype

Beskriver hva en verdi kan brukes til

Beskrivelse

Alle verdier har en datatype som beskriver hva den kan brukes til. De vanligste datatypene er:

- tall: Brukes i all matematikk, se for eksempel [aritmetiske operatorer](#aritmetisk operatorer).
- strenger: Brukes til å representere tekst-verdier.
- boolean: Brukes til å beskrive om noe er *sant* (`True`) eller *falskt* (`False`).
- lister: Brukes å samle verdier i en ordnet rekkefølge.
- [assosiative lister](#assosiativ liste). Brukes til å samle verdier som er assosiert med en nøkkel.

Definere

Operasjon som lager en variabel eller funksjon.

Beskrivelse

For å kunne bruke en variabel eller funksjon må de først defineres. Når en variabel defineres knyttes et variabelnavn til en verdi som lagres i arbeidsminnet. Når en funksjon defineres knyttes et funksjonsnavn til et sett med operasjoner. Navnet brukes så i programmet for å representere enten verdien til variabelen eller alle operasjonen til funksjonen.

I tilegg til å få et navn må en variabel også få en verdi (tilordnes) når den defineres. I noen programmeringsspråk kan man si ifra at man skal bruke en variabel uten å gi den en verdi. Da heter det å *deklarere* en variabel.

Tilordne

Gir eller endrer en verdi til en variabel

Beskrivelse

Når en verdi lagres i en variabel heter det at denne variabelen tilordnes en verdi. Dette gjøres med en tilordningsoperator, som i Python er `=`. På venstre side av tilordningsoperatoren må variabelens navn skrives. På høyre side må det stå en verdi eller et uttrykk som resulterer i en verdi. Denne verdien blir så lagret i variabelen.

Tall

Datatype som brukes til numeriske verdier

Beskrivelse

Verdier av datatypen tall brukes i programmering til utføre operasjoner som vi i hverdagen tenker på som matematiske operasjoner. Når to verdier skal legges sammen gjennom addisjon må disse verdien for

eksempel ha datatypen tall.

En del programmeringsspråk for eksempel Python skiller på datatypen heltall og flyttall. Flyttal brukes til å reprsentere tall med desimaler. Andre programmeringsspråk skiller ikke i utgangspunktet på heltall eller flyttal som f eks JavaScript eller Scratch.

Heltall

Datatype som brukes til å representere hele tall

Beskrivelse

Heletall er tall som ikke har desimaler. Heltall brukes for eksempel til å representere et antall for eksempel antallet spiller i et spill eller en posisjon i en rekkefølge. Heltall er mer naturlig for en datamaskin å arbeide med enn flyttal. Det er for at alle heltall lett kan representeres med et binært tall som består av 1er og 0er.

I programmeringsspråket Python heter datatypen for helttall `int`, som er kort for integer.

Flyttall

Datatype som brukes til å representere desimaltall

Beskrivelse

Et flyttal er et desimaltall. Innen all programmering bruker man `"."` istedefor `","` for å angi desimalen i et desimaltall eller flyttal.

Grunnen til at man velger å ikke bruke ordet desimaltall er for at desimaltall for det meste ikke går å representeres eksakt. For eksempel består desimaltallet $\sqrt{2}$ av uendelig mange desimaler og må i en datamaskin representeres på en avrundet og unøyaktig måte. I Python er $\sqrt{2}\simeq 1.4142135623730951$.

Flyttal er vanskeligere å representere eksakt for en datamaskin enn heltall. Dette er for at en datamaskin i utgangspunktet bare forstår 1er og 0er. Et heltall kan representeres med disse siffrene gjennom binære tall mens et flyttal må representeres på en annen måte for eksempel må alle desimaltall representeres med et endelig antall desimaler.

Ordet *flyt* (engelske *float*) kommer fra at plasseringen av desimaltegnen flyter. Vi kan for eksempel tenke oss at `0.03` kan representers med heltallet 3 hvor desimaltegnet er flyttet 2 plasser. Hvordan et flyttal faktisk representeres på går vi ikke inn på her men i Python kan dette med at flyttall er vanskelige å representer vises med følgende eksempel:

```
print(3*0.1)
```

Dette resulterer i utskriften: `0.30000000000000004` som nesten er `0.3` men det *er* ikke eksakt `0.3`.

Binære tall

Tall som representeres med sifferene 1 og 0

Beskrivelse

Binære tall er tall som representerers med sifferene 1 og 0 for eksempel 1011 er et binært tall. For å regne ut verdien til tallet ganger vi sifferverdien med plassverdien, akkurat som vi gjør med *vanlige tall* fra titall systemet. Plassverdien til de 4 første posisjonene i et binært tall er: 8, 4, 2, 1. Verdien til 1011 kan vi derfor regne ut til å være: $1\cdot8 + 0\cdot4 + 1\cdot 2 + 1\cdot 1 = 11$. Inne i en datamaskin lagres **allt** som binæretall. Dette er for at en datamaskin består av veldig mange strømbrytere som enten kan være av (0) eller på (1). TODO: Skal vi skrive mer dette temaet?

Streng

Datatype som brukes til å representere tekst-verdier

Beskrivelse

Innen programmering kalles en tekst for en streng (*string* eng.). Ordet kommer fra at man lagrer en *streng* med enkeltbokstaver. Mange ganger kan det være nyttig å tenke på en streng som en liste med enkelt bokstaver (*characters* eng.) men til forskjell fra lister kan ikke enkelt elementene endres uten å lage en ny streng.

Strenger kan legges sammen med streng operatoren +. Denne operasjonen kalles da å konkatenere to strenger, men oftest sier man gjerne bare å *legge sammen* to strenger.

```
fornavn = "Elise"
etternavn = "Sandberg"
print("Eleven heter " + fornavn + " " + etternavn)
```

Det finnes flere ulike operasjoner som kan gjøres på og med strenger... TODO: Skal vi forklare alle de her?

Boolean

Datatype som brukes til å representere sanne og falske verdier

Beskrivelse

Verdier av datatypen boolean kalles boolske verdier og de anvends til å representere sanne og falske verdier. Man bruker boolske verdier til å svare på betingelser eller til å representere svar på *ja- og nei-spørsmål*. Boolske verdier er også resultatet av boolske operatorer.

Liste

En samling av verdier som er ordnet i en rekkefølge

Beskrivelse

En liste (*array* eng.) brukes når du har mange verdier som hører sammen. Verdiene i en liste kalles elementer og er ordnet i en rekkefølge. Verdien til hvert element hentes ut fra listen gjennom å bruke nummeret i rekkefølgen elementet har i listen. Dette nummeret kalles for indeksen til elementet. Du kan lagre listen i en variabel slik at du kan bruke verdiene seinere i et program.

```
deltagere = ["Rebecca", "Erik", "Selma", "Amanda"]
deltager = deltagere[2]
deltagere[1] = "Svein"
```

`deltagere` er en list-variabel med fire elementer. Hvert element representerer navnene til deltagerne i en konkuranse. `deltager` er en variabel som har det *tredje* (ikke det andre!) elementet fra listen. Til slutt endres verdien til det andre elementet til `"Svein"`. Deltager `"Erik"` er altså byttet ut med `"Svein"`.

Element - liste

Verdiene i en liste kalles for elementer

Indeks - liste

En indeks brukes når man skal hente ut eller endre på elementene i en liste. Indeksen betegner plasseringen til verdien i listen og må altid angis med et heltall. Det første elementet har indeksen `0`, det andre har indeksen `1` og så videre. På rad to over brukes indeksen `2` til å hente ut verdien til det tredje elementet i listen `deltager`. På rad tre over brukes indeksen `1` til å endre det andre elementet i listen til verdien `"Svein"`.

Assosiativ liste

En samling verdier som hver er assosiert med en nøkkel

Beskrivelse

En assosiativ liste (*maps*, *dictionaries* eng.) brukes når du har mange verdier som hører sammen og hvor hver verdi knyttes til en (nøkkel). For eksempel kan man lagre verdier knyttet til en elev i en assosiativ liste hvor nøklene kan være `fornavn`, `etternavn`, `klasse` og `resultater`. Et element i en assosiativ liste er altså et nøkkel - verdi par hvor nøkkelen altid er en streng og verdien kan ha en hvilkorlig datatype.

Assosiative lister finnes i de fleste programmeringsspråk men heter da gjerne noe annet.

- python: `dict` - kort for *dictionary*
- JavaScript: `Object`
- Scratch: Det finnes ikke assosiative lister i Scratch (enn?!)

```
elev = {
    "fornavn": "Salma",
    "etternavn": "Eriksen",
    "klasse": "8A",
    "resultater": [4, 5, 3, 6, 5, 4]
}

# Hente ut element-verdier
fornavn = elev["fornavn"]
resultater = elev["resultater"]
```

```
# Endre element-verdier
elev["klasse"] = "9A"
```

Element - assositativ liste

Et element i en assositativ liste er en nøkkel - verdi par. Nøkkelen sier noe om hva verdien er (tenk variabelnavn) og verdien er en verdi som er akseptert i programmeringsspråket. I eksemplet over ser vi at verdiene er 3 strenger og en liste.

Operatorer

Enkelt tegn som bearbeider en eller flere verdier

Beskrivelse

En operator er et enkelt tegn som brukes til å bearbeide en eller to verdier. Operatorer er ofte en kilde til misforståelse når man lær seg programmering da de likner på tegn som brukes i hverdagsspråket og innen matematikk og i tilegg likner de ulike operatorene på hverandre.

Det som forener alle operatorer er at de bearbeider (opererer på) en eller flere (gjerne to) verdier og de returnerer *alltid* en verdi. Addisjons-operatoren `+` opererer på to tall og returnerer resultatet av addisjonen. For eksempel `3 + 4` adderer verdiene `3` og `4` og returnerer `7`. Addisjons-operatoren er en aritmetisk operator.

Det finns flere ulike typer operatorer. Disse skilles ved at de bearbeider og returnerer ulike typer verdier:

- aritmetiske operatorer: bearbeider og returnerer tall-verdier.
- strengoperatorer: bearbeider streng og tall-verdier og returnerer nesten alltid streng-verdier.
- boolske operatorer: bearbeider og returnerer boolske-verdier.
- sammenlignings operatorer: bearbeider ulike typer verdier men returnerer alltid en boolsk-verdi.
- tilordningsoperatorer: tilordner en variabel som er på venstre side av operatoren en verdi.

Operator-rekkefølge

Operatorer kan brukes til å sette sammen uttrykk: `7-2*3+2`. For at slike sammensatte uttrykk skal evalueres entydig finnes det regler for i hvilken rekkefølge operatorene skal utføres i (*operator precedence* eng.). I uttrykket `4-2*3+2` brukes de tre aritmetiske operatorene `-`, `+` og `*` og de utføres i samme rekkefølge som man gjør innen matematikken. Hvert programmeringsspråk har egne regler for hvilke rekkefølge operatorer utføres i. Disse bør en avansert programmererer kjenne til. Rekkefølgen for de aritmetiske operatorene er de samme for de aller fleste programmeringsspråk:

1. parentser
2. potenser
3. multiplikasjon og divisjon
4. addisjon og substraksjon Når to operatorer som har samme rekkefølge kommer etter heverandre i et uttrykk utføres operatorene fra venstre til høyre. `7-2*3+2` som: 1.`7-2*3+2`: Utfører ganger-operatoren 2.`7-6+2` 3.`1+2` 4.`3` Merk at hvis ikke operatorene utføres fra venstre til høyre ville f eks rad 2 bli noe annet: `7-6+2` $\rightarrow$ `7-8` som blir `-1`.

Unære og binære operatorer

En operator som bearbeider to verdier kalles en binær operator og en operator som bearbeider en verdi kalles en unær operator. `+` er et eksempel på en binær aritmetisk operator da den brukes til å legge sammen to verdier. `-` er et eksempel på en aritmetisk operator som både kan være unær og binær. `-` er unær når den representerer et negativt fortegn: `-5`, og binær når den representerer substraksjon: `7-4`.

Vanlige missforståelser ved bruk av operatorer.

- Tilordningsoperator må stå til venstre for et uttrykk. Man kan altså ikke skrive `4*6=x`
- Tilordningsoperatoren blandes ofte sammen med likhets operatoren
- `4a` betyr ikke `4*a`. Mener du variabelen `a` ganger 4 skriver du det eksplisit ved å bruker multiplikasjons-operatoren `*`.

Aritmetiske operatorer

Tegn som bearbeider en eller to tall-verdier

Beskrivelse

Aritmetiske operatorene er de som tilsvarer regnetegnene og fortegnene innen matematikken og bearbeider derfor bare tall-verdier.

- `+`: addisjon; `5+2` resulterer i `7`
- `-`: substraksjon (binært) og negasjon (unært); `5-2` resulterer i `3`
- `*`: multiplikasjon; `5*2` resulterer i `10`.
- `/`: divisjon; `5/2` resulterer i `2.5`

Operatorene over finnes i både Python og Scratch. Innen Python har vi i tilegg disse aritmetiske operatorene:

- `**`: potens; `5**2` resulterer i `25`.
- `//`: heltalsdivisjon; `5//2` resulterer i `2`.
- `%`: modulus; rest ved heltalsdivisjon, `5%2` resulterer i `1`.

Strengoperatorer

Tegn som bearbeider en eller to streng-verdier

Beskrivelse

Strengoperatorer brukes til å bearbeide strenger.

Boolske operatorer

Tegn som bearbeider en eller to boolske-verdier

Beskrivelse

Den enkleste måten å bearbeide en eller to verdier er å bruke en operator.

Sammenligningsoperator

Tegn som sammenligner to verdier og returnerer en boolsk verdi

Beskrivelse

En sammenligningsoperator sammenligner to verdier. Basert på verdiene og type operator returneres så resultatet av sammenligningen som en boolskverdi: `true` eller `false`.

Det finnes 5 ulike sammenligningsoperatorer:

`==`: likhet; `4==6` returnerer verdien `false` da 4 ikke er lik 6. `<`: mindre enn; `4<6` returnerer verdien `true` da 4 er mindre enn 6. `<=`: mindre enn eller lik; `7<=7` returnerer verdien `true` da 7 er mindre enn eller lik 7. `<`: større enn; `4>6` returnerer verdien `false` da 4 ikke er større enn 6. `>=`: større enn eller lik; `7>=4` returnerer verdien `true` da 7 er større enn eller lik 4.

Tilordningsoperatorer

Tegn som tilordner verdien til en variabel

Beskrivelse

Er man ny til programmering blandes dette tegnet ofte sammen med likhetstegnet fra matematikk som tilsynelatende er det samme. I matematikk betyr likhetstegnet at det som står på venstre side om likhetstegnet er lik det som står på høyre side.

Funksjon

Operasjoner som lagres til et navn

Beskrivelse

En funksjon brukes til å lagre et sett med operasjoner til et navn. Operasjonene som lagres i en funksjon må utføre en spesifikk oppgave slik at man bruker funksjonens navn istedenfor alle operasjonene når programmet skal utføre oppgaven. Når funksjonen brukes heter det at man kaller funksjonen. Når en funksjon blir kallet blir alle operasjonene til funksjonen utført og slik blir også oppgaven til funksjonen utført.

En funksjon likner på en variabel med at den har et navn og noe blir lagret til navnet, men en funksjon lagrer operasjoner hvor en variabel bare lagrer verdier.

Funksjonsnavn

Navnet til en funksjon

Beskrivelse

En funksjon har nesten altid et navn. Syntaksen til et programmeringsspråk legger noen krav til navnet. Disse krav er de samme som for variabelnavn.

Tips til funksjonsnavn

Bruk funksjonsnavn som beskriver hva funksjonen skal gjøre. Ettersom en funksjon utfører noe er det altid lurt å bruke et verb som første ord i navnet. For eksempel er `beregn_nytt_lån` eller `hent_resultater` gode navn til to funksjoner.

Kalle funksjon

Utføre operasjoner knyttet til en funksjon

Beskrivelse

Argument

Verdier man sender til en funksjon når man kaller den

Beskrivelse

Kommentar

Tekst som ikke utfører noen operasjoner

Beskrivelse

If-setning

En operasjon som gjør et valg

Beskrivelse

En if-setning brukes når man gjør et valg i sin kode. Valget gjøres på bakgrunn av verdien til en betingelse. Hvis betingelsen har verdien sant ytføres første kodeblokken. TODO: Fortsett...

Valg

Tas når kode kjøres på grunnlag av en betingelse

Beskrivelse

Når kode skal kjøres på grunnlag av en betingelse gjør man et valg. Oftest

Kodeblokk

Linjer med kode som hører sammen

Beskrivelse

TODO: Her mener vi scope...

Betingelse

Noe som må være sant for at kode skal utføres

Beskrivelse

I programmering

Løkke

Operasjoner som repeteres flere ganger.

Beskrivelse

For-løkke

Beskrivelse

While-løkke

Beskrivelse

Importere

Beskrivelse

Bibliotek

Beskrivelse

Tilfeldig tall

Et tall som velges tilfeldig

Beskrivelse